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

2. Data Management SS2020: Exercise 02 – Queries and APIs

Published: April 07, 2020
Deadline: April 28, 2020, 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), as well as 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 1.3, with the following small modifications. **Person name aliases as well as conference editors are ignored in this exercise.** The schema should also include all primary keys, foreign keys, as well as `NOT NULL` and `UNIQUE` constraints. Additionally, please ensure that a single paper references either a conference or journal, but not both. 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 1.3, we will provide a recommended `CreateSchema.sql` by April 10.

Partial Results: SQL script `CreateSchema.sql`.

2.2. Data Ingestion via ODBC/JDBC and SQL (8/25 points)

Write a program `IngestData` in a programming language of your choosing (but we recommend using languages such as Python, Java, C#, or C++) that loads the data, from the provided data files [1], into the schema created in Task 2.1. The program should be invoked as follows[2]:

```
IngestData ./confs.csv ./journals.csv ./persons.csv ./pubs.csv ./theses.csv \
  <host> <port> <database> <user> <password>
```

Note that the partially denormalized inputs require deduplication of countries and institutions. It is up to you if you handle this requirement via (1) program-local data structures (e.g., lookup tables like `Country-CountryID`) and call-level interfaces like ODBC/JDBC, or (2) ingestion into temporary tables and transformations in SQL.

Partial Results: Source code `IngestData.*`, and a script to compile and run the program.

[1] `https://github.com/tugraz-isds/datasets/tree/master/dblp_publications`
[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 (9/25 points)

Having populated the created database[3] in Task 2.2, it is now ready for query processing. Create SQL queries to answer the following questions and tasks:

- **Q01:** Where did the conference `SIGMOD 2019` (short name, year) take place?
  (return city and country)
- **Q02:** Which persons are affiliated with `Graz University of Technology`?
  (return name, website; sorted ascending by name)
- **Q03:** How many theses were published per year?
  (return year, count; sorted ascending by year)
- **Q04:** Which journal issues contained more than 70 papers?
  (return title, volume, issue, year; sorted descending by year and issue)
- **Q05:** Which cities hosted more than 2 conferences?
  (return city, country, count; sorted decreasing by count)
- **Q06:** Create a histogram that counts the number of papers for all groups of papers with equal number of authors.
  (return number of authors, number of papers; sorted ascending by number of authors)
- **Q07:** How many distinct theses and papers did persons currently affiliated with Austrian institutions publish? (return single count)
- **Q08:** How many persons did not publish any journal or conference paper in the year they published their PhD thesis? (return single count)

Partial Results: SQL script `Queries.sql`, with comments indicating the query numbers and the obtained results.

2.4. Query Plans and Relational Algebra (5/25 points)

Finally, pick two of the queries Q01 through Q08, analyze and explain them, as well as draw the related query trees.

- Obtain a detailed explanation of the physical execution plan using `EXPLAIN`. Then annotate how the operators of this plan correspond to operations of extended relational algebra.
- Draw the query tree by hand or with a tool support[4].

Partial Results: SQL script `ExplainQueries.sql` (with comments describing the relationship to relational algebra) and two images, one for each query tree `Q<X>.jpg`.

[3] If you did not succeed doing Task 2.2, leave out the results and provide the plain SQL queries.
[4] You can obtain a visualization of the physical execution plan by using pgAdmin's visual explain tool.

2.5. Extra Credit (5 extra points)

Given the populated database from Task 2.2 and a basic understanding of query processing from Task 2.3, write a program `PrintPubs`—again, in a programming language of your choosing—that reconstructs and prints the publication list of a given person name. This list should contain all papers (each having an author list, title, conference/journal shortname, year, and pages) in reverse chronological order. Note that the reconstruction can be done via a SQL query or by the application program. The program should be invoked as follows:

```
PrintPubs <person_name> <host> <port> <database> <user> <password>
```

Partial Results: Source code `PrintPubs.*`, and a script to compile and run the program.

A. Recommended Schema and Examples

As an alternative to your own relational schema from Exercise 1, we will provide a recommended schema (by Apr 10) 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.