Hussain Hussain, Adrian Spataru
Based on previous exercises from Prof. Matthias Böhm
Graz University of Technology
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science

2. Data Management WS2022: Exercise 02 – Queries and APIs

Published: November 07, 2022
Deadline: December 04, 2022, 11.59pm

This exercise on query languages and APIs aims to provide practical experience with the open-source database management system (DBMS) PostgreSQL, the Structured Query Language (SQL), and call-level APIs such as ODBC and JDBC (or their Python equivalents). The expected result is a zip archive named `DBExercise02_<studentID>.zip`, submitted in TeachCenter.

2.1. Database and Schema Creation via SQL (3/25 points)

As a preparation step, setup the DBMS PostgreSQL (free, pre-built packages are available for Windows, Linux, Solaris, BSD, macOS) or use the provided Docker container. The task is to create a new database named `db<student_ID>` and setup the provided schema[1].

2.2. Data Ingestion via ODBC/JDBC and SQL (10/25 points)

Write a program `IngestData.py` in a Python (version 3.6 or above) that loads the data from the provided data files[2], and ingests them into the schema created in Task 2.1. Your code will be invoked as follows[3]:

```
./IngestData.py users.csv artists.csv festival.csv countries.csv \
 festival_users.csv playcount.csv festival_genre.csv friends.csv \
  <host> <port> <database> <user> <password>
```

All inserts should be performed via call-level interfaces like ODBC, JDBC, or Python's DB-API. We provide a template python file[4] where the SQL connection is set up. You can use the template for this task.

Partial Results: Source code `IngestData.py`.

[1] `https://github.com/tug-db/exercises/blob/main/2022WS/02_ExerciseQueriesAPIs/Supplements/schema.sql`
[2] `https://github.com/tug-db/datasets/tree/main/LFM-1b`
[3] The concrete paths are irrelevant. In this example, the `./` just refers to a relative path from the current working directory and the backslash is a Linux line continuation.
[4] `https://github.com/tug-db/exercises/blob/main/2022WS/02_ExerciseQueriesAPIs/Supplements/IngestData.py`

2.3. SQL Query Processing (10/25 points)

Having populated the created database in Task 2.2, it is now ready for query processing. Create SQL queries to answer the following questions and tasks (Q01-O06: 1 point, Q07/Q08: 2 points). The expected results per query will be provided on the course website. For any queries requiring you to return a real number, you should round the number to two decimal places.

- **Q01:** Which festivals are held in Austria? (return Festival.FestivalName)
- **Q02:** Which users listen to artists that play the genre 'Hip Hop'? (return Users.UID)
- **Q03:** For each festival, compute the number of standard tickets bought by users above the age of 30. (return Festival.FestivalName, TicketCount; sort in a descending order of the TicketCount)
- **Q04:** For each artist, compute the average age of users who listen to that artist. (return Artist.ArtistName, AverageAge)
- **Q05:** For each country with population of 5 million or more, report the artist with the highest total playcount by users of that country. In case of multiple artists with the highest playcount, return all. (return Country.CountryName, Artist.ArtistName, TotalPlaycount; sorted in ascending order of Country.CountryName)
- **Q06:** Find all the users who do have friends but exclusively from their country. (return User.UID, Country.CountryName)
- **Q07:** For each festival beginning in November 2022, find the artist with the top total playcount, that plays one of the genres featured on that festival. In case of multiple top artists, return all. (return Festival.FestivalName, Artist.ArtistName, TotalPlayCount, Genre.GenreName)
- **Q08:** We say that a user listens to a genre if they listen to any artist who plays that genre. For each Genre, what is the average number of friends that listeners of that genre have. (return Genre.GenreName, FriendCount)

Partial Results: SQL script for each query `Q01.sql`, `Q02.sql`, ..., `Q08.sql`.

2.4. Query Plans and Relational Algebra (2/25 points)

Obtain a detailed explanation of the physical execution plan of **Q02** using `EXPLAIN`. Then annotate how the operators of this plan correspond to operations of extended relational algebra.

Partial Results: SQL script `ExplainQ02.sql` with output and annotations in comments.