Data Modeling with Cassandra

REVIEW

HISTORY

Meets Specifications

Congratulations 🎓

You have passed this project.

The structure of this project is impressive and you've done a great job with the combination of the PARTITION KEY and CLUSTERING COLUMNS which was appropriately used to uniquely identify each row according to the requirements. I hope you must have learnt some great concepts and got a feel of the industry standard project on what Data Engineers work on and what problems do they face. Keep up the good work.

Be a lifelong leaner
Stay Safe and Stay Udacious

Additional Readings

7 mistakes when using Apache Cassandra
Tips for securing Cassandra
DynamoDB vs. Cassandra: from "no idea" to "it's a no-brainer"

ETL Pipeline Processing

✓

Student creates `event_data_new.csv` file.

Good job! I see the `event_datafile_new.csv` file, which indicates you followed the ETL pipeline to create the csv file and the count `6821` is also correct.

✓

Student uses the appropriate datatype within the `CREATE` statement.

You used the right data types in your create statement which reflects a good understanding of the data and how to model it using the right Cassandra data types

Additional Readings

Cassandra Data Types | Built-in, Collection, User-defined
Pre-defined data type in Apache Cassandra

Data Modeling

✓

Student creates the correct Apache Cassandra tables for each of the three queries. The `CREATE TABLE` statement should include the appropriate table.

The rule we are testing here is one table per query, and you passed it successfully !
You created a table for each one of the three queries to match that rule as Cassandra doesn't offer a lot of flexibility of a generic table with created and mapped to the query requirement

Additional Readings

Logical Data Modeling
Apache Cassandra Data Model: Components And Statements

✓

Student demonstrates good understanding of data modeling by generating correct SELECT statements to generate the result being asked for in the question.

The SELECT statement should NOT use `ALLOW FILTERING` to generate the results.

The select statements are very good as you fulfilled the following points
1- You used columns by names and not select * which is better in terms of readability of the query and performance as you are fetching only the columns you need
2- You didn't use ALLOW FILTERING in your select queries which means that you are not spanning the whole cluster to fetch the records required
3- You used - from your point of view - the right where conditions that match the partitioning and clustering logic in your create statements.

Good job!

Additional Readings

How to avoid Cassandra ALLOW FILTERING?
ALLOW FILTERING vs NOT; Cassandra Data Model Question

✓

Student should use table names that reflect the query and the result it will generate. Table names should include alphanumeric characters and underscores, and table names must start with a letter.

Table names are indicative and are related to the conditions used or the data retrieved. You named them according to the query perspective , whether the data is related to a session, or retrieved info about the artist, or retrieved info about the user. Nice job in this point !

Additional Readings

Valid characters in Table names and keyspaces

✓

The sequence in which columns appear should reflect how the data is partitioned and the order of the data within the partitions.

Good work! the ordering in create statements matched the ordering in PRIMARY KEY section and the ordering in the insertion matches that too.
This point is important. As you know, Apache Cassandra is a partition row store, which means the partition key determines which any particular row is stored on which node. In case of composite partition key, partitions are distributed across the nodes of the cluster and how they are chunked for write purposes. Any clustering column(s) would determine the order in which the data is sorted within the partition.

PRIMARY KEYS

✓

The combination of the PARTITION KEY alone or with the addition of CLUSTERING COLUMNS should be used appropriately to uniquely identify each row.

The COMPOSITE KEY combination for the tables are in place. You have explicitly defined `itemInSession` as CLUSTERING COLUMN for the second table to fulfill the sorting requirement. Good use of `user_id` in the third query to uniquely identify each record per song per user.

Additional Readings

Should every table in Cassandra have a partition key?
Cassandra - querying on clustering keys

Presentation

✓

The notebooks should include a description of the query the data is modeled after.

Good job. I see you created header with description for each query in the comments.

Suggestion

You can use PrettyTable to display data in a visually appealing ASCII tabular format

```
t = PrettyTable(['Artist', 'Song', 'Length'])
for row in rows:
    t.add_row([row.artist, row.song, row.length])
print(t)
```

✓

Code should be organized well into the different queries. Any in-line comments that were clearly part of the project instructions should be removed so the notebook provides a professional look.

Good job. You separated into three queries and also removed any unnecessary comments.

DOWNLOAD PROJECT

RETURN TO PATH