Exercise 2.3: Denormalizing

In this exercise, you will:

- Create tables to support querying for videos by actor or genre

Background

With all of the success you've been having on the video sharing development team, you have been promoted and assigned to work on a high-priority project to incorporate movie content into the KillrVideo application.

Your new team is normalizing their video and actor metadata into separate tables and currently are stuck figuring out how to join tables in Cassandra. Having been around the Cassandra block a few times, you know that JOINs are expensive and not supported. It is up to you to show your team the optimal way of performing these queries.

The video metadata is similar to what was in the video sharing domain:

Column Name	Data Type
video_id	timeuuid
added_date	timestamp
description	text
encoding	video_encoding
tags	set<text>
title	text
user_id	uuid

There is also the additional following metadata:

Column Name	Data Type
actor	text
character	text
genre	text

With this metadata, the data model must support the following queries:

- Q1: Retrieve videos an actor has appeared in (newest first).
- Q2: Retrieve videos within a particular genre (newest first).

Creating a new videos_by_actor table

1. Navigate to /labwork/denormalization.

2. In 'cqlsh', create a new table called 'videos_by_actor' which will support query Q1.

Before we do that though, look a little more closely at the table above. Our 'encoding' column is actually something called a User Defined Type (or UDT for short). Fear not! We will be talking about these in the next exercise. For now, copy and paste this code to create the UDT so that our create table works correctly.

3. Creating the 'encoding' UDT.

```
CREATE TYPE IF NOT EXISTS video_encoding (
   encoding TEXT,
   height INT,
   width INT,
   bit_rates SET<TEXT>
);
```

4. We provided most of the CREATE TABLE for you except the PRIMARY KEY.

```
CREATE TABLE videos_by_actor (
  actor text,
  added_date timestamp,
  video_id timeuuid,
  character_name text,
  description text,
  encoding frozen<video_encoding>,
  tags set<text>,
  title text,
  user_id uuid,
  PRIMARY KEY ( )
) WITH CLUSTERING ORDER BY ( );
```

5. Load 'videos_by_actor.csv' into the 'videos_by_actor' table using the COPY command.

```
COPY videos_by_actor
(actor,added_date,video_id,character_name,description,encoding,ta
gs,title,user_id) FROM 'videos_by_actor.csv' WITH HEADER = true;
```

6. Run a query to retrieve the video information for a particular actor (Tom Hanks, Denzel Washington, or see if your favorite actor is in there).

7. Try SELECTing just the actor and the added_date columns. Notice the order of added_dates.

Create a videos_by_genre table

8. In 'cqlsh', create a new table called 'videos_by_genre' which will support query Q2. We provided most of the CREATE TABLE for you except the PRIMARY KEY.

   ```
   CREATE TABLE videos_by_genre (
     genre text,
     added_date timestamp,
     video_id timeuuid,
     description text,
     encoding frozen<video_encoding>,
     tags set<text>,
     title text,
     user_id uuid,
     PRIMARY KEY ( )
   ) WITH CLUSTERING ORDER BY ( );
   ```

9. Load 'videos_by_genre.csv' into the 'videos_by_genre' table using the COPY command.

   ```
   COPY videos_by_genre
   (genre,added_date,video_id,description,encoding,tags,title,
   user_id) FROM 'videos_by_genre.csv' WITH HEADER = true;
   ```

10. Run a query to retrieve the video information for a particular genre (Future noir, Time travel).

    ```
    SELECT * FROM videos_by_genre WHERE genre = 'Musical' LIMIT 10;
    ```

11. Exit cqlsh.