CSCI 620/Section 02/Mior, Introduction to Big Data, Spring 2195

Assignment 2 – SQL

Description

Create the following database schema in Postgres with appropriate types and populate it with IMDB data. Primary keys are in bold. Foreign keys are indicated with FK.

- Movie (**id**, type, title, originalTitle, startYear, endYear, runtime, avgRating, numVotes)
- Genre (**id**, genre)
- Movie_Genre (**genre**, **movie**)
  - genre FK Genre(id)
  - movie FK Movie(id)
- Member (**id**, name, birthYear, deathYear)
- Movie_Actor (**actor**, **movie**)
  - actor FK Member(id)
  - movie FK Movie(id)
- Movie_Writer (**writer**, **movie**)
  - writer FK Member(id)
  - movie FK Movie(id)
- Movie_Director (**director**, **movie**)
  - director FK Member(id)
  - movie FK Movie(id)
- Movie_Producer (**producer**, **movie**)
  - producer FK Member(id)
  - movie FK Movie(id)
- Role (**id**, role)
- Actor_Movie_Role (**actor**, **movie**, **role**)
  - (actor,movie) FK Movie_Actor (actor, movie)
  - role FK Role(id)

It is recommended that, to load roles, you disable the foreign keys of the database. (You should still ensure that the data is valid by removing any roles which do not satisfy the foreign key constraints).

Your tasks

1. Provide a program to load the IMDB data from the text files into the database. Your program needs to load the whole database in approximately twelve hours using commodity hardware.
   (10 points)

2. Provide a program to retrieve the following data from the database. Each will consist of a single SQL query and you need to report evidence that you retrieved what was expected. Report the time your queries took to run.
   (9 points per query)

   2.1. Number of invalid Movie_Actor relationships with respect to roles.
   2.2. Alive actors whose name starts with “Phi” and did not participate in any movie in 2014.
   2.3. Producers who have produced the most talk shows in 2017 and whose name contains “Gill”.
   2.4. Alive producers with the greatest number of long-run movies produced (runtime greater than 120 minutes).
   2.5. Alive actors who have portrayed Jesus Christ (look for both words independently).

3. Visualize and provide a brief explanation of the execution plan for each of the previous queries. (It’s okay if you don’t understand all the details.)

   (Hint: http://tatiyants.com/pev/ will allow you to copy and paste EXPLAIN output from Postgres to visualize query plans. You can also explore interactively to receive more information on the meaning of each step in the plan.)
   (10 points)

4. Provide relational algebra expressions for all the previous queries.
   (3 points per query)

5. Create indexes and full-text indexes where they are required. Document your decisions and provide the scripts to generate them. Re-run all the previous queries and report performance improvement and provide a brief explanation about why such an improvement including references to execution plans.
   (Hint: To reason about the performance gain you may need to restart the database server in order to clear query caches.)
   (20 points)