PRACTICAL-6

AIM:

To implement basic CRUD operations (create, read, update, delete) in MongoDB and Cassandra.

THEORY:

MongoDB:

• MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License (SSPL).

• Main Features:

o Ad-hoc queries

▪ MongoDB supports field, range query, and regular-expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions. Queries can also be configured to return a random sample of results of a given size.

o Indexing

▪ Fields in a MongoDB document can be indexed with primary and secondary indices or index.

o Replication

▪ MongoDB provides high availability with replica sets. A replica set consists of two or more copies of the data. Each replica-set member may act in the role of primary or secondary replica at any time. All writes and reads are done on the primary replica by default. Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically conducts an election process to determine which secondary should become the primary. Secondaries can optionally serve read operations, but that data is only eventually consistent by default.

▪ If the replicated MongoDB deployment only has a single secondary member, a separate daemon called an arbiter must be added to the set. It has a single responsibility, which is to resolve the election of the new primary. As a consequence, an idealized distributed MongoDB deployment requires at least three separate servers, even in the case of just one primary and one secondary.

o Load balancing

▪ MongoDB scales horizontally using sharding. The user chooses a shard key, which determines how the data in a collection will be distributed. The data is split into ranges (based on the shard key) and distributed across multiple shards. (A shard is a master with one or more replicas.). Alternatively, the shard key can be hashed to map to a shard – enabling an even data distribution.

▪ MongoDB can run over multiple servers, balancing the load or duplicating data to keep the system up and running in case of hardware failure.

o File storage

▪ MongoDB can be used as a file system, called GridFS, with load balancing and data replication features over multiple machines for storing files.

▪ This function, called grid file system, is included with MongoDB drivers. MongoDB exposes functions for file manipulation and content to developers. GridFS can be accessed using mongofiles utility or plugins for Nginx and lighttpd. GridFS divides a file into parts, or chunks, and stores each of those chunks as a separate document.

o Aggregation

▪ MongoDB provides three ways to perform aggregation: the aggregation pipeline, the map-reduce function, and single-purpose aggregation methods.

▪ Map-reduce can be used for batch processing of data and aggregation operations. But according to MongoDB's documentation, the Aggregation Pipeline provides better performance for most aggregation operations.

▪ The aggregation framework enables users to obtain the kind of results for which the SQL GROUP BY clause is used. Aggregation operators can be strung together to form a pipeline – analogous to Unix pipes. The aggregation framework includes the $lookup operator which can join documents from multiple collections, as well as statistical operators such as standard deviation.

o Server-side JavaScript execution

▪ JavaScript can be used in queries, aggregation functions (such as MapReduce), and sent directly to the database to be executed.

o Capped collections

▪ MongoDB supports fixed-size collections called capped collections. This type of collection maintains insertion order and, once the specified size has been reached, behaves like a circular queue.

o Transactions

▪ MongoDB claims to support multi-document ACID transactions since the 4.0 release in June 2018. This claim was found to not be true as MongoDB violates snapshot isolation.

Cassandra:

• Apache Cassandra is a free and open-source, distributed, wide-column store, NoSQL database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. Cassandra offers support for clusters spanning multiple datacenters, with asynchronous masterless replication allowing low latency operations for all clients. Cassandra was designed to implement a combination of Amazon's Dynamo distributed storage and replication techniques combined with Google's Bigtable data and storage engine model.

Main features:

o Distributed

▪ Every node in the cluster has the same role. There is no single point of failure. Data is distributed across the cluster (so each node contains different data), but there is no master as every node can service any request. o Supports replication and multi data center replication

▪ Replication strategies are configurable. Cassandra is designed as a distributed system, for deployment of large numbers of nodes across multiple data centers. Key features of Cassandra’s distributed architecture are specifically tailored for multiple-data center deployment, for redundancy, for failover and disaster recovery.

o Scalability

▪ Designed to have read and write throughput both increase linearly as new machines are added, with the aim of no downtime or interruption to applications. o Fault-tolerant

▪ Data is automatically replicated to multiple nodes for fault-tolerance. Replication across multiple data centers is supported. Failed nodes can be replaced with no downtime.

o Tunable consistency

▪ Cassandra is typically classified as an AP system, meaning that availability and partition tolerance are generally considered to be more important than consistency in Cassandra, Writes and reads offer a tunable level of consistency, all the way from "writes never fail" to "block for all replicas to be readable", with the quorum level in the middle.

o MapReduce support

▪ Cassandra has Hadoop integration, with MapReduce support. There is support also for Apache Pig and Apache Hive.

o Query language

▪ Cassandra introduced the Cassandra Query Language (CQL). CQL is a simple interface for accessing Cassandra, as an alternative to the traditional Structured Query Language (SQL).

o Eventual consistency

▪ Cassandra manages eventual consistency of reads, upserts and deletes through Tombstones

IMPLEMENTATION:

We can download MongoDB and Cassandra with the help of following tutorials.
https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/
https://phoenixnap.com/kb/install-cassandra-on-windows

MongoDB:

- We can start the MongoDB service using “mongo” command

- We can check available databases using “show dbs” command.

- We can create database using “use” keyword easily.

```
> use shopping
switched to db shopping
```

- We can insert the data in database using “insertOne” command.

```
> db.products.insertOne({name:'product1', price:40, color:'pink'});
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6131e482ae9c55a33bbd67be")
}
```

```
> db.products.insertOne({name:'product2', price:60, color:'blue'});
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6131e496ae9c55a33bbd67bf")
}
```

- We can check all the records of database using “find” and “pretty()” to format the output.

```
> db.products.find();
{ "_id" : ObjectId("6131e482ae9c55a33bbd67be"), "name" : "product1", "price" : 40, "color" : "pink" }
{ "_id" : ObjectId("6131e496ae9c55a33bbd67bf"), "name" : "product2", "price" : 60, "color" : "blue" }
> db.products.insertMany([{name:'product3', price:20, color:'green'}, {name:'product4', price: 15, color:'yellow'}]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6131e4e1ae9c55a33bbd67c0"),
                ObjectId("6131e4e1ae9c55a33bbd67c1")
        ]
}
> db.products.find().pretty();
{
        "_id" : ObjectId("6131e482ae9c55a33bbd67be"),
        "name" : "product1",
        "price" : 40,
        "color" : "pink"
}
{
        "_id" : ObjectId("6131e496ae9c55a33bbd67bf"),
        "name" : "product2",
        "price" : 60,
        "color" : "blue"
}
{
        "_id" : ObjectId("6131e4e1ae9c55a33bbd67c0"),
        "name" : "product3",
        "price" : 20,
        "color" : "green"
}
{
        "_id" : ObjectId("6131e4e1ae9c55a33bbd67c1"),
        "name" : "product4",
        "price" : 15,
        "color" : "yellow"
}
```

- We can also update our record using “updateOne”.

```
> db.products.updateOne({name:'product2'}, {$set: {price:50, qty:10}});
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
```

Cassandra:

- We can create and use keyspace in Cassandra using following command.

```
CREATE KEYSPACE keysp WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '3'}  AND durable_writes = true;
```

```
cqlsh:keysp> INSERT INTO emp (emp_id, emp_name, emp_city, emp_sal, emp_phone) VALUES (101, 'Robin', 'Hyderabad', 50000, 123456789);
cqlsh:keysp> INSERT INTO emp (emp_id, emp_name, emp_city, emp_sal, emp_phone) VALUES (102, 'Timmy', 'Delhi', 53000, 256478269);
cqlsh:keysp> INSERT INTO emp (emp_id, emp_name, emp_city, emp_sal, emp_phone) VALUES (103, 'Heli', 'Mumbai', 49000, 793067431);
cqlsh:keysp> SELECT * FROM emp;

 emp_id | emp_city  | emp_name | emp_phone | emp_sal
--------+-----------+----------+-----------+---------
    102 |     Delhi |    Timmy | 256478269 |   53000
    101 | Hyderabad |    Robin | 123456789 |   50000
    103 |    Mumbai |     Heli | 793067431 |   49000

(3 rows)
```

- We can also update and delete the record using corresponding queries.

```
cqlsh:keysp> UPDATE emp SET emp_sal=48000 WHERE emp_id=103;
cqlsh:keysp> SELECT * FROM emp;

 emp_id | emp_city  | emp_name | emp_phone | emp_sal
--------+-----------+----------+-----------+---------
    102 |     Delhi |    Timmy | 256478269 |   53000
    101 | Hyderabad |    Robin | 123456789 |   50000
    103 |    Mumbai |     Heli | 793067431 |   48000

(3 rows)
cqlsh:keysp> DELETE FROM emp WHERE emp_id=102;
cqlsh:keysp> SELECT * FROM emp;

 emp_id | emp_city  | emp_name | emp_phone | emp_sal
--------+-----------+----------+-----------+---------
    101 | Hyderabad |    Robin | 123456789 |   50000
    103 |    Mumbai |     Heli | 793067431 |   48000

(2 rows)
```

CONCLUSION:

In this practical, we learnt about mongoDB and Cassandra and performed basic CRUD operations in both the databases