Universität Salzburg | Übungsleiter: Hütter/Kwitt/Resmerita

Datenbanken 1 – PS (501.073)

Projektabgabe – Teil 3 | Abzugeben bis **16.07.2020 (22:00)**

Im vorletzen Teil des Projekts stellen wir Ihnen die Datenbank aus den letzten Projektabgaben, sowie ein Python Programm `query.py` zur Verfügung. Mittels

1. `create.sql`
2. `pop.sql`
3. `drop.sql`

kann die Datenbank (1) erstellt, (2) befüllt, sowie (3) gelöscht werden. Weitere Details finden Sie im Abschnitt *Testen Ihrer SQL Statements auf Ihrem lokalen Rechner*.

Das Python Programm `query.py` ließt eine Datei von SQL Statements. Die einzelnen Queries sind durch Kommentarzeilen (`-- QUERY n`) voneinander getrennt.

Hier ein **Beispiel**:

```
python query.py sql_queries.sql 1 nickname=mynick
```

Die SQL Datei `sql_queries.sql` sieht dabei beispielsweise so aus:

```
-- QUERY 1
select nickname
from person
where nickname=%(nickname)s;
-- QUERY 2
select * from text;
```

Die Reihenfolge der Queries ist unerheblich, Sie können also auch Query 2 vor Query 1 angeben.

Das Programm `query.py` ließt zuerst den Namen der Datei mit SQL Statements (hier: `sql_queries.sql`), dann einen Integer Wert zwischen 1 und 12, der das Statement in der Datei `sql_queries.sql` identifiziert. Danach werden die Parameter in der Form `parametername=parameterwert` angegeben – siehe Beispiel. In den Queries werden die Parameter dann in der Form `%(parametername)s` verwendet.

Ihre Aufgabe ist es, die SQL Statements in eine Datei `sql_queries.sql` zu schreiben.
(**Bitte genau diesen Dateinamen verwenden**)

Folgend sind die **12** zu implementierenden SQL Anfragen aufgelistet. In den Kästchen finden Sie jeweils das geforderte Ausgabeformat.

1. Geben Sie die Nicknamen aller jener Personen aus, deren Vorname gleich `%(pattern)s` ist (Tipp: "... = `%(pattern)s`").

   ```
   nickname
   ```

2. Geben Sie die Nicknamen aller jener Personen mit Vornamen 'Jerry' aus, welche nach dem 01.01.1999 geboren wurden.

   ```
   nickname
   ```

3. Geben Sie alle jene Vornamen aus die häufiger als 2x vorkommen.

   ```
   firstname
   ```

4. Geben Sie das Geburtsdatum und den Vornamen der zwei jüngsten Personen aus.

   ```
   birthdate | firstname
   ```

5. Gruppieren Sie alle Postings nach Ort (also Attribut `location`) und zählen Sie die Postings. Ausgegeben werden soll die maximale Anzahl an Postings (über alle Orte betrachtet).

   ```
   maxNrOfPostings
   ```

6. Geben Sie die durchschnittliche Anzahl an Tags pro Posting aus.

   ```
   avgNrOfTagsPerPosting
   ```

7. Finden Sie die Nicknamen aller jener Personen, für die die Person mit Nicknamen '2coolLogic' "likes" vergeben hat. Zählen Sie die "likes" und geben Sie die Nicknamen sowie die Anzahl an "likes" in absteigender Reihenfolge aus.

   ```
   nickname | count
   ```

8. Was ist die minimale Passwort Länge die von Personen verwendet wird?

   ```
   minPwdLength
   ```

9. In Informationen ist die Informationsgrösse in **Bytes** angegeben. Wieviele Informationen gibt es die > 4.8 KB (KiloByte) gross sind.

   ```
   count
   ```

10. Geben Sie alle Links aus, die den Sub-string "wiki" beinhalten.

    ```
    link
    ```

11. Betrachten Sie alle Links die den Sub-string “wiki” beinhalten und zählen Sie wie häufig diese Links vorkommen. Geben Sie jenen Link aus welcher am häufigsten vorkommt.

```
link
```

12. Berechnen Sie die Altersdifferenz (in Jahren) zwischen der ältesten Person und einer Person die mittels Nicknamen spezifiziert wird, also mittels `nickname = %(pattern)s`. *Hinweis*: Verwenden Sie die Funktion `date_part` von Postgres (siehe Dokumentation).

```
AgeDiff
```

Testen Ihrer SQL Statements auf Ihrem lokalen Rechner

Hinweis: Die folgenden Statements und Pfade beziehen sich auf eine Beispielinstallation von Postgres. Im Programm `query.py` stellen wir die Verbindung zur Datenbank mit dem Python Modul `psycopg2` her. Unter Linux (bzw. Mac) können Sie dieses Modul beispielsweise mit

```
pip install psycopg2
```

oder (auf Debian-basierten Linuxdistributionen)

```
apt-get install python-psycopg2
```

installieren. Weitere Informationen finden Sie online unter:

- `https://wiki.postgresql.org/wiki/Using_psycopg2_with_PostgreSQL`
- `http://initd.org/psycopg/docs/`

Zum Testen ihrer SQL Statements installieren und starten Sie einen lokalen PostgreSQL Server. Entsprechende Anleitungen sind im Internet zu finden, für Debian-basierte Systeme z.B. `https://wiki.debian.org/PostgreSql`. Danach ist eine Datenbank zu erstellen, z.B. `mydb`. Weiters erstellen Sie die entsprechenden Tabellen und befüllen diese.

```
createdb mydb
psql -d mydb -f drop.sql
psql -d mydb -f create.sql
psql -d mydb -f pop.sql
```

Nun können Sie `query.py` wie folgt ausführen:

```
python query.py --connection-string "host='localhost' dbname='mydb'" sql_queries.sql 1 nickname=anick
```

Alternativ, können Sie auch den Source Code von `query.py` editieren und die Variable

```
default_conn_string = "host='localhost' dbname='mydb'"
```

entsprechend setzen; dann brauchen Sie den Parameter `--connection-string` NICHT mehr anzugeben.

Evaluierung

Wir evaluieren Ihre Lösung (automatisiert) gegen unsere Musterlösung, d.h., gegen die korrekten Tupel (Anzahl und Reihenfolge), sowie die korrekte Anordnung der Attribute.