CASSANDRA - CREATE TABLE

Creating a Table using Cqlsh

You can create a table using the command **CREATE TABLE**. Given below is the syntax for creating a table.

Syntax

```
CREATE (TABLE | COLUMNFAMILY) <tablename>
('<column-definition>' , '<column-definition>')
(WITH <option> AND <option>)
```

Defining a Column

You can define a column as shown below.

```
column name1 data type,
column name2 data type,

example:

age int,
name text
```

Primary Key

The primary key is a column that is used to uniquely identify a row. Therefore,defining a primary key is mandatory while creating a table. A primary key is made of one or more columns of a table. You can define a primary key of a table as shown below.

```
CREATE TABLE tablename(
   column1 name datatype PRIMARYKEY,
   column2 name data type,
   column3 name data type.
   )
```

or

```
CREATE TABLE tablename(
   column1 name datatype PRIMARYKEY,
   column2 name data type,
   column3 name data type,
   PRIMARY KEY (column1)
   )
```

Example

Given below is an example to create a table in Cassandra using cqlsh. Here we are:

- Using the keyspace tutorialspoint
- Creating a table named **emp**

It will have details such as employee name, id, city, salary, and phone number. Employee id is the primary key.

```
cqlsh> USE tutorialspoint;
cqlsh:tutorialspoint>; CREATE TABLE emp(
   emp_id int PRIMARY KEY,
```

```
   emp_name text,
   emp_city text,
   emp_sal varint,
   emp_phone varint
   );
```

Verification

The select statement will give you the schema. Verify the table using the select statement as shown below.

```
cqlsh:tutorialspoint> select * from emp;

 emp_id | emp_city | emp_name | emp_phone | emp_sal
--------+----------+----------+-----------+---------

(0 rows)
```

Here you can observe the table created with the given columns. Since we have deleted the keyspace tutorialspoint, you will not find it in the keyspaces list.

Creating a Table using Java API

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

Step1: Create a Cluster Object

First of all, create an instance of the **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, 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 named **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 creating a table named **emp.** You have to store the query in a string variable and pass it to the execute method as shown below.

```
//Query
String query = "CREATE TABLE emp(emp_id int PRIMARY KEY, "
   + "emp_name text, "
   + "emp_city text, "
   + "emp_sal varint, "
   + "emp_phone varint );";
session.execute(query);
```

Given below is the complete program to create and use a keyspace in Cassandra using Java API.

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

public class Create_Table {

   public static void main(String args[]){

      //Query
      String query = "CREATE TABLE emp(emp_id int PRIMARY KEY, "
         + "emp_name text, "
         + "emp_city text, "
         + "emp_sal varint, "
         + "emp_phone varint );";

      //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(query);

      System.out.println("Table 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_Table.java
$java Create_Table
```

Under normal conditions, it should produce the following output:

```
Table created
```