Universität Salzburg | Übungsleiter: Schäler/Kwitt/Resmerita

Datenbanken 1 – PS (501.073)

Projektabgabe – Teil 3 | Abzugeben bis **05.07.2021 (22:00)**

Im letzten 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 10, 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 **10** (zehn) zu implementierenden SQL Anfragen aufgelistet. In den Kästchen finden Sie jeweils das **geforderte Ausgabeformat**.

1. Geben Sie die Anzahl der Personen aus.

   ```
   count
   ```

2. Geben Sie den Nicknamen aller Personen aus, deren Vorname den Buchstaben 'y' enthält. Anmerkung: Werden '%' Symbole in der Query verwendet, müssen diese *escaped* werden, nämlich mit einem zusätzlichen '%'.

   ```
   nickname
   ```

3. Berechnen Sie für jeden Monat die Gesamtanzahl an Postings. Die Monate sollen als Zahl kodiert sein, z.B. 2 für Februar und 12 für den Dezember. Sortieren Sie die Ausgabe nach Monat (aufsteigend).

   ```
   count | month
   ```

4. Geben Sie Vor- und Nachnamen alle Follower von ‚Isotiene' duplikatfrei aus.

   ```
   firstname | lastname
   ```

5. Geben Sie die Namen aller Personen aus, die ‚Isotiene' und ‚Evesonel' folgen.

   ```
   firstname | lastname
   ```

6. Bestimmen Sie den Nicknamen aller Personen, die noch nichts gepostet haben.

   ```
   nickname
   ```

7. Geben Sie eine Liste der 10 wichtigsten Postings aus. Die Wichtigkeit (`importance`) eines Postings berechnet sich aus der Summe der Retweets und Likes.

   ```
   nickname | postingid | importance
   ```

8. Geben Sie alle Postings aus, die häufiger geretweetet als geliked werden. Tipp: Joins eliminieren Tupel ohne Joinpartner, also z.B. ohne Likes

   ```
   nickname | postingid
   ```

9. Bestimmen Sie für jede Person ihre Reichweite. Die Reichweite einer Person *a* ist eine Zahl, die sich wie folgt berechnet: a) die Anzahl der Personen, die *a* direkt folgen, b) Anzahl der Personen die jemandem folgen, der *a* direkt folgt. Beide teile werden zum `influence` addiert.

   Beispiel: folgt *B* direkt *A*, ist *B* Teil der Reichweite von *A*. Folgt *C* direkt *B*, folgt *C* auch *A*. Gibt es eine weitere Person *D*, die *C* folgt (aber nicht *B* und *A*), ist *D* nicht Teil des Reichweite von *A*. Jede Person zählt nur einmal; man selbst zählt nicht als eigene Reichweite.

   ```
   nickname | influence
   ```

10. Bestimmen Sie die Person, die insgesamt für Ihre Postings, die meisten Likes erhalten hat.

    ```
    nickname | likes
    ```

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 gegen unsere Musterlösung, d.h., gegen die korrekte Anzahl (und Reihenfolge) der zurückgegebenen Tupel und die korrekte Anordnung der Attribute.