CSC 370 — Database Systems
Fall 2016
Assignment No. 1. Version 1

Note 1 **This assignment is to be done individually**

Note 2 Working with other people is prohibited.

- Due date: Sept 20, 2016, at the beginning of the class.
- This assignment is worth 1% of your total course mark.
- Summit in paper your queries, their results, and their corresponding relational algebra
- Submit electronically in a single **text** file all the SQL queries.

Objectives

After completing this assignment, you will have experience:

- Writing Relational Algebra Expressions
- Creating basic SQL statements
- Interacting with postgresql using its command line interface.

Section 2. IMDB

A. In the *imdb* database I have created a set of tables that correspond to the IMDB data. Look at their names and schemas, and their foreign key constraints. These are some specifics about the data:

- Productions. Correspond to all types of productions. Use the field *attr* to determine the type of production (NULL for movies, *TV* for movies-made-for-TV (they are not considered movies in the queries below), *TV-series* for tv-shows, *TV-ep* for episodes of tv-shows, *V* for "videos" (I am not sure what this is) and *VG* for video games.
- Roles. When an actor/actress appears as different characters in the same movie, the field *character* contains them all, separated by "/". For example: the dual role of Dustin Hoffman in Tootsie is encoded as *Michael Dorsey/Dorothy Michaels*.
- Persons. Information about people. *pindex* is a field that allows to distinguish people with the same firstname/lastname.
- Directors. Who is a director of a movie. *pid* refers to the key of a record in *Persons*.

B. To connect to the database:

- To connect to the database you need to be connected within the faculty. You can do this by login in to one of the Linux servers in the Department, such as *linux.csc.uvic.ca*.
- To connect to the database you should use: host *studentdb.csc.uvic.ca*, database name *imdb*, your username is the same as in connex. Your password will be discussed in class. For example, this is how I would connect to the DBMS from *linux.csc.uvic.ca*:

```
dmg@linux6:~$ psql -h studentdb.csc.uvic.ca -U dmg imdb --password
Password for user dmg:
psql (9.3.14)
SSL connection (cipher: DHE-RSA-AES256-GCM-SHA384, bits: 256)
Type "help" for help.

imdb=# \q
```

- Connect to the DBMS using psql.
- Some queries will take some time to execute. I recommend you learn to use screen.
- In psql, you can:
  - List the tables using `\d`
  - List the schema of a table using `\d table`
  - You can output the results of a query using `\o filename`. You stop such output using `\o`
  - Read the manual for psql for more information.
- For this assignment, use only tables that are type *table* (output of `\d`)

Your task, should you choose to accept it

Answer the following questions, both in relational algebra, and SQL. **Relational algebra queries should match SQL**. For SQL queries provide the query and the result (if the result contains more than 15 rows, show only the first 10 and the **total number**). One query per question. Your query should only use the information provided in the question.

For this assignment you are only allowed to use projection, selection, union, intersection and difference. No other operation should be used. **You should not use cross products or joins**.

1 List the *id* and *year* of movies that contain the string *Harry Potter* in the title and were made after the *year* 2011. Hint: use the regular expression operator ~.

```
| id                                                                 | year |
|--------------------------------------------------------------------+------|
| Drunk Harry Potter (2013)                                          | 2013 |
| Harry Potter v. Voldemort (2012)                                   | 2012 |
| Harry Potter and the Escape from Gringotts (2014)                  | 2014 |
| Harry Potter and the Unlikely Collaboration (2013)                 | 2013 |
| Harry Potter Casts a revealing spell (2016)                        | 2016 |
| Harry Potter’s Parent Teacher Conference (2012)                    | 2012 |
| Hufflepuff: A Harry Potter Rap Parody (2015)                       | 2015 |
| Life After Hogwarts: Episode 1 - Harry Potter Goes to Therapy (2012) | 2012 |
| Nizard Harry Potter Rap (2012)                                     | 2012 |
```

$$\Pi_{id,year}\sigma_{title\ 'HarryPotter'\ AND\ attr\ IS\ NULL\ AND\ year>2011}P$$

```
SELECT id, year FROM productions
WHERE title ~ 'Harry Potter' AND
      attr is null AND
       year > 2011;
```

2 List the movies (productions where *attr* is null) in which Maryl Streep had a role (*id* equal *Streep, Meryl*) where made before 1980 (*year*). List its *id* and her *character*. Use the relation *roles*.

```
| id                                   | character     |
|--------------------------------------+---------------|
| Everybody Rides the Carousel (1975)  | Stage 6       |
| Julia (1977)                         | Anne Marie    |
| Kramer vs. Kramer (1979)             | Joanna Kramer |
| Manhattan (1979)                     | Jill          |
| The Deer Hunter (1978)               | Linda         |
| The Seduction of Joe Tynan (1979)    | Karen Traynor |
```

$$Before = \Pi_{id}\sigma_{attr\ IS\ NULL\ ANDyear<1980}P$$

$$\Pi_{id,character}\sigma_{pid='Streep,\ Meryl'\ AND\ id\ IN\ Before}$$

```
WITH Before AS
    (SELECT id FROM productions
     WHERE attr IS NULL and
          year < 1980)
SELECT id, character FROM roles WHERE pid = 'Streep, Meryl' AND
    id IN (TABLE Before)
```

3 One of the most famous roles of Clint Eastwood was "The Man with no Name" in the Dollar Trilogy, directed by Sergio Leone– pid *Leone, Sergio (I)*. But in each of the three movies he had a character name. For each production in which Eastwood acted, and Leone directed, list the *id* of the production, and the *character*, and the *billing* of his role.

$$Sergio = \Pi_{id}\sigma_{pid='Leone,Sergio(I)'}D$$

$$\Pi_{id,character,billing}\sigma_{pid='Eastwood,Clint'\wedge id\ in\ Sergio}R$$

```
WITH Sergio AS (
   select id from directors where pid = 'Leone, Sergio (I)')
SELECT id, character, billing FROM roles
WHERE pid = 'Eastwood, Clint' and
   id in (TABLE Sergio);
```

```
                   id                    | character | billing
-----------------------------------------+-----------+---------
 Il buono, il brutto, il cattivo. (1966) | Blondie   |       2
 Per un pugno di dollari (1964)          | Joe       |       1
 Per qualche dollaro in piu (1965)       | Monco     |       1
(3 rows)
```

4 Find the *id* of productions where both, Leonard Nimoy (*pid Nimoy, Leonard*) and Stephen Hawking (*Hawking, Stephen*) had roles.

```
| id                                                 |
|----------------------------------------------------|
| The Science of Star Trek (1995) (TV)               |
| How William Shatner Changed the World (2005) (TV)  |
```

$$\Pi_{id}\sigma_{pid='Hawking,\ Stephen'}R \bigcap \Pi_{id}\sigma_{pid='Nimoy,\ Leonard}R$$

```
SELECT id FROM roles WHERE pid = 'Hawking, Stephen'
INTERSECT
SELECT id FROM roles WHERE pid = 'Nimoy, Leonard'
```

5 Surprisingly, The Matrix Reloaded–*id The Matrix Reloaded (2003)*–only shared 5 actors/actresses with its predecessor, the Matrix–*id The Matrix (1999)*. List their *id* of the production, the *pid* of the actor/actress and their *character*. Hint: first find the *pids* of the persons who acted in the two movies, then use this information to retrieve the id of the production and the name of their role (from roles).

$$PidsBoth = \Pi_{id}\sigma_{id='The\ Matrix\ \mathrm{(1999)}'}R \bigcap \Pi_{id}\sigma_{id='The\ Matrix\ Reloaded\ \mathrm{(2003)}'}R$$

$$\Pi_{id,pid,character}\sigma_{id\ IN\ PidsBoth\ \wedge\ id\ IN\ ('id='The\ Matrix\ \mathrm{(1999)}','The\ Matrix\ Reloaded\ \mathrm{(2003)}')}R$$

```
WITH PidsBoth AS (
   SELECT pid FROM roles
              WHERE id = 'The Matrix (1999)'
              INTERSECT
              SELECT  pid FROM roles
              WHERE id = 'The Matrix Reloaded (2003)'
)
SELECT id, pid, character FROM roles
WHERE pid IN (TABLE PidsBoth)
          AND id in ('The Matrix (1999)', 'The Matrix Reloaded (2003)');

              id             |         pid         |  character
----------------------------+---------------------+-------------
 The Matrix (1999)          | Reeves, Keanu       | Neo
 The Matrix Reloaded (2003) | Reeves, Keanu       | Neo
 The Matrix (1999)          | Fishburne, Laurence | Morpheus
 The Matrix Reloaded (2003) | Fishburne, Laurence | Morpheus
 The Matrix (1999)          | Weaving, Hugo       | Agent Smith
 The Matrix Reloaded (2003) | Weaving, Hugo       | Agent Smith
 The Matrix (1999)          | Foster, Gloria (I)  | Oracle
 The Matrix Reloaded (2003) | Foster, Gloria (I)  | The Oracle
 The Matrix (1999)          | Moss, Carrie-Anne   | Trinity
 The Matrix Reloaded (2003) | Moss, Carrie-Anne   | Trinity
(10 rows)
```

Hints

1. When you don't know exactly the value of a field, you can use the regular expression operator `attr ~ 'regexp'` to find the regular expression *regexp* in the *attr*. For example:

   ```
   SELECT * FROM productions WHERE title ~ '^Star Wars'  and attr is null;
   ```

   will match movies with title that starts with the string *Star Wars*.

2. Remember to add a semicolon at the end of your sql statement. Postgres will process a statement only after it finds a semicolon.

3. Use the command line program `screen`. Do *man screen*.
4. Remember, your relational algebra should match your SQL.
5. psql will allow you to use emacs commands to traverse its history. You can also configure it to use vi commands. Read the documentation of psql.
6. In postgresql, usernames are case sensitive.
7. Once you are logged in, you need to type the semicolon at the end or every query. If you don't, psql will not process your query and way for more input. If your prompt is different from `<username>=`, then your query is not yet complete.
8. You can always use `Ctrl-c` to stop your query (either you are in the middle of typing it, or it is being processed.
9. To exist psql type `Ctrl-d`. It is bad practice to simply kill the terminal, since the dbms will still hold resources for a while until it can detect that the client is dead.