Univ.-Prof. Dr.-Ing. Matthias Boehm
Graz University of Technology
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMK endowed chair for Data Management

2. Data Management SS21: Exercise 02 – Queries and APIs

Published: April 7, 2021 (updates: Apr 17 fix returns Q07, Apr 08 disambiguate Q07/Q08)
Deadline: April 27, 2021, 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). The task is to create a new database named `db<student_ID>` and setup the normalized relational schema from Task 1.2. The schema should also include all primary keys, foreign keys, as well as `NOT NULL` and `UNIQUE` constraints. Your SQL script should be robust in case of partially existing tables and drop them before attempting to create the schema. If you do not want to use your own schema from Task 1.2, we will provide a recommended `CreateSchema.sql` by April 8.

Partial Results: SQL script `CreateSchema.sql`.

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

Write a program `IngestData` in a programming language of your choosing (but we recommend Python, Java, C#, or C++) that loads the data from the provided, denormalized data files [1], and ingests the data into the normalized schema created in Task 2.1. Please, further provide a script `runIngestData.sh` that sets up all necessary prerequisites, compiles and runs your program (with the provided arguments), and can be invoked as follows[2]:

```
./runIngestData.sh ./AthleteEvents.csv ./HostCities.csv ./NOCRegions.csv \
  <host> <port> <database> <user> <password>
```

It is up to you if you handle the necessary normalization of the input data via (1) program-local data structures (e.g., lookup tables like `City-CityID`) and call-level APIs, or (2) ingestion into temporary tables and transformations in SQL.

Partial Results: Source code `IngestData.*` and script `runIngestData.sh`.

[1] `https://github.com/tugraz-isds/datasets/tree/master/summer_olympics`
[2] 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.

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 (note that the sorting order only matters if explicitly requested).

- **Q01:** Obtain the detailed information of the athlete `Usain St. Leo Bolt`.
  (return name, gender, day of birth, height)
- **Q02:** Compute the distinct cities that hosted the Olympic games.
  (return distinct {city name, country name} pairs, sorted ascending by country name)
- **Q03:** List the athletes of team Austria 2012.
  (return name, gender, day of birth, sorted ascending by gender, day of birth)
- **Q04:** Compute for each sport type, the number of distinct event types as well as its last (i.e., maximum) year of occurrence.
  (return sport name, count, last occurrence, sorted descending by count)
- **Q05:** Determine the years in which female athletes whose names contain `Ledecky` won (in total) greater or equal than five gold or silver medals. (return year)
- **Q06:** How many medals of each type did `Michael Fred Phelps; II` win?
  (return medal, count, sorted descending by count)
- **Q07:** Determine the top-10 athletes that participated between 1948 and 2016 in the most event occurrences but never won a single medal. (return athlete key, name, day of birth, participation count, sorted descending by count, and ascending by name).
- **Q08:** Compute the athlete-centric Olympic medal table of 2016, which counts every medal awarded (i.e., multiple medals in team events). (return country name, NOC, #gold, #silver, #bronze, #total, sorted descending by {#gold, #silver, #bronze, #total})

Partial Results: A(n) SQL script (text file) for each query `Q01.sql`, `Q02.sql`, ..., `Q08.sql`.

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

Obtain a detailed `EXPLAIN` of the execution plan of **Q05**. Then annotate how the physical operators of this plan correspond to logical operators of extended relational algebra.

Partial Results: SQL script `ExplainQ05.sql` with text explain output and annotations.

A. Recommended Schema and Examples

As an alternative to your own relational schema from Exercise 1, we will provide a recommended schema (by April 8) as a fresh start for this exercise. Even when using the provided schema, please include it in the submission. Furthermore, we also provide an additional example Python script that demonstrates how to access PostgreSQL through a call-level interface from an application program. This script assumes that Python 3 and pip is already installed. Note that both the schema and Python scripts are made available on the course website.