Exercises

Michail Kazimianec
Werner Nutt
Damiano Somenzi

2. Database Construction in PostgreSQL

These exercises involve the construction of a database in PostgreSQL and cover the following topics:

1. Usage of the PostgreSQL database management system;
2. Creation of tables as well as primary and foreign keys;
3. Filling tables with data and modifying the content of tables.

The target database for the exercises is a demographic database of three entity sets which correspond to countries, regions and cities. Figure 1 shows the ER diagram of this database. The entity types are COUNTRY, REGION, and CITY. REGION and CITY are weak entity types, which are in 1 to N relationships with their identifying entity types. Each country has a unique ISO code (e.g., `it` for Italy, `fr` for France, etc.). Within a country, regions are uniquely identified by their name. Region names do not necessarily uniquely identify a region, as can be seen by the regions of Galicia in Spain and Galicia, which is a part of Ukraine. Similarly, cities are uniquely identified by their name within a region.

To regions and cities statistic information is attached, namely, (i) the area of a region or city, (ii) the population of a region or city, and (iii) the date of the last poll when the population was surveyed. In addition, we store whether a city is the capital of a country.

1 Accessing a Database Using `psql`

The most basic interface to database management systems are command line tools that allow one to issue commands, set parameters, and pose queries. The command line tool for PostgreSQL is called `psql`.

Exercise 1 Connect to your PostgreSQL database from a Linux shell using the `psql` interactive terminal program. Do it several times, while changing the order of parameters for the command `psql`. Is it always possible to login successfully? If not, explain why.

Tips: The syntax of the `psql` command is:

`psql -h` ⟨*hostname*⟩ `-d` ⟨*database*⟩ `-U` ⟨*username*⟩

or simply

`psql -h` ⟨*hostname*⟩ ⟨*database*⟩ ⟨*username*⟩.

Useful to know: The `psql` environment has a number of internal commands that are not SQL commands. They begin with the backslash character, `\`. Some of these commands are listed in the welcome message. For example, you can get help on the syntax of various PostgreSQL commands by typing `\h`. To leave `psql`, use `\q`.

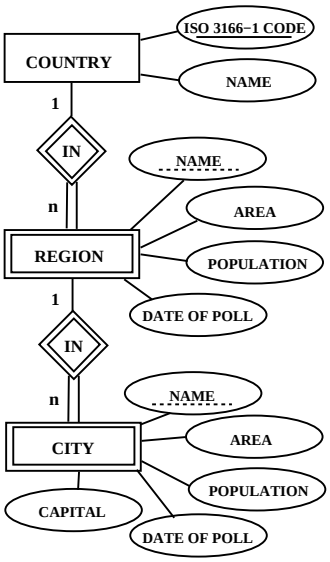

Figure 1: ER Diagram of the Target Database

2 Creating Tables

Exercise 2 Translate the entity types COUNTRY and REGION into relations and create tables for the resulting relations without specifying integrity constraints.

Tips: To create a table in a database use the `CREATE TABLE` command. The example below illustrates the command for the table corresponding to the entity type COUNTRY:

```
CREATE TABLE country (
  iso_code  CHAR(2), -- the code as char data type
  name      VARCHAR(30) -- the name as varchar data type
);
```

Use `\d` to view the schema of the current database and `\d` ⟨*table_name*⟩ to view the schema of a table.

PostgreSQL has a number of data types which can be assigned to attributes. The key data types for the example database are listed in Table 1.

Data Type	Description
`CHAR(...)`	fixed-length character string
`VARCHAR(...)`	variable-length character string with limit
`REAL`	single precision floating-point number
`INTEGER`	signed four-byte integer
`DATE`	calendar date (year, month, day)
`BOOLEAN`	two possible values – TRUE or FALSE

Table 1: Key Data Types for the Target Database

Useful to know: You can type your SQL command into the `psql` environment breaking the lines. Commands are not terminated by `psql` until the semicolon is typed. White spaces (i.e., spaces, tabs, and newlines) may be used freely in SQL commands. This allows you to type SQL commands aligned differently than in the example above, even all in one line. Two dashes are used to introduce a comment.
SQL is case-insensitive, that is, it doesn't matter whether you use upper or lower case letters in a command. We have adopted the convention write SQL keywords in upper case and user defined identifiers (like table and attribute names) in lower case.

3 Primary and Foreign Keys

Exercise 3 Alter the tables for the entity types COUNTRY and REGION so that they include primary and foreign keys.
Create a table for the entity type CITY, including all integrity constraints specified by the ER diagram.

Tips: Integrity constraints can be added to an existing table with the `ALTER TABLE` command. The following example shows how to add a foreign key to the `region` table:

```
ALTER TABLE region ADD FOREIGN KEY (iso_code)
                       REFERENCES country(iso_code)
```

Integrity constraints can also be directly specified by the `CREATE TABLE` command. The example below creates a table `country` with a primary key:

```
CREATE TABLE country (
  iso_code  CHAR(2) PRIMARY KEY,
  name      VARCHAR(30)
);
```

4 Inserting Records into Tables

Exercise 4 Add a record for the city of Rome, which is the capital of Italy and is located in the Lazio region. All the missing data, such as population or area, must remain unspecified in the database.

Exercise 5 Add a record for the city of Paris, which is the capital of France and is located in the Ile-de-France region.

Consult the tables above to find all the possible information to be included into your records.

Name	Country	Area (km^2)	Population	Date of Poll
Lazio	Italy	17208.0	5,304,778	2006-01-01
Ile-de-France	France	12011.9	11,226,000	2003-01-01

Table 2: Regions

Name	Region	Area (km^2)	Population	Date of Poll
Paris	Ile-de-France	86.9	2,142,800	2004-01-01
Rome	Lazio	1284.9	2,547,677	2006-01-01

Table 3: Cities

Tips: The following example command adds information about Italy to the `country` table:

```
INSERT INTO country (iso_code, name) VALUES ('it', 'Italy');
```

To view all the records of a table use

```
SELECT * FROM ⟨table_name⟩;
```

5 Modifying and Deleting Records

Exercise 6 Modify the record of the Ile-de-France region in the database so that it contains the current population of 11,291,020 people.

Exercise 7 Delete the record of the Ile-de-France region from the database.

Tips: The following example command modifies the area of the city of Rome:

```
UPDATE city SET area = 90.0 WHERE name = 'Rome';
```

This command deletes Rome from the database:

```
DELETE FROM city WHERE name = 'Rome';
```

6 `CHECK` and `NOT-NULL` Constraints

Exercise 8 Modify the table declarations of your database such that (i) each region and each city entry has an area and a population figure, which is greater than 0; (ii) for each region and city the date of the last poll is present.

Tips: The following command adds a `NOT NULL` constraint for the `name` column of the **country** table:

```
ALTER TABLE country ALTER COLUMN name SET NOT NULL;
```

The following command adds a `CHECK` constraint for the `area` column of the `region` table:

```
ALTER TABLE region ADD CHECK (area > 0.0);
```

7 Using PostgreSQL Scripts

7.1 Scripts in PostgreSQL

Exercise 9 Update the database with the data from the script file

`demographic_data.sql`.

The file can be found on the Exercises web page of the course.

Exercise 10 Backup the schema and the data of the database. Backup schema and data must be stored in different archive files.

Tips: The syntax of `psql` to execute a script is as follows:

`psql -h` ⟨*`hostname`*⟩ `-d` ⟨*`database`*⟩ `-U` ⟨*`username`*⟩ `-f` ⟨*`file`*⟩.

Useful to know: You can use the `pg_restore` utility to restore a PostgreSQL database dumped by `pg_dump`.

7.2 The pgAdmin III Graphical Front-end

Exercise 11 Connect to your database using the pgAdmin III program. Drop all tables of the database and then restore them from your backup files using pgAdmin III.

Tips: Type `pgadmin3` in the Linux command line shell in order to start the program.

Useful to know: The tool pgAdmin III is designed to answer the needs of users, from writing simple SQL queries to developing complex databases. The graphical interface supports all PostgreSQL features and makes administration easy. The application also includes a syntax highlighting SQL editor. The program is available both for Linux and Windows platforms.