CASSANDRA - CREATE DATA

Creating Data using Cqlsh

You can insert data into the columns of a row in a table using the command **INSERT**. Given below is the syntax for creating data in a table.

```
INSERT INTO <tablename>
(<column1 name>, <column2 name>....)
VALUES (<value1>, <value2>....)
USING <option>
```

Example

Let us assume there is a table called **emp** with columns emp_id, emp_name, emp_city, emp_phone, emp_sal and you have to insert the following data into the **emp** table.

emp_id	emp_name	emp_city	emp_phone	emp_sal
1	ram	Hyderabad	9848022338	50000
2	robin	Hyderabad	9848022339	40000
3	rahman	Chennai	9848022330	45000

Use the commands given below to fill the table with required data.

```
cqlsh:tutorialspoint> INSERT INTO emp (emp_id, emp_name, emp_city,
   emp_phone, emp_sal) VALUES(1,'ram', 'Hyderabad', 9848022338, 50000);

cqlsh:tutorialspoint> INSERT INTO emp (emp_id, emp_name, emp_city,
   emp_phone, emp_sal) VALUES(2,'robin', 'Hyderabad', 9848022339, 40000);

cqlsh:tutorialspoint> INSERT INTO emp (emp_id, emp_name, emp_city,
   emp_phone, emp_sal) VALUES(3,'rahman', 'Chennai', 9848022330, 45000);
```

Verification

After inserting data, use SELECT statement to verify whether the data has been inserted or not. If you verify the emp table using SELECT statement, it will give you the following output.

```
cqlsh:tutorialspoint> SELECT * FROM emp;

 emp_id |  emp_city | emp_name |  emp_phone | emp_sal
--------+-----------+----------+------------+---------
      1 | Hyderabad |      ram | 9848022338 | 50000
      2 | Hyderabad |    robin | 9848022339 | 40000
      3 |   Chennai |   rahman | 9848022330 | 45000

(3 rows)
```

Here you can observe the table has populated with the data we inserted.

Creating Data using Java API

You can create data in a table using the execute method of Session class. Follow the steps given below to create data in a table using java API.

Step1: Create a Cluster Object

Create an instance of **Cluster.builder** class of **com.datastax.driver.core** package as shown below.

```
//Creating Cluster.Builder object
Cluster.Builder builder1 = Cluster.builder();
```

Add a contact point *IPaddressofthenode* using the **addContactPoint** method of **Cluster.Builder** object. This method returns **Cluster.Builder**.

```
//Adding contact point to the Cluster.Builder object
Cluster.Builder builder2 = build.addContactPoint("127.0.0.1");
```

Using the new builder object, create a cluster object. To do so, you have a method called **build** in the **Cluster.Builder** class. The following code shows how to create a cluster object.

```
//Building a cluster
Cluster cluster = builder.build();
```

You can build a cluster object using a single line of code as shown below.

```
Cluster cluster = Cluster.builder().addContactPoint("127.0.0.1").build();
```

Step 2: Create a Session Object

Create an instance of Session object using the connect method of Cluster class as shown below.

```
Session session = cluster.connect( );
```

This method creates a new session and initializes it. If you already have a keyspace, then you can set it to the existing one by passing the KeySpace name in string format to this method as shown below.

```
Session session = cluster.connect(“ Your keyspace name ” );
```

Here we are using the KeySpace called **tp**. Therefore, create the session object as shown below.

```
Session session = cluster.connect(“ tp” );
```

Step 3: Execute Query

You can execute CQL queries using the execute method of Session class. Pass the query either in string format or as a **Statement** class object to the execute method. Whatever you pass to this method in string format will be executed on the **cqlsh**.

In the following example, we are inserting data in a table called **emp**. You have to store the query in a string variable and pass it to the execute method as shown below.

```
String query1 = “INSERT INTO emp (emp_id, emp_name, emp_city, emp_phone, emp_sal)
   VALUES(1,'ram', 'Hyderabad', 9848022338, 50000);” ;

String query2 = “INSERT INTO emp (emp_id, emp_name, emp_city, emp_phone, emp_sal)
   VALUES(2,'robin', 'Hyderabad', 9848022339, 40000);” ;

String query3 = “INSERT INTO emp (emp_id, emp_name, emp_city, emp_phone, emp_sal)
   VALUES(3,'rahman', 'Chennai', 9848022330, 45000);” ;

session.execute(query1);
session.execute(query2);
session.execute(query3);
```

Given below is the complete program to insert data into a table in Cassandra using Java API.

```
import com.datastax.driver.core.Cluster;
import com.datastax.driver.core.Session;

public class Create_Data {

   public static void main(String args[]){

      //queries
      String query1 = "INSERT INTO emp (emp_id, emp_name, emp_city, emp_phone,
emp_sal)"

         + " VALUES(1,'ram', 'Hyderabad', 9848022338, 50000);" ;

      String query2 = "INSERT INTO emp (emp_id, emp_name, emp_city,
         emp_phone, emp_sal)"

         + " VALUES(2,'robin', 'Hyderabad', 9848022339, 40000);" ;

      String query3 = "INSERT INTO emp (emp_id, emp_name, emp_city, emp_phone,
emp_sal)"

         + " VALUES(3,'rahman', 'Chennai', 9848022330, 45000);" ;

      //Creating Cluster object
      Cluster cluster = Cluster.builder().addContactPoint("127.0.0.1").build();

      //Creating Session object
      Session session = cluster.connect("tp");

      //Executing the query
      session.execute(query1);

      session.execute(query2);

      session.execute(query3);

      System.out.println("Data created");
   }
}
```

Save the above program with the class name followed by .java, browse to the location where it is saved. Compile and execute the program as shown below.

```
$javac Create_Data.java
$java Create_Data
```

Under normal conditions, it should produce the following output:

```
Data created
```