SQL lab #1

DBM1

October 7, 2015

You have until the 13th of October, 11:59pm, to complete that lab and send your report to vincent.primault@insa-lyon.fr, as a single PDF file.

You have access to a database containing information about shows and actors, with 3 million shows and 3 million actors playing 4 million roles. The schema is already created and the tables are populated with real-world data coming from IMDB[1].

SQLDeveloper You will use SQLDeveloper as an Oracle client. Before being able to launch SQLDeveloper, you must edit the file located at *.sqldeveloper/4.1.0/product.conf* and add the following lines:

```
SetJavaHome /usr/java/jdk1.8.0_60
AddVMOption -Duser.language=en
```

Oracle server You will connect to an Oracle instance located on server *if-oracle*, with username "EtudIST", SID "IF" and provided password.

Exercise 1 Understanding the schema

This database contains the following tables: *Shows*, *Actors*, *Characters*, *Genres*, *Belongs* and *Plays*. Your first step is get knowledge about the database structure, by using the existing product. This process is called retro-engineering.

1. For each table, find out its attributes. For each attribute, give its name, type and whether or not it can be null. Find out the relations you expect to have between the different tables. The final result should be a figure representing the database schema, similar to the one presented in the SQL lecture (slide 7).
2. Explain how is stored a series and its episodes.
3. Can you give some pros and/or cons of storing movies, series and episodes in the same table?

Exercise 2 Single and multi-table queries

Now that you have some knowledge about the database structure, you can query it. Write the SQL query providing an answer to each of the following questions. You are free to choose the most meaningful attributes to select, but avoid using "SELECT *" everywhere. **Be careful about which entity the query is about (movies, shows, episodes, etc.). When talking about shows, it includes everything except episodes.** You may have to take into account performance considerations for the most difficult (i.e., last) queries.

1. Give the movies made in the 21st century.
2. Give the series still in production series having "Science" in their name.
3. Give the female actors whose first name has five letters ordered by their last name.

[1] `http://www.imdb.com/`

4. Give the list of non-suspended episodes (title, season, episode number) of your very own favorite series, ordered by season and episode.
5. Give the actors whose first name is Daniel playing in action movies.
6. Give the list of actors, the movies they played in and the associated character (hint: the character can be unknown).
7. Give the pairs of different movies made in the same year in which Nicolas Cage plays.
8. Propose an interesting query about "Game of Thrones", involving at least two different tables.

Exercise 3 Statistical queries

You can resume with some more SQL queries providing aggregated results.

1. Give the number of shows per genre.
2. Give the number of shows associated with no genre.
3. Give the five series with the greatest number of episodes.
4. Give the number of episodes per season of each series.
5. Give the average number of seasons per series.
6. Give the start year and end year of finished series with more than 20 episodes.

Exercise 4 Understanding existing queries

For each of the following query, formulate in english what it does. Explain how it works (e.g., the goal of each subquery).

1.
```sql
select first, last
from Plays
join Actors on Actors.id = Plays.actor
where character in (
  select character
  from Plays
  where character is not null
  group by character
  having count(1) >= 2
);
```

2.
```sql
select first, last
  from Actors
  where id in (
    select actor
    from Shows s1
    join Shows s2 on s1.series=s2.id and s2.title='The Big Bang Theory'
    join Plays on Plays.show = s1.id
    group by actor
    having count(1) = (
      select count(1)
      from Shows s3
      join Shows s4 on s4.series = s3.id
      where s3.kind = 'series' and s3.title = 'The Big Bang Theory'
        and s4.suspended = 0
    )
  );
```