Lesson 3 Exercise 3 Clustering Column-ANSWER KEY

March 10, 2022

1 Lesson 3 Exercise 3 Solution: Focus on Clustering Columns

1.0.1 Walk through the basics of creating a table with a good Primary Key and Clustering Columns in Apache Cassandra, inserting rows of data, and doing a simple CQL query to validate the information.

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 [1]: import cassandra
```

1.0.2 Create a connection to the database

```
In [2]: 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 Create a keyspace to work in

```
In [3]: 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 [4]: try:
            session.set_keyspace('udacity')
        except Exception as e:
            print(e)
```

1.0.4 Imagine we would like to start creating a new Music Library of albums.

1.0.5 We want to ask 1 question of our data:

1. Give me all the information from the music library about a given album `select * from album_library WHERE album_name="Close To You"`

1.0.6 Here is the Data:

1.0.7 How should we model this data? What should be our Primary Key and Partition Key?

1.0.8 Since the data is looking for the `ALBUM_NAME` let's start with that. From there we will need to add other elements to make sure the Key is unique. We also need to add the `ARTIST_NAME` as Clustering Columns to make the data unique. That should be enough to make the row key unique.

`Table Name: music_library column 1: Year column 2: Artist Name column 3: Album Name Column 4: City PRIMARY KEY(album name, artist name)`

```
In [5]: query = "CREATE TABLE IF NOT EXISTS music_library "
        query = query + "(album_name text, artist_name text, year int, city text, PRIMARY KEY (a
        try:
            session.execute(query)
        except Exception as e:
            print(e)
```

1.0.9 Insert the data into the table

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

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

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

        try:
            session.execute(query, ("Beatles For Sale", "The Beatles", 1964, "London"))
```

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

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

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

1.0.10 Validate the Data Model -- Did it work?

```
select * from album_library WHERE album_name="Close To You"
```

```
In [ ]: query = "select * from music_library WHERE album_NAME='Close To You'"
        try:
            rows = session.execute(query)
        except Exception as e:
            print(e)

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

1.0.11 Success it worked! We created a unique Primary key that evenly distributed our data, with clustering columns

1.0.12 For the sake of the demo, drop the table

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

1.0.13 Close the session and cluster connection

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

In [ ]:
```