SLIIT
Discover Your Future

NoSQL Databases

Document Databases – MongoDB

Year 3 – Semester 2

B.Sc. (Hons) in Information Technology Specializing in Software Engineering

SLIIT
Discover Your Future

Contents

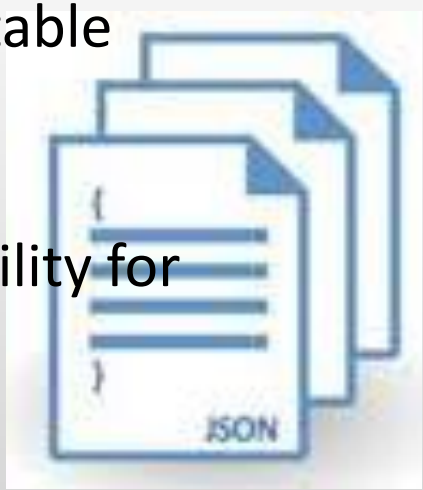
- What is a Document database ?
- What is MongoDB ?
- Why use MongoDB ?
- What is MongoDB great for and not ?
- MongoDB Basics
- MongoDB – CRUD operations
- Get started with MongoDB

SLIIT
Discover Your Future

What is a Document database ?

Document Databases

- Document database is a type of NoSQL database.
- A document database is used for storing, retrieving, and managing semi-structured data.
- Unlike traditional relational databases, the data model in a document database is not structured in a table format of rows and columns.
- The schema can vary, providing far more flexibility for data modeling than relational databases.



Document Databases: Representatives

MS Azure DocumentDB

SLIIT
Discover Your Future

*MongoDB (from "hu**mongo**us") is a scalable, high - performance, open source, schema - free, document – oriented database.*

Mongodb.org

MongoDB

- A NoSQL database of type document oriented
- Eschews the traditional table based relational database structure in favor of JSON like documents with dynamic schemas (MongoDB calls the format BSON)
- First developed by MongoDB Inc in October 2007
- Shifted to open source in 2009

SLIIT
Discover Your Future

MongoDB

- As of May 2018, it is the fifth most popular type of database management system, and the most popular for document stores.

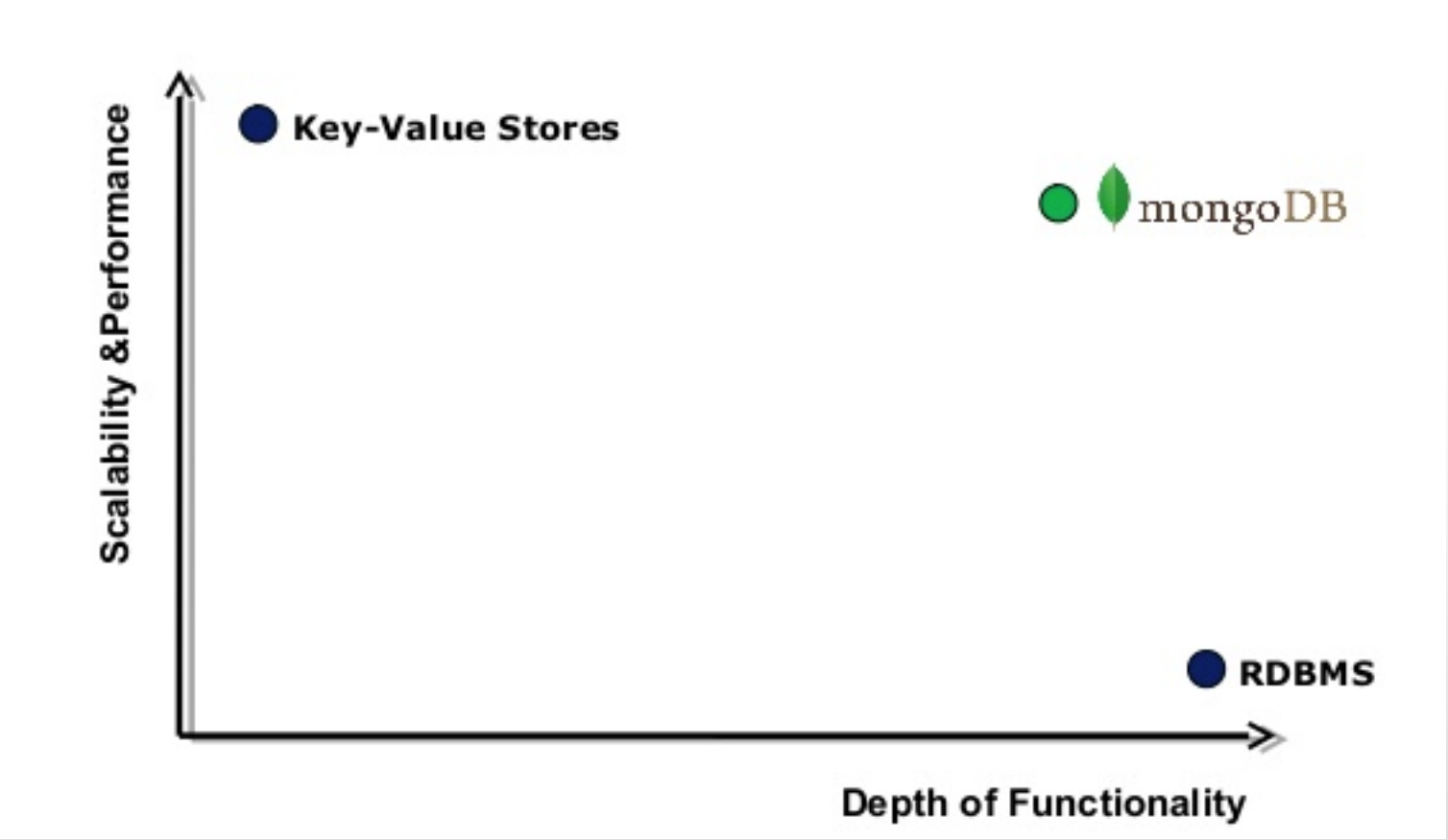
342 systems in ranking, May 2018

Rank			DBMS	Database Model	Score		
May 2018	Apr 2018	May 2017			May 2018	Apr 2018	May 2017
1.	1.	1.	Oracle	Relational DBMS	1290.42	+0.63	-63.90
2.	2.	2.	MySQL	Relational DBMS	1223.34	-3.06	-116.69
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1085.84	-9.67	-127.96
4.	4.	4.	PostgreSQL	Relational DBMS	400.90	+5.43	+34.99
5.	5.	5.	MongoDB	Document store	342.11	+0.70	+10.53
6.	6.	6.	DB2	Relational DBMS	185.61	-3.34	-3.23
7.	↑ 9.	↑ 9.	Redis	Key-value store	135.35	+5.24	+17.90
8.	↓ 7.	↓ 7.	Microsoft Access	Relational DBMS	133.11	+0.89	+3.24
9.	↓ 8.	↑ 11.	Elasticsearch	Search engine	130.44	-0.92	+21.62
10.	10.	↓ 8.	Cassandra	Wide column store	117.83	-1.26	-5.28

https://db-engines.com/en/ranking

SLIIT
Discover Your Future

Where MongoDB stands ?



SLIIT
Discover Your Future

Why use MongoDB ?

Why use MongoDB ?

- SQL was invented in the 70's to store data.
- MongoDB stores documents (or) objects.
- Now-a-days, everyone works with objects (Python/Ruby/Java/etc.)
- And we need Databases to persist our objects.
- Then why not store objects directly ?
- Embedded documents and arrays reduce need for joins. No Joins and No-multi document transactions.

SLIIT
Discover Your Future

What is MongoDB great for and not?

What is MongoDB great for ?

- RDBMS replacement for Web Applications.
- Semi-structured Content Management.
- Real-time Analytics & High-Speed Logging.
- Caching and High Scalability

  Web 2.0, Media, SAAS, Gaming

  HealthCare, Finance, Telecom, Government

Not great for ?

- Highly Transactional Applications.
- Problems requiring SQL.

Some companies using MongoDB in production

SLIIT
Discover Your Future

MongoDB Basics

MongoDB Basics

- A MongoDB instance has databases.

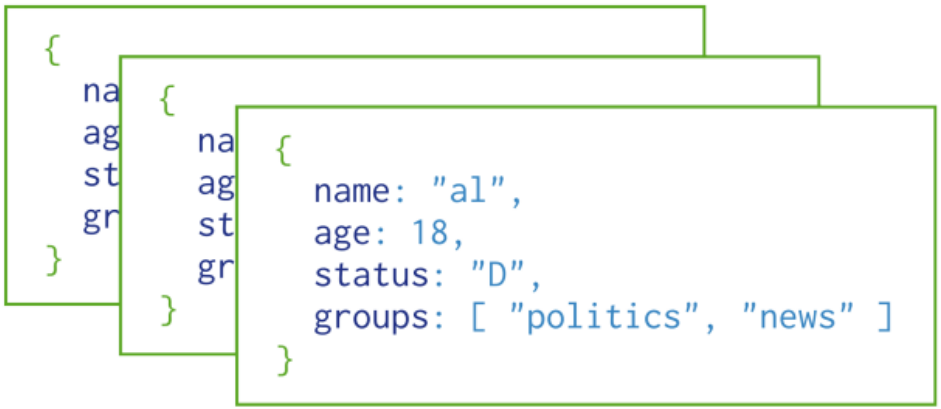
  - high level containers

- A database have collections.
- Collections are made up of documents.
- Documents have fields.
- When asking data from MongoDB, the result set is returned as a cursor.

  - can count, skip ahead, before actually pulling down data

SLIIT
Discover Your Future

MongoDB Basics



Each document within a collection can have its own unique set of fields.

```
{
  name: "sue",                    <-- field: value
  age: 26,                        <-- field: value
  status: "A",                    <-- field: value
  groups: [ "news", "sports" ]    <-- field: value
}
```

MongoDB CRUD

MongoDB CRUD

- MongoDB provides rich semantics for reading and manipulating data.
- CRUD = Create, Read, Update and Delete

SLIIT
Discover Your Future

CRUD : Create

```
db.users.insert (
   {
      name: "sue",
      age: 26,
      status: "A"
   }
)
```

collection

field: value

field: value

field: value

document

The following diagram shows the same query in SQL:

```
INSERT INTO users
            ( name, age, status )
VALUES      ( "sue", 26, "A" )
```

table

columns

values/row

SLIIT
Discover Your Future

CRUD : Read

```
db.users.find(                          ← collection
    { age: { $gt: 18 } },               ← query criteria
    { name: 1, address: 1 }             ← projection
).limit(5)                              ← cursor modifier
```

The next diagram shows the same query in SQL:

```
SELECT _id, name, address   ← projection
FROM   users                ← table
WHERE  age > 18             ← select criteria
LIMIT  5                    ← cursor modifier
```

SLIIT
Discover Your Future

CRUD : Update

```
db.users.update(                          <-- collection
    { age: { $gt: 18 } },                 <-- update criteria
    { $set: { status: "A" } },            <-- update action
    { multi: true }                       <-- update option
)
```

The following diagram shows the same query in SQL:

```
UPDATE users          <-- table
SET    status = 'A'   <-- update action
WHERE  age > 18       <-- update criteria
```

SLIIT
Discover Your Future

CRUD : Delete

```
db.users.remove(          <-- collection
    { status: "D" }       <-- remove criteria
)
```

```
DELETE FROM users         <-- table
WHERE  status = 'D'       <-- delete criteria
```

*Note that insert, update, delete have been deprecated as of current Mongo documentation

SLIIT
Discover Your Future

Get started with MongoDB

SLIIT
Discover Your Future

Get started with MongoDB

- You can either install MongoDB n your machine or visit : https://www.tutorialspoint.com/mongodb_terminal_online.php

- Global commands : ***help, exit***, etc .
- Commands execute against the current database are executed against the db object,

  for example :

  - db.help() : returns a list of commands that you can use against db object

Create Database

- To create a wonderland database :

  use wonderland

  (creates the database and switches to it)

- To get the collections in the current database :

  db.getCollectionNames()

Insert Data

- To insert a document into the collection:

```
db.unicorns.insert(
        {
                name : 'Aurora',
                gender : 'f',
                weight : 450
        })
```

SLIIT
Discover Your Future

List Documents in a Collection

- Try out :

  db.unicorns.find()

- One more field is added : _id

  - Every document must have a unique _id field

  - Can generate your own or have MongoDB generate automatically for you

SLIIT
Discover Your Future

Query Selector

- First use:

  db.unicorns.remove({})

  to delete documents in the unicorns collection

- Use :

  { field : value }

  to select documents that match the condition

- If matching multiple conditions is desired:

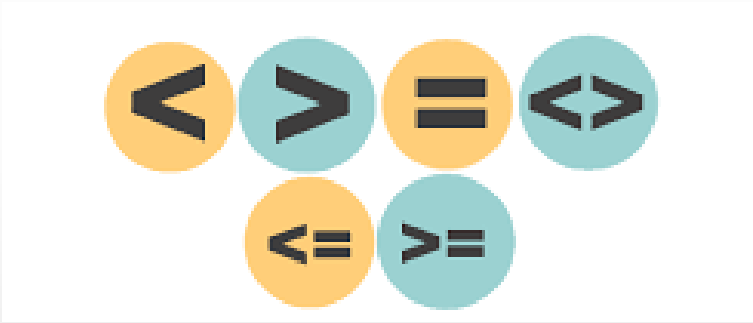
  { field1 : value1, field2 : value2 … }

  this implies the **AND** statement

SLIIT
Discover Your Future

Comparison operators in MongoDB

- $lt less than
- $lte less than or equal to
- $gt greater than
- $gte greater than or equal to
- $ne not equal to



SLIIT
Discover Your Future

Questions

SLIIT
Discover Your Future

Questions

1) Find the male unicorns weigh more than 700 pounds.

2) Find the unicorns that have no vampire field.

3) Find the unicorns that like apples or oranges.

4) Find the female unicorns that either love apples or weigh less than 500 pounds.

CRUD : Update

- Intuitively, updating unicorn Roooooodles' weight to 590 can be :

```
db.unicorns.update(
        { name : "Roooooodles" },
        { weight : 590 })
```

What is the result ?

CRUD : Update

- Try the following command :

db.unicorns.find({ weight : 590 })

Which gives a result similar to,

```
{
  _id: ObjectId('68ef3e7b881b15e8a2ce5f4f'),
  name: 'Roooooodles',
  dob: ISODate('1979-08-18T18:44:00.000Z'),
  loves: [ 'apple' ],
  gender: 'm',
  vampires: 99
}
```

CRUD : Update

- To fix the problem, we should do :

```
db.unicorns.update({ name : "Roooooooodles" },
                   { $set : { weight : 590 }})
```

More Update operators

- **$inc** : increment a field by a certain positive or negative amount

- **$push** : add a value to the existing field

SLIIT
Discover Your Future

Questions

Questions

5) Decrease unicorn Pilot's number of vampires by 2.

6) Add "sugar" to the list of food unicorn Aurora loves to eat.

SLIIT
Discover Your Future

Projection

- Find() can take a second argument, which is the project list.
- Example :

db.unicorns.find({},{ name : 1, _id : 0 })

- The values following fields names are boolean :

1 means including the field

0 means excluding the field

- Note that except excluding _id, the list cannot have a mixture of exclusion and inclusion.

Upserts

- An upsert updates the document if found or inserts it if not.

- To enable upserting we pass a third parameter to update {upsert : true}

Upserts - Example

This will not do anything :

```
db.unicorns.update(
        {name : "Walala"},
        {$inc : { vampires : 1 }})
```

Instead do this :

```
db.unicorns.update(
        {name : "Walala"},
        {$inc : { vampires : 1 }},
        { upsert : true })
```

Multiple Updates

- By default, update will only update a single document. Passing the third parameter {multi : true} will enable the multiple update.

SLIIT
Discover Your Future

Questions

Questions

7) Give all of the unicorns vaccine (set vaccinated to be true)

8) Sort the unicorns based on weights decreasingly.

9) Sort the unicorns based on the names increasingly, then the number of vampires decreasingly.

10) Get the second and third heaviest unicorns.

11) Count the number of unicorns who have more than 50 vampires.

References

- Karl Seguin, *The Little MongoDB Book*, http://openmymind.net/mongodb.pdf
- Kristina Chodorow, *MongoDB:The Definite Guide*, O'Reilly
- MongoDB CRUD Operations, https://docs.mongodb.org/master/MongoDB-crud- guide-master.pdf

Thank you …