Lesson 3 Exercise 1 Three Queries Three Tables-ANSWER KEY

March 10, 2022

1 Lesson 3 Exercise 1 Solution: Three Queries Three Tables

1.0.1 Walk through the basics of creating a table in Apache Cassandra, inserting rows of data, and doing a simple CQL query to validate the information. You will practice Denormalization, and the concept of 1 table per query, which is an encouraged practice with 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 [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 Let's imagine we would like to start creating a Music Library of albums.

1.0.5 We want to ask 3 questions of our data

1. Give every album in the music library that was released in a given year `select * from music_library WHERE YEAR=1970` #### 2. Give every album in the music library that was created by a given artist
`select * from artist_library WHERE artist_name="The Beatles"` #### 3. Give all the information from the music library about a given album `select * from album_library WHERE album_name="Close To You"`

1.0.6 Because we want to do three different queries, we will need different tables that partition the data differently.

- The music library table will be partitioned by year that will become the partition key, and artist name will be the clustering column to make each Primary Key unique.
- The artist library table will be partitioned by artist name that will be the partition key, and year will be the clustering column to make each Primary Key unique. More on Primary keys in the next lesson and demo.
- The album library table will be partitioned by album name that will be the partition key, and artist will be the clustering colum to make each Primary Key unique.

```
   Table Name: music_library column 1: Year column 2: Artist Name column 3: Album
Name PRIMARY KEY(year, artist name)
   Table Name: artist_library  column 1: Artist Name column 2: Year column 3: Album
Name PRIMARY KEY (artist name, year)
   Table Name: album_library  column 1: Album Name column 2: Artist Name column 3:
Year PRIMARY KEY (album_name, artist_name)
```

1.0.7 Create the tables

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

        query = "CREATE TABLE IF NOT EXISTS artist_library "
        query = query + "(artist_name text, year int, album_name text, PRIMARY KEY (artist_name,
        try:
            session.execute(query)
        except Exception as e:
            print(e)

        query = "CREATE TABLE IF NOT EXISTS album_library "
        query = query + "(artist_name text, album_name text, year int, PRIMARY KEY (album_name,
        try:
            session.execute(query)
        except Exception as e:
            print(e)
```

1.0.8 Insert data into the tables

```
In [6]: query = "INSERT INTO music_library (year, artist_name, album_name)"
        query = query + " VALUES (%s, %s, %s)"

        query1 = "INSERT INTO artist_library (artist_name, year, album_name)"
        query1 = query1 + " VALUES (%s, %s, %s)"

        query2 = "INSERT INTO album_library (album_name, artist_name, year)"
        query2 = query2 + " VALUES (%s, %s, %s)"

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

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

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

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

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

        try:
            session.execute(query1, ("The Beatles", 1970, "Let it Be"))
```

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

try:
    session.execute(query1, ("The Beatles", 1965, "Rubber Soul"))
except Exception as e:
    print(e)

try:
    session.execute(query1, ("The Who", 1965, "My Generation"))
except Exception as e:
    print(e)

try:
    session.execute(query1, ("The Monkees", 1966, "The Monkees"))
except Exception as e:
    print(e)

try:
    session.execute(query1, ("The Carpenters", 1970, "Close To You"))
except Exception as e:
    print(e)


try:
    session.execute(query2, ("Let it Be", "The Beatles", 1970))
except Exception as e:
    print(e)

try:
    session.execute(query2, ("Rubber Soul", "The Beatles", 1965))
except Exception as e:
    print(e)

try:
    session.execute(query2, ("My Generation", "The Who", 1965))
except Exception as e:
    print(e)

try:
    session.execute(query2, ("The Monkees", "The Monkees", 1966))
except Exception as e:
    print(e)

try:
    session.execute(query2, ("Close To You", "The Carpenters", 1970))
except Exception as e:
    print(e)
```

1.0.9 This might have felt unnatural to insert duplicate data into two tables. If I just normalized these tables, I wouldn't have to have extra copies! While this is true, remember there are no `JOINS` in Apache Cassandra. For the benefit of high availibity and scalabity denormalization must be how this is done.

1.0.10 Validate our Data Model

`select * from music_library WHERE YEAR=1970`

```
In [7]: query = "select * from music_library WHERE YEAR=1970"
        try:
            rows = session.execute(query)
            for row in rows:
                print (row.year, row.artist_name, row.album_name,)

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

```
1970 The Beatles Let it Be
1970 The Carpenters Close To You
```

1.0.11 Validate our Data Model

`select * from artist_library WHERE ARTIST_NAME = "The Beatles"`

```
In [8]: query = "select * from artist_library WHERE ARTIST_NAME='The Beatles'"
        try:
            rows = session.execute(query)
        except Exception as e:
            print(e)

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

```
The Beatles 1965 Rubber Soul
The Beatles 1970 Let it Be
```

1.0.12 Validate our Data Model

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

```
In [9]: query = "select * from album_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.year, row.album_name)
```

```
The Carpenters 1970 Close To You
```

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

```
In [10]: query = "drop table music_library"
         try:
             rows = session.execute(query)
         except Exception as e:
             print(e)

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

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

1.0.14 Close the session and cluster connection

```
In [11]: session.shutdown()
         cluster.shutdown()
```

```
In [ ]:
```