UV Distributed Information Management
Summer term 2021

Assignment 01

Summary:

Deadline: April 19, 2021, 11:55 pm (aka 23:55) CET.

Extended Deadline: April 26, 2021, 11:55 pm (aka 23:55) CET.

Submission: A compressed archive (e.g., a `zip`-file) of your Python code and the answers to the questionnaire.

Grading: 55% Python code, 45% answers (incl. meeting).

---

1 General Remarks

The purpose of this assignment is to get in touch with a wide-spread general-purpose database management system (DBMS) named *PostgreSQL*. PostgreSQL is an open-source database system that is easy to use and accessible using the Python programming language (i.e., through the `psycopg2` [1] module). Commands (for the `psql` command-line tool) and Python code are written in TrueType font, e.g., `SELECT x FROM t WHERE a = b`.

Please submit your final Python code and the answers to the questionnaire until April 19, 2021, 11:55 pm (aka 23:55) CET via Blackboard [2] (late submission until April 26, 2021, 11:55 pm CET). Furthermore, please keep in mind that the exams contribute 46% to your final grade, hence you need to submit at least one assignment (partially) to pass the course.

1.1 Support

If you have troubles understanding the assignment, please use one of the following communication channels to get help (in this order):

1. **Lecture:** mondays 08:00 - 10:00 am CET (exception: lecture-free periods).
2. **Slack:** `https://dbteaching.slack.com/archives/C01QQHPLFB2`.
3. **Email:** dkocher@cs.sbg.ac.at (as a last resort).

We recommend to start the assignment early in case you run into problems. The earlier you identify a problem the easier it is for the instructor and other students to provide help in time. **Remark:** Please notify the instructor as soon as possible if the assignment description is (partially) unclear or if there is another problem with the submission.

---

[1] `https://www.psycopg.org/` and `https://pypi.org/project/psycopg2/`
[2] `https://elearn.sbg.ac.at`

2 Assignment Description

This assignment can be split into 5 parts:

1. Installation of PostgreSQL (PSQL), cf. Section 2.1.
2. Create tables and fill tables with data, cf. Section 2.2.
3. Familiarize yourself with PSQL, cf. Section 2.3.
4. Write an example application in Python that uses the database, cf. Section 2.4.
5. Answer the questionnaire, cf. Section 2.5.

Only part 4 and 5 contribute to the overall grade of this assignment. We recommend to follow these steps and refer to the corresponding sections for details.

2.1 PostgreSQL Installation

The first step is to install a PostgreSQL server locally on your machine. This should be possible on almost all operating systems, but the specific steps may diverge for different operating systems. Once PostgreSQL is installed, you can create a database and import data. Sources on how to install PostgreSQL can be found in the supplementary material provided in Section 4. Although PostgreSQL also provides a graphical user interface named *pgAdmin*, we recommend to also install PostgreSQL's *Command-Line Tools*, which allow you to interact with the database system through the command line. In particular the `psql` command-line tool (aka *PSQL terminal* or *PSQL shell*) provides the most basic way to use the database. Contrarily, the *Stack Builder* is not required for this assignment.

During installation, PostgreSQL also creates a default database named `postgres`. Open the `psql` command-line tool to connect to it (indicated by the `postgres=#` in the command-line tool). If you are asked for a server, a database, a port, or credentials (username + password), use the following default configuration:

- Server: `localhost` (or leave blank)
- Database: `postgres` (or leave blank)
- Port: `5432` (or leave blank)
- Username: `postgres` (or leave blank)
- Password: The password you set during the installation.

All available databases can be viewed by executing the `\l` command (mind the backslash) in the `psql` command-line tool. If you are using a fresh PostgreSQL installation, you should see three databases: `postgres`, `template0`, and `template1`.

For this assignment, we create our own database named `assignment1`. This can be done by executing the following command in the `psql` command-line tool:

```
CREATE DATABASE assignment1;
```

Afterwards, the `\l` command should print an additional database named `assignment1`. As database for this assignment, please use either `postgres` or `assignment1`. In order to use the database `assignment1` for this assignment, you must connect to it *after* creating it (the creation itself does not connect to the newly created database; indicated by the fact that `postgres=#` is still displayed in your `psql` command-line tool). Therefore, please use the command `\c assignment1` to connect to the new database. After the connection is established, the command-line tool should display `assignment1=#` (instead of `assignment1=#`). **Remark:** The creation of a database can take some time, hence please wait for the command to finish.

2.2 Data Initialization

Once your database is ready, the database contains no data, i.e., is empty. By executing the `\d` command, the database will report that no relations (tables) are to be found. Therefore, we first have to populate the database with some data.

In the course of this assignment, we will use parts of the publicly available Internet Movie Database (IMDB) [3] . First, we have to download the data from our Nextcloud [4] . Secondly, we use the DDL of PostgreSQL to create the tables that will store the data. Finally, we fill the tables with data. The data will be provided through two plain files that are then imported such that their contents fill the respective tables.

We provide an SQL script (`create_db.sql`) that contains the statements that are required to create the tables. There are two possible ways to execute the statements. Either you execute the script directly from the `psql` command-line tool (using `\i create_db.sql`), or you copy and execute each statement separately using the `psql` command-line tool.

To fill the tables with data, we provide you with two plain files and recommend to execute the following two commands (one after another; mind the semi-colon at the end) in your `psql` command-line tool. **Remark:** Importing the files into the tables will take some time, hence please wait for the commands to finish.

```
\COPY titles FROM title.basics_no_header_array_format.tsv WITH DELIMITER E'\t';
\COPY names FROM name.basics_no_header_array_format.tsv WITH DELIMITER E'\t';
```

On some systems, the encoding must be specified explicitly when importing the data from file. If you run into problems when importing the files, please try the following commands:

```
\COPY titles FROM title.basics_no_header_array_format.tsv WITH DELIMITER E'\t' ENCODING 'UTF8';
\COPY names FROM name.basics_no_header_array_format.tsv WITH DELIMITER E'\t' ENCODING 'UTF8';
```

Afterwards, you can see all available tables by executing `\d`. There should be two tables in the database: `names` (of actors) and `titles` (of movies). You can also look at the schema of a particular table. For example, the command `\d names` shows the schema (and some additional information) of the table `names`.

2.3 PSQL Introduction

After the successful import of the data, try to further familiarize yourself with the `psql` command-line tool. We provide the following SQL queries that can be executed out of the box by entering them into the `psql` command-line tool (one after another; mind the trailing semi-colon):

Query Q1:

```
SELECT * FROM names WHERE primaryName = 'Christian Bale';
```

Query Q2:

```
SELECT * FROM titles WHERE primaryTitle = 'The Avengers' AND titleType = 'movie';
```

Query Q3:

```
SELECT primaryTitle FROM names, titles
WHERE names.birthYear = titles.startYear AND names.primaryName = 'Scarlett Johansson';
```

Query Q4:

[3] https://www.imdb.com/interfaces/

[4] https://kitten.cosy.sbg.ac.at/index.php/s/GStGW3Xpp49RkdD

```
EXPLAIN SELECT * FROM titles WHERE primaryTitle = 'The Avengers' and titleType = 'movie';
```

SELECT, FROM, and WHERE clauses have briefly been covered in the lecture. The * in the SELECT clause specifies that *all* columns of the table in the FROM are retrieved.

In query Q2, the WHERE clause consists of two conditions that are linked using the AND operation. This means that every tuple that is part of the result must satisfy both conditions. For example, a TV series that is named "The Avengers" is not found because it is not a movie.

Query Q3 *joins* the two tables based on the condition in the WHERE clause. A join links tuples of two (or more) tables. In this specific case, we link the year of birth of the actors (names.birthYear) with the year of publication of the movies (titles.startYear). Furthermore, we specify that only actors with the name "Scarlett Johansson" should be considered. Intuitively, this means we ask the database system to show all movies that started in the year of birth of Scarlett Johansson. In this case, the FROM clause contains two tables that are separated by a comma. Without the WHERE clause, this results in the Cartesian product, i.e., every row of the first table (names) would be linked with every row in the second table (titles). This can result in a very large result, therefore we restrict the join using the conditions in the WHERE clause.

The last query, Q4, is basically Q2 with a small extension: We put the EXPLAIN keyword[5] at the beginning of the query. EXPLAIN shows the so-called *query plan*, that is, information about the steps PostgreSQL plans to execute in order to determine the result (without executing the actual query). As an exercise (not part of this assignment), the other queries could also be extended with the EXPLAIN keyword.

2.4 Access the Data Using Python

Although the psql command-line tool can be used to execute queries, a database system is typically accessed by an application. Therefore, the fourth part of this assignment is to write a small Python application that executes the queries given in Section 2.3. We recommend to use the psycopg2 module (or driver) for Python to (a) establish a connection to your local database, (b) execute the queries and retrieve the results, and (c) close the connection to your local database.

Remark: Please do not confuse the psycopg2 module with the (relatively) new psycopg3 module for Python. We use to the psycopg2 module in combination with Python3.

Template Code There are many tutorials regarding the installation [6] and the usage of psycopg2 [7] in combination with PostgreSQL. Nonetheless, we provide a minimum template code in Python3 that can be used as a starting point.

Listing 1: Template code to access a database using psycopg2.

```python
#!/usr/bin/python3

import psycopg2 as pg2

if __name__ == "__main__":
  try:
    # Establish a connection to the database 'assignment1' with user 'dkocher'
    # and password 'mypw'.
    connection = pg2.connect("dbname='assignment1' user='dkocher' password='mypw'")
  except:
    print("Unable to establish a connection to {}".format('assignment1'))

  # Retrieve a so-called cursor in order to interact with the database.
```

[5] https://www.postgresql.org/docs/current/using-explain.html
[6] https://www.psycopg.org/ and https://pypi.org/project/psycopg2/
[7] https://wiki.postgresql.org/wiki/Psycopg2_Tutorial

```
  cursor = connection.cursor()

  try:
    # Send a simple SELECT query to the database.
    cursor.execute("""SELECT * FROM names WHERE primaryName = 'Christian Bale'""")

    # Retrieve the entire result of the query.
    # cursor.fetchone() would retrieve only the first tuple of the result.
    records = cursor.fetchall()

    # Print the retrieved result.
    for record in records:
      print("{}".format(record))
  except:
    print("Unable to execute simple SELECT query.")
  finally: # The finally-branch is always executed (independently of an exception).
    if cursor is not None:
      # Close the cursor.
      cursor.close()

    if connection is not None:
      # Close the connection.
      connection.close()
```

2.5 Questionnaire

The questionnaire contains questions about the assignment. These questions are potentially discussed during the after-assignment meetings. The questionnaire can be found in a separate textfile called `assignment1-questionnaire.txt`.

3 Submission

Please submit a single compressed archive (e.g., `.zip` or `.tar.gz`) that contains two files: (a) The code of your Python3 application and (b) the answers to the questionnaire.

Code Please submit a single Python3 file (`.py`) that contains the full code for this assignment, i.e., the connection to the database and the execution of the four queries. **Remark:** Please remove the database credentials (i.e., username and password) before you submit your code.

Questionnaire You can answer the questions directly in the textfile `assignment1-questionnaire.txt`. If you prefer to use a different application to answer the questions (e.g., Microsoft Word and the likes), you are welcome to do so. In any case, the submitted file must be in one of the following formats: `.txt`, `.pdf`, `.odt`, `.doc`, or `.docx`. **Remark:** The recommended formats are `txt` and `pdf`.

4 Supplementary Material

This section provides a list of pointers to material that may be helpful to solve the assignment.

- PostgreSQL: `https://www.postgresql.org/`
- Download PostgreSQL: `https://www.postgresql.org/download/`
- The full PostgreSQL documentation (in particular, Chapter 1 may be helpful): `https://www.postgresql.org/docs/current/`
- Other resources regarding installation and usage of PostgreSQL:
  - One of the many "Getting Started" guides: `https://www.youtube.com/watch?v=BLH3s5eTL4Y`

  - Installing PostgreSQL: https://www.postgresqltutorial.com/install-postgresql/
  - PostgreSQL on Windows: https://www.postgresql.org/download/windows/
- Documentation of the EXPLAIN statement: https://www.postgresql.org/docs/current/using-explain.html
- Python Modules: https://docs.python.org/3/installing/index.html
- The psycopg2 module: https://www.psycopg.org/
- Installation of the psycopg2 module for Python:
  - Official website: https://www.psycopg.org/install
  - The Python Package Index: https://pypi.org/project/psycopg2/
- The psycopg2 tutorial in the PostgreSQL wiki: https://wiki.postgresql.org/wiki/Psycopg2_Tutorial
- One of the many "Getting Started" guides: https://www.youtube.com/watch?v=2PDkXviEMD0