Kajol Tanesh Shah
A20496724
Fall 2022

CSP554—Big Data Technologies

Assignment #12 – Cassandra

Exercise 1) (4 points)
Read the article "A Big Data Modeling Methodology for Apache Cassandra" available on the blackboard in the 'Articles' section. Provide a ½ page summary including your comments and impressions.

Ans. Apache Cassandra is a leading transactional, scalable, and highly-available distributed database. It is known to manage some of the world's largest datasets on clusters with many thousands of nodes deployed across multiple data centers. The wide adoption of Cassandra in big data applications is attributed to, among other things, its scalable and fault-tolerant peer-to-peer architecture, versatile and flexible data model that evolved from the BigTable data model declarative and user-friendly Cassandra Query Language (CQL), and very efficient write and read access paths that enable critical big data applications to stay always on, scale to millions of transactions per second, and handle node and even entire data center failures with ease.

Cassandra Data Model:
A CQL table (hereafter referred to as a table) can be viewed as a set of partitions that contain rows with a similar structure. Each partition in a table has a unique partition key and each row in a partition may optionally have a unique clustering key. Both keys can be simple (one column) or composite (multiple columns). The combination of a partition key and a clustering key uniquely identifies a row in a table and is called a primary key. While the partition key component of a primary key is always mandatory, the clustering key component is optional.

Query Model:
Queries over tables are expressed in CQL, which has an SQL-like syntax. Unlike SQL, CQL supports no binary operations, such as joins, and has a number of rules for query predicates that ensure efficiency and scalability:- 1) Only primary key columns may be used in a query predicate 2) All partition key columns must be restricted by values 3) All, some, or none of the clustering key columns can be used in a query predicate 4) If a clustering key column is used in a query predicate, then all clustering key columns that precede this clustering column in the primary key definition must also be used in the predicate.

Conceptual Data Model:
Designing a Cassandra database schema requires an understanding of the to-be-managed data and how a data-driven application needs to access such data. The former is captured via a conceptual data model, such as an entity-relationship model. In contrast, the latter is captured via an application workflow diagram that defines data access patterns for individual application tasks.

Logical Data Modeling:
A logical data model corresponds to a Cassandra database schema with table schemas defining columns, primary, partition, and clustering keys. The query-driven conceptual-to-logical data model mapping is defined via data modeling principles, mapping rules, and mapping patterns.

Data Modeling Principles:
The following four data modeling principles provide a foundation for the mapping of conceptual to logical data models:-

- DMP1 (Know Your Data)
- DMP2 (Know Your Queries)
- DMP3 (Data Nesting)
- DMP4 (Data Duplication)

Mapping Rules:
The five mapping rules that guide a query-driven transition from a conceptual data model to a logical data model:-

- MR1 (Entities and Relationships) - Entity and relationship types map to tables, while entities and relationships map to table rows
- MR2 (Equality Search Attributes). Equality search attributes, which are used in a query predicate, map to the prefix columns of a table primary key
- MR3 (Inequality Search Attributes). An inequality search attribute, which is used in a query predicate, maps to a table clustering key column
- MR4 (Ordering Attributes). Ordering attributes, which are specified in a query, map to clustering key columns with ascending or descending clustering order as prescribed by the query.
- MR5 (Key Attributes). Key attribute types map to primary key columns

Mapping Patterns:
Mapping Patterns serve as the basis for automating Cassandra database schema design.

Physical Data Modeling:
The final step is the analysis and optimization of a logical data model to produce a physical data model.

Exercise 2) (3 points)

Step A – Start an EMR cluster
Start up an EMR/Hadoop cluster as previously, but instead of choosing the "Core Hadoop" configuration choose the "Spark" configuration (see below), otherwise proceed as before.

Step B – Install the Cassandra database software and start it
Open up a terminal connection to your EMR master node. Over the course of this exercise, you will need to open up three separate terminal connections to your EMR master node. This is the first, which we will call Cass-Term.

Enter the following two command:
wget https://archive.apache.org/dist/cassandra/3.11.2/apache-cassandra-3.11.2-bin.tar.gz

tar -xzvf apache-cassandra-3.11.2-bin.tar.gz

Note, this will create a new directory (apache-cassandra-3.11.2) holding the Cassandra software release.

Then enter this command to start Cassandra (lots of diagnostic messages will appear):
apache-cassandra-3.11.2/bin/cassandra &

Step C – Run the Cassandra interactive command line interface
Open a second terminal connection to the EMR master node. Going forward we will call this terminal connection: Cqlsh-Term.

Ans.

Step D – Prepare to edit your Cassandra code
Open a third terminal connection to the EMR master node. Going forward we will call this terminal connection: Edit-Term.

You will use this terminal window to run the 'vi' editor to create your Cassandra code files.

a) Create a file in your working (home) directory called init.cql using your Edit-term (or using your PC/MAC and then scp it to the EMR master node) and enter the following command. Use your IIT id as the name of your keyspace… For example, if your id is A1234567, then replace <IIT id> below with that value:

Ans.
CREATE KEYSPACE <A20496724> WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };

```
[hadoop@ip-172-31-6-128 ~]$ vi init.cql
[hadoop@ip-172-31-6-128 ~]$ cat init.cql
CREATE KEYSPACE A20496724 WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };
[hadoop@ip-172-31-6-128 ~]$
```

b) Then execute this file in the CQL shell using the Cqlsh-Term as follows…

Ans.

source ‘./init.cql’;

```
[hadoop@ip-172-31-6-128 ~]$ ls
apache-cassandra-3.11.2  apache-cassandra-3.11.2-bin.tar.gz  init.cql
[hadoop@ip-172-31-6-128 ~]$ apache-cassandra-3.11.2/bin/cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> source ‘./init.cql’;
Invalid syntax at line 1, char 8
  source ‘./init.cql’;
         ^
cqlsh>
cqlsh> source './init.cql';
cqlsh>
```

c) To check if your script file has created a keyspace execute the following in the CQL shell:

Ans.

describe keyspaces;

```
cqlsh> describe keyspaces;

system_schema  system_auth  a20496724  system  system_distributed  system_traces

cqlsh>
```

d) At this point you have created a keyspace unique to you. So, make that keyspace the default by entering the following into the CQL shell:

Ans.

USE a20496724;

```
cqlsh> use a20496724;
cqlsh:a20496724>
```

Now create a file in your working directory called ex2.cql using the Edit-Term (or PC/MAC and scp). In this file write the command to create a table named ‘Music’ with the following characteristics:

Attribute Name	Attribute Type	Primary Key / Cluster Key
artistName	text	Primary Key
albumName	text	Cluster Key

numberSold	int	Non Key Column
Cost	int	Non Key Column

Ans.
CREATE TABLE a20496724.Music (artistName text, albumName text, numberSold int, Cost int, PRIMARY KEY (artistName, albumName)) WITH CLUSTERING ORDER BY (albumName ASC);

```
[hadoop@ip-172-31-6-128 ~]$ vi ex2.cql
[hadoop@ip-172-31-6-128 ~]$ cat ex2.cql
CREATE TABLE a20496724.Music (artistName text, albumName text, numberSold int, Cost int, PRIMARY KEY (artistName, albumName)) WITH CLUSTERING ORDER BY (albumName ASC);
[hadoop@ip-172-31-6-128 ~]$ ls
apache-cassandra-3.11.2  apache-cassandra-3.11.2-bin.tar.gz  ex2.cql  init.cql
[hadoop@ip-172-31-6-128 ~]$
```

Execute ex2.cql in the CQL shell. Then execute the shell command 'DESCRIBE TABLE Music;' and include the output as the result of this exercise.
Ans.

```
cqlsh:a20496724> source './ex2.cql';
cqlsh:a20496724> DESCRIBE TABLE Music;

CREATE TABLE a20496724.music (
    artistname text,
    albumname text,
    cost int,
    numbersold int,
    PRIMARY KEY (artistname, albumname)
) WITH CLUSTERING ORDER BY (albumname ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

cqlsh:a20496724>
```

Exercise 3) (3 points)

Now create a file in your working directory called ex3.cql using the Edit-Term. In this file write the commands to insert the following records into table 'Music'…

artistName	albumName	numberSold	cost
Mozart	Greatest Hits	100000	10
Taylor Swift	Fearless	2300000	15
Black Sabbath	Paranoid	534000	12
Katy Perry	Prism	800000	16
Katy Perry	Teenage Dream	750000	14

Ans.
INSERT INTO A20496724.Music (artistName, albumName, numberSold, cost)
VALUES ('Mozart', 'Greatest Hits', 100000, 10);

INSERT INTO A20496724.Music (artistName, albumName, numberSold, cost)
VALUES ('Taylor Swift', 'Fearless', 2300000, 15);

INSERT INTO A20496724.Music (artistName, albumName, numberSold, cost)
VALUES ('Black Sabbath', 'Paranoid', 534000, 12);

INSERT INTO A20496724.Music (artistName, albumName, numberSold, cost)
VALUES ('Katy Perry', 'Prism', 800000, 16);

INSERT INTO A20496724.Music (artistName, albumName, numberSold, cost)
VALUES ('Katy Perry', 'Teenage Dream', 750000, 14);

a) Execute ex3.cql. Provide the content of this file as the result of this exercise.
Ans.

```
[hadoop@ip-172-31-6-128 ~]$ vi ex3.cql
[hadoop@ip-172-31-6-128 ~]$ ls
apache-cassandra-3.11.2  apache-cassandra-3.11.2-bin.tar.gz  ex2.cql  ex3.cql  init.cql
[hadoop@ip-172-31-6-128 ~]$ cat ex3.cql
INSERT INTO A20496724.Music (artistName, albumName, numberSold, cost)
VALUES ('Mozart', 'Greatest Hits', 100000, 10);
INSERT INTO A20496724.Music (artistName, albumName, numberSold, cost)
VALUES ('Taylor Swift', 'Fearless', 2300000, 15);
INSERT INTO A20496724.Music (artistName, albumName, numberSold, cost)
VALUES ('Black Sabbath', 'Paranoid', 534000, 12);
INSERT INTO A20496724.Music (artistName, albumName, numberSold, cost)
VALUES ('Katy Perry', 'Prism', 800000,  16);
INSERT INTO A20496724.Music (artistName, albumName, numberSold, cost)
VALUES ('Katy Perry', 'Teenage Dream', 750000,  14);

[hadoop@ip-172-31-6-128 ~]$
```

b) Execute the command ‘SELECT * FROM Music;’ and provide the output of this command as another result of the exercise.

Ans.

```
cqlsh:a20496724> source './ex3.cql';
cqlsh:a20496724> SELECT * FROM Music;

 artistname    | albumname      | cost | numbersold
---------------+----------------+------+------------
        Mozart | Greatest Hits |   10 |     100000
 Black Sabbath |      Paranoid |   12 |     534000
  Taylor Swift |      Fearless |   15 |    2300000
    Katy Perry |         Prism |   16 |     800000
    Katy Perry | Teenage Dream |   14 |     750000

(5 rows)
cqlsh:a20496724>
```

Exercise 4) (2 points)
Now create a file in your working directory called ex4.cql using the Edit-Term. In this file write the commands to query and output only Katy Perry songs. Execute ex4.cql. Provide the content of this file and output of executing this file as the result of this exercise.

Ans.
SELECT * FROM Music WHERE artistName = 'Katy Perry';

```
[hadoop@ip-172-31-6-128 ~]$ vi ex4.cql
[hadoop@ip-172-31-6-128 ~]$ cat ex4.cql
SELECT * FROM Music WHERE artistName = 'Katy Perry';
[hadoop@ip-172-31-6-128 ~]$
```

```
cqlsh:a20496724> source 'ex4.cql';

 artistname | albumname     | cost | numbersold
------------+---------------+------+------------
 Katy Perry |         Prism |   16 |     800000
 Katy Perry | Teenage Dream |   14 |     750000

(2 rows)
cqlsh:a20496724>
```

Exercise 5) (2 points)
Now create a file in your working directory called ex5.cql using the Edit-Term. In this file write the commands to query only albums that have sold 700000 copies or more. Execute ex5.cql. Provide the content of this file and the output of executing this file as the result of this exercise.

Ans.
SELECT * FROM Music WHERE numberSold >= 700000 ALLOW FILTERING;

```
[hadoop@ip-172-31-6-128 ~]$ vi ex5.cql
[hadoop@ip-172-31-6-128 ~]$ cat ex5.cql
SELECT * FROM Music WHERE numberSold >= 700000 ALLOW FILTERING;
[hadoop@ip-172-31-6-128 ~]$
```

```
cqlsh:a20496724> source 'ex5.cql';

 artistname   | albumname     | cost | numbersold
--------------+---------------+------+------------
 Taylor Swift |      Fearless |   15 |    2300000
   Katy Perry |         Prism |   16 |     800000
   Katy Perry | Teenage Dream |   14 |     750000

(3 rows)
cqlsh:a20496724>
```