Universität Salzburg | Übungsleiter: Schäler/Kwitt/Resmerita

Datenbanken 1 – PS (501.073)

Projektabgabe | Abzugeben bis **08.07.2022 (23:59)** via Blackboard

Im SQL Projekt stellen wir Ihnen eine Datenbank 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
```

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

```
-- QUERY 1
select nickname
from person
where nickname='mynick';
-- 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.

Ihre Aufgabe ist es, die SQL Statements in eine Datei `sql_queries.sql` zu schreiben. Die Auswertung der SQL Statements erfolgt automatisch, daher (**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 Namen aller Weine und deren Erzeuger aus, die aus der Region 'Kalifornien' stammen.

   ```
   name | weingut
   ```

2. Geben Sie die Namen aller Weine aus, die zwischen 2000 und 2010 (jeweils inklusive) produziert wurden.

   ```
   name
   ```

3. Geben Sie für jeden Gutachter, beschrieben durch `vorname` und `name`, die Anzahl der durchgeführten Gutachten als Attribut `anzahl` und die durchschnittlich vergebene Punktzahl aus.

   ```
   vorname | name | anzahl | punkte
   ```

4. Erstellen Sie eine neue Tabelle (oder Sicht) `Rotweine_plus` mit dem Schema (`name,jahrgang,weingut`), die alle Rotweine der Jahrgänge vor 2015 enthält und deren `name` mit der Zeichenkette 'PLUS' beginnt.

   ```
   name | jahrgang | weingut
   ```

5. Geben Sie den Namen und das zugehörige Weingut aller Weine duplikatfrei aus, die mindestens ein Gutachten mit der Bestnote 10 Punkte erhalten haben.

   ```
   name | weingut
   ```

6. Geben Sie den Namen aller Weine und die Anzahl der Rebsorten mit denen dieser jeweils hergestellt wurde aus, sofern er a) aus mindestens zwei unterschiedlichen Rebsorten hergestellt wurden oder b) keine Informationen über die enthaltenen Rebsorten verfügbar sind. Setzten Sie im Fall von b) den Wert des Attributs `anzahl` auf 0.

   ```
   name | anzahl
   ```

7. Geben Sie alle Gutachten (`gid, wid, punkte`) aus, bei denen das Gutachten von einem Gutachter erstellt wurde, der (oder die) den Wein selbst erzeugt hat. Dies ist der Fall, wenn der Wert das Attributs `Wein.weingut` mit dem Attribut `Gutachter.weingut` übereinstimmt.

   ```
   wid | gid | punkte
   ```

8. Berechnen Sie für jeden Wein eine korrigierte Durchschnittspunktzahl. Die Korrektur erfolgt in dem Gutachten von Gutachtern des den Wein erzeugenden Weingutes nicht berücksichtigt werden. Geben Sie für jeden Wein zusätzlich die originale Durchschnittspunktzahl (ohne Korrektur) an. Ein Ergebnistupel besteht aus der `wid` und dem Namen des Weins, sowie der korrigierten und der originalen Durchschnittspunktzahl.

   ```
   wid | name | avg_korrigiert | avg_original
   ```

9. Geben Sie das Weingut mit den meisten erzeugten Weinen, inklusive der Anzahl der erzeugten Weine, aus. Gehen Sie davon aus, dass es nur ein solches Weingut gibt.

   ```
   weingut | anzahl
   ```

10. In die Weindatenbank haben sich Fehler wie folgt eingeschlichen. Viele Weine enthalten als Teil Ihres Namens eine Rebsorte, aus der sie hergestellt werden. Geben Sie alle Weine aus, bei denen die Rebsorte 'Riesling' Teil des Namens ist, aber die so angebende Rebsorte nicht als Tupel in der Relation Hergestellt_aus aufgeführt ist.

name

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. Als Referenzsystem verwenden wir PostgreSQL in Version ≥ 13.