



# mongoDB

Prof. P. M. Jadav

Associate Professor,

Computer Engineering Department

Dharmsinh Desai University, Nadiad

# Content

- Background (NoSQL)
- Introduction
- Features of MongoDB
- Advantages of MongoDB
- RDBMS vs MongoDB
- Data Types in MongoDB
- MongoDB Installation
- MongoDB Components
- MongoDB Shell Commands

# Background (NoSQL)

- database used to manage huge sets of unstructured data
- data is not stored in tabular relations like relational databases
- designed to overcome the
  - Performance,
  - Scalability,
  - Data Modelling and
  - Distribution limitations

that are seen in the Relational Databases

# NoSQL Database Types

- **Document Databases**: key is paired with a complex data structure called as Document (Ex. **MongoDB**)
- **Graph stores**: used to store networked data. Where in we can relate data based on some existing data.
- **Key-Value stores**: a key is used to identify record (Ex. **Redis**)
- **Wide-column stores**: Used to store large data sets (Ex. **Cassandra** (Used in Facebook), **HBase** etc.)

# Introduction

- Open-source, document based NoSQL database
- developed by Eliot Horowitz and Dwight Merriman in the year 2007
- Stores the data in form of key-value pairs
- High performance and scalable
- Cross-platform database (Windows, Linux etc.)
- name derived from the word humongous to support the idea of processing large amount of data.

# Document-based database

- data structure with name-value pairs
- Hierarchical data storage
- JSON representation of custom Objects
- Schema-less
- Data is stored in **BSON** (Binary JSON)

# Sample Document

```
{
  _id      : ObjectId("5099803df3f4948bd2f98391"),
  name     : { first: "Alan", last: "Turing" },
  birth    : new Date('Jun 23, 1912'),
  death    : new Date('Jun 07, 1954'),
  contribs : [ "Turing machine", "Turing test", "Turingery" ],
  view     : NumberLong(1250000)
}
```

# Features of MongoDB

- I/O operations are lesser compare to RDBMS due to support of embedded documents
- select queries are faster due to faster indexing support
- rich query language
- Auto-replication feature leads to high availability
- Support of Automatic failover
- Horizontal scalability due to Sharding feature
- Support of multiple storage engine



# Advantages of MongoDB

- Schema-less database
- Dynamic query by document query language
- Scalable
- No complex joins are needed
- SQL injection is not possible
- Search by REGEX and fields
- No need of mapping application objects to data objects
- Index on any attribute
- Fast in-place update

# Organizations that use MongoDB

- Adobe
- LinkedIn
- McAfee
- FourSquare
- eBay
- MetLife
- SAP

# RDBMS Vs. MongoDB

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
Column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key <b>_id</b> )
ACID Property	CAP theorem (Consistency, Availability and Partition tolerance)

## id (primary key)

- **\_id** is a 12 bytes hexadecimal number
- assures the uniqueness of every document
- If you don't provide then MongoDB provides a unique id for every document
- These 12 bytes
  - first 4 bytes for the current timestamp
  - next 3 bytes for machine id
  - next 2 bytes for process id of MongoDB server and
  - remaining 3 bytes are simple incremental VALUE.

# MongoDB Data Types

- String** – String in MongoDB must be UTF-8 valid
- Integer** – 32/64 bit depending upon your server
- Boolean** – stores a boolean (true/ false) value
- Double** – This type is used to store floating point values
- Min/ Max keys** – used to compare a value against the lowest and highest BSON elements
- Arrays** – stores arrays/list/multiple values into one key
- Timestamp** – ctimestamp
- Object** – used for embedded documents

# MongoDB Data Types

**Null** – used to store a Null value

**Symbol** – identical to a string; however, it's generally reserved for languages that use a specific symbol type

**Date** – stores the current date-time in UNIX time format

**Object ID** – used to store the document's ID

**Binary data** – This datatype is used to store binary data

**Code** – stores JavaScript code into the document

**Regular expression** – stores regular expression

# MongoDB Installation

- Download and install the MongoDB community server from the following url:

<https://www.mongodb.com/download-center/v2/community>

- On Windows the mongod.exe executables will be in the folder something like this:

C:\Program Files\MongoDB\Server\4.0\bin\

# Setup MongoDB Environment

- MongoDB requires a **data directory** to store all data
- MongoDB's default data directory path is the absolute path **\data\db** on the drive from which you start MongoDB
- Create this folder by running the following command:

```
mkdir c:\data\db
```



# MongoDB Components

Component Set	Binaries
Server	<code>mongod.exe</code>
Client	<code>mongo.exe</code>
Router	<code>mongos.exe</code>
Monitoring Tools	<code>mongostat.exe</code> , <code>mongotop.exe</code>
Import-Export Tools	<code>mongodump.exe</code> , <code>mongoexport.exe</code> , <code>mongoimport.exe</code>
Miscellaneous Tools	<code>bsondump.exe</code> , <code>mongofiles.exe</code> , <code>mongoperf.exe</code>

# Running MongoDB

- Start the server:

`mongod`

- If your data path is different other than default, then run:

`mongod --dbpath d:\data\db`

- Start the shell to connect to server:

`mongo`

- Default port number of MongoDB server is 27017 if it is running on some other port no. (say 28012), then run following command:

`mongo --port 28012`

# MongoDB Shell Commands

- To clear the screen:

`cls`

- To view existing databases:

`show dbs`

- To create/connect to an existing database:

`use <db_name>`

- To check currently selected database:

`db`

- To drop database, first select the database and then drop it by:

`db.dropDatabase()`

# MongoDB Shell Commands

- To view collections:

`show collections`

- To create students collection:

`db.createCollection("students")`

- To insert a document to students collection:

`db.students.insert( {name: "Viren" } )`

If you insert a document in the collection and if, the collection doesn't exist, it will be created automatically

- To drop students collection:

`db.students.drop()`

# MongoDB Shell Commands

- To **query** data from collection:

```
db.students.find()
```

- To **display** the results in a **formatted** way:

```
db.students.find().pretty()
```

- To **query** the document on the basis of some condition:

```
db.students.find( {name: "Viren" } )
```

```
db.students.find( {roll_no: {$gt: 10} } )
```

```
db.students.find( {roll_no: {$gte: 10} } )
```

```
db.students.find( {roll_no: {$gte: 10} } ).limit(5)
```

```
db.students.find( {roll_no: {$gte: 10} } ).sort( { roll_no: -1 } )
```

# MongoDB Shell Commands

- To **query** the document on the basis of some condition:

```
db.students.find( {  
    $and : [ {roll_no: 10} , {name: "Viren"} ]  
})
```

```
db.students.find( {  
    $or : [ {roll_no: 10} , {name: "Viren"} ]  
})
```

# MongoDB Shell Commands

- To **update** the document on the basis of some condition:

`db.<collection_name>.update(SELECTION_CRITERIA, UPDATED_DATA)`

```
db.students.update(  
    {roll_no: 5} ,  
    {$set: {mobile: "9876512345"}}  
)
```

It will update only single document

# MongoDB Shell Commands

- To **update** multiple document on the basis of some condition:

```
db.students.update(  
    {roll_no: 5} ,  
    {$set: {mobile: "9876512345"} },  
    { multi: true }  
)
```

- To **create** an **index** (descending order) on field 'name':

```
db.students.createIndex( { name: -1 } )
```



# MongoDB Shell Commands

- Save a New Document without Specifying an `_id` Field:

```
db.students.save( {roll_no : 7} )
```

It will insert a new student document with a new `_id` field with a unique ObjectId value

- Save a New Document Specifying an `_id` Field:

```
db.students.save( { _id: 10, roll_no : 8} )
```

it will insert a new document if a document with `_id=10` doesn't exist, otherwise update the document with `_id=10`

# MongoDB Shell Commands

- To **remove** all documents from a collection:

```
db.students.remove( { } )
```

- To **remove** all documents that match a condition:

```
db.students.remove( { roll_no : { $gt > 10 } } )
```

- To **remove** a single document that match a condition:

```
db.students.remove( { roll_no : { $gt > 10 } }, true )
```

# References

- <https://docs.mongodb.com/manual/>
- <https://www.tutorialspoint.com/mongodb/>
- <https://www.studytonight.com/mongodb/introduction-to-mongodb>