Lesson 3 Demo 4 Using the WHERE Clause

May 11, 2021

1 Lesson 3 Demo 4: Using the WHERE Clause

1.0.1 In this demo we are going to walk through the basics of using the WHERE clause in Apache Cassandra.

We will use a python wrapper/ python driver called cassandra to run the Apache Cassandra queries. This library should be preinstalled but in the future to install this library you can run this command in a notebook to install locally: ! pip install cassandra-driver #### More documentation can be found here: https://datastax.github.io/python-driver/

Import Apache Cassandra python package

```
In [4]: import cassandra
```

1.0.2 First let's create a connection to the database

```
In [5]: from cassandra.cluster import Cluster
        try:
            cluster = Cluster(['127.0.0.1']) #If you have a locally installed Apache Cassandra i
            session = cluster.connect()
        except Exception as e:
            print(e)
```

1.0.3 Let's create a keyspace to do our work in

```
In [6]: try:
            session.execute("""
            CREATE KEYSPACE IF NOT EXISTS udacity
            WITH REPLICATION =
            { 'class' : 'SimpleStrategy', 'replication_factor' : 1 }"""
        )

        except Exception as e:
            print(e)
```

Connect to our Keyspace. Compare this to how we had to create a new session in PostgreSQL.

```
In [7]: try:
            session.set_keyspace('udacity')
        except Exception as e:
            print(e)
```

1.0.4 Let's imagine we would like to start creating a new Music Library of albums.

1.0.5 We want to ask 4 question of our data

1. Give me every album in my music library that was released in a given year `select * from music_library WHERE YEAR=1970` #### 2. Give me the album that is in my music library that was released in a given year by "The Beatles" `select * from music_library WHERE YEAR = 1970 AND ARTIST_NAME = "The Beatles"` #### 3. Give me all the albums released in a given year in a give location `select * from music_library WHERE YEAR = 1970 AND LOCATION = "Liverpool"` #### 4. Give me city that the albumn "Let It Be" was recorded `select city from music_library WHERE YEAR = "1970" AND ARTIST_NAME = "The Beatles" AND ALBUM_NAME="Let it Be"`

1.0.6 Here is our Collection of Data

1.0.7 How should we model this data? What should be our Primary Key and Partition Key? Since our data is looking for the YEAR let's start with that. From there we will add clustering columns on Artist Name and Album Name.

`Table Name: music_library column 1: Year column 2: Artist Name column 3: Album Name Column 4: City PRIMARY KEY(year, artist_name, album_name)`

```
In [8]: query = "CREATE TABLE IF NOT EXISTS music_library "
        query = query + "(year int, artist_name text, album_name text, city text, PRIMARY KEY (y
        try:
            session.execute(query)
        except Exception as e:
            print(e)
```

1.0.8 Let's insert our data into of table

```
In [9]: query = "INSERT INTO music_library (year, artist_name, album_name, city)"
        query = query + " VALUES (%s, %s, %s, %s)"

        try:
            session.execute(query, (1970, "The Beatles", "Let it Be", "Liverpool"))
        except Exception as e:
            print(e)

        try:
            session.execute(query, (1965, "The Beatles", "Rubber Soul", "Oxford"))
        except Exception as e:
            print(e)
```

```
try:
    session.execute(query, (1965, "The Who", "My Generation", "London"))
except Exception as e:
    print(e)

try:
    session.execute(query, (1966, "The Monkees", "The Monkees", "Los Angeles"))
except Exception as e:
    print(e)

try:
    session.execute(query, (1970, "The Carpenters", "Close To You", "San Diego"))
except Exception as e:
    print(e)
```

1.0.9 Let's Validate our Data Model with our 4 queries.

`select * from music_library WHERE YEAR=1970`

```
In [14]: query = "select * from music_library WHERE YEAR=1970"
         try:
             rows = session.execute(query)
         except Exception as e:
             print(e)

         for row in rows:
             print (row.year, row.artist_name, row.album_name, row.city)
```

```
1970 The Beatles Let it Be Liverpool
1970 The Carpenters Close To You San Diego
```

1.0.10 Success it worked! Let's try the 2nd query.

`select * from music_library WHERE YEAR = 1970 AND ARTIST_NAME = "The Beatles"`

```
In [12]: query = "select * from music_library WHERE YEAR=1970 AND ARTIST_NAME = 'The Beatles'"
         try:
             rows = session.execute(query)
         except Exception as e:
             print(e)

         for row in rows:
             print (row.year, row.artist_name, row.album_name, row.city)
```

```
1970 The Beatles Let it Be Liverpool
```

1.0.11 Success it worked! Let's try the 3rd query.

```
select * from music_library WHERE YEAR = 1970 AND LOCATION = "Liverpool"
```

```
In [13]: query = "select * from music_library WHERE YEAR = 1970 AND LOCATION = 'Liverpool'"
         try:
             rows = session.execute(query)
         except Exception as e:
             print(e)

         for row in rows:
             print (row.year, row.artist_name, row.album_name, row.city)
```

```
Error from server: code=2200 [Invalid query] message="Undefined column name location"
```

1.0.12 Error! You can not try to access a column or a clustering column if you have not used the other defined clustering column. Let's see if we can try it a different way.

```
select city from music_library WHERE YEAR = "1970" AND ARTIST_NAME = "The Beatles"
AND ALBUM_NAME="Let it Be"
```

```
In [20]: query = "select city from music_library WHERE YEAR = 1970 AND ARTIST_NAME = 'The Beatle
         try:
             rows = session.execute(query)
         except Exception as e:
             print(e)

         for row in rows:
             print (row.city)
```

```
Liverpool
```

1.0.13 For the sake of the demo, I will drop the table.

```
In [11]: query = "drop table music_library"
         try:
             rows = session.execute(query)
         except Exception as e:
             print(e)

         query = "drop table music_library1"
         try:
             rows = session.execute(query)
         except Exception as e:
             print(e)
```

1.0.14 And Finally close the session and cluster connection

```
In [12]: session.shutdown()
         cluster.shutdown()

In [ ]:
```