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 SS2022: Exercise 02 – Queries and APIs

Published: April 03, 2022
Deadline: May 03, 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]. You may partially customize this schema but it should be in third-normal form; include all primary keys, foreign keys, as well as `NOT NULL` and `UNIQUE` constraints; and be robust in case of partially existing tables and drop them before attempting to create the schema.

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 data files[2], and ingests them into the schema created in Task 2.1. Please, further provide a script `runIngestData.sh` (or `.bat`) that sets up prerequisites, compiles and runs your program, and can be invoked as follows[3]:

```
./runIngestData.sh ./Districts.csv ./Institutions.csv ./Streets.csv \
  ./PopulationByCitizenship.csv  ./PopulationByGender.csv \
  <host> <port> <database> <user> <password>
```

It is up to you if you perform necessary transformations of the denormalized input files via (1) program-local data structures (e.g., lookup tables like `Street-StKey`), or (2) ingestion into temporary tables and transformations in SQL. However, all inserts should be performed via call-level interfaces like ODBC, JDBC, or Python's DB-API.

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

[1] `https://mboehm7.github.io/teaching/ss22_dbs/CreateSchema.sql`

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

[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.

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 districts have the postal code `8051`? (return Districts.Name)
- **Q02:** Which institutions have an address on `Leonhardstraße`? (return Institutions.Name, Addresses.PostalCode, Addresses.StNumber)
- **Q03:** Compute, for each district, its relative area (in percent) of the total Graz area (sum of district areas). (return Districts.Name, relative area)
- **Q04:** Count, for each district, the number of streets that belong entirely to this district (filter out streets that belong to more than one district). (return Districts.Name, Districts.Area, street count; sorted descending by street count)
- **Q05:** How many distinct countries were represented (by people's citizenships) between `2010-01-01` and `2014-12-31` in each district? (return Districts.Name, country count; sorted descending by country count)
- **Q06:** Obtain the population count for all `N-EU` countries represented in `Jakomini` as of `2022-01-01`? (return Countries.Name, PopByCitizenship.PopCount; sorted descending by PopCount)
- **Q07:** Compute the top-10 countries (by people's citizenship) with the largest absolute change in total population count over time. (return Countries.Name, date maximum, maximum, date minimum, minimum, difference max-min; sorted descending by difference)
- **Q08:** Find all pairs of distinct districts that had at the same date, the same population count of the same gender (e.g. `Wetzelsdorf` and `Straßgang` both having `6970 males` as of `2008-04-01`). (return PopulationByGender.Date, Districts.Name 1, Districts.Name 2, PopulationByGender.Gender, PopulationByGender.PopCount)

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 **Q06** using `EXPLAIN`. Then annotate how the operators of this plan correspond to operations of extended relational algebra.

Partial Results: SQL script `ExplainQ06.sql` with output and annotations in comments.

A. Recommended Schema and Examples

Please include—even if unmodified—the schema (see Task 2.1) into your 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 are already installed. Note that the schema, Docker container, Python scripts, and expected results are made available on the course website.