CSCI 620/Section 02/Mior, Introduction to Big Data, Spring 2235

Assignment 2 – SQL

Description

Create the following database schema in Postgres with appropriate types and populate it with Stack Exchange data. Primary keys are in bold. Foreign keys are indicated with FK. Note that some of the data used in the first assignment will not be required.

- Users (**Id**, AccountId, DisplayName, AboutMe, CreationDate, Reputation)
- Posts (**Id**, ParentId, OwnerUserId, AcceptedAnswerId, Title, Body, Score, ViewCount, CreationDate)
    - OwnerUserId FK Users(Id)
    - ParentId FK Posts(Id)
    - AcceptedAnswerId FK Posts(Id)
- Tags (**Id**, TagName)
- PostTags(**PostId**, **TagId**)
    - PostId FK Posts(Id)
    - TagId FK Tags(Id)
- Badges(**Id**, UserId, Name, Date)
    - UserId FK Users(Id)
- Comments(**Id**, PostId, Score, Text, CreationDate, UserId)
    - PostId FK Posts(Id)
    - UserId FK Users(Id)

Your tasks

1. Provide a program to load the Stack Exchange data from the XML files used in the first assignment into the database. Your program needs to load all the relevant data in approximately twelve hours using commodity hardware. As in the first assignment, you can ignore rows which would violate foreign keys. You may find it easier to insert all rows without foreign key constraints, delete any which would violate the foreign keys, and then add the foreign key constraints at the end. **(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 (e.g. example rows returned by the query). Report the time your queries took to run. **(9 points per query)**

    2.1. Names of the top 10 most popular badges earned by users within a year of creating their accounts.

2.2. Display names of users who have never posted but have a reputation greater than 1,000.
2.3. Display name and reputation of users who have answered more than one question with the tag “postgresql”.
2.4. Display name of users who posted comments with a score greater than 10 within the first week of creating their accounts.
2.5. The tag names of the tags most commonly used on posts along with the tag “postgresql” and the count of each tag.

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, but you should attempt to provide a reasonable explanation of each step. This may require some research to understand the meaning of a step in the plan. **(10 points)**

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

4.1. Display names of users who have commented but never posted.
4.2. Display names of users who have not made any post with the tag “postgresql”
4.3. Display names of users who have commented on any post in 2017 and whose name contains “John”.
4.4. The title of posts with a score greater than 1,000 posted by users within the first year of creating their account.
4.5. Users who have made posts with both the tag “postgres” and the tag “mysql”.

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)**

(continued on next page)