Universität Salzburg | Übungsleiter: Kwitt/Resmerita/Hütter

Datenbanken 1 – PS (501.073)

Projektabgabe – Teil 5 | Abzugeben bis **29.06.2018 (22:00)**

Im letzen 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 3 count=5
```

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 den Nicknamen aller Personen aus, die mindestens einen Text gepostet haben. Jeder Nickname soll nur einmal ausgegeben werden. Ordnen Sie das Ergebnis nach Nickname.

```
nickname
```

2. Geben Sie die Anzahl an Personen aus, die nach dem '1.1.1995' geboren sind.

```
count
```

3. Geben Sie die Location aller Postings der Person "Bearbugar" aus, welche mehr als `%(count)s` Likes haben. Ordnen Sie das Ergebnis nach Location.

```
location
```

4. Geben Sie den am häufigsten verwendeten Tag der Person mit Nicknamen `%(nickname)s` aus. Ordnen Sie das Ergebnis nach Tag.

```
tag | count
```

5. Geben Sie alle Posts ohne Likes aus. Ordnen Sie das Ergebnis nach Nickname.

```
nickname | postingid
```

6. Geben Sie die `%(nth)s`-jüngste Person aus. Anmerkung: Falls es mehrere Personen mit dem gleichen Geburtsdatum gibt, haben diese die gleiche Position in der Sortierreihenfolge und sollen alle ausgegeben werden.

```
nickname | firstname | lastname | birthdate
```

7. Geben Sie von allen Personen, deren Nickname dem Muster `%(personpattern)s` (LIKE-Syntax) entspricht, den Nicknamen der Person mit der *geringsten* Anzahl an Followern aus. Sollten mehrere Personen dieses Kriterium erfüllen, sollen alle Nicknamen (alphabetisch sortiert) ausgegeben werden.

```
nickname | followercount
```

8. Geben Sie die Primärschlüsselattribute (PostingID, Nickname) aller Postings aus, die sowohl Data als auch Text beinhalten. Ordnen Sie das Ergebnis nach Nickname und PostingID.

```
postingid | nickname
```

9. Geben Sie die Primärschlüsselattribute (PostingID, Nickname) aller Postings aus, deren Informationen in Summe größer als `%(size)s` sind und mit dem Tag `%(tag)s` versehen sind. Ordnen Sie das Ergebnis nach Nickname und PostingID.

```
postingid | nickname | totalsize
```

10. Geben Sie alle Tags (aufsteigend sortiert) der Postings mit den meisten Likes aus. Sollten mehrere Postings die gleiche Anzahl an Likes haben, sollen alle Tags dieser Postings ausgegeben werden. Etwaige Duplikate sollen eliminiert werden.

```
tag
```

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 3 count=5
```

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.