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. Database Systems SS19: Exercise 02 – Query Languages and APIs

Published: April 08, 2019 (last update: April 22)
Deadline: May 07, 2019, 11.59pm CET (original deadline: April 30)

This exercise on query languages and APIs aims to provide practical experience with the open source DBMS PostgreSQL, the Structured Query Language (SQL), as well as APIs such as ODBC and JDBC. The expected result is a zip archive named `DB_Exercise02_<student_ID>.zip`, submitted in TeachCenter.

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

As a preparation step, setup the database management system PostgreSQL (free, pre-built packages are available for Windows, Linux, Solaris, BSD, macOS). Then, the task is to create a new database `db<student_ID>`, and setup the normalized relational schema from Task 1.3. This should also include all primary key, foreign key, not null, and unique constraints. Note the SQL script should be robust in case of partially existing tables, in which case these tables should be dropped before attempting to create the schema.

Partial Results: SQL script `CreateSchema.sql`.

2.2. Data Ingestion via ODBC/JDBC and SQL (9/25 points)

Given the provided data files[1] for 1954-2010 and 2014 (which are denormalized and thus, exhibit redundancy), write a program `IngestData` (in your favorite programming language such as C, C++, Java, or Python) that loads the data in two phases into the schema, created in Task 2.1:

```
IngestData ./1954_2010_Squads.csv ./1954_2010_Matches.csv \
  ./1954_2010_Goals.csv <host>:<port>/<database>
IngestData ./2014_Squads.csv ./2014_Matches.csv \
  ./2014_Goals.csv <host>:<port>/<database>
```

Note that the denormalized input data and two phase loading process requires a clean handling of existing data. The two-phase ingestion aims to emulate real-world scenarios like periodic ingestion into a central data warehouse. It's up to you if you realize this state handling requirement via (1) program-local data structures and 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/soccer_world_cup_1954_2014/subsets

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 queries to answer the following questions and tasks:

- **Q01:** In which tournaments did `Austria` participate between `1954` and `2014` (inclusive)?
- **Q02:** Was there a country that participated but did not score a single goal in the worldcup 2014?
- **Q03:** Which player(s) shot the quickest goal(s) (by time of the game) in 2014?
- **Q04:** Which participating countries did never reach the play-off phase?
- **Q05:** With how many distinct clubs where players of `Germany` affiliated between `2002` and `2014`?
- **Q06:** How many players from `Sturm Graz` ever participated in a world cup tournament.
- **Q07:** Which player(s) shot more than two goals in a single game in 2014?
- **Q08:** Rank countries that won at least one tournament by the number of won tournaments in decreasing order.
- **Q09:** Create the top-10 list of teams for won matches in history.
- **Q10:** Construct the final group table for Group G of the 2014 tournament (matches, wins, draws, losses, goal difference, points), ranked by points and goal differences.

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

2.4. Query Plans and Relational Algebra (3/25 points)

Finally, pick two of the queries Q01 through Q10, and obtain an explanation of the physical execution plan (after running `ANALYZE` on the relevant tables). This should include a SQL query to return the plan, as well as a graphical representation of this plan. Furthermore, explain how the operators of the returned plan correspond to operations of extended relational algebra.

Partial Results: SQL script `ExplainQueries.sql` (with comments describing the relationship to relational algebra), and two images of the visual plan explanation.

A. Relational Schema as Input for Task 2.1

As an alternative to your own schema from Exercise 1, the following relational schema can be used as a basis for this exercise.

```
DROP TABLE IF EXISTS Tournaments, Countries, Stadiums,
   Hosts, Groups, Teams, Clubs, Players, Matches, Goals;

CREATE TABLE Tournaments(
   TYear SMALLINT PRIMARY KEY,
   PtsWin SMALLINT NOT NULL
);
```

```
CREATE TABLE Countries(
   Cid SERIAL PRIMARY KEY,
   Name VARCHAR(256) NOT NULL UNIQUE
);

CREATE TABLE Stadiums(
   Sid SERIAL PRIMARY KEY,
   Name VARCHAR(256) NOT NULL UNIQUE,
   Cid INT REFERENCES Countries NOT NULL
);

CREATE TABLE Hosts(
   TYear SMALLINT REFERENCES Tournaments NOT NULL,
   Cid INT REFERENCES Countries NOT NULL,
   PRIMARY KEY(TYear, CID)
);

CREATE TABLE Groups(
   Gid SERIAL PRIMARY KEY,
   Name VARCHAR(64) NOT NULL,
   TYear SMALLINT REFERENCES Tournaments NOT NULL
);

CREATE TABLE Teams(
   Tid SERIAL PRIMARY KEY,
   Cid INT REFERENCES Countries NOT NULL,
   Gid INT REFERENCES Groups NOT NULL
);

CREATE TABLE Clubs(
   Ncid SERIAL PRIMARY KEY,
   Name VARCHAR(256) NOT NULL UNIQUE,
   Cid INT REFERENCES Countries NOT NULL
);

CREATE TABLE Players(
   Pid SERIAL PRIMARY KEY,
   Name VARCHAR(256) NOT NULL,
   Pos CHAR(2) NOT NULL CHECK(Pos IN('GK','DF','MF','FW')),
   JNum SMALLINT NOT NULL,
   Ncid INT REFERENCES Clubs,
   Tid INT REFERENCES Teams NOT NULL,
   UNIQUE(Tid, JNum)
);

CREATE TABLE Matches(
   Mid INT PRIMARY KEY,
   HomeTid INT REFERENCES Teams NOT NULL,
   VisitTid INT REFERENCES Teams NOT NULL,
   HomeScore SMALLINT NOT NULL,
   VisitScore SMALLINT NOT NULL,
   MatchDate DATE NOT NULL,
```

```
   MatchType VARCHAR(64) NOT NULL,
   Sid INT REFERENCES Stadiums NOT NULL
);

CREATE TABLE Goals(
   Gid SERIAL PRIMARY KEY,
   Mid INT REFERENCES Matches NOT NULL,
   Pid INT REFERENCES Players NOT NULL,
   GOwn BOOL NOT NULL,
   GTime SMALLINT NOT NULL
);
```