NOSQL ASSIGNMENT

***SUBMITTED BY*,**

RIYA VINCENT 2031

ROBIN MONACHAN 2032

SAKTHI PRIYA M 2033

SHAD IBRAHIM NP 2034

SHALIMA SHAJAHAN 2035

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

- **Scalability:**
  - Read and write throughput both increase linearly as new machines are added, with no downtime or interruption to applications.
- **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.
- **MapReduce support:**
  - Cassandra has Hadoop integration, with MapReduce support.
- **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 the 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

INITIAL PROCEDURE:

Step 1: Here we consider the database “TKM”
Step 2: Creating and updating a keyspace ,here keyspace is “students”
Step 3: Creating a table “detail”
Step 4: Describe table “detail”

```
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 students;
token@cqlsh:students> CREATE TABLE students.detail(id decimal PRIMARY KEY,name text,place text);
token@cqlsh:students> DESCRIBE students.detail;

CREATE TABLE students.detail (
    id decimal PRIMARY KEY,
    name text,
    place 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';
```

QUERIES

1. INSERTING DATA INTO A TABLE

Procedure:

- INSERT INTO detail (id, name, place) VALUES (231, 'riya', ‘kannur');
- INSERT INTO detail (id, name, place) VALUES (232, 'robin', ‘kollam');
- INSERT INTO detail (id, name, place) VALUES (233, 'sakthi', ‘Thrissur');
- INSERT INTO detail (id, name, place) VALUES (234, 'shad', ‘kozhikode');
- INSERT INTO detail (id, name, place) VALUES (235, 'shalima', ‘Ernakulam');

OUTPUT:

```
token@cqlsh:students> INSERT INTO detail(id, name, place) VALUES(231, 'riya', 'kannur');
token@cqlsh:students> INSERT INTO detail(id, name, place) VALUES(232, 'robin', 'kollam');
token@cqlsh:students> INSERT INTO detail(id, name, place) VALUES(233, 'sakthi', 'Thrissur');
token@cqlsh:students> INSERT INTO detail(id, name, place) VALUES(234, 'shad', 'kozhikode');
token@cqlsh:students> INSERT INTO detail(id, name, place) VALUES(235, 'shalima', 'Ernakulam');
```

2.RETRIEVAL OF ALL DATA

Procedure:

- SELECT * FROM students.detail;

OUTPUT:

```
token@cqlsh:students> INSERT INTO detail(id, name, place) VALUES(231, 'riya', 'kannur');
token@cqlsh:students> INSERT INTO detail(id, name, place) VALUES(232, 'robin', 'kollam');
token@cqlsh:students> INSERT INTO detail(id, name, place) VALUES(233, 'sakthi', 'Thrissur');
token@cqlsh:students> INSERT INTO detail(id, name, place) VALUES(234, 'shad', 'kozhikode');
token@cqlsh:students> INSERT INTO detail(id, name, place) VALUES(235, 'shalima', 'Ernakulam');
token@cqlsh:students> SELECT * FROM students.detail;

 id  | name    | place
-----+---------+-----------
 232 |   robin |    kollam
 234 |    shad | kozhikode
 235 | shalima | Ernakulam
 233 |  sakthi |  Thrissur
 231 |    riya |    kannur

(5 rows)
```

3.RETRIEVE THE DETAILS OF STAFFS USING CONDITIONS

Procedure:

- SELECT * FROM students.detail WHERE id=234 ALLOW FILTERING;
- SELECT * FROM students.detail WHERE name=’robin’ ALLOW FILTERING;

OUTPUT:

```
token@cqlsh:students> SELECT * FROM students.detail WHERE id = 234 ALLOW FILTERING;

 id  | name | place
-----+------+-----------
 234 | shad | kozhikode

(1 rows)
token@cqlsh:students> SELECT * FROM students.detail WHERE name = 'robin' ALLOW FILTERING;

 id  | name  | place
-----+-------+--------
 232 | robin | kollam

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

4.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 id,writetime(name) FROM detail;

OUTPUT:

```
token@cqlsh:students> SELECT id,writetime(name) FROM detail;

 id  | writetime(name)
-----+------------------
 232 | 1630509290457743
 234 | 1630509339607752
 235 | 1630509373907829
 233 | 1630509317217733
 231 | 1630509267982794

(5 rows)
```

5.RETRIEVE THE TTL VALUE

Procedure:

- SELECT id, TTL(name),place FROM detail WHERE id=231;

OUTPUT:

```
token@cqlsh:students> SELECT id,TTL(name),place FROM detail WHERE id=231;

 id  | ttl(name) | place
-----+-----------+--------
 231 |      null | kannur

(1 rows)
```

6. QUERY TO ADD THE IDENTIFIER USING A UUID

Procedure:

- ALTER TABLE detail ADD roll uuid;
- DESCRIBE detail;

OUTPUT:

```
token@cqlsh:students> ALTER TABLE detail ADD roll uuid;
token@cqlsh:students> DESCRIBE detail;

CREATE TABLE students.detail (
    id decimal PRIMARY KEY,
    name text,
    place text,
    roll uuid
) 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';
```

7.QUERY TO INSERT AN ROLL USING UUID() FUNCTION AND THEN VIEW THE RESULTS

uuid:
A universally unique identifier (UUID) is a 128-bit value in which the bits conform to one of several types, of which the most commonly used are known as Type 1 and Type 4. The CQL uuid type is a Type 4 UUID, which is based entirely on random numbers. UUIDs are typically represented as dash-separated sequences of hex digits. For example:

1a6300ca-0572-4736-a393-c0b7229e193e

The uuid type is often used as a surrogate key, either by itself or in combination with other values.

Procedure:

- UPDATE detail SET roll = uuid() WHERE id=233;
- SELECT name, roll FROM detail WHERE id=233;

OUTPUT:

```
token@cqlsh:students> UPDATE detail SET roll=uuid() WHERE id=233;
token@cqlsh:students> SELECT name,roll FROM detail WHERE id=233;

 name   | roll
--------+--------------------------------------
 sakthi | 32c79ceb-b23d-4272-9380-bf656f8bf514

(1 rows)
```

8.QUERY TO MODIFY TABLE TO ADD A SET OF EMAIL ADDRESSES

set

The set data type stores a collection of elements. The elements are unordered, but cqlsh returns the elements in sorted order. For example, text values are returned in alphabetical order. One advantage of using sets is the ability to insert additional items without having to read the contents first.

Procedure:

- ALTER TABLE detail ADD email set<text>;
- UPDATE detail SET email = { 'shalima123@gmail.com' } WHERE id =235;
- SELECT * FROM detail WHERE id = 235;

OUTPUT:

```
token@cqlsh:students> ALTER TABLE detail ADD email set<text>;
token@cqlsh:students> UPDATE detail SET email = { 'shalima123@gmail.com' } WHERE id=235;
token@cqlsh:students> SELECT * FROM detail WHERE id=235;

 id  | email                      | name    | place     | roll
-----+----------------------------+---------+-----------+------
 235 | {'shalima123@gmail.com'} | shalima | Ernakulam | null

(1 rows)
```

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

Procedure:

- UPDATE detail SET email = email + { ‘shalima_shajahan@gmail.com' } WHERE id=235;
- SELECT id,name,email FROM detail WHERE id=235;

OUTPUT:

```
token@cqlsh:students> UPDATE detail SET email = email + { 'shalima_shajahan@gmail.com' } WHERE id=235;
token@cqlsh:students> SELECT id,name,email FROM detail WHERE id=235;

 id  | name    | email
-----+---------+-----------------------------------------------------------
 235 | shalima | {'shalima123@gmail.com', 'shalima_shajahan@gmail.com'}

(1 rows)
```

10.QUERY TO MODIFY OUR STAFFS TABLE TO ADD A LIST OF PHONE NUMBERS AND ADD A PHONE NUMBER FOR MARY AND CHECK THAT IT WAS ADDED SUCCESSFULL

List

The list data type contains an ordered list of elements. By default, the values are stored in order of insertion.

Procedure:

- ALTER TABLE detail ADD phn_no list<decimal>;
- UPDATE detail SET phn_no = [9836472821] WHERE id=234;
- SELECT * FROM detail WHERE id=234;

OUTPUT: