### Лабораторная работа № 13

### Введение в MongoDB (NoSQL - Not only SQL)

В соответствии с рейтингами последних лет к лидирующим моделям баз данных относятся реляционные СУБД (relational DBMS): Oracle Database, MySQL, Microsoft SQL Server, PostgreSQL и документоориентированная СУБД (document-oriented database) MongoDB. Следует отметить, что DBMS MongoDB относится к нереляционным СУБД NoSQL с новыми методами хранения данных и средствами доступа к ним.

В зависимости от модели базы данных к основным СУБД NoSQL относятся:

* БД на основе пар «ключ - значение» (DynamoDB, Redis, Aerospike и др.)
* Колоночно-ориентированные СУБД или column-oriented DBMS (Cassandra, Hypertable, ConcourseDB и др)
* Документоориентированные СУБД (MongoDB, Azure DocumentDB, CouchDB и др.)
* Графовые БД (Neo4j, InfoGrid, Bigdata и др.)
* Базы данных XML (EMC Documentum xDB, eXist, Sedna и др.)

Если реляционную модель баз данных традиционно используют для оперативной обработки транзакций или в системах поддержки принятия решений, то модель NoSQL применяют для оперативной обработки данных. Технология NoSQL обеспечивает эффективное хранение и обработку огромных объемов неструктурированных данных, которые требуют высоких скоростей для выполнения операций чтения и записи.

Следует отметить, что базы данных NoSQL являются одной из технологий обработки Big data. Особенностью технологий NoSQL является неограниченное горизонтальное масштабирование и высокая производительность. Необходимо также отметить, что в некоторых реляционных СУБД, например, таких как PostgreSQL реализованы средства обработки неструктурированных данных в формате JSON.

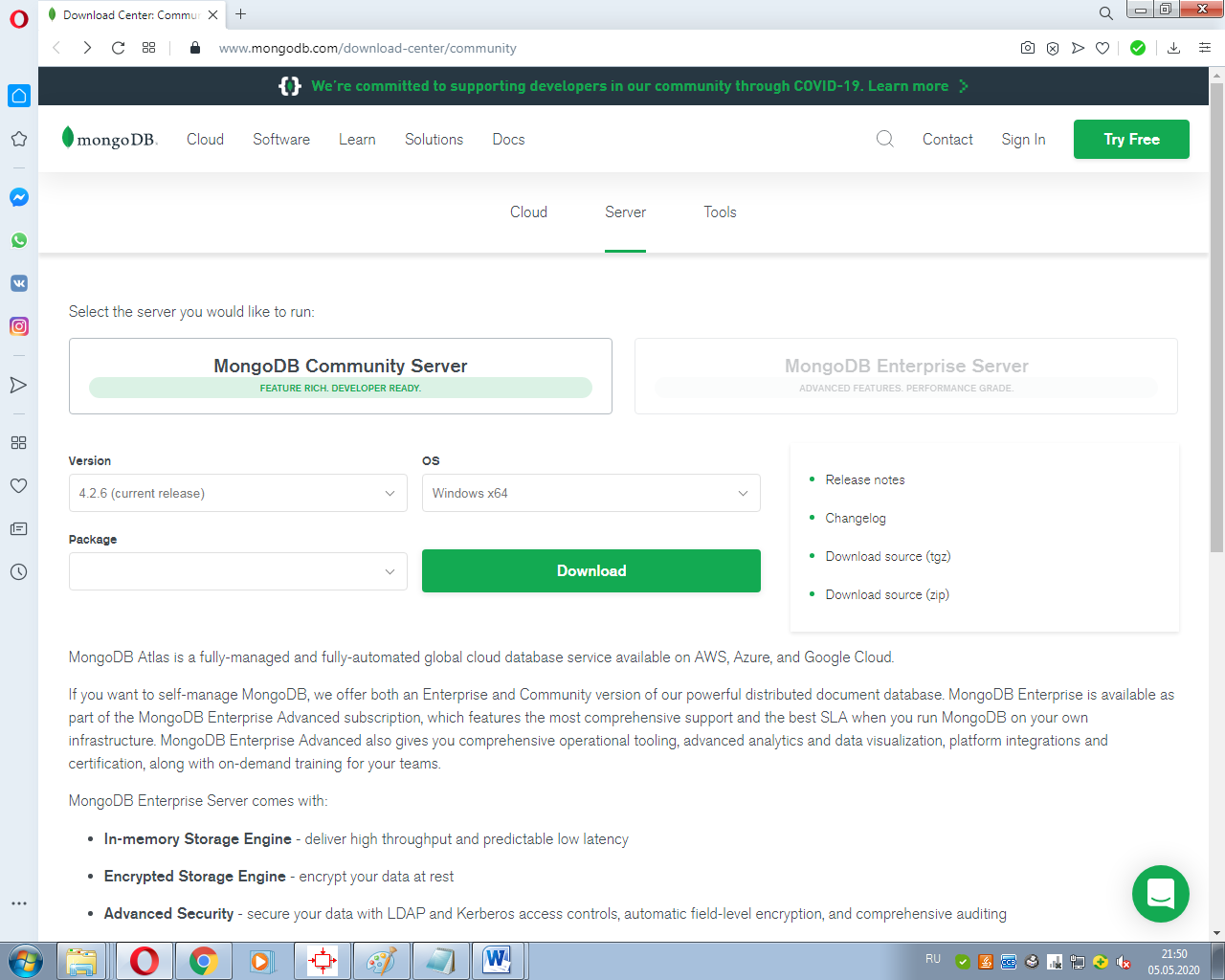
БД MongoDB востребованна при разработке Web-приложений на основе Node.js. (традиционный стек технологий для построения веб-приложений – LAMP: Linux, Apache, MySQL, PHP; стек технологий с использованием БД MongoDB - MEAN (Mongo, Express, Angular, Node).

**ЗАДАНИЕ №1. Ознакомиться с видео** [**https://www.youtube.com/watch?v=RBCVGRkDMQM**](https://www.youtube.com/watch?v=RBCVGRkDMQM)

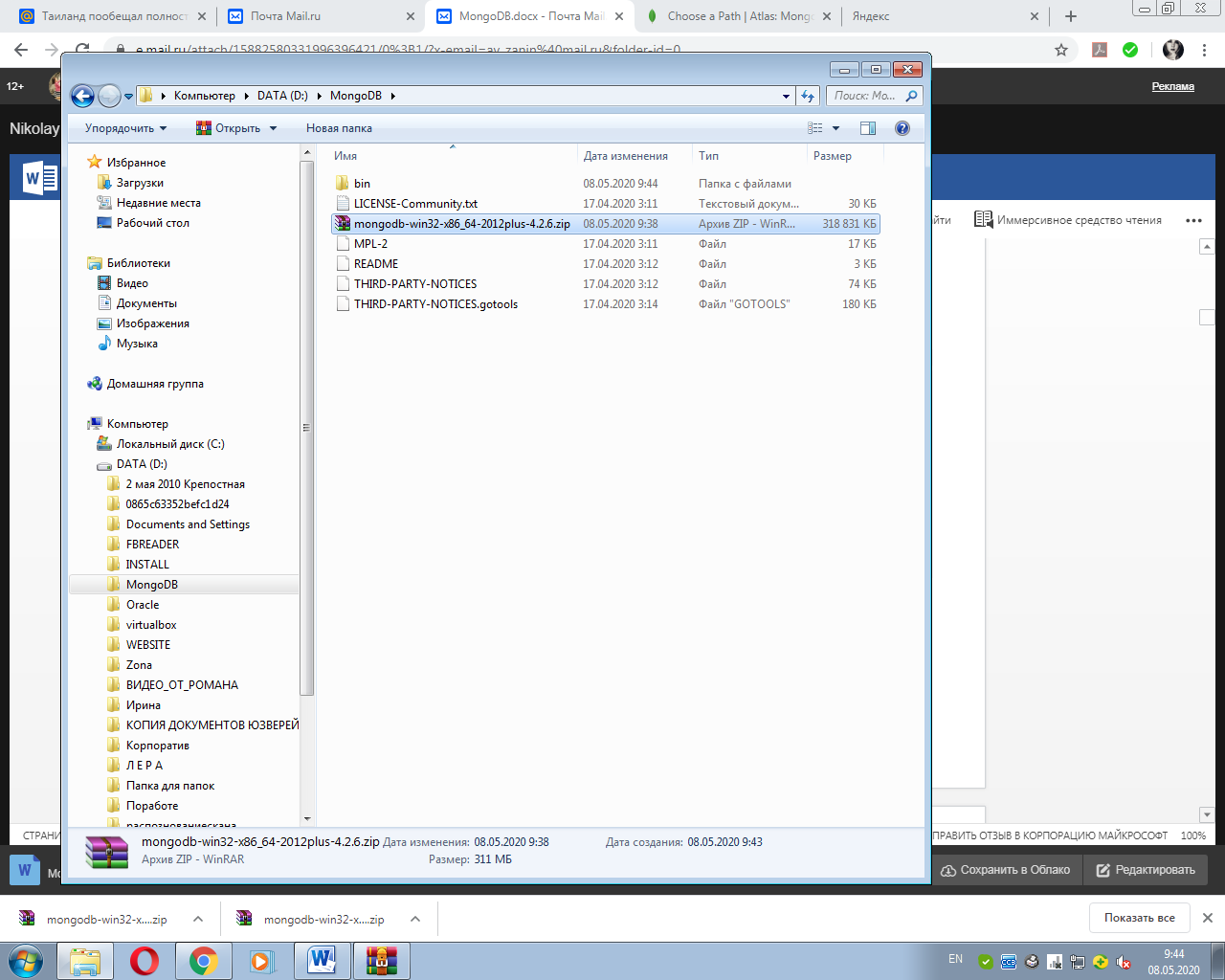
MongoDB — документоориентированная СУБД с открытым исходным кодом, не требующая описания схемы таблиц. Она классифицируется как NoSQL и использует BSON (бинарный JSON\*), т.е. каждая запись это документ, без жестко заданной схемы, который может содержать вложенные документы. Хорошо масштабируется, написана на языке C++ и поддерживает синтаксис JavaScript. Поддержка SQL отсутствует. У MongoDB есть драйверы для многих популярных языков программирования (Си, C++, C#, Go, Java, JavaScript, Perl, PHP, Python, Ruby и др.). Также есть неофициальные и поддерживаемые сообществом драйверы для прочих языков программирования.

**ЗАДАНИЕ №2 Установка и начало работы с MongoDB**

Для установки MongoDB загрузим один распространяемых пакетов с официального сайта <https://www.mongodb.com/download-center/community>



Сформируйте архив ZIP, версии 4.2.6 (пошаговую установку см. <https://metanit.com/nosql/mongodb/1.1.php>)



Создайте директорию c:\MongoDB. Распакуйте архив в эту директорию.

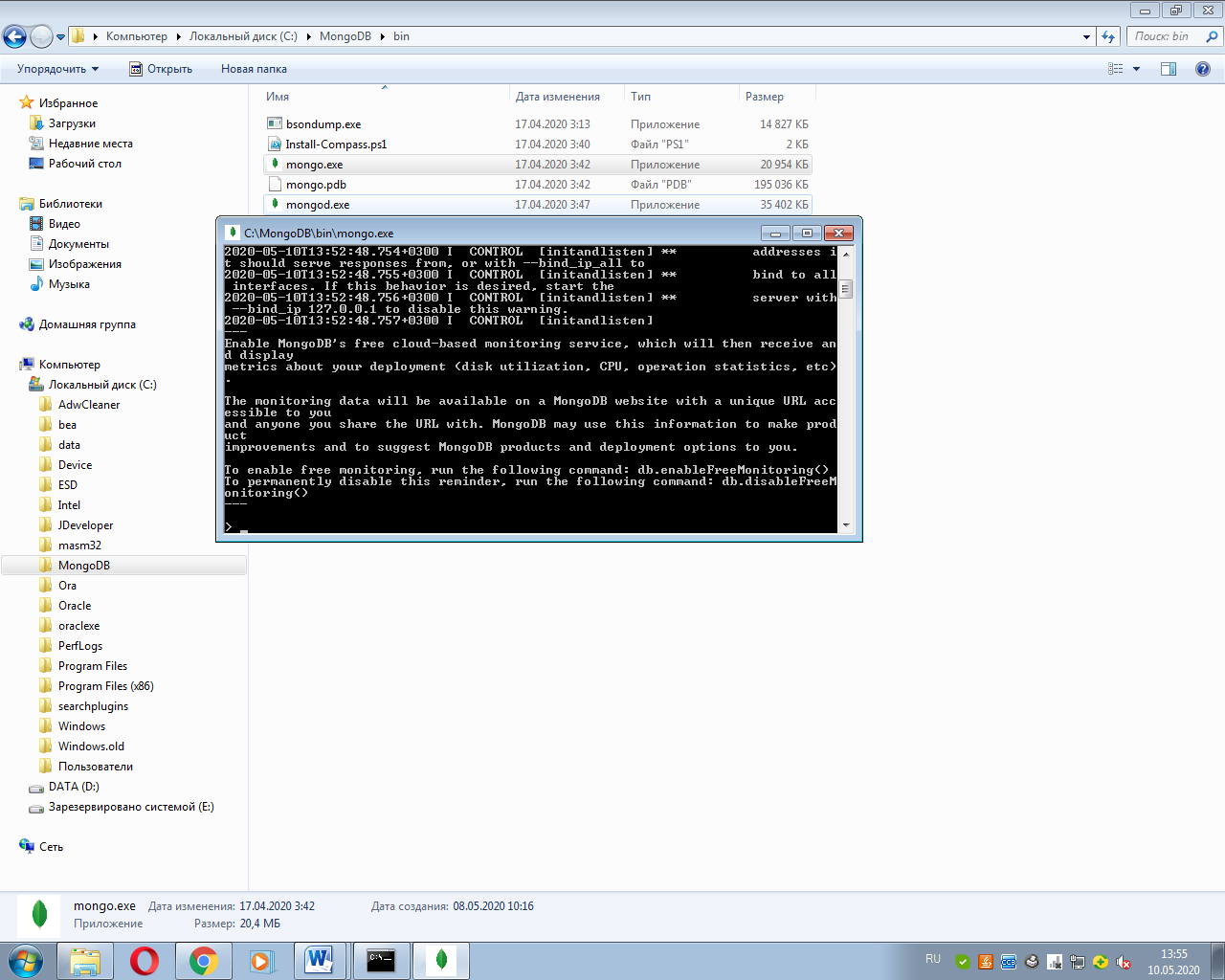
Создайте директорию для хранения базы с:\data\db

Запустите командную строку (в Windows) или консоль в Linux.

Для работы с БД установите нужный каталог cd c:\MongoDB\bin

Запустите БД c:\MongoDB\bin>mongod

Из каталога c:\MongoDB\bin (в Windows) запустите mongo.exe



Дальнейшая работа будет в интерактивном режиме из строки

>

( если возникли проблемы с установкой см. видео <https://www.youtube.com/watch?v=TIH3dcZjrzk>)

**ЗАДАНИЕ №3 Осуществить следующие манипуляции в командной строке (БД)**

- подсказка

>help

- вывести названия всех имеющихся бд

>show dbs

-создадим новую базу данных по имени users

> use users

>show dbs

-создадим еще одну новую базу данных по имени users1

> use users1

- узнать, какая бд используется в текущей момент

>db

- перейти в базу данных users

> use users

- удалить базу данных можно командой (важный момент - нужно находиться в той базе данных, которую необходимо удалить) :

>db.dropDatabase()

>show dbs

- перейти в базу данных users1 и удалить ее

> use users1

>db.dropDatabase()

>show dbs

- получить статистику о сервере MongoDB

>db.stats ()

**КОЛЛЕКЦИИ**

|  |  |
| --- | --- |
|  |  |

**\*JSON** (JavaScript Object Notation) - простой формат обмена данными, удобный для чтения и написания, как человеком, так и компьютером. Он основан на подмножестве [языка программирования JavaScript](http://javascript.crockford.com/). JSON - текстовый формат, полностью независимый от языка реализации, но он использует соглашения, знакомые программистам C-подобных языков, таких как C, C++, C#, Java, JavaScript, Perl, Python и многих других. Эти свойства делают JSON идеальным языком обмена данными. JSON основан на двух структурах данных: Коллекция пар ключ/значение. В разных языках, эта концепция реализована как *объект*, запись, структура, словарь, хэш, именованный список или ассоциативный массив. Упорядоченный список значений. В большинстве языков это реализовано как *массив*, вектор, список или последовательность. Это универсальные структуры данных. Почти все современные языки программирования поддерживают их в какой-либо форме.

Метод createCollection () db.createCollection (имя, параметры) используется для создания коллекции.

Синтаксис команды createCollection () следующий:

db.createCollection(name, options)

name это имя коллекции, которая будет создана. Параметры – это документ, который используется для указания конфигурации коллекции.

|  |  |  |
| --- | --- | --- |
| параметр | Тип | Описание |
| название | строка | Название коллекции, которая будет создана |
| Опции | Документ | (Необязательно) Укажите параметры, касающиеся объема памяти и индексации |

Параметр options является необязательным, поэтому необходимо указывать только название коллекции. Ниже приведен список опций, которые вы можете использовать –

|  |  |  |
| --- | --- | --- |
| поле | Тип | Описание |
| ограничен | логический | (Необязательно) Если установлено значение true, включает ограниченную коллекцию. Ограниченная коллекция – это коллекция фиксированного размера, которая автоматически перезаписывает свои самые старые записи, когда достигает максимального размера. Если вы указываете true, вам также нужно указать параметр размера. |
| autoIndexId | логический | (Необязательно) Если установлено значение true, автоматически создавать индекс для поля \_id.s Значение по умолчанию – false. |
| размер | число | (Необязательно) Указывает максимальный размер в байтах для ограниченной коллекции. Если значение capped равно true, то вам также необходимо указать это поле. |
| Максимум | число | (Необязательно) Указывает максимально допустимое количество документов в ограниченном собрании. |

При вставке документа MongoDB сначала проверяет поле размера коллекции с ограничениями, а затем проверяет поле max.

**ЗАДАНИЕ №4 Осуществить следующие манипуляции в командной строке (простые команды)**

> db.mycoll.help()

> db.mycoll. createCollection.help()

- Создать коллекцию c именем "mycollection" в БД test (коллекция может создаваться автоматически при добавлении документов, но можно создать явно)

>use test

>db.createCollection("mycollection")

- Создать коллекцию

>db.createCollection("mycol", { capped : true, autoIndexId : true, size :

6142800, max : 10000 } )

- Создать коллекцию:

db.createCollection("ps")

- Вывести названия всех имеющихся коллекций

>show collections

 - Удаление коллекции:

>db.ps.drop()

- Переименовать коллекцию (см. коллекции не было, создание – автоматически) :

>db.ps.renameCollection("peoples")

 >show collections

- Добавить один документ в коллекцию:

> db.insertOne.help()

>db.peoples.insertOne({"name": "Mike", "lastname": "Kozlov"})

-Добавить несколько документов в коллекцию:

>db.peoples.insertMany([{"name": "Mike", "lastname": "Kozlov"},{"name": "Ivan", "lastname": "Kruglov"}])

-Посчитать количество документов в коллекции и при выборке:

>db.peoples.count()

>db.peoples.find({name: "Mike"}).count()

>db.peoples.find({name: "Ivan"}).count()

-Добавить любое количество документов в коллекцию:

>db.peoples.insert([{"name": "Mike", "lastname": "Kozlov"}])

-Пример добавления с вложенными данными:

>db.peoples.insert({"name": "Mike", "lastname": "Kozlov", address: {"country": "Russia", "city": "Moscow", "index": 111511}})

-Выборка из коллекции:

>db.peoples.find()

-Выборка из коллекции уникальных значений поля "name":

>db.peoples.distinct("name")

-Выборка по одному и нескольким условиям:

>db.peoples.find({name: "Mike"})

>db.peoples.find({lastname: "Kozlov"})

-Выборка документов без какого-то свойства(полю присваиваем значение 0 либо true\false):

>db.peoples.find({name: "Mike"}, {lastname: 0})

>db.peoples.find({name: "Mike"}, {lastname: false})

-Выборка всех документов без какого-то свойства:

>db.peoples.find({}, {lastname: 0})

-Выборка по вложенному элементу:

>db.peoples.find({"address.country": "Russia"})

-Выборка с ограничением по количеству выводимых документов:

>db.peoples.find().limit(2)

-Выборка с сортировкой по полю(по возрастанию - 1, по убыванию - -1):

>db.peoples.find().sort({name: 1})

-Выборка с несколькими функциями:

>db.peoples.find().sort({name: 1}).limit(2)

-Выборка с помощью JavaScript:

>func = function() { return this.name=="Mike"; }

>db.peoples.find(func)

-Выборка с условными операторами:

$eq - =

$gt - >

$lt - <

$gte - >=

$lte - <=

$ne - <>

-Пример с двумя условиями больше и меньше указанных значений:

>db.peoples.find({"address.index": {$gt : 111500, $lt : 111700}})

 Изменение данных

Изменение документа:

Функция update принимает три параметра для обновления документа:

1 документ, который будет изменен

2 документ, на который будет изменен

3 Параметры обновления документов (upsert и multi)

3.1.1 upsert имеет значение true, то будет обновлен документ, если он найден, и создан новый, если документа нет

3.1.2 upsert имеет значение false, то не будет создан новый документ, если найденного документа нет

3.2.1 multi определяет, должен ли обновиться первый элемент в выборке (используется по умолчанию, если не указан)

3.2.2 multi определяет, должны обновляться все документы выборки

-Пример (найденный документ - {name : "Mike"}, будет заменен документом {name: "Mike", lastname: "Degtyarev"}):

>db.peoples.update( { name : "Mike" }, { name: "Mike", lastname: "Degtyarev" }, { upsert: true } )

-Изменение одного поля первого в выборке документа, укажем оператор $set, если документт не имеет изменяемого поля, создастся новое поле в документете:

>db.peoples.update({name : "Mike", lastname: "Degtyarev"}, {$set: {lastname: "Kozlov"}})

-В данном случае изменится первый в выборке документ, указав оператор multi:true, изменятся все документы из выборки:

>db.peoples.update({name : "Mike", lastname: "Degtyarev"}, {$set: {lastname: "Kozlov"}}, {multi:true})

Удаление данных

-Удаление поля, укажем оператор $unset:

>db.peoples.update({name : "Mike"}, {$unset: {lastname: "Kozlov"}})

Просмотрим документы

>db.peoples.find()

>db.peoples.update({name : "Mike"}, {$unset: {lastname: "Kozlov", name: "Mike"}})

 Просмотрим документы

>db.peoples.find(

-Удаление документа:

Посчитаем количество

>db.peoples.find({name: "Mike"}).count()

>db.peoples.deleteOne({name : "Mike"})

 Посчитаем количество

>db.peoples.find({name: "Mike"}).count()

-Удаление множества документов:

>db.peoples.remove({name : "Mike"})

-Удаление всех элементов коллекции:

db.peoples.remove({})

 Посчитаем количество

>db.peoples.find({name: "Mike"}).count()