SL- VI
Expt. 3

Aim: Design Data Model using NoSQL Database - ***Cassandra***
Steps:

First install and configure ***Cassandra*** from,
https://sl6it.wordpress.com/2015/12/10/4-installation-of-nosql-database-cassandra/

We will create data model for a social music service.

start cassandra daemon
~/cassandra/bin/cassandra -f

The cassandra daemon should start in the foreground
(**don't press ctrl + c**; as it'll terminate the daemon)

open a **new** terminal
~/cassandra/bin/cqlsh

It'll have output like this
cqlsh>

First, create a keyspace

```
CREATE KEYSPACE mykeyspace100
WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };
```

use the new keyspace:

```
USE mykeyspace100;
```

Create a songs table having a title, album, and artist column, plus a column (called data) for the actual audio file itself.

```
CREATE TABLE songs (
  id int PRIMARY KEY,
  title text,
  album text,
  artist text,
  data blob
 );
```

Create playlists table

```
CREATE TABLE playlists (
  id int,
  song_order int,
  song_id int,
  title text,
  album text,
  artist text,
  PRIMARY KEY  (id, song_order ) );
```

The combination of the **id** and **song_order (Compound Primary Key)** in the playlists table uniquely identifies a row in the playlists table.

Use the INSERT command to insert data in the playslists table.

```
INSERT INTO playlists (id, song_order, song_id, title, artist, album)
  VALUES (100, 1,1001, 'La Grange', 'Mike', 'Tres Hombres');


INSERT INTO playlists (id, song_order, song_id, title, artist, album)
  VALUES (100, 2,1002,'Moving in', 'Swift', 'We must Obey');


INSERT INTO playlists (id, song_order, song_id, title, artist, album)
VALUES (100, 3,1003, 'One day', 'Justin', 'Roll Away');


INSERT INTO playlists (id, song_order, song_id, title, artist, album)
  VALUES (100, 4,1004,'Ojo Rojo', 'Swift', 'No One');
```

Now use the SELECT query to display the table's data

```
SELECT * FROM playlists;
```

Create index on artist

```
CREATE INDEX ON playlists( artist );
```

Search albums and titles of artist **'Swift'**

```
SELECT album, title FROM playlists WHERE artist = 'Swift';
```

You can query a single sequential set of data on disk to get the songs for a playlist.

```
SELECT * FROM playlists WHERE id = 100
  ORDER BY song_order DESC LIMIT 50;
```

 # Adding a collection to a table

Collection in Cassandra is of three types
- Set, List, Map

Each element of a set, list, or map is internally stored as one Cassandra column.

Alter playlist table to add a collection set, ***tags***:

```
ALTER TABLE playlists ADD tags set<text>;
```

 # Updating a collection

To update a set, use the UPDATE command and the addition (+) operator to add an element or the subtraction (-) operator to remove an element.

Update the playlists table to insert the tags data:

```
UPDATE playlists   SET tags = tags + {'2010'} WHERE id = 100 AND song_order = 4;

SELECT * FROM playlists;

UPDATE playlists SET tags = tags + {'2007'}   WHERE id = 100 AND song_order = 2;

SELECT * FROM playlists;

UPDATE playlists SET tags = tags + {'classic'}   WHERE id = 100 AND song_order = 2;

SELECT * FROM playlists;

UPDATE playlists SET tags = tags + {'1973'}   WHERE id = 100 AND song_order = 1;

SELECT * FROM playlists;

UPDATE playlists SET tags = tags + {'blues'}   WHERE id = 100 AND song_order = 1;

SELECT * FROM playlists;

UPDATE playlists SET tags = tags + {'rock'}   WHERE id = 100 AND song_order = 4;

SELECT * FROM playlists;
```

Alter playlist table to add a list collection, ***reviews***:

```
ALTER TABLE playlists ADD reviews list<text>;
```

To update a list, a similar syntax using square brackets instead of curly brackets is used.

```
UPDATE playlists SET reviews =reviews + [ 'best lyrics' ] WHERE id=100 and song_order= 4;

SELECT * FROM playlists;

UPDATE playlists SET reviews =reviews + [ 'magical' ] WHERE id=100 and song_order= 4;

SELECT * FROM playlists;
```

Alter playlist table to add a map collection, ***venue (*** a schedule of live appearances).

```
ALTER TABLE playlists ADD venue map<timestamp, text>;
```

To update a map, use INSERT to specify the data in a map collection.

```
INSERT INTO playlists (id, song_order, venue)
VALUES (100, 4, { '2016-9-22 22:00'  : 'The Fillmore',  '2016-10-1 21:00' : 'The Apple Barrel'});

SELECT * FROM playlists;

INSERT INTO playlists (id, song_order, venue)
VALUES (100, 3, { '2016-1-12 22:00'  : 'Cactus Cafe',   '2016-01-22 20:00' : 'Mohawk'});

SELECT * FROM playlists;
```

Inserting data into the map replaces the entire map.

 # Indexing a collection

We can index collections and query the database to find a collection containing a particular value. Suppose we want to find songs tagged ***blues*** and that debuted at ***the Fillmore***.

First index the tags set and venue map.

```
CREATE INDEX ON playlists (tags);

CREATE INDEX mymapindex ON playlists (venue);
```

Filter data in a collection

```
SELECT album, tags FROM playlists;
```

Query for values in the tags set.

```
SELECT album, tags FROM playlists WHERE tags CONTAINS 'blues';
```

Query for values in the venue map.

```
SELECT artist, venue FROM playlists WHERE venue CONTAINS 'The Fillmore';
```

Exit cqlsh

```
cqlsh> exit
```

Reference:
http://docs.datastax.com/en/cql/3.1/cql/ddl/ddl_intro_c.html

--
Prof. S. T. Kolhe
(IT DEPT, SRES COE Kopargaon)