МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Е.И. Николаев

ОСНОВЫ NOSQL СУБД

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

для студентов специальности

09.03.02 Информационные системы и технологии

Ставрополь, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Программное и аппаратное обеспечение информационных систем неуклонно усложняются и одновременно расширяется сфера применения средств автоматизации, растет количество пользователей программных систем. Эти факторы предъявляют повышенные требования к процессам проектирования, разработки, внедрения и сопровождения приложений.

Экстенсивные пути развития информационных систем исчерпали себя и каждый специалист ищет пути устойчивого развития за счет распределения и распараллеливания вычислительной нагрузки. На первый план в сложном процессе развития ИТ-сферы выходят новые программные и аппаратные технологии: от распределенных систем хранения и управления данными до параллельных вычислительных систем, основанных на применении графических процессоров.

Пособие «Основы NoSQL СУБД» призвано обеспечить студентов всеобъемлющим теоретическим материалом, необходимым и достаточным для использования современных инструментальных средств в процессе разработки высокопроизводительных приложений и проектирования информационных систем с использованием перспективных технологий. В пособии детально рассмотрены основные концепции построения систем с использованием концепций параллелизма: типология и классификация многопроцессорных систем; принципы разработки программного обеспечения для параллельных и распределенных систем; программирование в рамках стандарта MPI; построение высокопроизводительных систем на базе GPU; разработка систем интенсивной и распределенной обработки данных; основы BigData.

Основная цель изучения дисциплины «Основы NoSQL СУБД»: изучение современных средств и методов разработки, проектирования и сопровождения высокопроизводительных информационных систем и

получение практических навыков построения параллельных приложений (многопоточных).

Задачами изучения дисциплины «Основы NoSQL СУБД» являются:

– изучение типологии многопроцессорных систем;

– изучение классификации и возможностей каждого семейства многопроцессорных систем;

– обучение основным принципам и технологиям параллельного программирования;

– изучение механизмов программирования с использованием механизма потоков;

– изучение основных принципов тестирования и отладки параллельных программ;

– изучение аппаратных средств высокопроизводительных ИС;

– изучение программных средств высокопроизводительных ИС.

Материал, усвоенный студентами с использованием данного пособия, позволит студентам реализовывать программное обеспечение информационных систем с использованием передовых технологий многопроцессорного, многопоточного и многоядерного программирования.

ЛАБОРАТОРНАЯ РАБОТА 1. ЗНАКОМСТВО С NOSQL СУБД MONGODB.

1. Цель и содержание

Цель лабораторной работы: знакомство с MongoDB.

Задачи лабораторной работы: научиться производить установку и запуск MongoDB.

2. Теоретическое обоснование

2.1 Введение в многомерный анализ данных. Знакомство с NoSQL.

Информационные системы любого крупного и серьезного предприятия в большинстве случаев содержат данные подверженные комплексному анализу, расчету тенденции, динамики и других характеристик. Подобного рода данные, как правило, хранятся в хранилищах данных и в конечном итоге требуют оперативного, быстрого доступа к ним.

В последнее время для задач оперативной обработки плохоструктурированных данных все большую популярность приобретают NoSQL (от англ.: «*Not Only SQL*» – «*Не только SQL*») БД. Они используются не только как элемент хранилища данных, но все чаще как само хранилище.

NoSQL – это ряд технологий, подходов, проектов направленных на реализацию моделей баз данных, имеющих существенные отличия от традиционных СУБД, работающих с языком SQL. Концепция NoSQL не отрицает SQL, она лишь стремится решить проблемы и вопросы, с которыми не достаточно хорошо справляется РСУБД. Чаще всего данные в NoSQL решении представляются в виде хеш-таблиц, деревьев, документов и других структур.

Концепция NoSQL предоставляет высокую доступность и устойчивость данных к разделению, но при этом в NoSQL не все хорошо с

согласованностью данных. Этот подход призван решать собственный класс задач.[1]

На данный момент существует большое количество NoSQL баз данных (полный список можно найти на сайте http://nosql-database.org/).

NoSQL БД, как правило, являются гибкими решениями, позволяющими масштабировать их на множество серверов с минимальными затратами времени и средств. Подобного рода решения подходят, в том числе, и для хранения слабо структурируемой, не структурируемой информации.

NoSQL БД обеспечивают высокую скорость выполнения, низкие затраты на масштабирование, хранение и обработку больших объемов данных.

Отличительной чертой NoSQL баз данных является отсутствие необходимости использовать реляционные модели данных.

Подобного рода базы данных в большинстве случаев не имеют GUI[2], либо он минималистичен. Взаимодействие с БД происходит через API[3], либо через сеть. Для работы с NoSQL БД требуется написать приложение, взаимодействующее с API БД и выполняющее требуемые функции. По сути, NoSQL БД выступает в роли хранилища данных разрабатываемого приложения, а вся логика работы с БД реализуется в приложении и возложена на плечи программиста. Хранилище выполняет основные функции по оперированию с данными: хранение, извлечение, поиск. Например, такая операция как соединение (JOIN), присутствующая в РСУБД, в NoSQL реализуется самим программистом. Так, в РСУБД база данных самостоятельно обрабатывает запрос, извлекая требуемые данные, в случае NoSQL БД, если необходимо произвести операцию соединения, то программист её реализует сам. Забегая вперед, скажем, что аналогом таблицы в NoSQL БД является коллекция.

---

[1] Что такое NoSQL и MongoDb? [Электронный ресурс]. – Режим доступа: http://www.andrey-vasiliev.com/no-sql/chto-takoe-nosql-i-mongodb/

[2] GUI – (от англ. «*graphical user interface*») графический пользовательский интерфейс

[3] API – (от англ. «*application programming interface*») интерфейс программирования приложений

Ввиду отсутствия необходимости использовать реляционные модели данных в NoSQL, данные могут храниться в абсолютно не структурированном виде. Так, в любой документ можно добавить произвольное поле. Документ может содержать вложенный документ, образуя иерархию. Поля в БД определяются на уровне документа. Это значит, что любой документ может иметь уникальный набор полей, отличный от других документов. На рисунке 1 представлен пример документа, включающего помимо полей вложенные документы. Так как база данных не хранит модель данных, изменение модели данных требует изменить программный код приложения, не внося никаких изменений в БД.

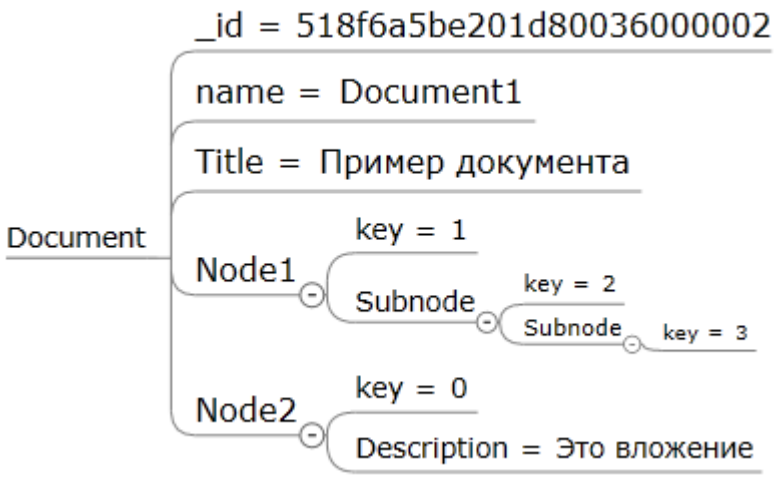


Рисунок 1 – Пример документа в NoSQL БД.

Обозначим некоторые характеристики, присущие NoSQL базам данных:

– в NoSQL СУБД не используется SQL (имеется в виду DML[4]);

– отсутствие необходимости хранить данные с определенной структурой;

– высокая производительность решений, высокая доступность;

– распределенное хранение данных;

[4] DML – (от англ. «*Data Manipulation Language*») язык управления (манипулирования) данными

– способность к горизонтальному масштабированию по требованию для некоторого набора операций на многих серверах;

– эффективное использование распределенных индексов и памяти для запросов;

– отсутствие транзакций;

– простые протоколы доступа к хранимым данным;

– отсутствие поддержки транзакционной целостности ACID (atomicity, consistency, isolation, durability – атомарность, согласованность, изолированность, долговечность).

Для того чтобы понять схему представления данных в NoSQL БД, необходимо вспомнить, как данные представлены в реляционных БД (РБД).

РБД представляет собой набор доменов (отношений, таблиц), имена которых совпадают с именами схем отношений в схеме БД. Отношения состоят из множества кортежей, соответствующих одной схеме отношения (строк). Кортеж (набор именованных значений заданного типа) состоит из множества атрибутов (столбцов). Между сущностями существуют связи. Если бы назначением базы данных было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой. Однако одно из основных требований к организации базы данных – это обеспечение возможности отыскания одних сущностей по значениям других, для чего необходимо установить между ними определенные связи[5]. Для структурирования данных существует процесс нормализации.

Нормализация – это процесс структурирования модели данных, обеспечивающий связность и отсутствие избыточности в данных.

В NoSQL БД данные хранятся в виде пар ключ-значение. Любой записи в БД соответствует ключ. БД состоит из коллекций. Коллекция является эквивалентом таблицы. База данных может иметь нуль или больше коллекций. Коллекция состоит из документов, следовательно, эквивалент

[5] Лекции по информатике. Основные понятия Баз Данных [Электронный ресурс]. – Режим доступа: http://gendocs.ru/v33068/лекции_по_информатике._основные_понятия_ баз_данных

кортежа – документ. Документы представлены как объекты, с полями и значениями, представляющими пары ключ-значение. Документы состоят из полей, которые подобны атрибутам. Как отмечалось ранее, коллекция не содержит информацию о структуре содержащихся в ней данных, подобного рода информацию содержит каждый отдельный документ.

База данных имеет индексы. Индексы в NoSQL БД почти эквивалентны индексам в РБД.

NoSQL БД, как правило, вместо данных возвращают курсор, с которым мы можем оперировать (подсчитывать записи, пропускать их) не загружая сами данные. Курсор – получаемый при выполнении запроса результирующий набор и связанный с ним указатель текущей записи.

Теперь попробуем представить базу данных, содержащую информацию о мобильных телефонах в реляционном и не реляционном виде. Допустим, описание телефона имеет следующие параметры: марка телефона, модель, семейство ОС и её версия.

В реляционном виде, после проведения операции нормализации, модель данных будет выглядеть так, как представлено на рисунке 2.

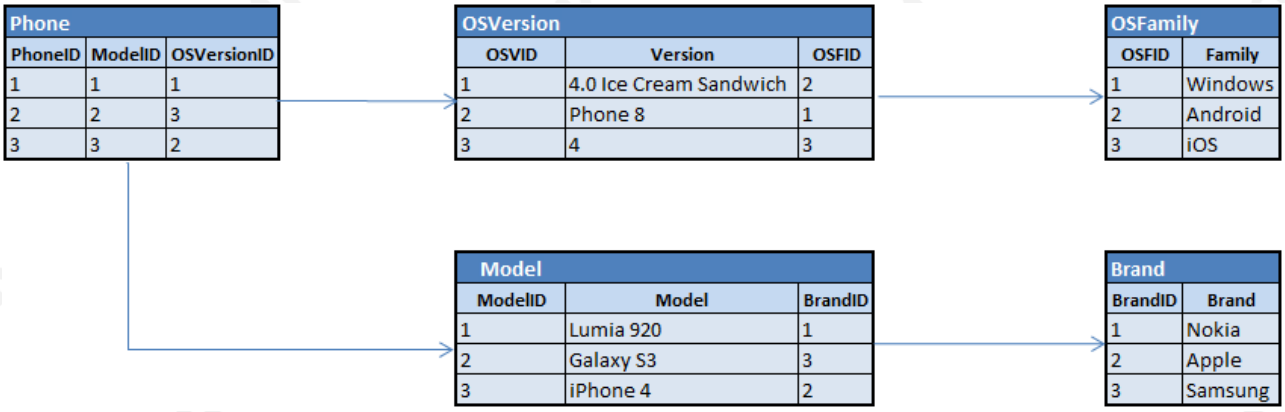


Рисунок 2 – Схема базы данных «телефон».

В не реляционном виде база данных будет хранить три документа, соответствующих каждой из хранимых записей. Причем каждый документ может содержать присущие только ему поля. Модель БД в не реляционном виде представлена на рисунке 3.

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 3 – База данных «телефон» в не реляционном виде.

2.2 Знакомство с MongoDB.

Одной из самых популярных NoSQL СУБД в настоящее время является MongoDB. MongoDB – это документно-ориентированная база данных с открытым исходным кодом.

Основными особенностями MongoDB являются:

– документно-ориентированное хранилище;

– полная поддержка индексов;

– репликация данных;

– высокая доступность данных;

– способность к горизонтальному масштабированию;

– авто-шардинг;

– поддержка запросов;

– поддержка Map/Reduce;

– поддержка GridFS[6].

Более подробное описание БД и её особенностей представлено на официальном сайте.

[6] MongoDB [Электронный ресурс]. – Режим доступа: http://www.mongodb.org/

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.

MongoDB обладает хорошей скоростью работы с данными (чтения/записи), хорошей масштабируемостью. Благодаря отличным реализациям репликации и шардинга, базу данных mongo легко реплицировать на кластер компьютеров или настроить шардинг (возможность разнести данные по нескольким серверам). Кроме того MongoDB обладает системой распределенных вычислений с высокой степенью отказоустойчивости.

Репликация – это тиражирование изменений данных с главного сервера БД на одном или нескольких зависимых серверах.

Шардинг – разделение данных на уровне ресурсов, разбиение данных по какому-либо признаку. Концепция шардинга заключается в логическом разделении данных по различным ресурсам.

Для управления документами используется нотация JSON, для их хранения – BSON (более подробно форматы хранения и обмена данными будут рассмотрены в лабораторной работе №3).

MongoDB не поддерживает модель транзакционной целостности ACID. Это означает, что в MongoDB отсутствует понятие «транзакция». Например, данные, изменяемые одним клиентом, одновременно могут читаться другим. Атомарность присутствует, но только на уровне целого документа.

2.3 Установка MongoDB.

Для начала работы с MongoDB её необходимо установить. К методическим указаниям прилагаются архивы с MongoDB, для различных операционных систем. Установка и работа с MongoDB в методических указаниях рассматривается на примере операционной системы Windows7.

Для других операционных систем семейства Windows действия по установке/настройке и работе с MongoDB аналогичны.

В подпапке «Windows» папки «MongoInstall» необходимо выбрать архив, подходящий к версии вашей ОС: «mongodb-win32-i386-2.4.3.zip» для 32-разрядной ОС и «mongodb-win32-x86_64-2.4.3.zip» для 64-разрядной ОС. Предпочтительной является версия для 64-х битной ОС, т.к. 32-х битная версия имеет ограничения функциональности. Максимальный размер БД в 32-разрядной версии приложения составляет 2 Гб.

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. Содержание папки «bin» представлено на рисунке 4.

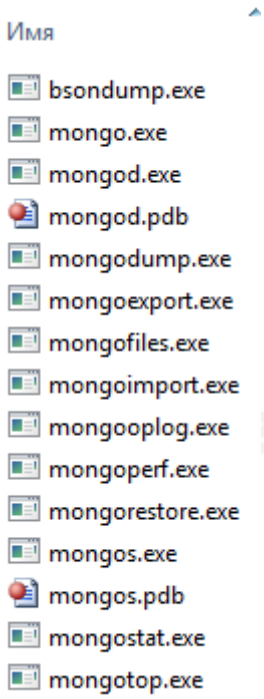


Рисунок 4– Приложения, входящие в состав MongoDB

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.

Основными приложениями являются 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» является сервером базы данных.

Приложение «mongos.exe» предназначено для шардинга данных. Оно представляет собой сервис маршрутизации для конфигурирования шардов, обрабатывает запросы от уровня приложений и определяет местоположение данных в кластере.

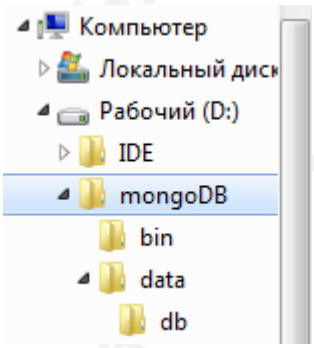
Приложение «mongo.exe» является консольным клиентом для «mongod.exe», и представляет собой интерфейс тестирования запросов для работы с базой данных и её администрирования.

Обратите внимание, что если ваш ПК работает под управление ОС Windows 7, вам необходимо установить обновление (http://support.microsoft.com/kb/2731284) для разрешения проблемы с маппингом памяти в консольных приложениях.[7]

Данное обновление автоматически устанавливается через центр обновлений Windows, если он у вас включен.

Обновление также прилагается к методическим указаниям: «3rd parties\Fix405791_x32_zip.exe» для 32х систем, и «3rd parties\Fix405791_x64_zip.exe» для 64х систем.

После извлечения файлов mongo, необходимо определить место хранения базы данных. По умолчанию MongoDB ищет файлы БД в папке «C:\data\db». Вы можете создать папку для хранения базы данных в любом месте. У вас должна получиться иерархия, подобная представленной на рисунке 5.



[7] Install MongoDB on Windows [Электронный ресурс]. – Режим доступа: http://docs.mongodb.org/manual/tutorial/install-mongodb-on-windows/

Рисунок 5 – Иерархия папок в MongoDB

MongoDB можно установить в любую папку на компьютере. Для установки достаточно скопировать папки MongoDB в требуемое место.

Mongo может работать в двух режимах:

– в качестве обычного приложения Windows;

– в качестве Windows-сервиса.

В случае использования mongo как сервиса, он автоматически запускается при старте системы.

2.4 Запуск MongoDB как Windows приложения

Для запуска MongoDB как Windows приложения необходимо запустить приложение «mongod.exe» с помощью командной строки. Запуск MongoDB для папки «D:\mongoDB\bin» можно произвести следующим образом:

```
D:\mongoDB\bin\mongod
```

или

```
D:
cd \mongoDB\bin
mongod
```

Если у вас папка с базой данных располагается в месте, отличном от стандартного размещения базы данных mongo, то вам необходимо произвести запуск mongo с параметром «--dbpath». Синтаксис команды следующий:

```
--dbpath <путь к папке>
```

Следующий пример показывает, как запустить «mongod.exe» с произвольным путем к папке с БД:

```
mongod --dbpath=D:\mongodb\db
D:\mongoDB\bin\mongod --dbpath=D:\mongodb\db
```

или

```
mongod --dbpath= ../mongodb/db
D:\mongoDB\bin\mongod --dbpath=../mongodb/db
```

В случае успешного запуска консоль перейдет в режим ожидания подключения как показано на рисунке 6.

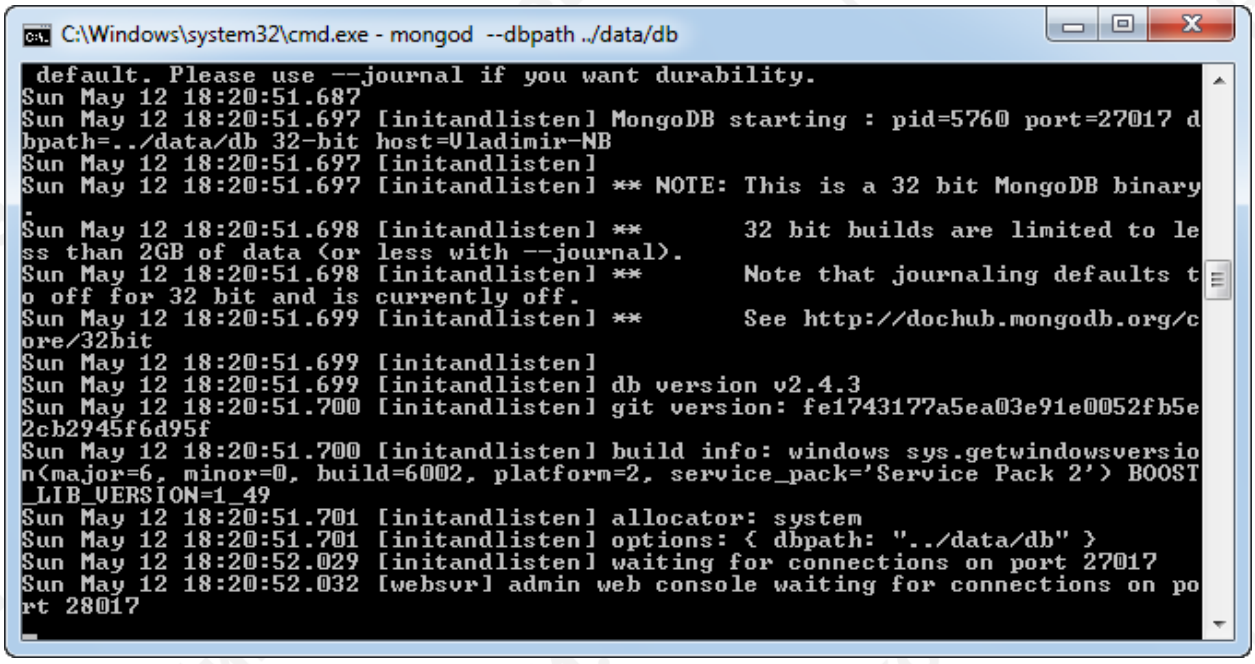


Рисунок 6 – Экранная форма консоли ожидающей подключения клиента

Если после попытки запуска приглашение на ввод команды повторилось, то произошла ошибка запуска. Одной из наиболее частых проблем является изначально неправильно указанный путь к папке с БД. На рисунке 7 показан пример ошибки с неправильно указанным путем к базе данных.

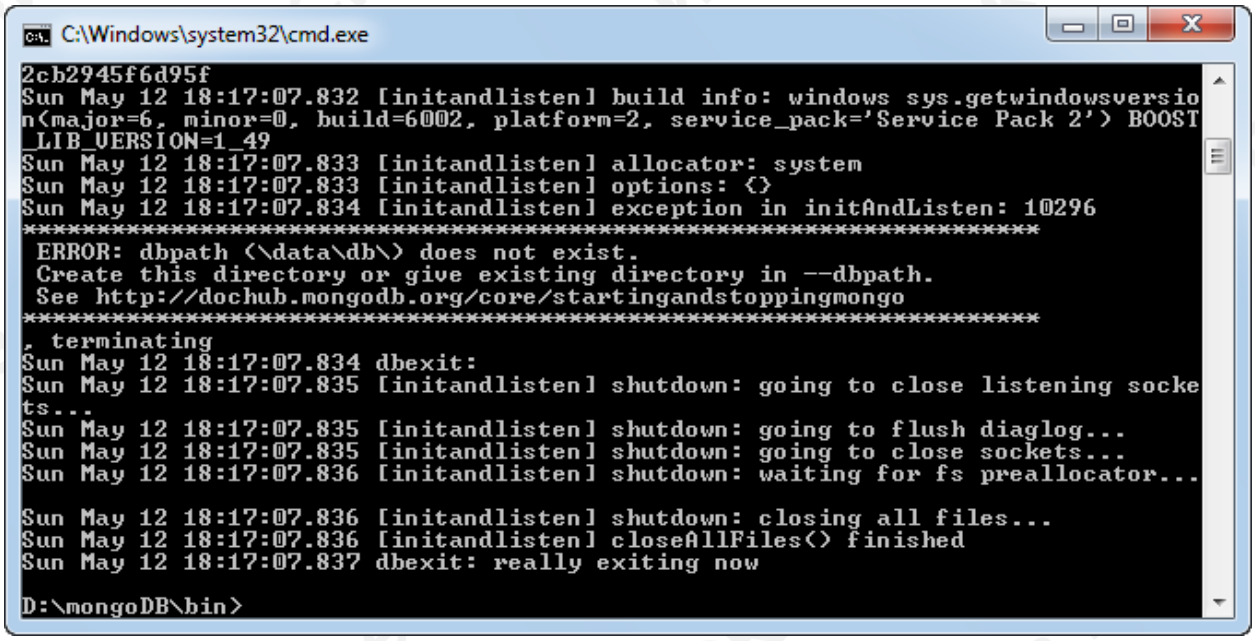


Рисунок 7 – Экранная форма консоли с сообщением об ошибке

Для решения проблемы необходимо указать явно путь к БД. Синтаксис команды «--dbpath» и пример её использования приведены выше.

После успешного запуска окно приложения «mongod.exe» можно свернуть.

2.5 Запуск MongoDB как сервис Windows

Обратите внимание, что для установки и запуска MongoDB в качестве сервиса Windows вам потребуются права администратора.

Для использования MongoDB в качестве сервиса необходимо произвести конфигурирование системы: задать пути для логирования и файлов конфигурации.

Первым делом 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 вам следует создать текстовый файл «mongod.cfg» в папке «D:\MongoDB», и поместить в него строку с путем к файлу отчета: «logpath=<путь к файлу с отчетом>», например:

```
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=<путь к БД>», например:

```
dbpath=D:\mongodb\data\db
```

Содержание файла конфигурации показано на рисунке 8.

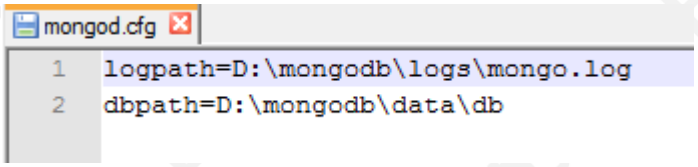


Рисунок 8 – Содержание файла конфигурации MongoDB.

После создания файла конфигурации необходимо произвести установку MongoDB.

Для установки запустите командную строку с правами администратора. Для этого введите «cmd» в строку поиска в меню «Пуск», активируйте контекстное меню приложения «cmd.exe» и выберите пункт меню «Запуск от имени администратора». Запуск командной строки от имени администратора представлен на рисунке 9.

Далее следует запустить «mongod.exe», передав ему в качестве параметра путь к файлу конфигурации и команду на установку. Формат команды: «<путь к mongo> – –config <путь к файлу конфигурации> –install», например:

```
mongod --config D:\mongodb\mongod.cfg -install
D:\mongodb\bin\mongod --config D:\mongodb\mongod.cfg -install
```

При удачной установке вы получите 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.

Также возможно установить mongo как Windows-сервис без создания файла конфигурации, для этого необходимо произвести установку следующей командой:

D:\mongodb\bin\mongod.exe --dbpath=D:\mongodb --logpath=D:\mongodb\log.txt –install

Где:

–-dbpath=<ваш путь к базе данных>,

--logpath=<ваш путь к папке с отчетами>.

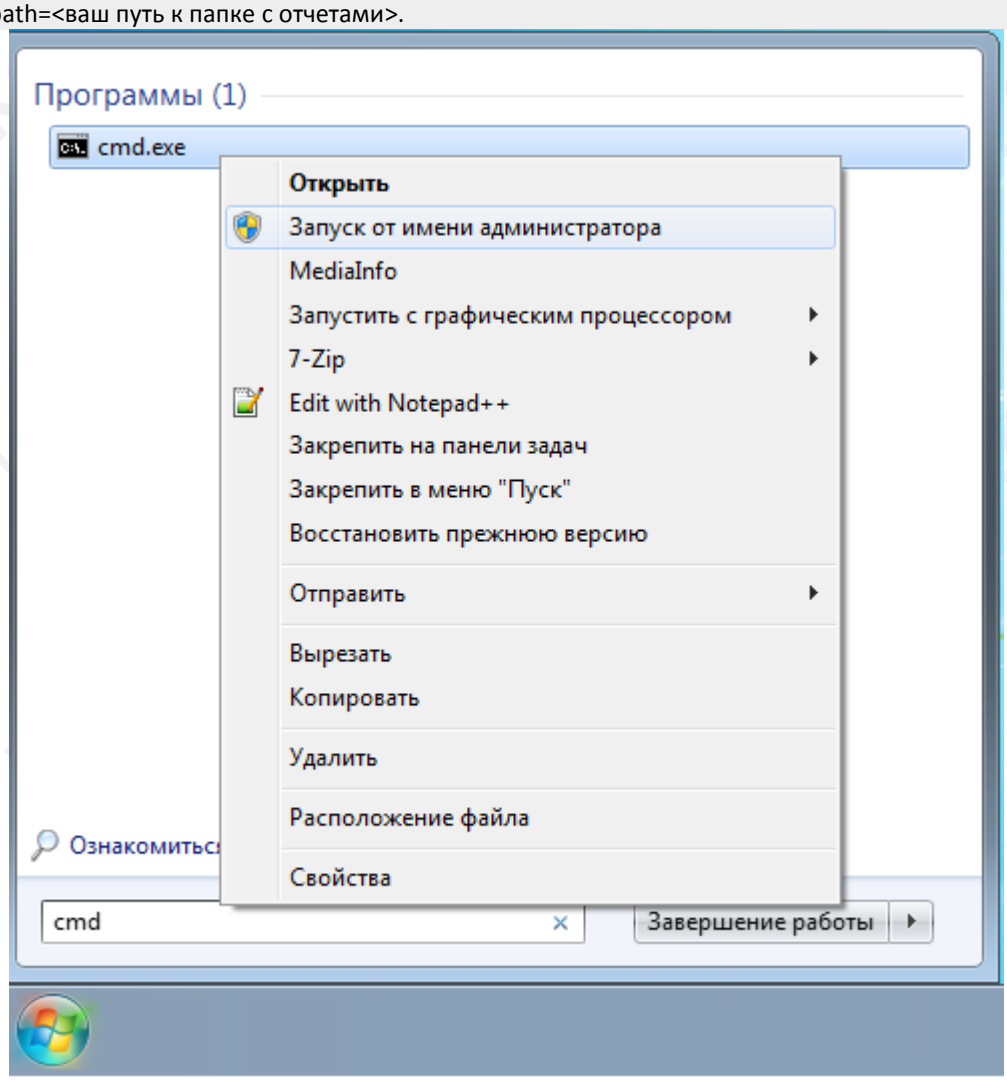


Рисунок 9 – Запуск командной строки от имени администратора

2.6 Подключение к MongoDB

Поле запуска сервера необходимо запустить клиент. Для запуска клиента вам необходимо запустить приложение «mongo.exe» через командную строку.

При запуске клиент автоматически подключится к серверу.

Как показано на рисунке 10, 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:

```
db.getName()
```

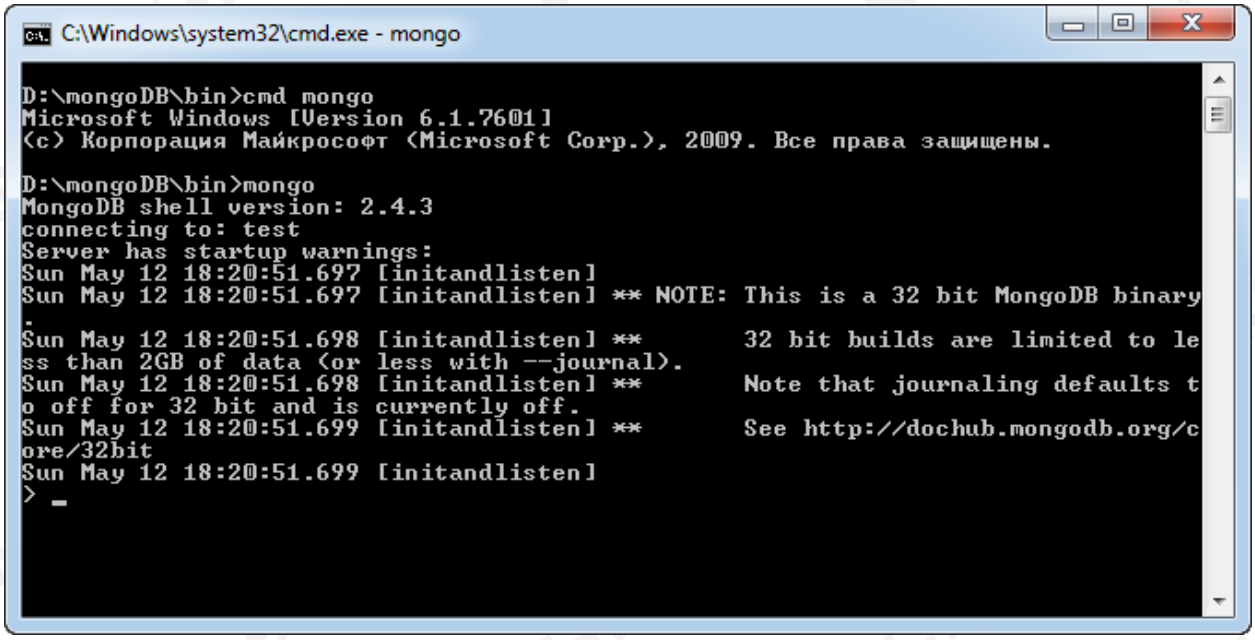


Рисунок 10 – Экранная форма окна приложения «mongo.exe»

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 следующую команду:

```
db.test.find()
```

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу

лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

1. Установите MongoDB одним из описанных выше способов.
2. Произведите подключение к тестовой базе данных.
3. Добавьте произвольные данные в БД.
4. Извлеките добавленные на предыдущем шаге данные.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Что означает термин NoSQL?
2. Какие преимущества предоставляют NoSQL базы данных в сравнении с реляционными базами данных?
3. Какими особенностями обладает MongoDB?
4. Сколькими способами можно произвести установку MongoDB? Кратко опишите эти способы.

5. На какие группы делятся приложения, входящие в состав MongoDB?

6. Создает ли MongoDB по умолчанию какую-либо базу данных? Если создает, назовите её имя.

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 2. ЗНАКОМСТВО С КОНСОЛЬЮ MONGODB И СПОСОБАМИ ВЗАИМОДЕЙСТВИЯ С БД.

1. Цель и содержание

Цель лабораторной работы: знакомство с консолью mongo, знакомство с утилитами для взаимодействия с базой данных mongo.

Задачи лабораторной работы: научиться взаимодействовать с базой данных mongo посредством консоли и пользовательских интерфейсов сторонних фирм.

2. Теоретическое обоснование

2.1 Обзор способов взаимодействия с MongoDB

Как было отмечено в лабораторной работе №1, использование MongoDB приложениями происходит через прикладной интерфейс программирования. Компания 10gen (разработчик MongoDB) помимо самой БД также разрабатывает и поддерживает драйвера, необходимые для разработки приложений с использованием MongoDB. Взаимодействие с БД через API является основным способом взаимодействия с базой данных.

Помимо использования драйверов с сервером баз данных mongo можно взаимодействовать через графические утилиты, разрабатываемые сторонними фирмами.

Различные графические утилиты, предназначенные для взаимодействия с базами данных mongo, позиционируются как средства администрирования.

Список административных интерфейсов представлен по адресу: http://docs.mongodb.org/ecosystem/tools/administration-interfaces/.

Через них можно выполнять как запросы к базе данных, так и административные функции.

MongoDB в своем составе не содержит графического интерфейса администрирования. Большинство административных функций выполняется через командную строку.

С MongoDB также можно взаимодействовать посредством http. Взаимодействие происходит с использованием либо HTTP, либо REST интерфейса[8]. Более подробная информация о взаимодействии с mongo через HTTP находится в документации, и представлена по адресу: http://docs.mongodb.org/ecosystem/tools/http-interfaces/.

2.2 Интерфейс командной строки

«Mongo.exe» – это интерактивная оболочка JavaScript интерфейса для MongoDB, которая предоставляет мощный интерфейс для системных администраторов, а также позволяет разработчикам тестировать запросы и операции непосредственно с базой данных. «Mongo.exe» предоставляет также полнофункциональную JavaScript среду для использования с MongoDB.

«Mongo.exe» является консольным приложением, её внешний вид представлен на рисунке 1.

В консоли mongo есть несколько глобальных команд, например «help», «use».

Команда «help» позволяет получить краткую справку по командам консоли mongo.

Команда «use» позволяет выбрать используемую базу данных.

Синтаксис команды: «use <имя базы данных>».

Примечание: выбираемая база данных может и не существовать на момент ввода команды, в таком случае MongoDB автоматически создаст БД при создании первой коллекции.

Не глобальные команды применяются к коллекциям или базе данных. Команды, которые используются применительно к текущей базе данных,

[8] **REST** (от англ. «*Representational State Transfer*» – «*передача состояния представления*») – стиль построения архитектуры распределенного приложения, простой интерфейс управления информацией.

начинаются с указания объекта базы данных: «db», например «db.help()» или «db.stats()». Команды, которые используются применительно к конкретной коллекции, помимо объекта базы данных используют также имя коллекции: «db.<имя коллекции>», например, «db.test.help()» или «db.test.count()».

Команда «help», применительно к любому объекту, позволяет получить список команд этого объекта.

Перечень команд консоли представлен на странице http://docs.mongodb.org/manual/reference/method/.

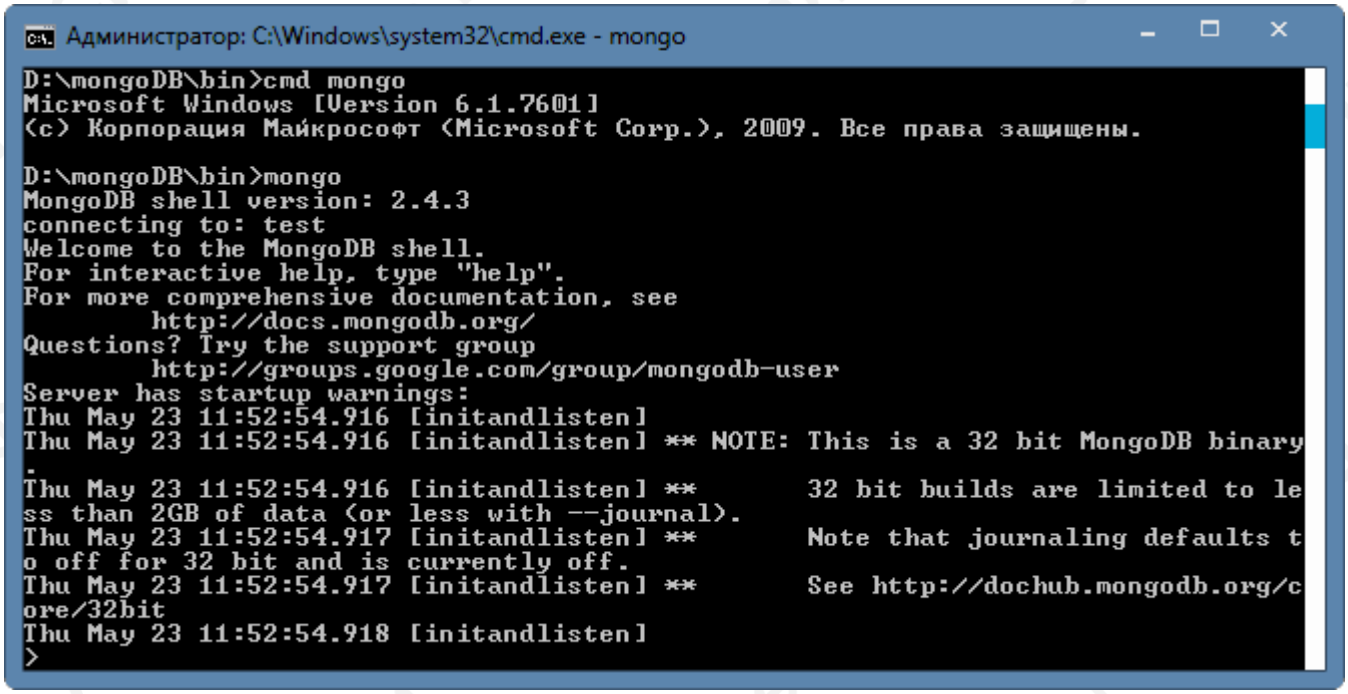


Рисунок 1 – Экранная форма консоли MongoDB

Для получения имени используемой БД необходимо ввести команду «getName» применительно к текущему объекту БД:

```
db.getName()
```

Добавление документа в БД производится с помощью команды

```
db.test.save( {name: "test parameter"})
```

Синтаксис команд, применимых к коллекции, следующий: «<объект бд>.<коллекция>.<функция> (<список параметров>)».

Список параметров задается в формате JSON, и будет рассмотрен в следующей лабораторной работе.

Для добавления записи необходимо указать имя записи и её значение через двоеточие. Обратите внимание, что каждая команда «save» создает новый документ.

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 следующую команду:

```
db.test.find()
```

2.3 Знакомство с MongoExplorer

MongoExplorer – это инструмент управления MongoDB.

Основные особенности приложения:

– MongoExplorer легок в использовании;

– отображает все коллекции и документы базы данных;

– использует удобное дерево для отображения документов;

– полностью поддерживает drag’n’drop;

– in-place редактирование документов.

MongoExplorer представляет собой приложение, написанное на SilverLight. MongoExplorer предоставляет основные возможности по взаимодействию с БД mongo: добавление, редактирование и удаление коллекций/документов/полей. Интерфейс приложения не перегружен и идеален для знакомства с БД. В отличие от командной строки, взаимодействие с БД происходит с использованием, привычного для большинства людей, графического интерфейса.

Для использования приложения MongoExplorer вам необходим установленный на компьютере Microsoft Silverlight. С методическими указаниями поставляется Silverlight версии 5. Скачать актуальную версию Silverlight можно на сайте: http://www.microsoft.com/silverlight/

Как показано на рисунке 2, в интерфейсе MongoExplorer можно выделить пять основных панелей:

1. Панель работы с коллекциями.
2. Панель работы с документами.
3. Панель ввода запросов.
4. Панель команд.
5. Область работы с документом.

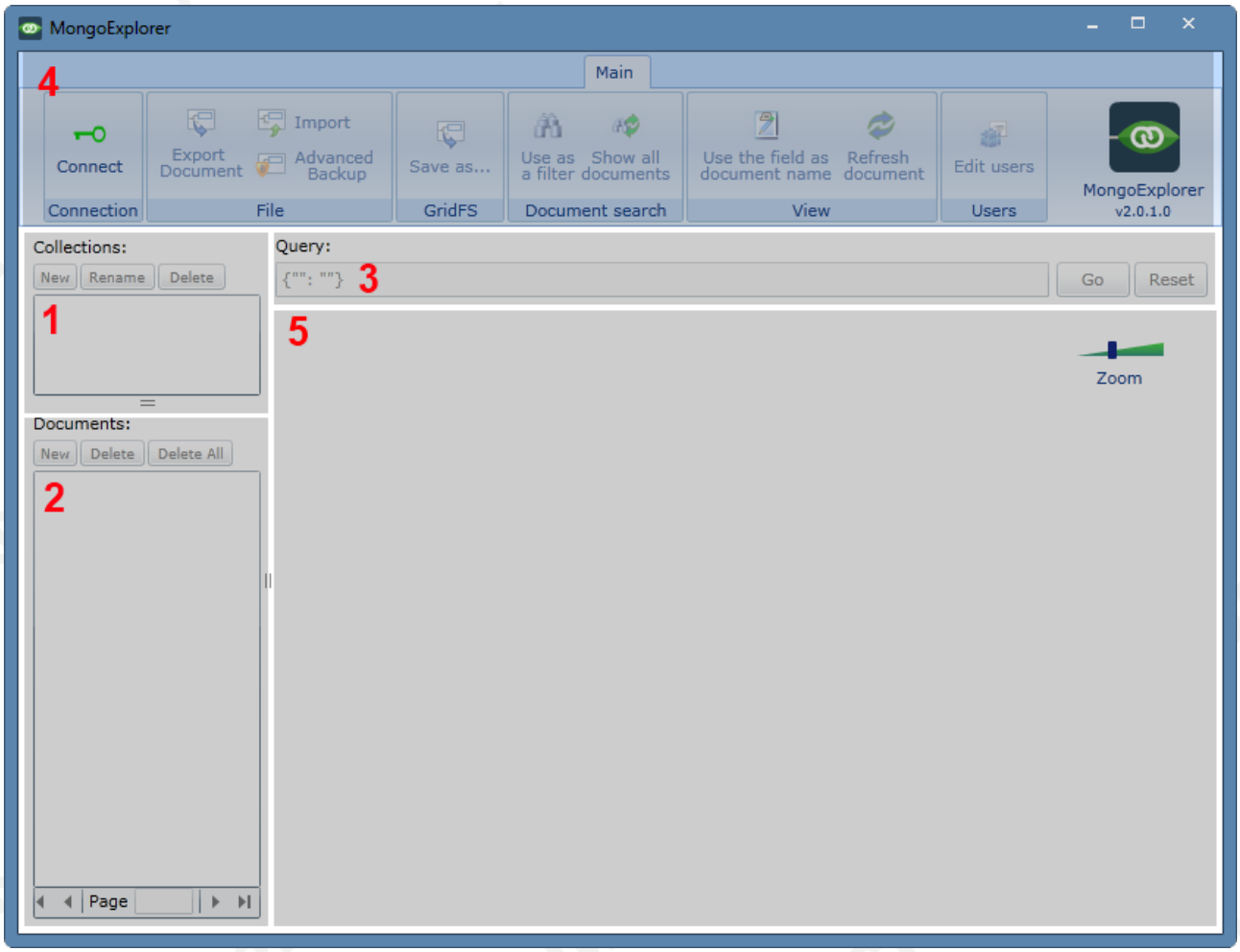


Рисунок 2 – Экранная форма приложения Mongo Explorer,

Рассмотрим каждую панель подробнее.

Панель работы с коллекциями содержит перечень коллекций, присутствующих в базе данных, а также предоставляет средства для создания, переименования и удаления коллекций.

Панель работы с документами, также содержит перечень всех документов базы данных, и кнопки управления документами: создание документов и их удаление.

Панель ввода запросов представляет собой поле, предназначенное для ввода запроса. Результат исполнения запроса отображается в панелях 2 и 5.

Панель команд содержит следующие команды:

Подключение к базе данных (и отключение от нее).

Экспорт документа в JSON формате.

Импорт документа в JSON формате.

Создание резервной копии БД.

Сохранение файла на диск (при использовании GridFS[9]).

Поиск по документу и использованием фильтра.

Отображение всех документов.

Использовать значение поля в качестве имени документа.

Обновить документы.

Работа со списком пользователей.

Область работы с документом представляет документ в графическом виде.

2.4 Взаимодействие с базой данных mongo через Mongo Explorer

Для подключения необходимо кликнуть по кнопке «Connect», находящейся в панели команд.

В появившемся окне, представленном на рисунке 3, необходимо указать имя базы данных, к которой производится подключение. Далее необходимо кликнуть по кнопке «ОК». Если никаких сообщений об ошибках не поступало, то подключение прошло успешно.

---

[9] GridFS – это спецификация для хранения файлов, размер которых превышает размер BSON-документа (16 МБ).
GridFS – это возможность MongoDB хранить файлы любых видов и размеров в самой БД, используя при этом преимущества шардинга и репликации.

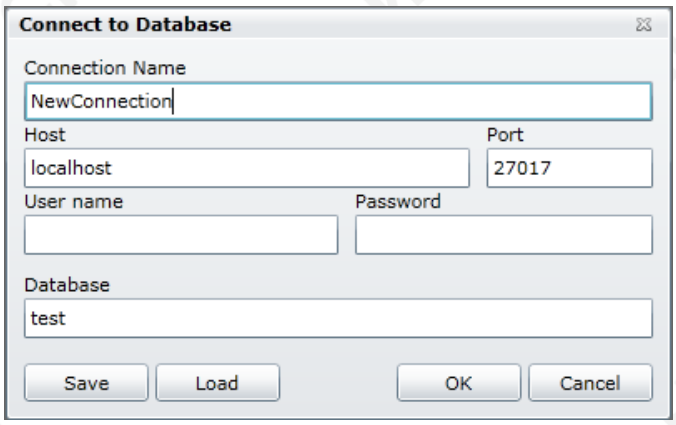


Рисунок 3 – Экранная форма окна подключения к базе данных.

После подключения к базе данных в области коллекций отобразится список существующих в базе данных коллекций.

Для просмотра документов принадлежащих коллекции необходимо кликнуть на имени коллекции.

Список коллекций базы данных и документов, принадлежащих первой коллекции, показан на рисунке 4.

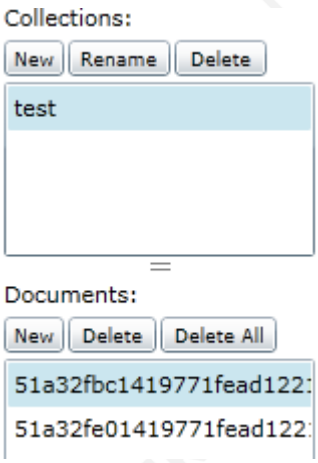


Рисунок 4 – Список коллекций базы данных

Для добавления поля в документ необходимо:

– выбрать документ в списке документов

– активировать контекстное меню на имени документа в области документа, как показано на рисунке 5.

– выбрать пункт меню «Insert child»

– в появившемся окне ввести имя и выбрать тип записи. Экранная форма окна представлена на рисунке 6.

– для ввода значения записи необходимо произвести двойной щелчок мышью на значении записи, и изменить её, как показано на рисунке 7.

Удаление записи производится с помощью пункта «Remove» контекстного меню, либо с помощью кнопки «Delete».

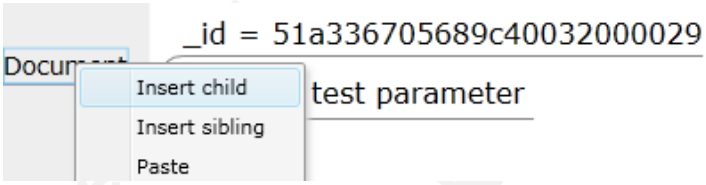


Рисунок 5 – Добавление поля к документу

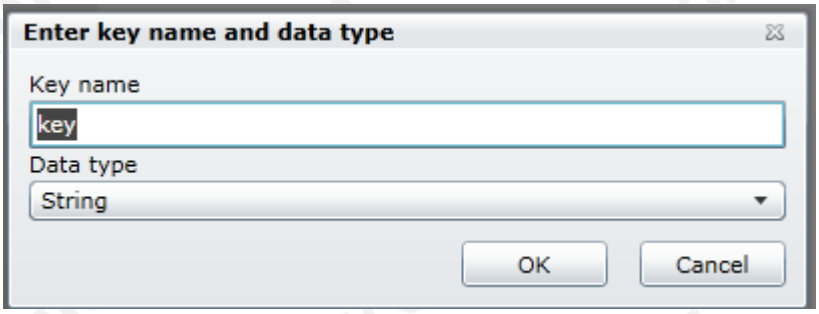


Рисунок 6 – Ввод имени, и выбор типа добавляемой записи

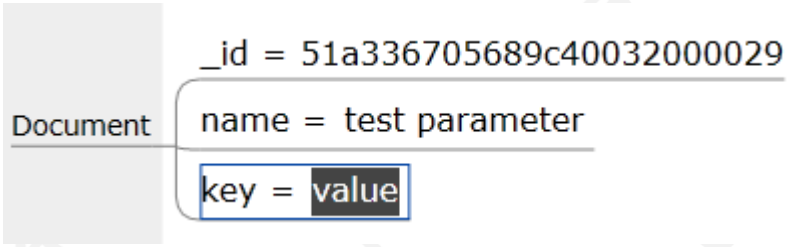


Рисунок 7 – Изменение значения записи

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше,

оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

Индивидуальное задание

1. Произведите подключение к тестовой базе данных.

2. Добавьте произвольные данные в базу данных с использованием командной строки.

3. Извлеките добавленные на предыдущем шаге данные с помощью командной строки.

4. Добавьте произвольные данные в базу данных с использованием Mongo Explorer.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Какие существуют способы взаимодействия с БД mongo?

2. Существует ли возможность взаимодействовать с БД mongo посредством WEB?

3. Какими особенностями обладает Mongo Explorer?

4. Опишите интерфейс приложения Mongo Explorer.

5. Каков формат команд командной строки?

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 3. ФОРМАТЫ ОБМЕНА ДАННЫМИ В MONGODB. МОДЕЛИРОВАНИЕ ДАННЫХ.

1. Цель и содержание

Цель лабораторной работы: знакомство с форматами обмена и представления данных в MongoDB, знакомство с моделированием данных.

Задачи лабораторной работы: научиться переводить модели данных реляционных БД в документно-ориентированный вид.

2. Теоретическое обоснование

2.1 Форматы обмена данными: JSON и BSON.

2.1.1 Формат обмена данными JSON

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 это реализовано 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 это реализовано как массив, вектор, список или последовательность.

Представленные структуры данных являются универсальными. Теоретически, все современные языки программирования поддерживают их в той или иной форме. Так как JSON используется для обмена данными между различными языками программирования, то имеет смысл строить его на этих структурах.

В JSON используются их следующие формы:

Объект – это неупорядоченное множество пар имя/значение, заключённое в фигурные скобки { }. Между именем и значением стоит символ *':'* (двоеточие), а пары имя/значение разделяются запятыми.

Массив (одномерный) – это множество значений, имеющих порядковые номера (индексы). Массив заключается в квадратные скобки [ ]. Значения отделяются запятыми.

Значение может быть строкой в двойных кавычках, числом, значением ***true*** или ***false***, объектом, массивом, или значением ***null***. Эти структуры могут быть вложены друг в друга.

Строка – это упорядоченное множество из нуля или более символов юникода, заключенное в двойные кавычки, с использованием escape-последовательностей начинающихся с обратной косой черты (backslash). Символы представляются простой строкой.

Имя – это строка.

Строка очень похожа на строку в языках C и Java. Число тоже очень похоже на C или Java-число, за исключением того, что используется только десятичный формат. Пробелы могут быть вставлены между любыми двумя символами[10].

Следующий пример показывает JSON-представление некоторых объектов.

```
{
    "hello": "world"
}
{
    "BSON": ["awesome", 5.05, 1986]
}
```

В лабораторной работе №1 приведен пример хранения информации о телефонах в виде отдельных документов. На рисунке 1 представлена база данных «телефон» в виде совокупности документов.

[10] JSON [Электронный ресурс]. – Режим доступа: http://json.org/

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 1 – База данных «телефон» в не реляционном виде.

Поле «Name» содержит имя документа.

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"
}
```

2.1.2 Формат обмена данными BSON

BSON (от англ. «*Binary JavaScript Object Notation»)* – бинарная версия JSON. Также как и JSON BSON поддерживает встраивание документов и массивы в другие документы и массивы. BSON также содержит расширения, которые позволяют оперировать с данными, не являющимися частью спецификации JSON. BSON не имеет схемы данных, что дает ему некоторые преимущества в гибкости и некоторые недостатки в эффективности использования дискового пространства (накладные расходы BSON связаны с хранением имен полей в сериализуемых данных).[11]

В BSON может храниться нуль или более пар ключ/значение. Данные хранятся как единое целое, как один документ. Все операции по модификации данных затрагивают изменение всего документа.

Представленные выше JSON документы в BSON будет выглядеть следующим образом:

```
{"hello": "world"}                    →    "\x16\x00\x00\x00\x02hello\x00
                                            \x06\x00\x00\x00world\x00\x00"


"\x31\x00\x00\x00\x04BSON\x00\x26\x00
                                          \x00\x00\x020\x00\x08\x00\x00
```

[11] BSON – Binary JSON [Электронный ресурс]. – Режим доступа: http://bsonspec.org/

```
{“BSON”: [“awesome”, 5.05, 1986]}      →
\x00awesome\x00\x011\x00\x33\x33\x33
                                              \x33\x33\x33
                                              \x14\x40\x102\x00\xc2\x07\x00\x00
                                              \x00\x00”
```

Примечание:
Обратите внимание, что строки в кавычках представляют терминальные символы, и их следует интерпретировать с семантикой «С» (например «\x01» представляет байты «0000 0001»)

2.2 Моделирование данных

Нереляционные СУБД позволяют проектировать модель предметной области в виде набора объектов. При этом информация об одной сущности, разбросанная по различным таблицам РБД, в нереляциионной БД будет собрана в одном объекте.

Главным отличие БД mongo от РБД является отсутствие аналога операции соединения (JOIN). Если существует необходимость использовать соединения в базе данных, то они реализуются в программном коде приложения. Для того чтобы найти данные, связанные с каким-либо документом, как правило, необходимо выполнить второй запрос.

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.

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

Индивидуальное задание

1. Составить схему РБД в соответствии с вашим вариантом.
2. Создать БД по созданной вами схеме в любой СУБД.

3. Заполнить базу данных произвольными данными.

4. Представить созданную на предыдущем шаге БД в нереляционном виде, записанную в JSON.

Варианты индивидуального задания

1. Фирма, торгующая автомобилями.
2. Магазин, торгующий цифровыми фотоаппаратами.
3. Фирма, занимающаяся производством USB-гаджетов.
4. Предприятие, производящее аудио-системы.
5. Магазин, торгующий ноутбуками.
6. Предприятие, содержащее парк самолетов.
7. Магазин программного обеспечения.
8. Производитель кухонной техники.
9. Дистрибьютор мотоциклов.
10. Магазин компьютерной периферии.
11. Магазин одежды.
12. Интернет-магазин наручных часов.
13. Магазин, специализирующийся на планшетных компьютерах.
14. Магазин спортивных велосипедов.
15. Фирма по установке климатической техники.
16. Подразделение банка, хранящее информацию о держателях банковских карт.
17. Фирма, специализирующаяся на поставках зеркальных фотоаппаратов.
18. Ювелирный магазин.
19. Фирма по продаже скутеров.
20. Магазин обуви.
21. Производитель моноблоков.
22. Магазин по продаже телевизоров.

23. Поставщик бытовой техники.

24. Арендодатель игровых приставок.

25. Магазин спортивных товаров.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Дайте определения терминам JSON и BSON.

2. На каких структурах строится JSON?

3. Какие форма представления данных используются в JSON?

4. Возможно ли ссылаться из одних документов MongoDB на другие? Если возможно, то какие механизмы используются для этого?

5. Возможно ли в MongoDB использовать массив документов?

6. Существует ли возможность в MongoDB использовать вложенные документы?

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 4. ДОКУМЕНТЫ И ИНДЕКСЫ В MONGODB

1. Цель и содержание

Цель лабораторной работы: знакомство с документами и индексами MongoDB.

Задачи лабораторной работы: научиться создавать, обновлять и удалять документы в MongoDB, научиться работать с индексами.

2. Теоретическое обоснование

2.1 Работа с документами в MongoDB

MongoDB управляет наборами JSON-подобных документов, хранимых в двоичном виде в формате BSON. Размер любого документа не может превышать 16 МБ. Ограничение максимального размера документа позволяет гарантировать, что ни один документ не может использовать чрезмерное количество оперативной памяти, чрезмерное количество трафика при передаче. Для хранения документов больше, чем 16 МБ, MongoDB предоставляет GridFS API. [12]

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.

[12] MongoDB Limits and Thresholds [Электронный ресурс]. – Режим доступа: http://docs.mongodb.org/manual/reference/limits/#limit-bson-document-size

Для создания коллекции используется метод «createCollection(<имя коллекции>)». Метод createCollection применяется для объекта базы данных, например:

```
db.createCollection("phones")
```

Примечания:
– Если коллекция не существует, то она создастся при вставке в неё документа
– Если документ не содержит _id поле, то MongoDB добавит _id поле и присвоить ему уникальный для документов ObjectId.
– Если документ определяет новое поле, то метод вставки документа (insert) вставит документ с новым полем.

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 критерий (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.<коллекция>.update(<запрос>, <данные, на которые нужно заменить>)».

Первым параметром метод принимает критерий (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 нужно 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***.

2.2 Индексы MongoDB

Индексы в MongoDB, так же 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 <индексируемое поле> должен включать имя индексируемого поля, а параметр <ключ> 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 стоп-слов 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 1.

Таблица 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()
```

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя

персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

Индивидуальное задание

1. Создайте базу данных, с которой вы в дальнейшем будете работать
2. Создайте коллекцию, в которой у вас будут храниться документы.
3. Наполните коллекцию документами в соответствии с вашим индивидуальным вариантом. Создайте не менее 10 документов. Документы коллекции должны содержать, по крайней мере, один массив, и один вложенный документ.
4. Создайте составной и полнотекстовый индексы для документов в вашей коллекции.
5. Создайте запрос, производящий замену документа.
6. Создайте запрос, производящий обновление любого поля документа и запрос, добавляющий в него новые поля.
7. Создайте запрос для получения всех документов коллекции и определите, используется ли для него индекс.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Какие способы создания БД присутствуют в MongoDB?

2. Возможно ли неявное создание коллекции?

3. Каков синтаксис функции обновления документа?

4. Приведите пример запроса для обновления поля документа?

5. Какие модификаторы функции обновления вы знаете? Перечислите их назначение.

6. С помощью какой функции вы можете удалить документ?

7. Для чего используется индекс в БД mongo?

8. Как создается и удаляется индекс в MongoDB?

9. Перечислите опции создания индекса?

10. Какие типы индексов можно создать в MongoDB?

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание. Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания. Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 5. ЗАПРОСЫ И ЗАПРОСЫ С УСЛОВИЕМ В MONGODB

1. Цель и содержание

Цель лабораторной работы: знакомство с запросами в MongoDB.

Задачи лабораторной работы: научиться выполнять различные виды запросов к базе данных mongo.

2. Теоретическое обоснование

2.1 Запросы в MongoDB

2.1.1 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.

2.1.2 Сортировка, постраничный вывод результата 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)
…
```

2.1.3 Запросы с условием

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> – 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:

<Выражение i> - 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}}]})
```

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров.

Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

Индивидуальное задание

1. Создайте пару простых запросов для выборки данных из БД.
2. Создайте сложные запросы с каждым из перечисленных модификаторов.
3. Создайте запросы с использованием методов сортировки, ограничения и пропуска данных.
4. Всего у вас должно получиться не менее десяти уникальных запросов.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Приведите синтаксис метода «Find».

2. Возможно, ли производить поиск по нескольким полям?

3. Каким образом можно произвести сортировку по нескольким полям? Приведите пример запроса.

4. Можно ли смешивать включения и исключения полей при составлении запроса?

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 6. ЗАПРОСЫ: МОДИФИКАТОРЫ МАССИВОВ. ПОЗИЦИОННЫЕ МОДИФИКАТОРЫ МАССИВОВ

1. Цель и содержание

Цель лабораторной работы: знакомство с модификаторами массивов.

Задачи лабораторной работы: научиться работать с массивами.

2. Теоретическое обоснование

2.1 Модификаторы массивов в запросах

Для добавления элемента в массив используется модификатор «$push», который используется как параметр метода «update». Синтаксис модификатора:

{ $push: { <массив>: <значение> },

где:

<массив> – массив, в который происходит добавление элемента;

<значение> – значение добавляемого элемента.

Пример использования модификатора:

```
db.phones.update({ name: "NL920" },{ $push: { cost: 19990 } })
```

Данный запрос добавит элемент «19990» в массив «cost».

Примечания
Массив будет создан, в случае если он отсутствует.
Если обновляемое поле не является массивом, то в результате выполнения метода возникнет ошибка.
Если значение является массивом, $push добавляет весь массив как отдельный элемент. Для того чтобы добавить несколько элементов необходимо использовать модификатор $each:
Следующий пример добавляет числа из массива [90, 92, 85] как отдельные элементы в массив cost:
db.phones.update({name: "NL920"},{$push: {cost: {$each: [19900, 20000, 20100]}}})[13]

Для удаления элемента из массива используется модификатор «$pop». Синтаксис модификатора:

{ $pop: { <поле>: <опц. удаления> } },

---

[13] $push – MongoDBManual 2.4.4[Электронный ресурс]. – Режим доступа: http://docs.mongodb.org/manual/reference/operator/push/

где:

<поле> – имя массива;

<опц. удаления> – 1 – если требуется удалить последний элемент в массиве, -1 – если требуется удалить первый элемент в массиве.

Удаление первого элемента в массиве выглядит следующим образом:

```
db.phones.update({ name: "NL920" },{ $pop: { cost: 1 } })
```

Модификатор «$push» добавляет элементы в массив, не проверяя их на уникальность. Для того чтобы добавить в массив только уникальные элементы используется модификатор «$addToSet». Синтаксис модификатора аналогичен синтаксису «$push».

Для удаления элемента массива по определенному критерию используется модификатор «$pull». Синтаксис модификатора:

{ $pull: { <массив>: <запрос> } },

где:

<массив> – имя массива;

<запрос> – запрос для отыскания требуемого элемента.

Например, для удаления элемента «20100» из массива «cost» используется запрос:

```
db.phones.update({ name: “NL920” },{ $pull: { cost: 20100 } })
```

2.2 Позиционные модификаторы массивов

Для доступа к конкретному элементу массива используется два способа: по конкретной позиции, и с помощью использования позиционного оператора (символ ‘$’).

Для доступа к элементу массива по номеру, после запроса на выборку документа в качестве имени изменяемого поля используется: «<массив>.<№ элемента>»

Например, для увеличения нулевого элемента массива на 100 используется следующий запрос:

```
db.phones.update({ "name": "NL920" }, {$inc: {"cost.0": 100}})
```

Если требуется изменить элемент массива, не зная его позиции в массиве, то его необходимо отыскать с помощью запроса, и применить требуемый модификатор. В данном случае в качестве имени изменяемого поля будет выступать: «<массив>.<символ '$'>».

Например, для того чтобы увеличить элемент массива равный «19900» на 200 требуется ввести в консоль:

```
db.phones.update({ "cost": 19900 }, {$inc: {"cost.$": 200}})
```

2.3 Запросы в массивах

Иногда требуется составить запрос, в котором происходит поиск по массиву значений. В таком случае применяются запросы в массивах. Выше было показано, как можно отыскать элемент в массиве. Для того чтобы найти элемент массива равный «19990», используется следующий запрос:

```
db.phones.find({ "cost": 19900 })
```

Данный запрос успешно найдет документы, в которых массив с именем «cost» будет содержать элемент «19990».

Для того чтобы выбрать документы больше, чем по одному элементу массива, можно использовать оператор «$all».

Синтаксис оператора:

{ <поле>: { $all: [<значение1>, <значение 2>, …]}},

где:

<поле> – поле, по которому проводится поиск;

<значение 1, 2, …> – значения массива, которые нужно найти.

Для получения документов, в которых массив содержит элементы «19990», «20000» и «20100» используется запрос:

```
db.phones.find({cost: {$all: [19990, 20000, 20100]}})
```

Для получения документов по полному совпадению элементов в массиве необходимо просто перечислить искомые элементы также как и при вставке:

```
db.phones.find({cost: [19990, 20000, 20100]})
```

Для управления количеством возвращаемых элементов в массиве используется оператор «$slice». Синтаксис оператора:

{$slice: <количество элементов>}.

Для возвращения элементов массива, начиная с его конца, оператору передается отрицательное значение, с начала – положительное.

Оператор $slice может действовать подобно «skip» и «limit», только в отношении массивов. В этом случае оператор имеет следующий синтаксис:

{$slice: [<skip>, <limit>]},

Где:

<skip> – пропускаемое количество элементов;

<limit> – возвращаемое количество элементов.

В качестве примера рассмотрим запрос для получения первых десяти элементов массива «cost», который выглядит следующим образом:

```
db.phones.find( {}, {cost : {$slice: 10}}
```

Если нужно получить последние десять записей, то число «10» в запросе будет отрицательным.

Для ограничения/пропуска элементов, или для получения элементов из середины массива необходимо использовать вторую форму оператора «$slice»:

```
db.phones.find( {}, {cost : {$slice: [5, 10]}}
```

Данный запрос пропустит 5 начальных элементов массива и вернет элементы с 6 по 16.

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

Индивидуальное задание

1. Создайте несколько запросов для вставки данных в массив.

2. Создайте запросы, производящие обновление данных в массиве: как по позиции элемента в массиве, так и по его значению.

3. Создайте запросы, удаляющие элементы из массива: по позиции элемента в массиве и по его значению.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Приведите синтаксис «$push» и «$pop».

2. Каким образом можно вставить в массив несколько элементов? Приведите пример запроса.

3. Для чего используются модификаторы массивов в запросах?

4. Какие способы обновления данных в массиве вы знаете? Приведите примеры.

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 7. РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ В MONGODB

1. Цель и содержание

Цель лабораторной работы: знакомство с регулярными выражениями MongoDB.

Задачи лабораторной работы: научиться использовать регулярные выражения при составлении запросов.

2. Теоретическое обоснование

2.1 Регулярные выражения в MongoDB

Регулярные выражения в MongoDB используют Perl-синтаксис (Perl-совместимые регулярные выражения). Регулярные выражения являются мощным инструментом для составления шаблонных запросов. Представляют собой строку, состоящую из метасимволов и задающую правила поиска.

Регулярные выражения – формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов. В MongoDB регулярные выражения используются для составления запросов. При составлении регулярных выражений необходимо руководствоваться следующими правилами:

1. Любой символ обозначает себя самого, если это не метасимвол. Если вам нужно отменить действие метасимвола, то поставьте перед ним '\'.

2. Строка символов обозначает строку этих символов.

3. Множество возможных символов (класс) заключается в квадратные скобки '[]', это значит, что в данном месте может стоять один из указанных в скобках символов. Если первый символ в скобках '^' – это значит, что ни один из указанных символов не может стоять в данном месте выражения. Внутри класса можно употреблять символ '-', обозначающий диапазон

символов. Например, a-z - один из малых букв латинского алфавита, 0-9 - цифра и т.д.

4. Все символы, включая специальные, можно обозначать с помощью '\' как в языке C.

5. Альтернативные последовательности разделяются символом '|' Заметьте, что внутри квадратных скобок это обычный символ.

6. Внутри регулярного выражения можно указывать "подшаблоны" заключая их в круглые скобки и ссылаться на них как '\номер'. Первая скобка обозначается как '\1'. [14]

При составлении запросов используются следующие опции:

i	не различать строчные и заглавные буквы
m	многострочная строка

В шаблонах используются следующие метасимволы:

\	следующий метасимвол считается обычным символом
^	начала строки
.	один произвольный символ. Кроме символа конца строки (\n)
$	конец строки
|	или
()	группировка
[]	класс символов

Метасимволы имеют модификаторы, которые пишутся после них:

*	повторяется 0 или большее число раз
+	повторяется 1 или большее число раз
?	1 или 0 раз
{n}	повторяется точно n раз
{n,}	повторяется, по меньшей мере, n раз

[14] Введение в Perl. Регулярные выражения (шаблоны) [Электронный учебник]. – Режим доступа: http://citforum.ru/internet/perl_tut/re.shtml

{n,m}	повторяется не меньше n, но и не больше m

Фигурные скобки во всех других случаях считаются обычными символами.

По умолчанию действие метасимволов распространяется столько раз, сколько это возможно, не учитывая результат действия следующих метасимволов.

При составлении регулярных выражений можно использовать бэкслэш-символы. Наиболее часто встречающиеся символы приведены в таблице 1.

Таблица 1 – Наиболее часто встречающиеся бэкслеш-символы

Обозначение	Описание
\t	символ табуляции
\n	новая строка
\r	перевод каретки
\v	вертикальная табуляция
\e	escape
\033	восьмеричная запись символа
\x1A	шестнадцатеричная
\l	нижний регистр следующего символа
\u	верхний регистр следующего символа
\L	все символы в нижнем регистре до \E
\U	в верхнем регистре следующего символа
\E	ограничитель смены регистра
\s	один space символ (пробел, табуляция, новая строка)
\S	один не space символ
\d	одна цифра
\D	одна не цифра
\w	алфавитно-цифровая последовательность и символ ‘_’
\W	не алфавитно-цифровая последовательность и символ ‘_’

Все вышеперечисленные бэкслэш-символы при распознании регулярного выражения воспринимается как один символ. Для того чтобы обозначить последовательности необходимо комбинировать бэкслэш-символы с модификаторами метасимволов, например:

\d+ - одна или больше цифр, то же, что и [0-9]+

\w+ – слово, тоже что и [a-zA-Z0-9]+

Для закрепления результата рассмотрим несколько примеров.

Пример 1: Необходимо составить запрос, для отыскания всех документов с именем «document» или «Document» или «DOCUmenT» или другим, чередующим строчные и заглавные буквы:

```
db.test.find( { "name": /document/i})
```

Рассмотрим данный запрос внимательнее. Регулярное выражение начинается и заканчивается символом '/'. Далее следует имя запрашиваемого документа: «document». Далее, опция «i» – не различать строчные и заглавные буквы.

Пример 2: Запрос, отыскивающий все документы, содержащие в имени последовательность символов «umen» в любом регистре.

```
db.test.find( { "name": /\w+umen\w/i})
или
db.test.find( { "name": /umen/i})
```

Запрос состоит из:

/ – начало запроса;

\w+ – любая цифробуквенная последовательность;

umen – часть слова;

\w – цифробуквенная последовательность;

/ – конец запроса;

i – не различать строчные и заглавные символы.

Пример 3: Запрос для нахождения всех документов, начинающихся с последовательности «doc»:

```
db.test.find( { "name": /^doc\w+/i})
```

В данном запросе явно указывается начало строки «^doc» и любая дальнейшая цифробуквенная последовательность («\w+»).

Пример 4: Составить запрос для отыскания всех документов, имеющих сложное имя:

Первые три символа являются буквами.

Четвертый символ – цифра.

Пятый символ – любой.

Любая последовательность цифробуквенных символов.

Последний символ в имени документа – цифра 5.

Пример имени документа удовлетворяющий приведенным требованиям: «Doc7sH_u67i5».

Запрос для отыскания данного документа:

```
db.test.find( { name: /\w{3}\d\w+5$/ })
```

Пример 5: Ознакомьтесь с часто встречающимися регулярными выражениями, которые представлены в таблице 2.

Таблица 2 – Наиболее часто встречающиеся регулярные выражения и их составные части.

Регулярное выражение	Описание
/men/	Проверяет, есть ли в строке подстрока «men»
/^doc/	Проверяет, начинается ли строка с «doc»
/ent$/	Проверяет, оканчивается ли строка на «ent»
/d|o|c/ [doc]	Проверяет, содержит ли строка символ «d» или «o» или «c»
/do{1, 3}c	Проверяет, содержит ли строка символ «c», следующие за ним 1-3 символа «o», за которыми следует символ «c»
/docu…t/	Проверяет, содержит ли строка символы «docu» и «t», разделенные тремя символами
/\d/	Проверяет, содержит ли строка цифру

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

Индивидуальное задание

1. Создайте различные типы запросов с использованием регулярных выражений:

– поиск документа начинающегося с определенной последовательности символов;

– поиск документов содержащих определенную последовательность символов;

– поиск документов со сложным именем (шаблонное имя);

– составьте запросы с регулярными выражениями, примеры которых не описаны в теоретическом описании лабораторной работы.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Для чего используются регулярные выражения?

2. Какие метасимволы используются при составлении регулярных выражений?

3. Можно ли к метасимволам применять какие-либо модификаторы, если можно то какие?

4. Какими правилами необходимо руководствоваться при составлении регулярных выражений?

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 8. РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛЕНИЯ. MAPREDUCE В MONGODB

1. Цель и содержание

Цель лабораторной работы: изучить модель распределенных вычислений MapReduce.

Задачи лабораторной работы: научиться выполнять распределенные операции над документами.

2. Теоретическое обоснование

2.1 Модель распределенных вычислений MapReduce.

MapReduce – это модель распределённых вычислений, параллельной обработки очень больших объемов данных (измеряемыми петабайтами), в компьютерных кластерах.

Компьютеры, входящие в вычислительный кластер называются «узлами». Существуем два типа узлов: главный узел и рабочий.

Работа MapReduce состоит из двух шагов: Map и Reduce (отображение и свертка).

Мар-шаг производит предварительную обработку входных данных. Для этого главный узел получает входные данные задачи, разделяет их на части и передает рабочим узлам для предварительной обработки. На этом шаге входные данные преобразуются в пары ключ/значение (и ключ, и значение могут быть составными). Для этого функция Map применяется к каждому элементу исходной коллекции. Графически работа функции показана на рисунке 1.

После первого шага, алгоритм сортирует все пары ключ/значение, и группирует все значения по ключу. Данный шаг продемонстрирован на рисунке 2.

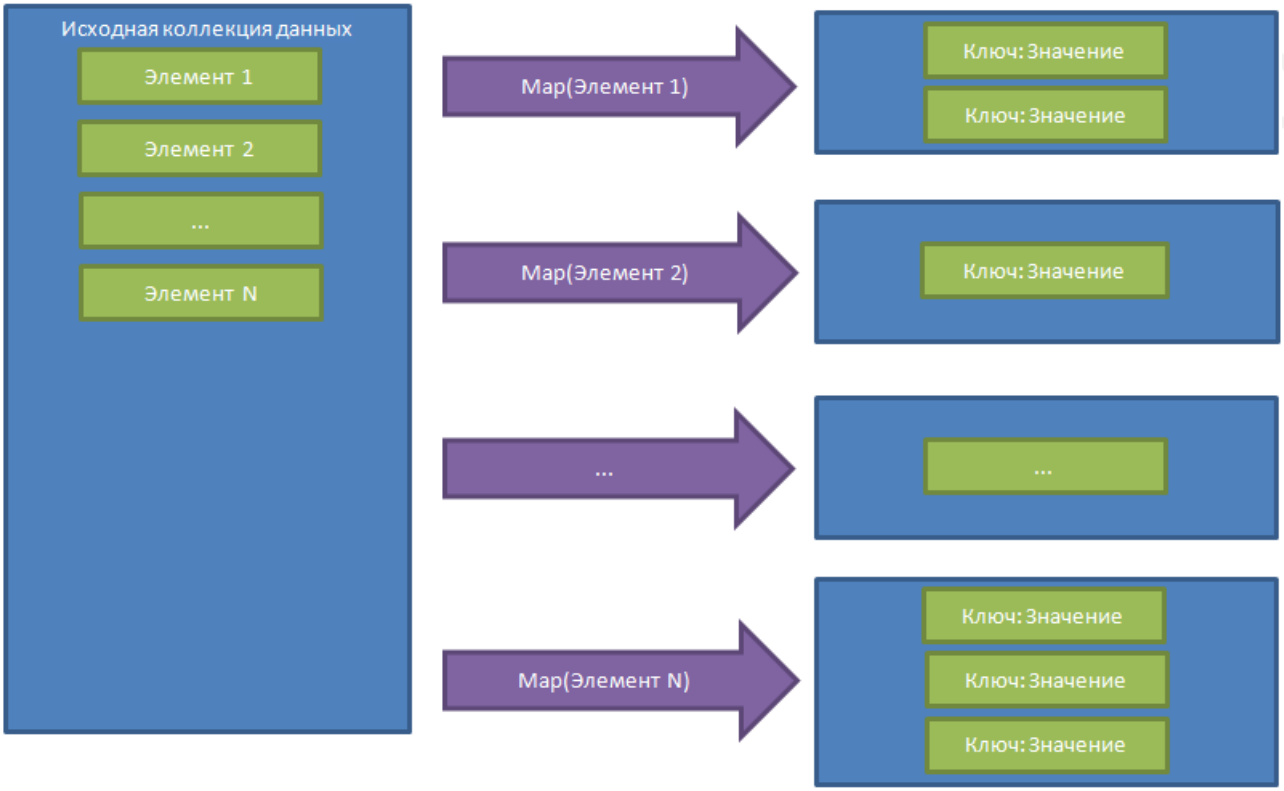


Рисунок 1 – Графическое изображение работы Map-шага

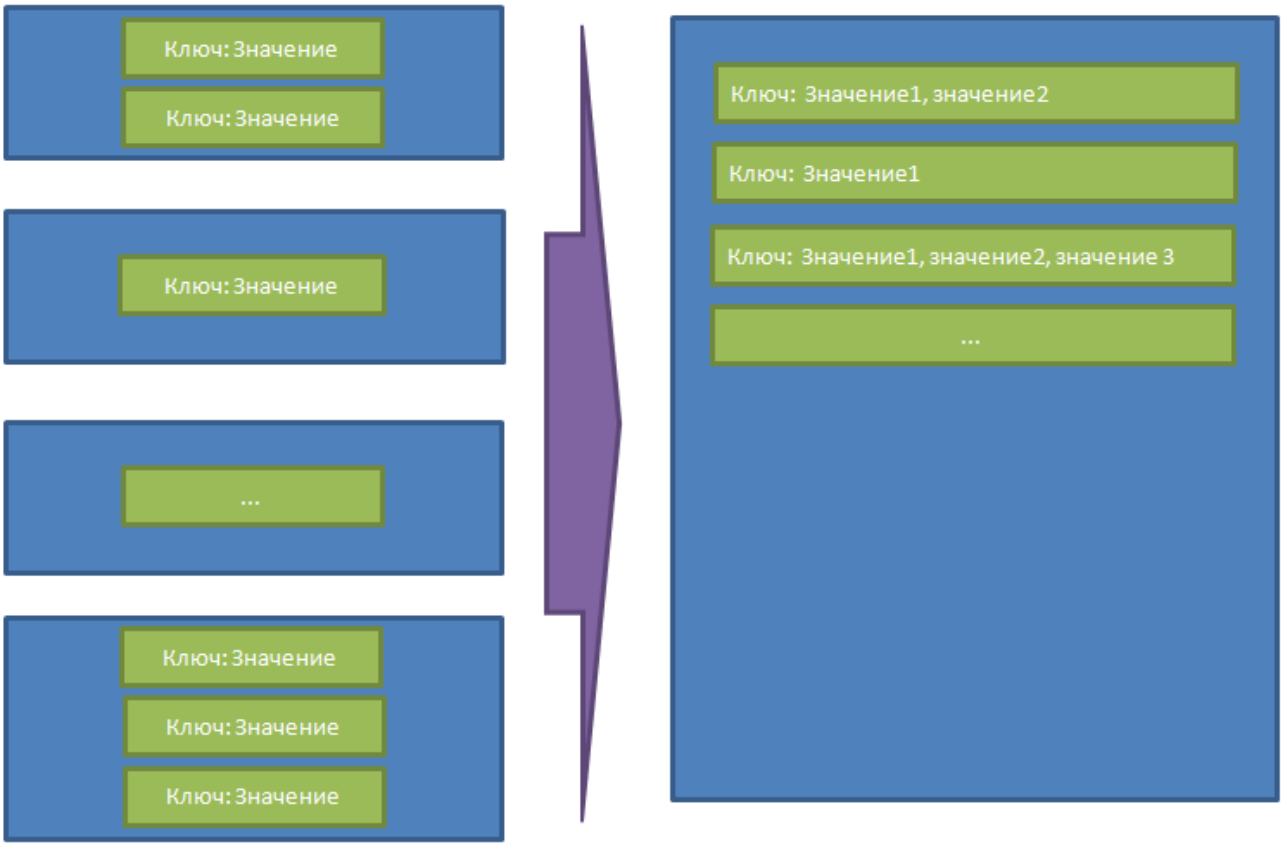


Рисунок 2 – Графическое изображение группировки данных по ключу

На втором шаге (Reduce-шаг) происходит свертка предварительно обработанных данных. При свертке на входе получаются ключ и массив значений, порожденный для этого ключа, а на выходе – финальный результат. Функция Reduce выполнится для каждого сгруппированного экземпляра пары ключ/значение. Графически результат работы Reduce-шага представлен на рисунке 3.

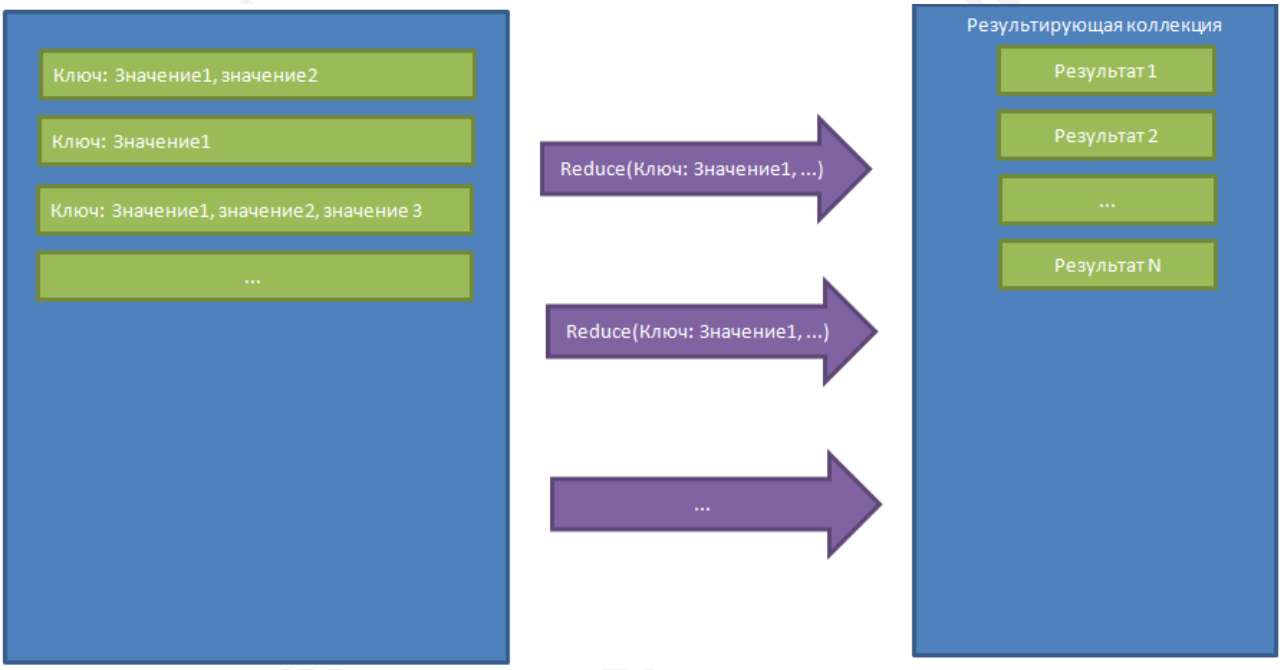


Рисунок 3 – Графическое изображение работы Reduce-шага

После завершения Reduce-шага главный узел получает ответы от рабочих узлов и на их основе формирует результат – решение задачи.

И Map и Reduce шаги выполняются распределено на компьютерах кластера. Все операции независимы друг от друга и могут производиться параллельно.

Основным преимуществом MapReduce по сравнению с традиционными решениями по обработке данных является производительность. Производительность достигается за счет распараллеливания вычислений и их независимости друг от друга. Кроме того, функции Map и Reduce

описываются на языке программирования высокого уровня, что, несомненно, предоставляет широкие возможности по обработке данных.[15]

Для лучшего понимания принципов работы MapReduce рассмотрим пример. Необходимо сгенерировать отчет по продажам мобильных телефонов. Желаемый результат – отчет в разрезе года, месяца, дня и количества. Данные о продажах хранятся в коллекции «selling»:

```
Наименование    Дата
Nexus One       Jan 20 2013 12:00
Nexus One       Jan 20 2013 13:00
iPhone 4        Jan 20 2013 13:30
Nexus One       Jan 20 2013 14:00
iPhone 4        Jan 21 2013 15:30
iPhone 4        Jan 21 2013 15:40
Nexus One       Jan 21 2013 16:20
iPhone 4        Jan 21 2013 17:00
Nexus One       Jan 21 2013 18:00
Nexus One       Jan 22 2013 19:00
```

На выходе нам необходимо получить отчет следующего содержания:

```
Наименование    Год     Месяц   День    Количество
Nexus One       2013    1       20      3
iPhone 4        2013    1       20      1
iPhone 4        2013    1       21      3
Nexus One       2013    1       21      2
Nexus One       2013    1       22      1
```

Первым делом выполняется функция Map (отображение). Функция Map должна создать пары ключ/значение для каждого документа. Функция будет вызвана для каждого документа коллекции «selling». Задача функции – создать ключ, состоящий из наименования, года, месяца и дня, и значение – единицу.

Исходный код функции на JavaScript:

```
function() {
```

[15] Understanding MapReduce [Электронный ресурс]. – Режим доступа: http://silviomassari.wordpress.com/2011/07/06/understanding-mapreduce-mongodb/

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

В данном коде this ссылается на текущий рассматриваемый документ. Результатом выполнения шага является новая коллекция:

```
{name: “Nexus One”, year: 2013, month: 0, day: 20} => [{count: 1}, {count: 1}, {count: 1}]
{name: “iPhone 4”, year: 2013, month: 0, day: 20} => [{count: 1}]
{name: “iPhone 4”, year: 2013, month: 0, day: 21} => [{count: 1}, {count: 1}, {count: 1}]
{name: “Nexus One”, year: 2013, month: 0, day: 21} => [{count: 1}, {count: 1}]
{name: “Nexus One”, year: 2013, month: 0, day: 22} => [{count: 1}]
```

Функция emit порождает новое значение, которое группируется по ключу.

Следующим шагом выполняется Reduce-функция. Она принимает результат работы предыдущего шага и суммирует значения полей «count». Исходный код функции Reduce:

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

Результатом выполнения функции является следующая коллекция:

```
{name: “Nexus One”, year: 2013, month: 0, day: 20} => [{count: 3}]
{name: “iPhone 4”, year: 2013, month: 0, day: 20} => [{count: 1}]
{name: “iPhone 4”, year: 2013, month: 0, day: 21} => [{count: 3}]
{name: “Nexus One”, year: 2013, month: 0, day: 21} => [{count: 2}]
{name: “Nexus One”, year: 2013, month: 0, day: 22} => [{count: 1}]
```

В MongoDB результат представлен в виде документов со следующей структурой:[16]

```
_id: { name: "Nexus One", year: 2013, month: 0, day: 20}, {count: 3}
```

2.2 Работа с MapReduce в MongoDB

Для обработки коллекций данных с помощью MapReduce необходимо в консоли mongo выполнить команду «mapReduce» применительно к обрабатываемой коллекции. Синтаксис команды:

db.<коллекция>.mapReduce(<map>, <reduce>,

{

<out> (коллекция),

<query> (Документ),

<sort> (Документ),

<Limit> (Число),

<finalize> (Функция),

<scope> (Документ),

<jsMode> (Логическое),

<verbose> (Логическое)

},

Где:

<map> – функция Map написанная на JavaScript;

<reduce> – функция Reduce написана на JavaScript;

<out> – указывает, куда выводить результат выполнения MapReduce;

<query> – определяет критерии отбора с использованием операторов запроса для входных документов Map функции;

<sort> – сортирует входные документы;

---

[16] Little MongoDB book [Электронный ресурс]. – Режим доступа: http://jsman.ru/mongo-book/Glava-6-MapReduce.html

<limit> – задает максимальное число документов для возврата из коллекции;

<finalize> – JavaScript функция, которая вызывается после исполнения Reduce-шага;

<scope> – определяет глобальные переменные, которые доступны в функциях Map, Reduce и Finalize;

<jsMode> – указывает, требуется ли преобразовывать промежуточные данные в BSON формат;

<verbose> – указывает, следует ли включать информацию о времени выполнения в результирующую информацию.

Первые три параметра функции являются обязательными, остальные – опциональные.

При передаче в качестве параметра out имени коллекции все данные из неё будут удалены. Для того чтобы добавить данные в существующую коллекцию необходимо использовать опцию «merge»: {out: {merge: <коллекция>}}

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров.

Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

1. Создайте коллекцию для хранения исходных документов для MapReduce.

```
db.createCollection("selling")
```

2. Наполните коллекцию данными.

```
db.selling.insert({name: "Nexus One", date: new Date(2013, 0, 20, 12, 00)})
db.selling.insert({name: "Nexus One", date: new Date(2013, 0, 20, 13, 00)})
db.selling.insert({name: "iPhone 4", date: new Date(2013, 0, 20, 13, 30)})
db.selling.insert({name: "Nexus One", date: new Date(2013, 0, 20, 14, 00)})
db.selling.insert({name: "iPhone 4", date: new Date(2013, 0, 21, 15, 30)})
db.selling.insert({name: "iPhone 4", date: new Date(2013, 0, 21, 15, 40)})
db.selling.insert({name: "Nexus One", date: new Date(2013, 0, 21, 16, 20)})
db.selling.insert({name: "iPhone 4", date: new Date(2013, 0, 21, 17, 00)})
db.selling.insert({name: "Nexus One", date: new Date(2013, 0, 21, 18, 00)})
db.selling.insert({name: "Nexus One", date: new Date(2013, 0, 22, 19, 00)})
```

3. Создайте map-функцию в консоли (консоль позволяет вводить многострочные конструкции).

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

```
        emit(key, {count: 1});
}
```

Можно скопировать текст функции и вставить в консоль. Описание функции Map в консоли представлено на рисунке 4.

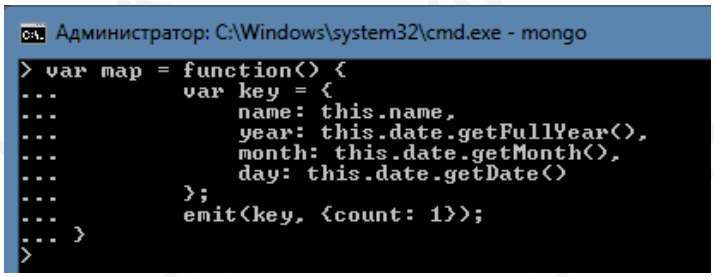


Рисунок 4 – Описание функции Map в консоли MongoDB

4. Создайте Reduce функцию в консоли.

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

Также как и в предыдущем шаге можно скопировать текст функции и вставить в консоль. Описание функции Reduce в консоли представлено на рисунке 5.

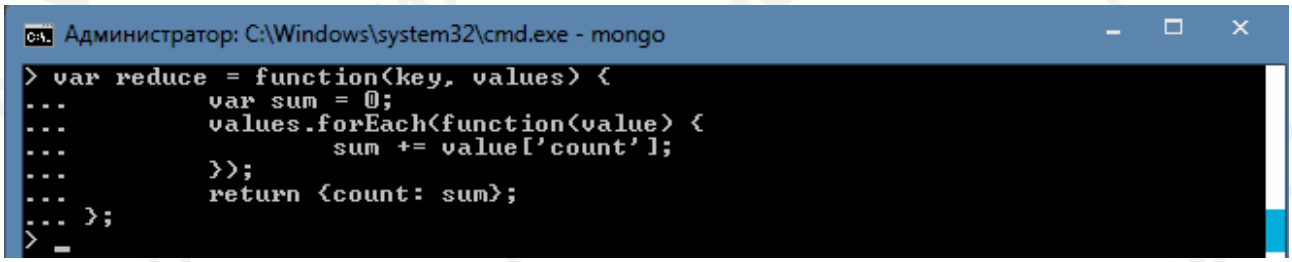


Рисунок 5 – Описание функции Reduce в консоли MongoDB

5. Выполните команду mapReduce над коллекцией «selling». Укажите параметр «inline: 1» для вывода результата в консоль.

```
db.selling.mapReduce(map, reduce, {out: {inline: 1}})
```

6. Ознакомьтесь с результатом выполнения mapReduce.

```
> db.selling.mapReduce(map, reduce, {out: {inline: 1}})
{
	"results" : [
		{
			"_id" : {
				"name" : "Nexus One",
				"year" : 2013,
				"month" : 0,
				"day" : 20
			},
			"value" : {
				"count" : 3
			}
		},
		{
			"_id" : {
				"name" : "Nexus One",
				"year" : 2013,
				"month" : 0,
				"day" : 21
			},
			"value" : {
				"count" : 2
			}
		},
		{
			"_id" : {
				"name" : "Nexus One",
				"year" : 2013,
				"month" : 0,
				"day" : 22
			},
			"value" : {
				"count" : 1
			}
		},
		{
			"_id" : {
				"name" : "iPhone 4",
				"year" : 2013,
				"month" : 0,
				"day" : 20
			},
			"value" : {
				"count" : 1
			}
		},
		{
			"_id" : {
				"name" : "iPhone 4",
				"year" : 2013,
				"month" : 0,
				"day" : 21
			},
			"value" : {
				"count" : 3
			}
		}
	],
```

```
        "timeMillis" : 1,
        "counts" : {
                "input" : 10,
                "emit" : 10,
                "reduce" : 3,
                "output" : 5
        },
        "ok" : 1,
}
```

7. Выполните команду mapReduce над коллекцией «selling». Для вывода результата в коллекцию параметр «out» принимает имя результирующей коллекции.

```
db.selling.mapReduce(map, reduce, {out: “sellingResult”})
```

8. Удостоверьтесь в том, что выполнение функции произошло без ошибок и результат соответствует ожидаемому.

Запросите список документов коллекции «sellingResult»:

```
db.sellingResult.find()
```

Сравните список полученных документов с ожидаемым результатом:

```
{ "_id" : { "name" : "Nexus One", "year" : 2013, "month" : 0, "day" : 20 }, "val
ue" : { "count" : 3 } }
{ "_id" : { "name" : "Nexus One", "year" : 2013, "month" : 0, "day" : 21 }, "val
ue" : { "count" : 2 } }
{ "_id" : { "name" : "Nexus One", "year" : 2013, "month" : 0, "day" : 22 }, "val
ue" : { "count" : 1 } }
{ "_id" : { "name" : "iPhone 4", "year" : 2013, "month" : 0, "day" : 20 }, "valu
e" : { "count" : 1 } }
{ "_id" : { "name" : "iPhone 4", "year" : 2013, "month" : 0, "day" : 21 }, "valu
e" : { "count" : 3 } }
```

Индивидуальное задание

1. Создайте коллекцию документов для обработки её с помощью MapReduce.

2. Наполните коллекцию документами.

3. Произведите обработку коллекции с использованием модели распределенных вычислений MapReduce.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Что означает термин MapReduce?

2. Из каких шагов состоит работа MapReduce?

3. Какими преимуществами обладает MapReduce по сравнению с обычными вычислениями?

4. Опишите работу с MapReduce в MongoDB.

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 9. АДМИНИСТРИРОВАНИЕ СУБД

1. Цель и содержание

Цель лабораторной работы: ознакомиться с администрированием баз данных mongo.

Задачи лабораторной работы: научиться работать с ролями в БД, научиться производить резервировании баз данных.

2. Теоретическое обоснование

2.1 Управление и диагностика

С помощью административного интерфейса входящего в состав MongoDB можно выполнять многие административные функции. Список некоторых команд представлен далее:

– help – просмотр справки;

– db.help() – получение справки по методам объекта «db»;

– db.commandHelp(<команда>) – справка по команде, имя которой передается в качестве параметра;

– show dbs – список существующих баз данных;

– show collections – список коллекций текущей базы;

– use <имя БД> – сделать текущей базу данных;

– db.getName() – посмотреть имя текущей БД;

– db – посмотреть имя текущей БД;

– db.version() – вывести версию программы;

– db.getMongo() – получить список текущих подключений.

Для копирования базы данных с одного сервера на другой используются следующие команды:

– db.cloneDatabase(<hostname>) – производит копирование удаленной базы данных в текущую. <hostname> – имя хоста, с которого производится копирование БД;

– db.copyDatabase(<origin>, <destination>, <hostname>) – производит копирование удаленной базы данных с хоста «<hostname>» с именем «<origin>», в базу данных с именем «<destination>»;

– db.dropDatabase() – удаляет текущую базу данных.

Для диагностики состояния базы данных используются следующие команды:

– db.printCollectionStats() – выводит информацию по всем коллекциям БД;

– db.stats() – выводит информацию о базе данных;

– db.serverStatus() – выводит информацию о статусе сервера: информацию о хосте, на котором выполняется процесс «mongod» и информацию о процессе и его настройках;

– db.repairDatabase() – восстанавливает базу данных после аварийного завершения процесса, сбоя питания или в других случаях, когда повреждено содержимое БД. Команда проверяет все данные на испорченность, удаляет найденные некорректные данные и собирает файлы данных. Если сбой произошел на рабочем узле, то нужно запустить «mongod» с параметром «-repair», а после окончания работы перезапустить его в нормальном режиме;

– db.shutdownServer() – остановить сервер базы данных;

– logpath <file> – указывается, в какой файл будет происходить вывод сообщений логирования. По умолчанию, MongoDB выводит сообщения на стандартный вывод (в консоль).

Если MongoDB использует реплики, то получение информации о них возможно с помощью следующих функций: db.getReplicationInfo(), db.printReplicationInfo(), db.printSlaveReplicationInfo().

2.2 Резервирование

Существует несколько способов для создания резервной копии данных, находящихся в базе данных mongo. Наилучшим является способ, который при резервировании не вызывает простоя (блокировки) сервиса. Таким способом является использование утилиты «mongodump.exe», входящей в состав MongoDB.

Для восстановления резервной копии, полученной с использованием «mongodump.exe», используется утилита «mongorestore.exe», которая также входит в состав MongoDB.

Утилита «mongodump.exe» имеет следующие параметры запуска:

--help – вывод справки;

--verbose, -v – степень детализации; выводить больше информации (например, для большей детализации необходимо использовать -vvvvv);

--host – хост mongo;

--port – порт сервера, также возможно использовать – host hostname:port;

--ipv6 – включить поддержку IPv6 (по умолчанию отключена);

--username, -u – имя пользователя;

--password, -p – пароль;

--dbpath – прямой доступ к файлам базы mongo по указанному пути, вместо подключения к серверу «mongod.exe». Для использования необходима блокировка директории с данными;

--db, -d – указывает базу данных;

--collection, -c – указывает используемые коллекции;

--directoryperdb – если задан аргумент «--dbpath», то каждая БД находится в отдельной директории;

--out, -o – директория вывода;

--query, -q – запрос JSON для ограничения документов, попадающих в вывод «mongodump.exe» (ограничение дампа документов).[17]

Утилита mongodump.exe может подключаться к указанному хосту, для дампа БД. Также можно задать базу данных, резервную копию которой необходимо сделать, либо её коллекцию (или коллекции). Если база данных не указывается, то подразумеваются все базы данных. Если директория, в которую должен происходить вывод данных, не указана (параметр «--out»), то вывод произойдет в поддиректорию «dump», директории запуска утилиты.

Результатом работы утилиты является дамп базы/баз данных, представляющий собой директории с именами БД, хранящие файлы в формате BSON.

Утилита «mongorestore.exe» умеет следующие параметры запуска:

--help – вывод справки;

--verbose, -v – степень детализации; выводить больше информации (например, для большей детализации необходимо использовать -vvvvv);

--host – хост mongo;

--port – порт сервера, также возможно использовать – host hostname:port;

--ipv6 – включить поддержку IPv6 (по умолчанию отключена);

--username, -u – имя пользователя;

--password, -p – пароль;

--dbpath – прямой доступ к файлам базы mongo по указанному пути, вместо подключения к серверу «mongod.exe». Необходима блокировка директории с данными, поэтому не может быть использована, если монго-сервер использует базу по указанному пути;

--db, -d – указывает базу данных;

--collection, -c – указывает используемые коллекции;

---

[17] Mongodump – MongoDBManual [Электронный ресурс]. – Режим доступа: http://docs.mongodb.org/manual/reference/program/mongodump/

--directoryperdb – если задан аргумент dbpath, то каждая БД находится в отдельной директории;

--objcheck – проверка объектов на валидность перед вставкой;

--filter <json> – использование фильтра перед вставкой;

--drop – удалить каждую коллекцию из целевой БД перед восстановлением;

--noIndexRestore – не восстанавливать индекс.

Для дампа либо восстановления базы данных на локальной машине достаточно указать базу данных и директорию/файл для дампа/восстановления.[18]

Например, для создания резервной копии всех баз данных на локальном компьютере в директорию «dump», утилита вызывается без параметров:

```
mongodump
```

Для создания резервной копии базы данных «test» в папку «d:\mongodb\backup\» используются следующие параметры запуска:

```
mongodump --db test --out d:\mongodb\backup\
```

Утилиту «mongorestore.exe» также можно запускать без параметров, при условии, что дамп данных находится в папке «dump» в директории утилиты. Для восстановления одной базы данных, либо коллекций, утилита вызывается с параметрами аналогичными параметрам «mongodump.exe».

Иногда требуется произвести блокировку базы данных для прямого доступа к файлам. В этом случае необходимо заблокировать БД командой

```
db.fsyncUnlock()
или
use admin
db.runCommand({fsync: 1,lock: 1})
```

[18] Mongorestore – MongoDBManual [Электронный ресурс]. – Режим доступа: http://docs.mongodb.org/manual/reference/program/mongorestore/

Для разблокирования БД используется команда

```
db.fsyncUnlock()
```

```
   Примечание:
   Обратите внимание на то, что в режиме блокировки все транзакции на
запись остановлены, и вызов любой функции, вносящей изменения в БД,
приведет к зависанию программы инициирующей вызов до тех пор, пока
блокировка с БД не будет снята.
```

При использовании репликации в MongoDB целесообразно делать резервные копии на рабочих серверах, а не на главном, т.к. в таком случае снижение производительности будет минимальным.

2.3 Работа с пользователями в MongoDB

До версии 2.4 в MongoDB учетные записи пользователей имели только следующие привилегии:

– только чтение базы данных;

– чтение и запись.

Начиная с версии 2.4, в MongoDB используются расширенные роли пользователей. Теперь роли пользователя не ограничены только чтением, либо записью всех баз данных, появилась более гибкая настройка. Одному пользователю возможно назначать различные роли для каждой базы данных. Ограничения для отдельных коллекций либо документов по-прежнему не доступны.

В MongoDB роли только предоставляют доступ, но никогда не ограничивают его. Именно поэтому пользователю в MongoDB нельзя запретить доступ к коллекции либо документу.

MongoDB поддерживает модель работы с пользователями, которая присутствовала в версии 2.2 (и более ранних версиях), в которой для добавления пользователя в базу данных используется команда «addUser» применительно к объекту БД. Синтаксис команды:

db.addUser(<имя>, <пароль>, {<только чтение>: true | false}),

где:

<имя> – имя пользователя;

<пароль> – пароль пользователя;

<только чтение> – ***true*** – только чтение базы данных, ***false*** – чтение и запись.

Данная команда добавляет пользователя, устанавливая ему режим доступа ко всей базе данных.

Для добавления пользователя с назначением ему роли используется команда «addUser», принимающая в качестве параметра сформированный документ «system.users». Формат документа:

```
{
   user: "<username>",
   pwd: "<hash>",
   roles: []
}
```

В документе поле «user» должно содержать имя пользователя, «pwd» – пароль, а массив «roles» – роли пользователя. Поле «pwd», после сохранения документа, содержит хэш пароля пользователя.

Например, команда для добавления пользователя с привилегиями на чтение и запись применительно к текущей БД имеет вид:

```
db.addUser( { user: "ivan", pwd: "pass", roles: [ "readWrite" ] } )
```

В случае если необходимо указать пользователя, описанного в другой БД, документ принимает следующий формат:

```
{
   user: "<username>",
   userSource: "<database>",
   roles: []
}
```

Поле «userSource» указывает базу данных, содержащую описание привилегий пользователя. Поля «pwd» и «userSource» не могут содержаться в одном документе.

Если необходимо для пользователя указать список привилегий для других баз данных, то они перечисляются во вложенном документе «otherDBRoles»:

```
{
   user: "<username>",
   userSource: "<database>",
   otherDBRoles: {
      <database0> : [],
      <database1> : []
   },
   roles: []
}
```

Документ «otherDBRoles» содержит список имен баз данных с указанием ролей для каждой, в нем <database0> – имя базы данных, для которой далее указаны роли и список ролей в массиве, <database1> – имя следующей БД, и т.д.

Следующий пример описывает пользователя с именем «admin» ролью «clusterAdmin» и дополнительными ролями «read» и «dbadmin» для баз данных «config» и «records»:

```
{
  user: "admin",
  userSource: "$external",
  roles: [ "clusterAdmin"],
  otherDBRoles:
  {
    config: [ "read" ],
    records: [ "dbadmin" ]
  }
}
```

Для добавления пользователя с описанной выше ролью используется запрос:

```
db.addUser({ user: "admin", userSource: "$external", roles: [
"clusterAdmin"],  otherDBRoles: { config: [ "read" ], records: [ "dbadmin" ]
}})
```

В MongoDB могут быть использованы следующие роли:

Пользовательские роли (применительно к текущей БД):

– read. Дает возможность пользователю получить данные из любой коллекции базы данных. Роль дает возможность пользоваться следующими функциями: «find», «aggregate», «checkShardingIndex», «collStats», «count», «dataSize», «dbHash», «dbStats», «distinct», «mapReduce» (только вывод в консоль) и некоторыми другими.

– readWrite. Дает возможность пользователю производить любые операции чтения/записи.

Роли администрирования баз данных (применительно к текущей БД):

– dbAdmin. Предоставляет возможность выполнять следующий набор административных операций в рамках БД: «clean», «collMod», «collStats», «create», «db.createCollection», «dbStats», «drop», «dropIndexes», «ensureIndex», «indexStats», «reIndex», «renameCollection» и некоторые другие. Только dbAdmin может получать данные из коллекции «system.profile»

– userAdmin. Позволяет пользователям читать и записывать данные в коллекцию «system.users» любой базы данных. Может изменять разрешения для существующих пользователей и создавать новые.

Административные роли:

– clusterAdmin. Предоставляет возможность администрировать кластер баз данных, дает возможность работать с репликами и шардами. Данная роль не предоставляет доступа к локальным конфигурациям баз данных.

Привилегии для любых баз данных:

– readAnyDatabase. Аналог «read», применительно к любой БД.

– readWriteAnyDatabase. Аналог «readWrite», применительно к любой БД.

– userAdminAnyDatabase. Аналог «userAdmin», применительно к любой БД.

– dbAdminAnyDatabase. Аналог «dbAdmin», применительно к любой БД.

Просмотр списка всех пользователей осуществляется с помощью команды «show users».

Удаление пользователя – командой «db.removeUser(<имя пользователя>)».

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

Индивидуальное задание

1. Получите диагностическую информацию о вашей базе данных и содержащихся в ней коллекциях.

2. Создайте резервную копию данных вашей БД.

3. Восстановите базу данных из резервной копии.

4. Создайте для базы данных несколько пользователей, имеющих различные роли.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Какие утилиты используются для резервирования и восстановления баз данных?

2. Какие параметры запуска имеет утилита «mongodump.exe»?

3. Какие параметры запуска имеет утилита «mongorestore.exe»?

4. Приведите синтаксис команды для добавления пользователя в БД.

5. Каким образом производится смена пароля для пользователя?

6. Перечислите роли, которые можно назначать пользователям в БД.

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 10. ШАРДИНГ В MONGODB. РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛЕНИЯ. MAPREDUCE НА НЕСКОЛЬКИХ СЕРВЕРАХ.

1. Цель и содержание

Цель лабораторной работы: изучить модель шардинга MongoDB.

Задачи лабораторной работы: научиться создавать и конфигурировать шард-сервера.

2. Теоретическое обоснование

2.1 Шардинг в MongoDB

Шардинг (партицированние) – разделение данных на несколько серверов (узлов) с сохранением определенного порядка.[19]

Шардинг используется для выравнивания нагрузки на запрос записей. При использовании шардинга данные исходной коллекции распределяются по нескольким серверам с использованием определенного шард-ключа. Шардинг позволяет создавать коллекции неограниченного размера, содержащие огромное количество записей, что важно для работы с большими данными.

Для каждой коллекции, которая распределена по шардам, определяется ключ шардинга. Пространство ключей для ключа шардинга делится на непрерывные диапазоны ключей (чанки), которые размещены по соответствующим шардам.

Схематически архитектура шардинга MongoDB представлена на рисунке 1.

---

[19] http://gliffer.ru/articles/nosql--sharding-mongodb-na-paltsah/

Клиентское приложение – это приложение, которое подключается к MongoDB и использует его как хранилище данных. MongoDB может одновременно работать с несколькими клиентами.

Сервер маршрутизации (процесс «mongos.exe») выполняет функции доступа к кластеру. «Mongos.exe» ведет себя подобно «mongod.exe», обеспечивая прозрачное взаимодействие с хранилищем данных. Клиент, взаимодействуя с «mongos.exe» не имеет представления о том, что находится за ним: локальная база данных или распределенное хранилище. «Mongos.exe» хранит сведения о том, какие документы хранятся, в каких узлах (шардах).

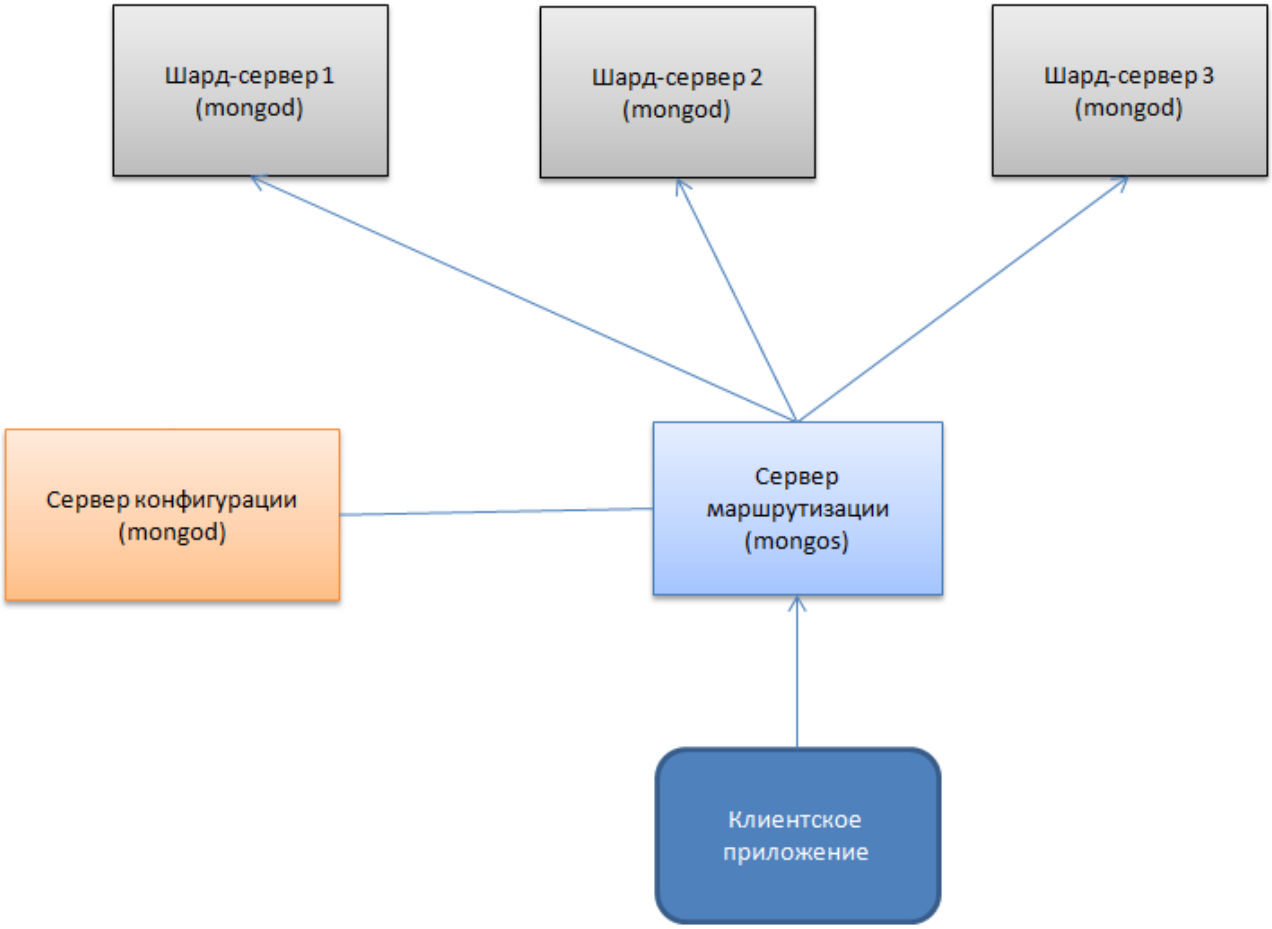


Рисунок 1 – Архитектура шардинга MongoDB

Шард-сервер состоит из шарда и процесса, управляющего шардом. Шард – это определенный набор данных. «Mongod.exe» – это процесс, управляющий данными шарда. Один шард может обслуживаться

несколькими mongod-процессами. Также каждый шард может обслуживаться распределенными mongod-процессами (репликами).

Как было указано ранее, чанки – это непрерывные куски данных какой-либо коллекции. Каждый чанк характеризуется следующими параметрами:

- минимальный шард-ключ (MinKey);
- максимальный шард-ключ (MaxKey);
- имя шарда, где находится чанк.

Когда чанк становится большим и превышает максимальный размер чанка, он делится пополам (этот процесс называется балансировкой и выполняется автоматически). Когда совокупный размер чанков шарда больше доступного для шарда, чанк переносится на другой шард.

Сервер конфигурации является процессом (или процессами), который хранит метаданные о шардах (описание чанков).

Коротко процесс обработки запроса на выборку данных из распределенной коллекции выглядит следующим образом: клиентский процесс обращается к серверу маршрутизации. Сервер маршрутизации обращается к серверу конфигурации и выясняет, в каких чанках хранятся требуемые данные и на каких шардах расположены эти чанки. После того как сервер маршрутизации получил данные о шардах и чанках, он обращается к ним (процесс «mongod») и получает искомые данные. Далее сервер маршрутизации возвращает данные клиенту.

Если запросы на изменение/вставку/удаление содержат ключ шардинга, то сервер маршрутизации, основываясь на информации, полученной от сервера конфигурации, передаст запрос к основному серверу, содержащему чанк с данными, диапазон ключей которого покрывает ключ шардинга искомых документов.

В случае запроса без указания ключа шардинга (ключ для шардинга не является частью критерия выбора) сервер маршрутизации отправит запрос к каждому шарду. Каждый шард произведет локальный поиск, а результат будет собран на сервере маршрутизации и возвращен клиенту.

2.2 MapReduce над распределенной коллекцией

Операция MapReduce также как и выборки данных происходит прозрачно для клиента без указания того, с каким хранилищем в данный момент происходит работа: с распределенным хранилищем или с локальным.

Распределенный MapReduce в MongoDB состоит из следующих шагов:

1. Клиент после определения функций map/reduce и исходных коллекций отправляет запрос к серверу маршрутизации.
2. С помощью сервера конфигурации определяется расположение искомых чанков.
3. Сервер маршрутизации передает запрос соответствующим шардам (или шарду, как и с обычными запросами шарды, вовлеченные в операцию, зависят от того, указан ли ключ шардинга в запросе)
4. База данных каждого шарда выполняет запрос и функцию map, которая выдает набор пар ключ/значение.
5. Вызывается функция reduce для результата, полученного на предыдущем шаге.

Функция reduce также может быть вызвана на предыдущем шаге, если объем вывода функции map полностью заполняет буфер памяти. В этом случае вызов функции reduce частично уменьшит объем занимаемой памяти.

6. После окончания работы предыдущего шага сервер маршрутизации уведомляет шарды, которые будут хранить обработанные данные.
7. База данных шарда, хранящего результирующую коллекцию, запросит частично уменьшенные данные от шардов на основе своего диапазона ключей.
8. База данных шарда снова выполняет функцию reduce над уже частично уменьшенным результатом. Затем результат сохраняется в базу данных, расположенную на этом(их) шарде(ах).

9. Выполняется функция финализации, если она представлена пользователем.

2.3 Выбор ключа для шардинга

Выбор ключа для шардинга является важной темой. Шардинговый ключ необходимо выбирать таким образом, чтобы данные можно было равномерно распределить между шардами. Примером ключа шардинга может быть: диапазон имен, номера штрих-кодов, и т.д. Общие рекомендации сложно дать, т. к. в каждом конкретном случае шардинговый ключ подбирается в зависимости от задачи.

Иногда, когда документы не содержат полей, которые могут быть использованы в качестве ключа для шардинга, в документы добавляет поле, которое содержит случайные данные и используется в качестве ключа для шардинга. Примером такого поля может служить хеш-сумма документа. Иногда в качестве ключа для шардинга используется поле «_id». Однако использование «_id» объекта по умолчанию в качестве ключа для шардинга является не очень хорошей идеей.

Для каждого конкретного случаю подходит свой ключ. Можно выделить следующие типы ключей:

- Случайный;

Ключ генерируется случайно, с использованием различных генераторов случайных чисел/последовательностей, либо различных хэш-сумм. Такой ключ отлично подходит для распределенного чтения и записи на всех шардах. Необходимо иметь ввиду, что для того чтобы получить выгоду от использования шардинга при выполнении чтения данных, необходимо использовать свой ключ в запросах. В противном случае mongoDB будет опрашивать все шарды. При использовании ключа для шардинга в запросе,

mongoDB может сразу обратиться к требуемому чанку на заранее известном шарде.

- Ключ в определенном диапазоне;

К данным типам ключей относятся ключи, которые имеют заранее определенный диапазон. Например, для ведения журнала приложения может быть использован ключ, привязанный к типу записи и имеющий следующие возможные значения: «Ошибка», «Предупреждение», «Сообщение». Очевидно, что в таком случае в базе данных будет всего три чанка, и они могут достигать очень больших размеров. К данной группе ключей также относятся ключи, ориентированные на разбиение данных в соответствии с алфавитом, либо другим критерием имеющим определенный диапазон.

- Составной ключ;

Если необходимо использовать ключи, описанные в предыдущем пункте, и количество их значений ограничено, то для уменьшения размера чанков, можно использовать составной ключ. В приведенном выше примере, если вы хотите использовать ключ с заранее определенными состояниями, вы можете его сделать составным и добавить метку с датой создания. Теперь в случае превышения максимального размера чанка, mongoDB может разделить его и вынести в отдельный чанк.

- Ключи, основанные на временных метках;

При использовании подобного рода ключей в качестве ключа выступают какие-либо данные, привязанные ко времени. Например, дата создания документа. Однако следует иметь ввиду, что, так как в данном типе ключа отсутствует всякая случайность, а временные метки всегда возрастают, то вы будете всегда производить запись в один чанк. Чанк может измениться, но все ваши записи всегда будут производиться в одно и тоже место. Это может быть проблемой и узким местом приложения, если вы имеете требовательные к ресурсам операции записи.[20]

[20] http://www.quora.com/MongoDB/How-should-shard-keys-be-assigned-with-MongoDB

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

Для выполнения лабораторной работы студентам необходимо разделиться на группы по три человека. Один студент (1) создает и настраивает сервер конфигурации, сервер маршрутизации и добавляет базу данных, второй (2) и третий (3) студенты подготавливают свой компьютер для использования в качестве шарда. После выполнения задания студенты меняются ролями.

Подготовка шард-сервера (2), (3)

1. Запустите «mongod» со следующими параметрами:

```
mongod --shardsvr <options>
Где options – другие опции настройки «mongod» (например: путь к каталогу с БД)
```

После успешного запуска «mongod» на ваших компьютерах работают шард-сервера.

2. Узнайте IP-адрес вашего компьютера.

```
IP-адрес компьютера можно посмотреть в свойствах сетевого подключения либо введя команду «ipconfig» в командной строке.
```

Создание сервера конфигурации (1)

```
Сервера конфигурации и маршрутизации расположены на локальной машине имеющий IP-адрес 127.0.0.1.
```

1. Для создания сервера конфигурации необходимо запустить «mongod» с параметром «--shardsvr»:

```
mongod --shardsvr
```

Создание сервера маршрутизации (1)

1. Для создания сервера конфигурации введите в консоль:

```
mongos --configdb 127.0.0.1:<port>
Где port – порт, на котором запущен «mongos» (по умолчанию: 27018)
```

Конфигурирование кластера (1)

1. Восстановите из резервной копии базу данных, прилагаемую к методическим указаниям, используя знания, полученные в ходе выполнения предыдущих работ.

2. Запустите процесс «mongo».

3. Войдите под именем администратора

```
use admin
```

3. Узнайте у студентов, работающих в вашей группе IP-адреса их компьютеров, предположим это: 192.168.1.2 и 192.18.1.3

4. Добавьте шарды в кластер:

```
db.runCommand( { addshard : "192.168.1.2:27018" } );
db.runCommand( { addshard : "192.18.1.3:27018" } );
```

5. Сообщите серверу имя базы данных для шардинга

```
> db.runCommand( { enablesharding : "pdb" } );
```

6. Сообщите серверу имя коллекции для шардинга:

В качестве ключа для шардинга выберете одно из полей БД и обоснуйте свой выбор.

```
> db.runCommand( { shardcollection : "pdb.phones", : {<key>:<keyValue>} })
Где:
        <key> - имя ключа для шардинга,
        <keyValue> - значение ключа для шардинга.
```

7. Создайте map и reduce функции, основываясь на знаниях, полученных в ходе выполнения предыдущих лабораторных работ.

8. Выполните команду mapReduce над коллекцией «phones». Результат работы выведите в консоль и сохраните в виде отдельной коллекции. Результатом работы MapReduce должен быть подсчет общего количества телефонов каждой фирмы, хранящихся в базе данных.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Что означает термин MapReduce?

2. Из каких шагов состоит работа MapReduce?

3. В чем особенности выполнения mapReduce над распределенными коллекциями?

4. Что такое шардинг?

5. Опишите преимущества и недостатки различных типов ключей, используемых для шардинга?

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 11. УСТАНОВКА И НАСТРОЙКА MONGODB ДЛЯ РЕШЕНИЯ УЧЕБНОЙ ЗАДАЧИ.

1. Цель и содержание

Цель лабораторной работы: самостоятельно установить и настроить MongoD, используя наилучшую стратегию для конкретной учебной задачи.

Задачи лабораторной работы: научиться самостоятельно проектировать, устанавливать и настраивать средства информационных технологий.

2. Теоретическое обоснование

Изучите материал лабораторной работы №1.

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу

лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

Необходимо самостоятельно установить и настроить MongoDB для решения задачи предметной области в соответствии с индивидуальным вариантом.

Варианты заданий:

1. Персональный блог.
2. Социальная сеть.
3. Система сбора информации о держателях пластиковых карт.
4. Система сбора информации о держателях сотовых телефонов.
5. Интернет-магазин.
6. Система анализа товарооборота предприятия.
7. Электронная система дистанционного обучения.
8. Система сбора информации о предпочтениях пользователей.
9. Система анализа интернет-трафика.
10. Анализ продаж интернет-магазина.
11. Анализ трафика транспортного предприятия.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Что означает термин NoSQL?

2. Какие преимущества предоставляют NoSQL базы данных в сравнении с реляционными базами данных?

3. Какими особенностями обладает MongoDB?

4. Сколькими способами можно произвести установку MongoDB? Кратко опишите эти способы.

5. На какие группы делятся приложения, входящие в состав MongoDB?

6. Создает ли MongoDB по умолчанию какую-либо базу данных? Если создает, назовите её имя.

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 12. РАБОТА С КОНСОЛЬЮ MONGODB.

1. Цель и содержание

Цель лабораторной работы: самостоятельно использовать команды консоли MongoDB для конкретной учебной задачи.

Задачи лабораторной работы: научиться самостоятельно проектировать, устанавливать и настраивать средства информационных технологий.

2. Теоретическое обоснование

Изучите материал лабораторной работы №2.

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу

лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

Необходимо продемонстрировать выполнение команд консоли на примере задачи предметной области в соответствии с индивидуальным вариантом.

Варианты заданий:

1. Персональный блог.
2. Социальная сеть.
3. Система сбора информации о держателях пластиковых карт.
4. Система сбора информации о держателях сотовых телефонов.
5. Интернет-магазин.
6. Система анализа товарооборота предприятия.
7. Электронная система дистанционного обучения.
8. Система сбора информации о предпочтениях пользователей.
9. Система анализа интернет-трафика.
10. Анализ продаж интернет-магазина.
11. Анализ трафика транспортного предприятия.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Какие существуют способы взаимодействия с БД mongo?
2. Существует ли возможность взаимодействовать с БД mongo посредством WEB?
3. Какими особенностями обладает Mongo Explorer?
4. Опишите интерфейс приложения Mongo Explorer.
5. Каков формат команд командной строки?

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 13. ПОСТРОЕНИЕ МОДЕЛИ ДАННЫХ. ФОРМАТИРОВАНИЕ ИСХОДНЫХ ДАННЫХ

1. Цель и содержание

Цель лабораторной работы: самостоятельно конвертировать набор данных в MongoDB для конкретной учебной задачи.

Задачи лабораторной работы: научиться самостоятельно проектировать, устанавливать и настраивать средства информационных технологий.

2. Теоретическое обоснование

Изучите материал лабораторной работы №3.

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу

лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

Необходимо выполнить загрузку данных из различных источников в базу MongoDB в соответствии с индивидуальным вариантом. Варианты заданий:

1. Персональный блог.
2. Социальная сеть.
3. Система сбора информации о держателях пластиковых карт.
4. Система сбора информации о держателях сотовых телефонов.
5. Интернет-магазин.
6. Система анализа товарооборота предприятия.
7. Электронная система дистанционного обучения.
8. Система сбора информации о предпочтениях пользователей.
9. Система анализа интернет-трафика.
10. Анализ продаж интернет-магазина.
11. Анализ трафика транспортного предприятия.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Дайте определения терминам JSON и BSON.

2. На каких структурах строится JSON?

3. Какие форма представления данных используются в JSON?

4. Возможно ли ссылаться из одних документов MongoDB на другие? Если возможно, то какие механизмы используются для этого?

5. Возможно ли в MongoDB использовать массив документов?

6. Существует ли возможность в MongoDB использовать вложенные документы?

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 14. ИСПОЛЬЗОВАНИЕ ДОКУМЕНТОВ И ИНДЕКСОВ ДЛЯ РЕШЕНИЯ УЧЕБНОЙ ЗАДАЧИ

1. Цель и содержание

Цель лабораторной работы: самостоятельно реализовать набор операций с документами в MongoDB для конкретной учебной задачи.

Задачи лабораторной работы: научиться самостоятельно проектировать, устанавливать и настраивать средства информационных технологий.

2. Теоретическое обоснование

Изучите материал лабораторной работы №4.

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу

лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

Необходимо разработать скрипты для выполнения основных операций в MongoDB в соответствии с индивидуальным вариантом. Варианты заданий:

1. Персональный блог.
2. Социальная сеть.
3. Система сбора информации о держателях пластиковых карт.
4. Система сбора информации о держателях сотовых телефонов.
5. Интернет-магазин.
6. Система анализа товарооборота предприятия.
7. Электронная система дистанционного обучения.
8. Система сбора информации о предпочтениях пользователей.
9. Система анализа интернет-трафика.
10. Анализ продаж интернет-магазина.
11. Анализ трафика транспортного предприятия.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Какие способы создания БД присутствуют в MongoDB?

2. Возможно ли неявное создание коллекции?

3. Каков синтаксис функции обновления документа?

4. Приведите пример запроса для обновления поля документа?

5. Какие модификаторы функции обновления вы знаете? Перечислите их назначение.

6. С помощью какой функции вы можете удалить документ?

7. Для чего используется индекс в БД mongo?

8. Как создается и удаляется индекс в MongoDB?

9. Перечислите опции создания индекса?

10. Какие типы индексов можно создать в MongoDB?

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 15. ИСПОЛЬЗОВАНИЕ ЗАПРОСОВ ДЛЯ РЕШЕНИЯ УЧЕБНОЙ ЗАДАЧИ

1. Цель и содержание

Цель лабораторной работы: самостоятельно реализовать набор запросов в MongoDB для конкретной учебной задачи.

Задачи лабораторной работы: научиться самостоятельно проектировать, устанавливать и настраивать средства информационных технологий.

2. Теоретическое обоснование

Изучите материал лабораторной работы №5.

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу

лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

Необходимо разработать запросы для реализации аналитических отчетов в MongoDB в соответствии с индивидуальным вариантом. Варианты заданий:

1. Персональный блог.
2. Социальная сеть.
3. Система сбора информации о держателях пластиковых карт.
4. Система сбора информации о держателях сотовых телефонов.
5. Интернет-магазин.
6. Система анализа товарооборота предприятия.
7. Электронная система дистанционного обучения.
8. Система сбора информации о предпочтениях пользователей.
9. Система анализа интернет-трафика.
10. Анализ продаж интернет-магазина.
11. Анализ трафика транспортного предприятия.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Приведите синтаксис метода «Find».

2. Возможно, ли производить поиск по нескольким полям?

3. Каким образом можно произвести сортировку по нескольким полям? Приведите пример запроса.

4. Можно ли смешивать включения и исключения полей при составлении запроса?

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 16. ИСПОЛЬЗОВАНИЕ МАССИВОВ ДЛЯ РЕШЕНИЯ УЧЕБНОЙ ЗАДАЧИ

1. Цель и содержание

Цель лабораторной работы: самостоятельно реализовать операции с использованием массивов в MongoDB для конкретной учебной задачи.

Задачи лабораторной работы: научиться самостоятельно проектировать, устанавливать и настраивать средства информационных технологий.

2. Теоретическое обоснование

Изучите материал лабораторной работы №6.

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу

лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

Необходимо реализовать операции с использованием массивов в MongoDB в соответствии с индивидуальным вариантом. Варианты заданий:

1. Персональный блог.
2. Социальная сеть.
3. Система сбора информации о держателях пластиковых карт.
4. Система сбора информации о держателях сотовых телефонов.
5. Интернет-магазин.
6. Система анализа товарооборота предприятия.
7. Электронная система дистанционного обучения.
8. Система сбора информации о предпочтениях пользователей.
9. Система анализа интернет-трафика.
10. Анализ продаж интернет-магазина.
11. Анализ трафика транспортного предприятия.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Приведите синтаксис «$push» и «$pop».

2. Каким образом можно вставить в массив несколько элементов? Приведите пример запроса.

3. Для чего используются модификаторы массивов в запросах?

4. Какие способы обновления данных в массиве вы знаете? Приведите примеры.

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

ЛАБОРАТОРНАЯ РАБОТА 17. ПРИМЕНЕНИЕ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ ДЛЯ РЕШЕНИЯ УЧЕБНОЙ ЗАДАЧИ

1. Цель и содержание

Цель лабораторной работы: самостоятельно реализовать операции с использованием массивов в MongoDB для конкретной учебной задачи.

Задачи лабораторной работы: научиться самостоятельно проектировать, устанавливать и настраивать средства информационных технологий.

2. Теоретическое обоснование

Изучите материал лабораторной работы №7.

3. Аппаратура и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, MongoDB 2.4.3.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу

лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения лабораторной работы

Необходимо реализовать операции с использованием массивов в MongoDB в соответствии с индивидуальным вариантом. Варианты заданий:

1. Персональный блог.
2. Социальная сеть.
3. Система сбора информации о держателях пластиковых карт.
4. Система сбора информации о держателях сотовых телефонов.
5. Интернет-магазин.
6. Система анализа товарооборота предприятия.
7. Электронная система дистанционного обучения.
8. Система сбора информации о предпочтениях пользователей.
9. Система анализа интернет-трафика.
10. Анализ продаж интернет-магазина.
11. Анализ трафика транспортного предприятия.

6. Содержание отчета и его форма

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Экранные формы, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

7. Вопросы для защиты работы

1. Для чего используются регулярные выражения?
2. Какие метасимволы используются при составлении регулярных выражений?
3. Можно ли к метасимволам применять какие-либо модификаторы, если можно то какие?
4. Какими правилами необходимо руководствоваться при составлении регулярных выражений?

8. Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание.

Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент:

1) отвечает на контрольные вопросы;

2) поясняет ход выполнения индивидуального задания;

3) поясняет результаты, полученные в результате выполнения индивидуального задания.

Ход защиты лабораторной работы контролируется преподавателем.

СПИСОК ЛИТЕРАТУРЫ

1. Левитин А.В. Алгоритмы. Введение в разработку и анализ. / М.: Вильямс, 2008 – 576 с.

2. Голицына О.Л., Попов И.И. Основы алгоритмизации и программирования (2-е издание). / М.: Инфа-М, 2011 – 432 с.

3. Ф. Новиков. Дискретная математика для программистов. / СПб: Питер, 2009 – 304 с.

4. Алексеев В.Е., Таланов В.А. Графы и алгоритмы. Структуры данных. Модели вычислений. Учебник. / М.: Бином. Лаборатория знаний, 2009 – 320 с.

5. Д. Кнут. Искусство программирования. т. 3. Сортировка и поиск. 2-е изд. / М.: Вильямс, 2010 – 530 с.

6. С. Окулов. Программирование в алгоритмах (3-е издание). / М.: Бином. Лаборатория знаний, 2010 – 383 с.

7. Б. Хусаинов. Структуры и алгоритмы обработки данных. Примеры на языке Си. / М.: Финансы и статистика, 2010 – 464 с.

8. Р. Седжвик. Фундаментальные алгоритмы на С. Часть 5. Алгоритмы на графах. / М.: ДиаСофт ЮП, 2009 – 480 с.

9. Уайт, Т. Hadoop. Подробное руководство. / Том Уайт. – СПб.: Питер, 2013. – 672 с. – ISBN 978-5-496-00662-0.

10. Бэнкер, К. MongoDB в действии, пер. А. Слинкин / Кайл Бэнкер. – М.: ДМК Пресс, 2014. – 394 с. – ISBN 978-5-94074-831-1, 978-1-93518-287-0, 978-5-97060-057-3.

11. Лэм, Ч. Hadoop в действии. / Чак Лэм. – М.: ДМК Пресс, 2012. – 424 с. – ISBN 978-5-94074-785-7.