Aufgabenblatt 3
SQL

- Abgabetermin: **Sonntag, 03.06.2018, 23:59**
- Zur Prüfungszulassung muss ein Aufgabenblatt mit mind. 25% der Punkte bewertet werden und alle weiteren Aufgabenblätter mit mindestens 50% der Punkte.
- Die Aufgaben sollen in Zweiergruppen bearbeitet werden.
- Abgabesytem unter
  `https://www.dcl.hpi.uni-potsdam.de/submit/`
  - ausschließlich pdf-Dateien *im A4-Format*
  - *eine Datei pro Aufgabe* namens Aufgabe-<aufgabenNr>.pdf
  - *jedes* Blatt beschriftet mit Namen

Vorbemerkung und Hinweise zur Bearbeitung

In dieser Übung verwenden wir eine lokale PostgreSQL Installation um Ergebnisse von SQL Anfragen direkt betrachten zu können. Hinweise zur Installation folgen auf der nächsten Seite.

Als Datensatz verwenden wir Daten, die aus der IMDB extrahiert wurden. Die Daten findet ihr unter `Lehrveranstaltungen\FG Informationssysteme\VL DBS I\Uebung 2018\imdb_database_dump.zip` als PostgreSQL dump. Eine Anleitung für die Installation von PostgreSQL und das Importieren der Daten folgt auf der nächsten Seite.
Hilfreiche/Weiterführende Links zu PostgreSQL:

- `https://www.postgresql.org/docs/9.5/static/index.html`
- `https://www.postgresql.org/docs/9.5/static/queries.html`
- `https://www.postgresql.org/docs/9.5/static/ddl.html`

Hinweise zum IMDb-Datensatz

- Die Tabellen `actor`, `actress`, `producer` und `genre` enthalten eine Spalte mit dem Namen "id". Hierbei handelt es sich um Zeilennummern und nicht um Werte, die etwa einen Film oder eine Person eindeutig identifizieren. Die Tabellen `actor` und `actress` speichern daher eher Rollen von Schauspielern in verschiedenen Filmen als Schauspieler selbst.
- Nimm an, dass Schauspieler, Schauspielerinnen und Produzenten eindeutig über ihre Namen identifiziert werden können.
- Nimm an, dass die "movie_id"-Attribute jeweils das "mid"-Attribut referenzieren. Eine Fremdschlüssel-Eigenschaft gilt allerding nicht automatisch in dem zur Verfügung gestellten IMDb-Datensample. Es gibt also z.B. Schauspieler und Genres, deren zugehörige Filme NICHT existieren!

Hinweise zur Bearbeitung der Aufgaben

- Benenne aggregierte Spalten so um, dass sinnvolle Spaltennamen ausgegeben werden.
- Anfragen auf Schauspielerinnen *und* Schauspielern sind explizit in der Frage formuliert. D. h. falls eine Frage nur *Schauspieler* erwähnt, soll auch nur die Tabelle `actor` angefragt werden.
- Wenn nach dem Ergebnis einer SQL-Anfrage gefragt ist, dann gibt maximal 15 Tupel und die Anzahl aller Tupel an.

PostgreSQL Installation

Für die Aufgabenblätter 3-5 benötigt ihr eine lokale Installation von Postgresql. Wir haben im folgenden einige Anleitungen zur Installation zusammengefasst. Sollte es bei euch zu Problemen bei der Installation kommen, versucht euch zunächst selbst zu behelfen (im Netz gibt es viele Anleitungen/Tutorials). Bei schwer lösbaren Problemen könnt ihr uns natürlich gerne kontaktieren. Für die Bearbeitung der Aufgaben braucht ihr lediglich die PSQL-Shell (Kommandozeile). Grafische Oberflächen sind nicht nötig (ihr dürft sie aber natürlich verwenden).

a) Hinweise zu Windows Setup und häufigen Problemen:

- Laden Sie den Installer z.B. von dieser Webseite runter:

  `https://www.postgresql.org/download/windows/`

  Während der Installation gibt es die Option, Stack Builder mit zu installieren. Das ist für diese Übung **nicht** notwendig.

  Für die Übung wird nur die SQL Shell "psql" benötigt. Nach Start der Shell muss Enter gedrückt werden, bis man die Passworteingabe erreicht. Danach sollte es folgendermaßen oder ähnlich aussehen wie in Abbildung 1

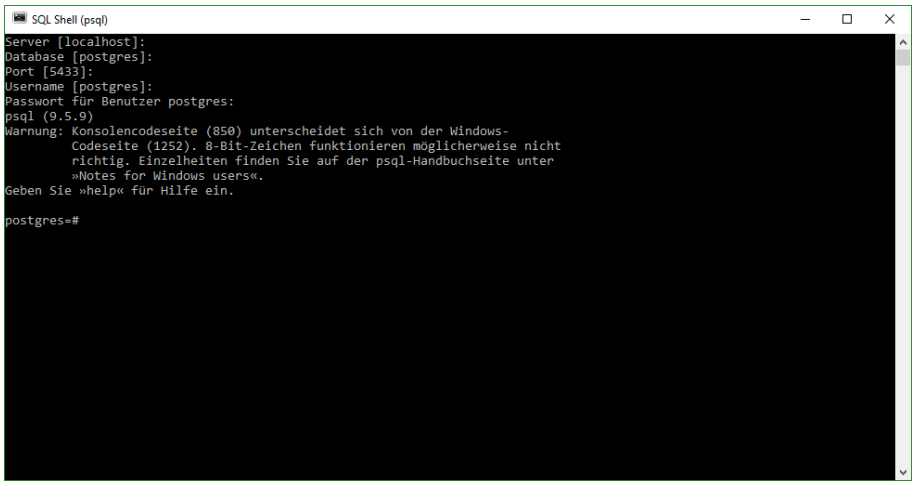

Abbildung 1: Windows SQL Shell nach erfolreicher Installation und Passworteingabe.

  Zugriffsprobleme können im Allgemeinen durch die Ausführung als Administrator gelöst werden. Da der Backslash für Kommandos reserviert ist, müssen Backslashes in Pfaden durch Slashes ersetzt werden.

b) Hinweise zu Linux Setup (Ubuntu 16.04 LTS):

- Eine Shell öffnen und *sudo apt-get install postgresql-contrib-9.5* ausführen
- Nun *sudo -u postgres psql* um PostgreSQL unter dem User postgres zu starten
- Nun sind sie mit der psql Shell verbunden. Sobald Sie eine Datenbank angelegt haben können Sie sich mit dieser direkt aus dem Terminal mit *psql <dbname>* verbinden.

c) Hinweise zu Mac Setup:

- Für die Installation wird empfohlen, den beliebten Packagemanager Homebrew zu nutzen, Ablauf:
  - Terminal öffnen
  - *brew -v* um zu überprüfen, ob Homebrew bereits installiert ist, ggf. installieren siehe `https://brew.sh/`

- *brew install postgresql@9.6* um PostgreSQL 9.6 zu installieren
- am Ende des ausgegebenen Installationslogs wird empfohlen, PostgreSQL zum Path hinzuzufügen; für die Bash-Shell sieht das dann beispielsweise so aus: *echo export PATH="/usr/local/opt/postgresql@9.6/bin:\$PATH" >> /.bashrc*
- Terminalsession beenden und eine neue starten, damit der neu hinzugefügte Path übernommen wird
- PostgreSQL-Installation ist damit abgeschlossen, ab jetzt kann man PostgreSQL folgendermaßen im Hintergrund starten und auch wieder stoppen
  - *brew services start postgresql@9.6*
  - *brew services stop postgresql@9.6* (beim Stoppen werden die angelegten Datenbanken *nicht* gelöscht)
- *psql postgres* um sich mit PostgreSQL zu verbinden
- *\q* um die Verbindung zu beenden

Import der Daten

a) Zip Datei `Lehrveranstaltungen\FG Informationssysteme\VL DBS I\Uebung 2018\imdb_database_dump.zip` herunterladen und entpacken

b) PostgreSQL Kommandozeile öffnen

c) Mittels *\i <pathToFile/imdb_database_dump.sql>* das SQL-Skript ausführen. Dies erzeugt die Datenbank 'dbs1_imdb', befüllt sie mit Daten und verbindet euch mit ihr. Das Kommando *\d* zeigt fälschlicherweise noch keine Tabellen an. Dies wird behoben indem ihr euch mittels *\c <dbname>* mit einer beliebigen anderen Datenbank verbindet und anschließend wieder zur IMDB-Datenbank wechselt.

d) Nun könnt ihr testen ob alles funktioniert hat, indem ihr überprüft ob die in Abbildung 2 gezeigten Queries die gleichen Resultate liefern. Lediglich der Owner wird bei euch anders sein.

```
dbs1_imdb=# \d
         List of relations
 Schema |   Name   | Type  | Owner
--------+----------+-------+-------
 public | actor    | table | leon
 public | actress  | table | leon
 public | genre    | table | leon
 public | laptop   | table | leon
 public | movie    | table | leon
 public | moviexml | table | leon
 public | producer | table | leon
(7 rows)

dbs1_imdb=# select count(*) from actress;
 count
-------
  3479
(1 row)

dbs1_imdb=# select * from movie where title = '4 aventures de Reinette et Mirabelle';
                  mid                  |                title                | year
---------------------------------------+-------------------------------------+------
 4 aventures de Reinette et Mirabelle (1987) | 4 aventures de Reinette et Mirabelle | 1987
(1 row)
```

Abbildung 2: Test Queries with Output.

Aufgabe 1: Deutsch → SQL

Nenne für jede der folgenden natürlichsprachlichen Fragen eine geeignete SQL-Anfrage und führe sie auf den Daten der IMDb aus. Gib auf deiner Abgabe die Anfrage und deren Ergebnis an.

a) Wieviele Schauspielerinnen gibt es? **2 P**

b) Gib alle Producer aus, die keine zugehörigen Einträge in der Filmtabelle haben (nach Producernamen sortiert)! Jeder Producer soll dabei nur einmal ausgegeben werden. **3 P**

c) Gib die Titel aller Filmpaare aus, in denen mindestens ein gemeinsamer Schauspieler mitspielt! Sortiere das Ergebnis nach dem Titel des zweiten Films. **3 P**

d) Gib die Namen der Personen (Schauspieler und Produzenten) an, die an der Serie "Edge of Night, The" beteiligt waren, und zwar einmal nach Mengen- und ein weiteres Mal nach Multimengensemantik. Hinweis: `UNION` benutzen. **4 P**

e) Erstelle eine Top-3 Liste der Filme mit den meisten Schauspielern und Schauspielerinnen! Sortiere entsprechend. Hinweis: Recherchiere hierzu die `FETCH FIRST` Klausel. **4 P**

f) Erstelle eine Top-3 Liste der Schauspieler und Schauspielerinnen mit den meisten Filmen! Sortiere entsprechend. **4 P**

Aufgabe 2: Deutsch → SQL

Die Aufgabenstellung aus Aufgabe 1 gilt weiter.

a) Gib alle Schauspieler an, die *auch* in Filmen des Genres „Action" spielen und deren Name mit „T" beginnt. **3 P**

b) Gib alle Schauspieler an, die *nur* in Filmen des Genres „Action" spielen und deren Name mit „T" beginnt. **4 P**

c) Gib die Namen aller Producer an, die 2001 Filme in beliebten Genres gedreht haben. Ein „beliebtes Genre" sei ein Genre in dem mindestens 200 Filme gedreht wurden. **5 P**

d) Formuliere *eine* Anfrage, die die Jahreszahl und die Anzahl der in diesem Jahr veröffentlichten Filme abfragt für

- das höchste vorkommende Jahr und
- das Jahr mit den meisten veröffentlichten Filmen

5 P

Aufgabe 3: Relationale Algebra → SQL

Formuliere die folgenden drei Anfragen der relationalen Algebra als SQL-Anfragen!

Verwendetes Schema:

- Stadt (StadtName, LandID, p1950, p2000, p2015)
  wobei `p1950`, `p2000` und `p2015` die Bevölkerungszahlen in diesen Jahren darstellen
- Land (LandID, Name, Kontinent, Hauptstadt, Bevoelkerung)
- Geographie (LandID, Landfläche, Wasserfläche, Küstenlänge, urbar)
  wobei `urbar` die urbare Fläche des Landes beschreibt

a) $\pi_{Name,Kontinent}(\sigma_{Bevoelkerung>200.000.000}(Land))$ **2 P**

b) $\pi_{Name}(\sigma_{(Bevoelkerung<2*p1950)\vee(Bevoelkerung<4*p2000)}(\sigma_{StadtName=Hauptstadt}(Stadt \bowtie Land)))$ **3 P**

c) $\pi_{Name}(Land \bowtie Geographie) -$
$\pi_{Name}(Land \bowtie (\sigma_{G1.urbar<G2.urbar}(\rho_{G1}(Geographie) \times \pi_{urbar}(\rho_{G2}(Geographie)))))$ **4 P**

Aufgabe 4: SQL → Deutsch

Gib natürlichsprachlich wieder, wonach folgende SQL-Anfragen suchen:

a)
```
WITH ProdAct AS
     (
     SELECT Prod.Name AS PName,
         Act.Name AS AName,
         COUNT(Act.MOVIE_ID) AS CountM
     FROM (
         SELECT NAME, MOVIE_ID FROM ACTOR
         UNION ALL
         SELECT NAME, MOVIE_ID FROM ACTRESS
     ) AS Act
     INNER JOIN MOVIE AS Mov ON Act.MOVIE_ID = Mov.MID
     INNER JOIN PRODUCER AS Prod ON Mov.MID = Prod.MOVIE_ID
     GROUP BY Prod.Name, Act.Name
     ORDER BY Prod.Name, Act.Name
)

SELECT ProdAct.AName, ProdAct.PName, CountM
FROM ProdAct,
     (
     SELECT AName, MAX(CountM) AS maxValue
     FROM ProdAct GROUP BY AName
     ) AS maxCount
WHERE ProdAct.AName = maxCount.AName AND
     ProdAct.CountM = maxCount.maxValue
```

5 P