ADVANCED DATABASE MANAGEMENT SYSTEMS LAB

ASSIGNMENT 1

SUBMITTED BY :

GROUP 4

CHRISTY RAJ (MCA216)

FATHIMA NOVRIN (MCA217)

GANGA KRISHNAN.G (MCA218)

HARI KRISHNAN S.R (MCA219)

HELNA MANOHARAN (MCA220)

TOPIC: Design database schemas and implement min 10 queries using Cassandra column based databases

Aim
To Design database schemas and implement queries using Cassandra databases

Objective(s)	
1	Study of NOSQL Cassandra.
2	Study the procedure to execute a query using Apache Cassandra.
3	Execute min 10 queries using Cassandra column based database.

1.STUDY OF NOSQL CASSANDRA

Cassandra

Apache Cassandra is an open source distributed database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure.

Features

1) **Scalability:**
   - Read and write throughput both increase linearly as new machines are added, with no downtime or interruption to applications.
2) **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.
3) **MapReduce support:**
   - Cassandra has Hadoop integration, with MapReduce support.
4) **Query language:**
   - Cassandra introduces CQL (Cassandra Query Language), a SQL-like alternative to the traditional RPC interface.
   - Keyspace: Keyspace is the outermost container for data.It is similar to the schema in a relational database.

Basic attributes of Keyspace are:

- Replication Factor: It is the number of machines in the cluster that will receive copies of the same data
- Replica Placement Strategy: It is a strategy to place replicas in the ring
- Simple Strategy,
- Old Network Topology Strategy
- Network Topology Strategy

Column Families: Column family is a NoSQL object that contains columns of related data. It is a tuple (pair) that consists of a key-value pair, where the key is mapped to a value that is a set of columns. It is similar to a table in a relational database and each key-value pair being a row.

Each column is a tuple (triplet) consisting of

- Column name
- Value
- Timestamp

2.STUDY THE PROCEDURE TO EXECUTE A QUERY USING APACHE CASSANDRA

Introduction

DataStax Community Edition must be installed on system before installing Cassandra. Verify the Cassandra installation using the following command:

- $ Cassandra version

- If Cassandra is already installed on system, then you will get the following response:

- Connected to Test Cluster at 127.0.0.1:9042.
- [cqlsh 5.0.1 | Cassandra 2.2.4 | CQL spec 3.3.1 | Native protocol v4] Use HELP for help.
- WARNING: pyreadline dependency missing. Install to enable tab completion. From source with checksum 79e53ce7994d1628b240f09af91e1af4

Creating KEYSPACE :

Cassandra is technology that can define databases and tables to analyze structured data. The theme for structured data analysis is to store the data in a tabular manner, and pass queries to analyze it.

Create KEYSPACE Statement

- Create KEYSPACE is a statement used to create a KEYSPACE in Cassandra. A KEYSPACE in Cassandra is aKEYSPACE or a collection of tables. The syntax for this statement is as follows:
  cqlsh>CREATE KEYSPACE ABC userdb replication={
  'class':'SimpleStrategy','replication_factor':'1'};

- Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists. We can use SCHEMA in place of DATABASE in this command.

- The following query is executed to create a database named userdb:

cqlsh> userdb;
Or
cqlsh> CREATE SCHEMA userdb;

The following query is used to verify a databases list:

cqlsh>:userdb> show tables; Improper show command. default
userdb

For creating Table:
Create Table
CREATE TABLE test_table (
id int, address text, name text,
PRIMARY KEY ((id))
);

CURD using cql Updating Table:

Update Table
insert into test_table (id, name, address) values (4, 'somnath', 'Sus');

CURD using cql Delete Table

Deleting rows from Table

delete from test_table where id =1;

The following queries are used to drop a database. Let us assume that the database name is userdb.
cqlsh:userdb> delete from Tablename where condition;

For describing tables

cqlsh:userdb> describe tables; show all table names
cqlsh:userdb>

For Help of any Topic
Cqshl> Help;

Display topics

Cqshl> Help topic name;

Help open in Browser.

3.Execute minimum 10 queries using Cassandra column-based database

QUERIES

1.CREATING A TABLE

PROCEDURE:

Step 1: Here we consider the database “Student”

Step 2: Creating and updating a keyspace ,here keyspace is “school”

Step 3: Creating a table “details”

Step 4: Describe table “details”

```
Connected to cndb at cassandra.ingress:9042.
[cqlsh 6.8.0 | DSE DB 4.0.0.6815 | CQL spec 3.4.5 | Native protocol v4]
Use HELP for help.
token@cqlsh> USE school;
token@cqlsh:school> CREATE TABLE school.details(rollno int PRIMARY KEY,name text,mark int,gender text);
token@cqlsh:school> DESCRIBE school.details;

CREATE TABLE school.details (
    rollno int PRIMARY KEY,
    gender text,
    mark int,
    name text
) WITH additional_write_policy = '99PERCENTILE'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.UnifiedCompactionStrategy'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99PERCENTILE';

token@cqlsh:school>
```

2.INSERTING DATA INTO A TABLE

Procedure:

- INSERT INTO details (rollno,name,mark,gender) VALUES (1,'Hari',100,'Male');
- INSERT INTO details (rollno,name,mark,gender) VALUES (2,'Ganga',10,'Female');
- INSERT INTO details (rollno,name,mark,gender) VALUES (3,'Fathima',50,'Female');

OUTPUT

```
token@cqlsh:school> INSERT INTO details (rollno,name,mark,gender) values(1,'Hari',100,'Male');
token@cqlsh:school> INSERT INTO details (rollno,name,mark,gender) values(2,'Ganga',10,'Female');
token@cqlsh:school> INSERT INTO details (rollno,name,mark,gender) values(3,'Fathima',50,'Female');
```

3.RETRIEVAL OF ALL DATA

Procedure:

- SELECT * FROM school.details;

OUTPUT:

```
token@cqlsh:school> SELECT * FROM school.details;

 rollno | gender | mark | name
--------+--------+------+---------
      1 |   Male |  100 |    Hari
      2 | Female |   10 |   Ganga
      3 | Female |   50 | Fathima

(3 rows)
token@cqlsh:school>
```

4.RETRIEVE THE DETAILS OF STAFFS USING CONDITIONS

Procedure:

- SELECT * FROM school.details WHERE gender=’Female’ ALLOW FILTERING;

OUTPUT

```
token@cqlsh:school> SELECT * FROM school.details WHERE gender='Female' ALLOW FILTERING;

 rollno | gender | mark | name
--------+--------+------+---------
      2 | Female |   10 |   Ganga
      3 | Female |   50 | Fathima

(2 rows)
```

6.TABLE UPDATION

Procedure :

- UPDATE school.details SET name=’Christy’ WHERE rollno=2;

OUTPUT:

```
token@cqlsh:school> UPDATE school.details SET name='Christy' WHERE rollno=2;
token@cqlsh:school> SELECT * FROM school.details;

 rollno | gender | mark | name
--------+--------+------+---------
      1 |   Male |  100 |    Hari
      2 | Female |   10 | Christy
      3 | Female |   50 | Fathima

(3 rows)
token@cqlsh:school>
```

6.ALTER TABLE command

- Adding new column'grade' into the table 'details'

Procedure:

- ALTER TABLE details ADD grade text;
- SELECT * FROM school.details;

OUTPUT

```
token@cqlsh:school> ALTER TABLE school.details ADD Grade text;
token@cqlsh:school> SELECT * FROM school.details;

 rollno | gender | grade | mark | name
--------+--------+-------+------+---------
      1 |   Male |  null |  100 |    Hari
      2 | Female |  null |   10 | Christy
      3 | Female |  null |   50 | Fathima

(3 rows)
token@cqlsh:school>
```

7.DROPPING A COLOUMN

Procedure:

ALTER TABLE school.details DROP gender;

OUTPUT:

```
token@cqlsh:school> ALTER TABLE school.details DROP gender;
token@cqlsh:school> SELECT * FROM school.details;

 rollno | grade | mark | name
--------+-------+------+---------
      1 |  null |  100 |    Hari
      2 |  null |   10 | Christy
      3 |  null |   50 | Fathima

(3 rows)
token@cqlsh:school>
```

8.RETRIEVING TIMESTAMPS

Timestamps

Each time you write data into Cassandra, a timestamp is generated for each column value that is updated. Internally, Cassandra uses these timestamps for resolving any conflicting changes that are made to the same value.

Procedure:

- SELECT rollno,name , writetime(name) FROM details;

OUTPUT

```
token@cqlsh:school> SELECT rollno,name,writetime(name) FROM details;

 rollno | name    | writetime(name)
--------+---------+------------------
      1 |    Hari | 1630592449941063
      2 | Christy | 1630592978595248
      3 | Fathima | 1630592643935201

(3 rows)
```

9.DELETING FROM THE TABLE

Procedure:

- DELETE FROM school.details WHERE rollno=3;

OUTPUT

```
token@cqlsh:school> DELETE FROM school.details WHERE rollno=3;
token@cqlsh:school> SELECT * FROM school.details;

 rollno | grade | mark | name
--------+-------+------+---------
      1 |  null |  100 |    Hari
      2 |  null |   10 | Christy

(2 rows)
token@cqlsh:school>
```

10.QUERY TO ADD ANOTHER EMAIL ADDRESS WITHOUT REPLACING THE WHOLE SET BY USING CONCATENATION

Procedure:

- ALTER TABLE school.details ADD Emails set<text>;
- UPDATE details SET emails={'novrin@fathima.com'} WHERE Rollno=2;

OUTPUT:

```
token@cqlsh:school> ALTER TABLE school.details ADD Emails set<text>;
token@cqlsh:school> UPDATE details SET emails= {'novrin@fathima.com'} WHERE rollno=2;
token@cqlsh:school> SELECT Emails from details WHERE rollno=2;

 emails
-----------------------
 {'novrin@fathima.com'}

(1 rows)
token@cqlsh:school>
```

Procedure:

- UPDATE stud_details SET Emails+{‘jackjohn@gmail.com} WHERE Rollno=104;

OUTPUT: