Course code : **CSE3009**
Course title : **No SQL Data Bases**
Module : **5**
Topic : **3**

Cassandra Query Language

Objectives

This session will give the knowledge about

- Cassandra Query Language

Cassandra Query Language

By default, Cassandra provides a prompt Cassandra query language shell (cqlsh) that allows users to communicate with it. Using this shell, you can execute Cassandra Query Language (CQL).

Using cqlsh, you can

- define a schema,
- insert data, and
- execute a query.

CQL Documented Shell Commands

- HELP − Displays help topics for all cqlsh commands.
- CAPTURE − Captures the output of a command and adds it to a file.
- CONSISTENCY − Shows the current consistency level, or sets a new consistency level.
- COPY − Copies data to and from Cassandra.
- DESCRIBE − Describes the current cluster of Cassandra and its objects.
- EXPAND − Expands the output of a query vertically.

CQL Documented Shell Commands

- EXIT − Using this command, you can terminate cqlsh.
- PAGING − Enables or disables query paging.
- SHOW − Displays the details of current cqlsh session such as Cassandra version, host, or data type assumptions.
- SOURCE − Executes a file that contains CQL statements.
- TRACING − Enables or disables request tracing.

CQL Data Types

Data Type	Constants	Description
ascii	strings	Represents ASCII character string
bigint	bigint	Represents 64-bit signed long
blob	blobs	Represents arbitrary bytes
Boolean	booleans	Represents true or false
counter	integers	Represents counter column

CQL Data Types

Data Type	Constants	Description
decimal	integers, floats	Represents variable-precision decimal
double	integers	Represents 64-bit IEEE-754 floating point
float	integers, floats	Represents 32-bit IEEE-754 floating point
inet	strings	Represents an IP address, IPv4 or IPv6
int	integers	Represents 32-bit signed int

CQL Data Types

Data Type	Constants	Description
text	strings	Represents UTF8 encoded string
timestamp	integers, strings	Represents a timestamp
timeuuid	uuids	Represents type 1 UUID
uuid	uuids	Represents type 1 or type 4
varchar	strings	Represents uTF8 encoded string
varint	integers	Represents arbitrary-precision integer

CQL Collection Types

Collection	Description
list	A list is a collection of one or more ordered elements.
map	A map is a collection of key-value pairs.
set	A set is a collection of one or more elements.

CQL Data Definition Commands

- **CREATE KEYSPACE** − Creates a KeySpace in Cassandra.
- **USE** − Connects to a created KeySpace.
- **ALTER KEYSPACE** − Changes the properties of a KeySpace.
- **DROP KEYSPACE** − Removes a KeySpace

- **CREATE TABLE** − Creates a table in a KeySpace.
- **ALTER TABLE** − Modifies the column properties of a table.
- **DROP TABLE** − Removes a table.
- **TRUNCATE** − Removes all the data from a table.
- **CREATE INDEX** − Defines a new index on a single column of a table.
- **DROP INDEX** − Deletes a named index.

CQL Data Definition Commands

CREATE KEYSPACE − Creates a KeySpace in Cassandra.

Syntax

- CREATE KEYSPACE <identifier> WITH <properties>
- CREATE KEYSPACE “KeySpace Name” WITH replication = {'class': ‘Strategy name’, 'replication_factor' : ‘No.Of replicas’} AND durable_writes = ‘Boolean value’; USE − Connects to a created KeySpace.

Example

```
CREATE KEYSPACE University WITH replication = {'class':'SimpleStrategy',
'replication_factor' : 3};
```

CQL Data Definition Commands

- **Strategy**: While declaring strategy name in Cassandra. There are two kinds of strategies declared in Cassandra Syntax.
  - **Simple Strategy**: Simple strategy is used when you have just one data center. In this strategy, the first replica is placed on the node selected by the partitioner. Remaining nodes are placed in the clockwise direction in the ring without considering rack or node location.
  - **Network Topology Strategy**: Network topology strategy is used when you have more than one data centers. In this strategy, you have to provide replication factor for each data center separately. Network topology strategy places replicas in nodes in the clockwise direction in the same data center. This strategy attempts to place replicas in different racks.
- **Replication Factor**: Replication factor is the number of replicas of data placed on different nodes. For no failure, 3 is good replication factor. More than two replication factor ensures no single point of failure.

CQL Data Definition Commands

- DESCRIBE keyspaces;
- SELECT * FROM system.schema_keyspaces;
- CREATE KEYSPACE cse WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'datacenter1' : 3 } AND DURABLE_WRITES = false;

Durable Writes

Durable Writes provides a means to instruct Cassandra whether to use "commitlog" for updates on the current KeySpace or not. This option is not mandatory. The default value for durable writes is TRUE.

- When Cassandra write:
  1. memtable (in memory)
  2. commit log (In persistence storage) before writing into, to safeguard the data, just incase of system restart.
  3. SSTABLE. (In persistence storage)

So, using "durable writes" it make sure the insert/update must write in commit log so the safeguard stay in place.

CQL Data Definition Commands

- USE University;
- ALTER KEYSPACE University WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'datacenter1' : 1 };
- ALTER KEYSPACE University WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'datacenter1' : 1 } AND DURABLE_WRITES = false;
- DROP KEYSPACE University;

CQL Data Definition Commands

- CREATE TABLE emp(emp_id int PRIMARY KEY, emp_name text,emp_city text, emp_sal int, emp_phone int);
- select * from emp;
- ALTER TABLE University.emp
  - ALTER emp_id TYPE int;
  - ADD emp_email text;
  - DROP emp_phone;
  - RENAME emp_city TO emp_address;
- DROP TABLE University.emp

CQL Data Definition Commands

- DESCRIBE COLUMNFAMILIES;
- TRUNCATE University.emp;
- INSERT INTO University.Emp(emp_id,emp_city,emp_name,emp_sal) VALUES(2,'ap','sample1', 2000);
- UPDATE University.Emp SET emp_name='Hayden' WHERE emp_id=1;
- UPDATE University.Emp SET emp_name='Hayden' WHERE emp_id=2;
- SELECT emp_name, emp_sal from University.emp;

CQL Data Definition Commands

- DELETE emp_sal FROM University.emp WHERE emp_id=3;
- DELETE FROM University.Emp WHERE emp_id=1;

- CREATE INDEX e_name ON University.emp(emp_name);
- DROP INDEX IF EXISTS University.e_name;
- INSERT INTO University.Emp(emp_id,emp_city,emp_name,emp_sal) VALUES(3,‘hyd','sample2', 2300) USING TTL 100;

BATCH

In Cassandra BATCH is used to execute multiple modification statements (insert, update, delete) simultaneously. It is very useful when you have to update some column as well as delete some of the existing.

BEGIN BATCH

<insert-stmt>/ <update-stmt>/ <delete-stmt>

APPLY BATCH

BATCH

Example

Assume there is a table in Cassandra called emp having the following data −

emp_id	emp_name	emp_city	emp_phone	emp_sal
1	ram	Hyderabad	9848022338	50000
2	robin	Delhi	9848022339	50000
3	rahman	Chennai	9848022330	45000

In this example, we will perform the following operations −

- Insert a new row with the following details (4, rajeev, pune, 9848022331, 30000).
- Update the salary of employee with row id 3 to 50000.
- Delete city of the employee with row id 2.

BATCH

BEGIN BATCH

... INSERT INTO emp (emp_id, emp_city, emp_name, emp_phone, emp_sal) values(4,'Pune','rajeev',9848022331, 30000);

... UPDATE emp SET emp_sal = 50000 WHERE emp_id =3;

... DELETE emp_city FROM emp WHERE emp_id = 2;

... APPLY BATCH;

CQL Collections

- CREATE TABLE University.student(stu_id INT, stu_name TEXT, stu_Email SET<TEXT>, PRIMARY KEY(stu_id));
- INSERT INTO University.student(stu_id,stu_name,stu_Email) VALUES(123,'test1',{'abc@gmail.com','xyz@hotmail.com'});
- INSERT INTO customer(cust_id ,Cust_name, Cust_address , Cust_mob, Cust_email , Bill_amount) VALUES (101, 'John',{'No':'123','Street': 'Roy street','City': 'Vijayawada','State': 'AP'}, ['9876534567','9878998787'], {'jon@gmail.com', 'john@gmail.com'}, 12000);

CQL User-defined Data Type

- CREATE TYPE university.card_details(num int,pin int,name text,cvv int);
- ALTER TYPE university.card_details ADD expire text;
- DESCRIBE TYPE card_details;
- DESCRIBE TYPES;
- DROP TYPE card_details;

CQL User-defined Data Type

- CREATE TYPE address (employee_id int, residence_address text, office_address text, city text);
- CREATE TABLE employee (employee_id int PRIMARY KEY, name text, address frozen<address>, salary text);

- INSERT INTO employee JSON '{"employee_id":1234, "name":"Akhil", "address":{"employee_id":1234,"residence_address":"65A Block","office_address":"75D Block","city":"Goa"}, "salary":"12000"}';

Summary

This session will give the knowledge about

- Cassandra Query Language