



UNIVERSITAT DE
BARCELONA

Master in Fundamental Principles of Data Science

Dr Rohit Kumar

Today's Objective

- Sample of How to query MongoDB
- Introduction to Cassandra



UNIVERSITAT DE
BARCELONA

MongoDB Query

Do at home

1) Run the following command and start mongoDB using docker

docker run -p 27017:27017 -v <an empty folder path to store data>:/data/db mongo

Make sure to replace ***<an empty folder path to store data>*** with an actual folder path in your disk.

2) Install MongoDB Compass

<https://docs.mongodb.com/compass/master/install/>

The mongo shell

- The mongo shell is an interactive JavaScript interface to MongoDB. You can use the mongo shell to query and update data as well as perform administrative operations.
- The mongo shell is included as part of the MongoDB Server installation. MongoDB also provides the mongo shell as a standalone package.
- You can run mongo shell without any command-line options to connect to a MongoDB instance running on your localhost with default port 27017:

```
mongo
```

The mongo shell

- To explicitly specify the port, include the `--port` command-line option.

```
mongo --port 28015
```

- You can use the command-line option `--host` `<host>:<port>`. For example, to connect to a MongoDB instance running on a remote host machine:

```
mongo --host mongodb0.example.com:28015
```

- You can use the `--username` `<user>` and `--password`, `--authenticationDatabase` `<db>` command-line options:

```
mongo --username alice --password --authenticationDatabase admin --host  
mongodb0.examples.com --port 28015
```

The mongo shell

- To display the database you are using, type `db`. The operation should return `test`, which is the default database.
- To switch databases, issue the `use <db>` helper, as in the following example:

```
use restaurants
```

Lets Query

- Download the restaurants data from git.
- Use MongoDB Compass for the following
 1. Create a database as ***ub***
 2. Create a collection as ***restaurants***
 3. Unzip and import the restaurants.json in the collection ***restaurants***.

Lets Query

Use Mongo Shell to run these queries

- Write a MongoDB query to display the fields `restaurant_id`, `name`, `borough` and `cuisine` for all the documents in the collection `restaurant`.
- Write a MongoDB query to display the fields `restaurant_id`, `name`, `borough` and `cuisine`, but exclude the field `_id` for all the documents in the collection `restaurant`.
- Write a MongoDB query to display all the restaurant which is in the borough `Bronx`.



Solution

```
db.restaurants.find({}, {"restaurant_id" :  
1, "name":1, "borough":1, "cuisine" :1});
```

```
db.restaurants.find({}, {"restaurant_id" :  
1, "name":1, "borough":1, "cuisine" :1, "_id":0});
```

```
db.restaurants.find({"borough": "Bronx"});
```

Lets Query

- Write a MongoDB query to display the first 5 restaurant which is in the borough Bronx.
- Write a MongoDB query to display the next 5 restaurants after skipping first 5 which are in the borough Bronx.
- Write a MongoDB query to find the restaurants who achieved a score more than 90.
- Write a MongoDB query to find the restaurants that achieved a score, more than 80 but less than 100.

Solution

- `db.restaurants.find({"borough": "Bronx"}).limit(5);`
- `db.restaurants.find({"borough": "Bronx"}).skip(5).limit(5);`
- `db.restaurants.find({grades : { $elemMatch: {"score": {$gt : 90}}}});`
- `db.restaurants.find({grades : { $elemMatch: {"score": {$gt : 80 , $lt : 100}}}});`



Lets Query

Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168.

```
db.restaurants.find(  
  {$and:  
    [  
      {"cuisine" : {$ne : "American "}},  
      {"grades.score" : {$gt : 70}},  
      {"address.coord" : {$lt : -65.754168}}  
    ]  
  }  
);
```

```
db.restaurants.find(  
  {  
    "cuisine" : {$ne : "American "},  
    "grades.score" : {$gt: 70},  
    "address.coord" : {$lt : -65.754168}  
  }  
);
```

Lets Query

- Write a MongoDB query to find the restaurants which contain 'Wil' as first three letters for its name.
- Write a MongoDB query to find the restaurants which contain 'ces' as last three letters for its name.
- Write a MongoDB query to find the restaurants which contain 'Reg' as three letters somewhere in its name.
- Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish.

Solution

- `db.restaurants.find({name: /^Wil/});`
- `db.restaurants.find({name: /ces$/});`
- `db.restaurants.find({"name": /. *Reg. */});`
- `db.restaurants.find(
 {
 "borough": "Bronx" ,
 $or : [
 { "cuisine" : "American " },
 { "cuisine" : "Chinese" }
]
 }
);`



UNIVERSITAT DE
BARCELONA

Apache Cassandra



Who is using it



Cassandra History

- Cassandra was first developed at Facebook for inbox search.
- Facebook open sourced it in July 2008.
- Apache incubator accepted Cassandra in March 2009.
- Cassandra is a top level project of [Apache](#) since February 2010.
- The latest version of Apache Cassandra is 3.2.1.

Apache Cassandra Features

- **Masterless Architecture:** Data can be written and read on any node.
- **Linear Scale Performance:** As more nodes are added, the performance of Cassandra increases.
- **No Single point of failure:** Cassandra replicates data on different nodes that ensures no single point of failure.
- **Fault Detection and Recovery:** Failed nodes can easily be restored and recovered.
- **Flexible and Dynamic Data Model:** Supports datatypes with Fast writes and reads. It is a column-oriented database.

Apache Cassandra Features

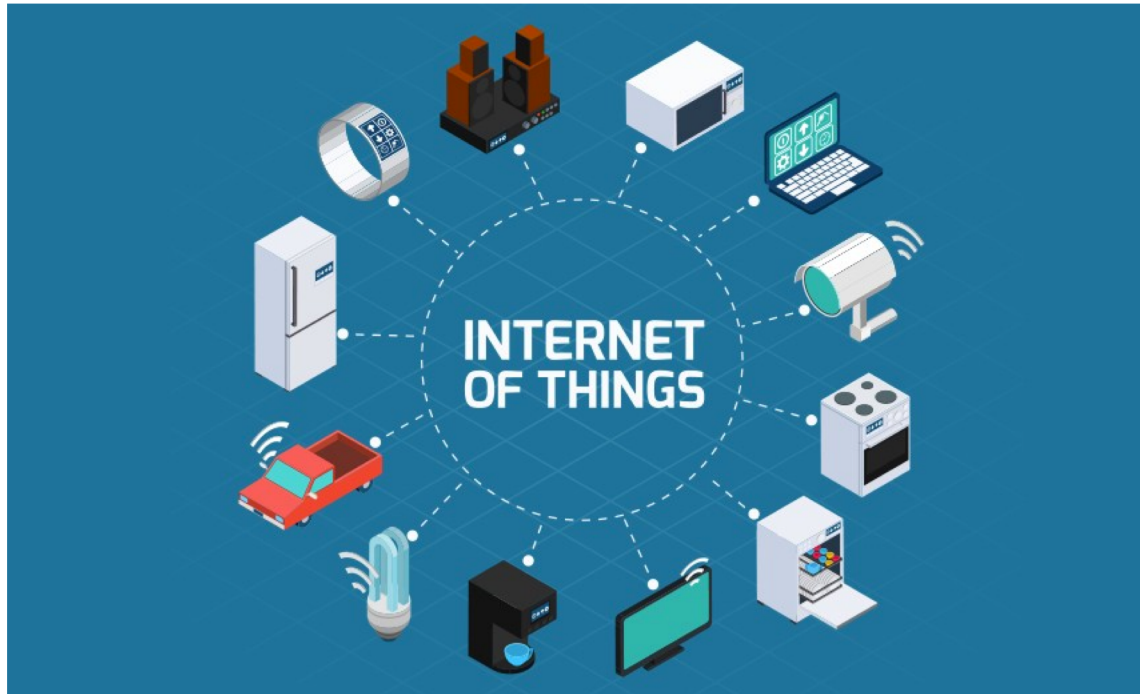
- **Data Protection:** Data is protected with commit log design and build in security like backup and restore mechanisms.
- **Multi Data Center Replication:** Cassandra provides feature to replicate data across multiple data center.
- **Data Compression:** Cassandra can compress up to 80% data without any overhead.
- **Cassandra Query language:** Cassandra provides query language that is similar like SQL language. It makes very easy for relational database developers moving from relational database to Cassandra.

Cassandra Use Cases/Application



Cassandra is a great database for the companies that provides Mobile phones and messaging services. These companies have a huge amount of data, so Cassandra is best for them.

Cassandra Use Cases/Application



Cassandra is a great database for the applications where data is coming at very high speed from different devices or sensors.

Run it now

1) Visit the page

<https://github.com/datastax/docker-images>

2) Follow the steps to “Create a DSE database container”.
This will create a server for you.

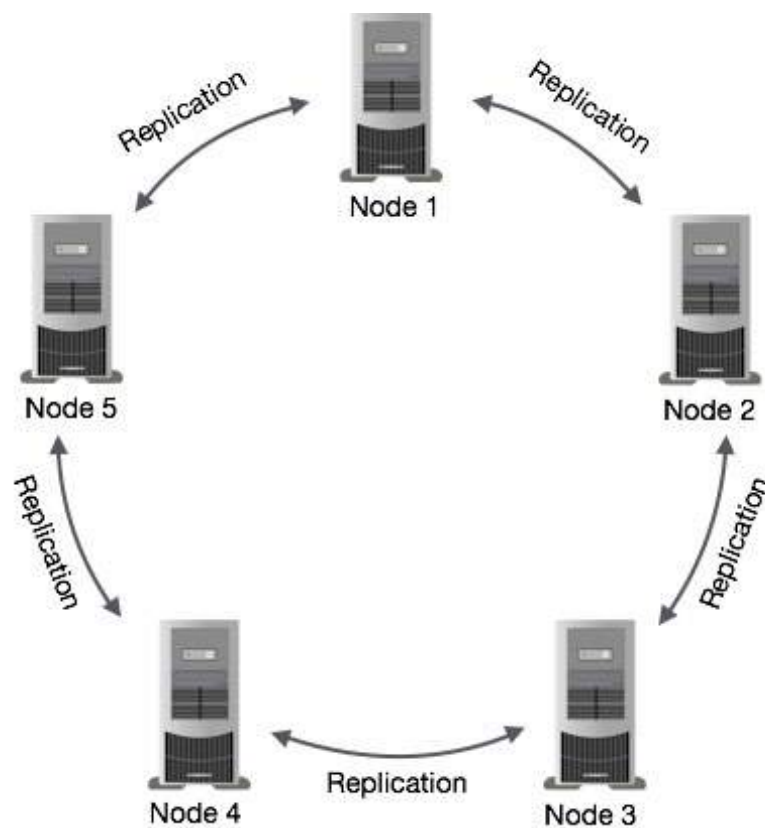
3) Follow the steps at “Creating a Studio Container”

Cassandra Architecture

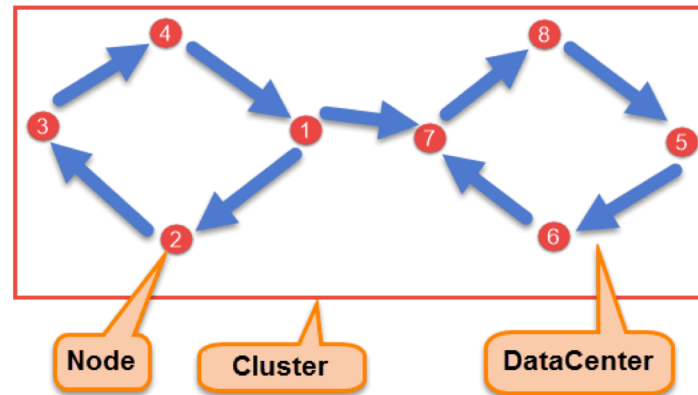
The design goal of Cassandra is to handle big data workloads across multiple nodes without any single point of failure. Cassandra has peer-to-peer distributed system across its nodes, and data is distributed among all the nodes in a cluster.

- All the nodes in a cluster play the same role. Each node is independent and at the same time interconnected to other nodes.
- Each node in a cluster can accept read and write requests, regardless of where the data is actually located in the cluster.
- When a node goes down, read/write requests can be served from other nodes in the network.

Cassandra Architecture



Cassandra Architecture



- **Node:** Node is the place where data is stored. It is the basic component of Cassandra.
- **Data Center:** A collection of nodes are called data center. Many nodes are categorized as a data center.
- **Cluster:** The cluster is the collection of many data centers.

Cassandra Architecture

- **Commit Log:** Every write operation is written to Commit Log. Commit log is used for crash recovery.
- **Mem-table:** After data written in Commit log, data is written in Mem-table. Data is written in Mem-table temporarily.
- **SSTable:** When Mem-table reaches a certain threshold, data is flushed to an SSTable disk file.

Data Replication

Cassandra places replicas of data on different nodes based on these two factors.

- Where to place next replica is determined by the **Replication Strategy**.
- While the total number of replicas placed on different nodes is determined by the **Replication Factor**.

For ensuring there is no single point of failure, **replication factor must be three**.

Data Replication

SimpleStrategy: SimpleStrategy is used when you have just one data center. SimpleStrategy places the first replica on the node selected by the partitioner. After that, remaining replicas are placed in clockwise direction in the Node ring.


NetworkTopologyStrategy: NetworkTopologyStrategy is used when you have more than two data centers. In NetworkTopologyStrategy, replicas are set for each data center separately. NetworkTopologyStrategy places replicas in the clockwise direction in the ring until reaches the first node in another rack.

RDBMS vs Cassandra

- **No joins:** You cannot perform joins in Cassandra.
- **No referential integrity:** In a relational database, you could specify foreign keys in a table to reference the primary key of a record in another table. But Cassandra does not enforce this.
- **Denormalization**
- **Query-first design**
- **Designing for optimal storage:** A key goal that you will see as you begin creating data models in Cassandra is to minimize the number of partitions that must be searched in order to satisfy a given query. Because the partition is a unit of storage that does not get divided across nodes, a query that searches a single partition will typically yield the best performance.

Cassandra hands on

- Go to <http://localhost:9091/connections>
- Update the default localhost by using host my-dse



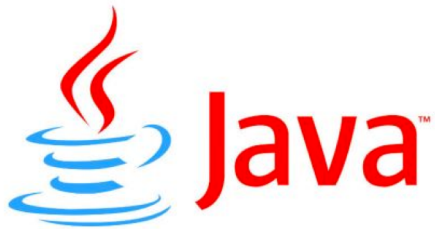
EDIT CONNECTION

Name *	Username
<input type="text" value="default localhost"/>	<input type="text"/>
Host/IP (comma delimited) *	Password
<input type="text" value="my-dse"/>	<input type="password"/>
Port *	
<input type="text" value="9042"/>	
<input type="checkbox"/> Use SSL	
<input type="button" value="Save"/>	<input type="button" value="Cancel"/>
<input type="button" value="Test"/>	

Hands On

- Open the notebook “Working with CQL v6.0.0”

Connectors and Drivers



Pymongo

```
import pymongo
```

```
myclient =  
pymongo.MongoClient("mongodb://localhost:27017/")
```

```
mydb = myclient["bts"]
```

```
mycol = mydb["customers"]
```