**CHOOSING INDEXES**

Update you MYSQL setting as follows to be able to import large sql files

Edit php.ini and restart the wamp.

post\_max\_size=1280M

upload\_max\_filesize=1280M

max\_execution\_time = 3000 //increase time as per your server requirement.

Once you have imported the movie database, your main task is to write SQL queries (task below). However, you will find that even simple queries will take a long time; this is because the movie database you created lacks indexes on frequently accessed columns of its tables.

Hence, you will need to choose indexes for the movie database and create them using MySQL CREATE INDEX  SQL statement:

CREATE [UNIQUE] INDEX index\_name ON table\_name(col\_1, col\_2, ...);

Run \help create index  at the MySQL command line to get more detailed information. If given, the optional keyword UNIQUE says that no row can exist in the table with duplicate values of all the index columns. If you did not mark those constraints already in the CREATE TABLE statements, telling SQLite about them here may let it choose a faster way to create and maintain the index, or to read through the indexed table when running a query.

**Your goal is to choose up to about 10 indexes**, such that once the indexes are created, queries 1-5 below should not take more than 1 minute on a modern computer, while queries 6-11 should run in 5-10 minutes each (could be slower if your machine is too slow). You have some flexibility about which indexes to choose**, there is no absolutely perfect solution**.

Before each CREATE INDEX statement write a one-line comment with the reason for choosing each index (e.g. "in order to speed up the selection in query 7" or "in order to speed up the join in query 11").

Put all your code for part B (CREATE INDEX code) in a file called  indexes.sql.

For each question below, write a single SQL query to answer that question. Put your queries in a file called queries.sql. Add a comment to each query indicating with the question it answers and the number of rows in the query result.

1. List the first and last names of all the actors who played in the movie 'Officer 444'. *[~13 rows expected]*
2. List all the directors who directed a *'Film-Noir'*movie in a leap year. (You need to check that the genre is *'Film-Noir'*and simply assume that every year divisible by 4 is a leap year.) Your query should return director name, the movie name, and the year. *[~113 rows]*
3. List all the actors who acted in a film before 1900 and also in a film after 2000. (That is: < 1900 and > 2000.) *[~53 rows]*

How can this be? Actors can't live more than 100 years, right? Please find the explanation. For that you need to investigate a bit, perhaps run 1-2 additional queries (include them in your queries.sql file). Once you identify one logical explanation why some actors appear in movies more than 100 years apart, write it in your SQL code, as a comment to the SQL query; keep your answer below 1-2 sentences.

1. List all directors who directed 500 movies or more, in descending order of the number of movies they directed. Return the directors' names and the number of movies each of them directed.*[~47 rows]*
2. We want to find actors that played five or more roles in the same movie during the year 2010. Notice that **CASTS** may have occasional duplicates, but we are not interested in these: we want actors that had five or more *distinct* roles in the same movie in the year 2010. Write a query that returns the actors' names, the movie name, and the number of distinct roles that they played in that movie (which will be ≥ 5). *[~24 rows]*
3. Consider again all actors that had five or more roles in a movie in 2010 from the previous question. Return the names of these actors and their roles. Your answer should have one tuple for each combination of (actor, movie, role) - so if an actor has 10 roles in a given movie, there should be 10 tuples for that actor and movie. *Approx. 140 rows, 1-2 minutes.*
4. For each year, count the number of movies in that year that had only female actors. Recall the meaning of the universal quantifier: a movie without any actors is also a movie with only female actors (since there are no male actors in such a movie!). *Approx. 130 rows, 1-2 minutes.*
5. Now make a small change: for each year, report the percentage of movies with only female actors made that year, and also the total number of movies made that year. For example, one answer will be:

1990 31.81 13522

meaning that in 1990 there were 13,522 movies, and 31.81% had only female actors. You do not need to round your answer. *Approx. 130 rows, 1-2 minutes.*

1. Find the film(s) with the largest cast. Return the movie title and the size of the cast. By "cast size" we mean the number of distinct actors that played in that movie: if an actor played multiple roles, or if the actor is simply listed more than once in CASTS, we still count her/him only once. You may *not* assume that only one film has the largest cast.

*1 row, 10 minutes. The cast size is around 1300.*

1. A decade is a sequence of 10 consecutive years. For example 1965, 1966, ..., 1974 is a decade, and so is 1967, 1968, ..., 1976. Find the decade with the largest number of films.

*1 row, 5 minutes. If you were to count those movies in the decade then you would get around 457,500.*

1. The Bacon number of an actor is the length of the shortest path between the actor and Kevin Bacon in the "co-acting" graph. That is, Kevin Bacon has Bacon number 0; all actors who acted in the same film as KB have Bacon number 1; all actors who acted in the same film as some actor with Bacon number 1 (but not with Bacon himself) have Bacon number 2, etc. Count how many actors have Bacon number is 2.

*1 row, 5 minutes The number of actors is around 521,900. (Hint: correlated subqueries are SLOW; uncorrelated subqueries are faster).*