What is MongoDB

- MongoDB is an open-source document database and leading NoSQL database
- Schema less
- Stores JSON objects
- document oriented database that provides
  - high performance
  - high availability
  - easy scalability

Why MongoDB

- Document Oriented Storage: Data is stored in the form of JSON style documents.
- Index on any attribute
- Geo Location support
- Replication and high availability
- Auto-sharding
- Rich queries
- Fast in-place updates
- Professional support by MongoDB

MongoDB Overview

- Database
  - Physical Container of Collection
- Collections
  - Collection is a group of MongoDB documents
  - equivalent of an RDBMS table
  - Collections do not enforce a schema.
- Document
  - set of key-value pairs.
  - Documents have dynamic schema

RDBMS and MongoDB

- Database ←→ Database
- Table ←→ Collection
- Row ←→ Document
- Column ←→ Field

Sample Document

```
{
    _id: ObjectId(7df78ad8902c)
    a: ‘1’,
    b: ‘2’
}
```

Advantages of MongoDB

- Schema less
- Structure of a single object is clear.
- No complex joins.
- Supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
- Tuning.
- Ease of scale-out: MongoDB is easy to scale.
- Conversion/mapping of application objects to database objects not needed.
- Uses internal memory for storing the (windowed) working set, enabling faster access of data

Create Collection

- Db.createcollection(“collection_name” Options)
- Example:

MongoDB shell version: 2.4.14

connecting to: test

> show dbs

local 0.078125GB

test 0.203125GB

> use test

switched to db test

> db.createCollection("test_collection")

{ "ok" : 1 }

>

Drop Collection

- db.COLLECTION_NAME.drop()

MongoDB shell version: 2.4.14

connecting to: test

> show collections

system.indexes

test_collection

> db.test_collection.drop()

true

Insert Document

- db.COLLECTION_NAME.insert(document)

```
>db.test_collection.insert({
  title: ‘nosql',
 description: 'basic description',
 by: 'rajeev',
 url: 'https://www.linkedin.com/in/rajeevguptajavatrainer/',
})
```

Query Document

- db.COLLECTION_NAME.find(document)

Example:

>db.test_collection.find()

>db.test_collection.find().pretty()

>db.test_collection.find({"title" : "nosql"})

> db.test_collection.find({"title" : "nosql"},{"by":1}).pretty()

{ "_id" : ObjectId("5791d58760a74da5b3e51eb9"), "by" : "rajeev" }

> db.test_collection.find({"title" : "nosql"},{"by":1,_id:0}).pretty()

{ "by" : "rajeev" }

Query Document

- db.COLLECTION_NAME.find(document)

Example:

>db.test_collection.find()

>db.test_collection.find().pretty()

>db.test_collection.find({"title" : "nosql"})

> db.test_collection.find({"title" : "nosql"},{"by":1}).pretty()

{ "_id" : ObjectId("5791d58760a74da5b3e51eb9"), "by" : "rajeev" }

> db.test_collection.find({"title" : "nosql"},{"by":1,_id:0}).pretty()

{ "by" : "rajeev" }

SQL vs Mongodb

SQL SELECT Statements	MongoDB find() Statements
SELECT * **FROM** users	**db**.**users**.**find**()
SELECT id, user_id, status **FROM** users	**db**.**users**.**find**({ }, { **user_id**: 1, **status**: 1 })
SELECT user_id, status **FROM** users	**db**.**users**.**find**({ }, { **user_id**: 1, **status**: 1, **_id**: 0 })
SELECT * **FROM** users **WHERE** status = "A"	**db**.**users**.**find**({ **status**: "A" })
SELECT user_id, status **FROM** users **WHERE** status = "A"	**db**.**users**.**find**({ **status**: "A" }, { **user_id**: 1, **status**: 1, **_id**: 0 })
SELECT * **FROM** users **WHERE** status != "A"	**db**.**users**.**find**({ **status**: { **$ne**: "A" } })
SELECT * **FROM** users **WHERE** status = "A" **AND** age = 50	**db**.**users**.**find**({ **status**: "A", **age**: 50 })
SELECT * **FROM** users **WHERE** status = "A" **OR** age = 50	**db**.**users**.**find**({ **$or**: [{ **status**: "A" } , { **age**: 50 }] })
SELECT * **FROM** users **WHERE** age > 25	**db**.**users**.**find**({ **age**: { **$gt**: 25 } })

SQL vs Mongodb

SELECT * **FROM** users **WHERE** age < 25	**db**.**users**.**find**({ **age**: { **$lt**: 25 } })
SELECT * **FROM** users **WHERE** age > 25 **AND** age <= 50	**db**.**users**.**find**({ **age**: { **$gt**: 25, **$lte**: 50 } })
SELECT * **FROM** users **WHERE** user_id **like** "%bc%"	**db**.**users**.**find**({ **user_id**: /bc/ })
SELECT * **FROM** users **WHERE** user_id **like** "bc%"	**db**.**users**.**find**({ **user_id**: /^bc/ })
SELECT * **FROM** users **WHERE** status = "A" **ORDER BY** user_id **ASC**	**db**.**users**.**find**({ **status**: "A" }).**sort**({ **user_id**: 1 })
SELECT * **FROM** users **WHERE** status = "A" **ORDER BY** user_id **DESC**	**db**.**users**.**find**({ **status**: "A" }).**sort**({ **user_id**: -1 })
SELECT COUNT(*) **FROM** users	**db**.**users**.**count**() *or* **db**.**users**.**find**().**count**()
SELECT COUNT(user_id) **FROM** users	**db**.**users**.**count**({ **user_id**: { **$exists**: **true** } }) *or* **db**.**users**.**find**({ **user_id**: { **$exists**: **true** } }).**count**()

Usefull MongoDB commands

- Db.createcollection(users)
- Db.users.insert({“name”: “XYZ”})
- db.users.createIndex({ user_id: 1 })
- db.users.update(

  { age: { $gt: 25 } },

  { $set: { status: "C" } },

  { multi: true }

)
- db.users.remove({ status: "D" })