COMP3311 20T3 Database Systems

Assignment 3
Python, Psycopg2 and IMDB

Last updated: **Thursday 19th November 10:21pm**
Most recent changes are shown in red ... older changes are shown in brown.

[Assignment Spec] [Database Design] [SQL Schema] [Sample Outputs]

Aims

This assignment aims to give you practice in

- manipulating a moderately large database (IMDB)
- implementing SQL views to satisfy requests for information
- implementing PLpgSQL functions to satisfy requests for information
- implementing Python scripts to extract and display data

The goal is to build some useful data access operations on the Internet Movie Database (IMDB), which contains a wealth of information about movies, actors, etc. You need to write Python scripts, using the Psycopg2 database connectivity module, to extract and display information from this database.

Summary

Submission:	Login to Course Web Site > Assignments > Assignment 3 > [Submit] > upload required files, or on a CSE server, `give cs3311 ass3 RequiredFiles`
Required Files:	`xtras.sql helpers.py best rels minfo bio`
Deadline:	15:00 Monday 23 November
Marks:	**14 marks** toward your total mark for this course
Late Penalty:	0.1 *marks* off the ceiling mark for each hour late

How to do this assignment:

- read this specification carefully and completely
- create a directory for this assignment
- unpack the supplied files into this directory
- login to `grieg` and run your PostgreSQL server
- remove your Assignment 2 database
- set up a copy of the supplied database (must be called imdb)
- complete the tasks below by editing the files
  - `xtras.sql` ... put any views or functions here
  - `helpers.py` ... put Python helper functions here
  - `best` ... Python script to list best movies
  - `rels` ... Python script to show world releases for a Movie
  - `minfo` ... Python script to show cast+crew for a Movie
  - `bio` ... Python script to show biography/filmography of a Name
- submit these files via WebCMS (you can submit multiple times)

Details of the above steps are given below. Note that you can put the files wherever you like; they do not have to be under your `/srvr` directory. You also edit your SQL files on hosts other than `grieg`. The only time that you need to use `grieg` is to manipulate your database.

Introduction

The Internet Movie Database (IMDB) is a huge collection of information about all kinds of video media. It has data about most movies since the dawn of cinema, but also a vast amount of information about TV series, documentaries, short films, etc. Similarly, it holds information about the people who worked on and starred in these video artefacts. It also hold viewer ratings and crticis reviews for video artefacts as well as a host of other trivia (e.g. bloopers).

The full IMDB database is way too large to let you all build copies of it, so we have have created a cut-down version of the database that deals with well-rated movies from the last 60 years. You can find more details on the database

schema in the [Database Design] page.

Some comments about the data in our copy of IMDB: there seems to be preponderance of recent Bollywood movies; some of the aliases look incorrect (e.g. for "Shawshank Redemption"); the data only goes to mid 2019, so you won't find recent blockbusters.

Doing this Assignment

This section describes how to carry out this assignment. Some of the instructions must be followed exactly; others require you to exercise some discretion. The instructions are targetted at people doing the assignment on Grieg. If you plan to work on this assignment at home on your own computer, you'll need to adapt the instructions to "local conditions".

If you're doing your assignment on the CSE machines, some commands must be carried out on `grieg`, while others can (and probably should) be done on a CSE machine other than `grieg`. In the examples below, we'll use `grieg$` to indicate that the comand must be done on `grieg` and `cse$` to indicate that it can be done elsewhere.

The first step in setting up this assignment is to set up a directory to hold your files for this assignment.

```
cse$ mkdir /my/dir/for/ass3
cse$ cd /my/dir/for/ass3
cse$ unzip /home/cs3311/web/20T3/assignments/ass3/ass3.zip
Archive: /home/cs3311/web/20T3/assignments/ass3/ass3.zip
  inflating: best
  inflating: bio
  inflating: helpers.py
  inflating: minfo
  inflating: rels
  inflating: xtras.sql
```

Note that the database dump is quite large. Do not copy into your assignment directory on the CSE servers, because you only need to read it once to build your database (see below). If you're working at home, you will need to copy it onto your home machine to load the database.

The next step is to set up your database:

```
... login to Grieg and source env as usual ...
grieg$ dropdb imdb  ... if you already had such a database
grieg$ createdb imdb
grieg$ bzcat /home/cs3311/web/20T3/assignments/ass3/database/imdb.dump.bz2 | psql imdb
grieg$ psql imdb
... examine the database contents ...
```

Note the database contains non-ascii characters, and so you will need to make sure that your PostgreSQL server uses `UTF8` encoding (and corresponding collation) before it will load.

Loading the database should take less than 10 seconds on Grieg, assuming that Grieg is not under heavy load. (If you leave your assignment until the last minute, loading the database on Grieg will be considerably slower, thus delaying your work even more. The solution: at least load the database *Right Now*, even if you don't start using it for a while.) (Note that the `imdb.dump` file is 20MB in size; copying the compressed version under your home directory or your `srvr/` directory, and then decompressing it, is not a good idea).

If you have other large databases under your PostgreSQL server on Grieg or you have large files under your `/srvr/`*`YOU`*`/` directory, it is possible that you will exhaust your Grieg disk quota. In particular, you will not be able to store a copy of the MyMyUNSW database as well as the IMDB database under your Grieg server. The solution: remove any existing databases before loading your IMDB database.

If you're running PostgreSQL at home, the file `ass3.zip` contains copies of the required files, but not the database, to get you started. You will need to download the database dump file separately. If you copy `ass3.zip` and `imdb.dump.bz2` to your home computer, unzip it, and perform commands analogous to the above, you should have a copy of the IMDB database that you can use at home to do this assignment.

Think of some questions you could ask on the database (e.g. like the ones in the Online Problem-solving Sessions) and work out SQL queries to answer them.

One useful query is

```
imdb=# select * from dbpop();
```

This will give you a list of tables and the number of tuples in each.

Your Tasks

Answer each of the following questions by writing a Python/Psycopg2 script. You can add any SQL views or PLpgSQL functions that are used in your Python scripts into the file `xtras.sql` You can add any functions that you want to share among the Python scripts into the file `helpers.py`. If you want to use any other Python modules, make sure that they are available on Grieg; your submitted code has to run on Grieg using the Python installation there. You can change the indentation from the two-space indent in the templates.

Hint: use PostgreSQL's case-insensitive regular expression pattern matching operator (`~*`) for matching partial names and titles. If you do this, it has the added advantage that your command-line arguments can be in any case you like, and you can even put regular expression chars into them.

Some of the questions below require you to display crew roles. In the database, these are stored as all lower-case with underscores replacing spaces, e.g. "`production_manager`". When these are displayed, the first letter should be capitalised and each underscore should be replaced by a space character.

Q1 (3 marks)

Complete the script called "`best`" so that it prints a list of the top *N* highest-rating movies (default *N* = 10). The script takes a single command-line argument which specifies how many movies should appear in the list. Movies should be ordered from highest to lowest rating, and should be displayed as, e.g.

```
$ ./best 5
9.8 Randhawa (2019)
9.6 Fan (2019)
9.6 Mama's Heart. Gongadze (2017)
9.5 Ananthu V/S Nusrath (2018)
9.5 Family of Thakurganj (2019)
```

Within groups of movies with the same rating, movies should be ordered by title. If the user supplies a number less than 1, print a usage message and exit. For more examples of how the script behaves, see [Sample Outputs].

Q2 (4 marks)

Complete the script called "`rels`" so that it prints a list of the different releases (different regions, different languages) for a movie. The script takes a single command-line argument which gives a part of a movie name (could be the entire name). If there are no movies matching the supplied partial-name, then you should print a message to this effect and quit the program, e.g.

```
grieg$ ./rels xyzzy
No movie matching 'xyzzy'
```

If the partial-name matches multiple movies, simply print a list of matching movies, ordered by year of release and then by title, e.g.

```
grieg$ ./rels mothra
Movies matching 'mothra'
===============
Mothra (1961)
Mothra vs. Godzilla (1964)
Godzilla and Mothra: The Battle for Earth (1992)
Godzilla, Mothra and King Ghidorah: Giant Monsters All-Out Attack (2001)
```

If the partial name matches exactly one movie, then print that movie's title and year, and then print a list of all of the other releases (aliases) of the movie. If there are no aliases, print "*Title* (*Year*) has no alternative releases". For each alias, show at least the title. If a region exists, add this, and if a language is spcified, add it as well, e.g.

```
grieg$ ./rels 2001
2001: A Space Odyssey (1968) was also released as
'2001' (region: XWW, language: en)
'Two Thousand and One: A Space Odyssey' (region: US)
'2001: Odisea del espacio' (region: UY)
'2001: Een zwerftocht in de ruimte' (region: NL)
```

Movie releases should be ordered accoring to the `ordering` attribute in the `Aliases` table.

If an alias has no region or language, then put the string in the `extra_info` field in the parentheses. If it has neither region, language or extra info, just print the local title.

Note that if there are two movies with exactly the same original title, you will not be able to view their releases, since the title alone does not distinguish them. We consider this problem in the next question.

For more examples of how the script behaves, see [Sample Outputs].

Q3 (5 marks)

Complete the script called "`minfo`" so that it prints a list of cast and crew for a movie. The script takes a command-line argument which gives a part of a movie name (could be the entire name). It also takes an optional command-line argument, which is a year and can be used to distinguish movies with the same title (or, at least, titles which match the partial movie name).

```
grieg $ ./minfo
Usage: minfo 'MovieTitlePattern' [Year]
```

If there are no movies matching the supplied partial-name, then you should print a message to this effect and quit the program, e.g.

```
grieg$ ./minfo xyzzy
No movie matching 'xyzzy'
```

If the partial-name matches multiple movies, simply print a list of matching movies, the same as in Q2. If you need to disambiguate, either use a larger partial-name or add a year to the command line. If a longer partial-name and year still doesn't disambiguate, print a list of matching movies as above.

If the command-line arguments identify a single movie, then print the movie details (title and year), followed by a list of the principal actors and their roles, followed by a list of the principal crew members and their roles. The list of actors should be sorted according to the `ordering` attribute in the `Principals` table (i.e the biggest star comes first) and then by the name of the role they played if the `ordering` attribute is equal (i.e. one actor plays multiple roles). The list of crew members should also be sorted according to the `ordering` attribute in the `Principals` table and then by role name, if the `ordering` attribute is the same (e.g. one person has multiple crew roles).

For more examples of how the script behaves, see [Sample Outputs].

Q4 (6 marks)

Complete the script called "`bio`" so that it prints a filmography for a given person (`Name`), showing their roles in each of the movies they are associated with. The script takes a command-line argument which gives a part of a person's name (could be the entire name). It also takes an optional command-line argument, which is a year (their birth year) and which can be used to distinguish people with similar names.

```
grieg$ ./bio
Usage: bio 'NamePattern' [Year]
```

If the name pattern doesn't match anyone in the `Names` table, then you should print a message to this effect and quit the program, e.g.

```
grieg$ ./bio Smmith
No name matching 'Smmith'
```

If the name pattern matches more than one person, then print a list of the matching people, with the years they were born and died in parentheses afer the name.

```
grieg: ./bio rooney
Names matching 'rooney'
===============
Darrell Rooney (???)
Frank Rooney (1913-)
Jennie Rooney (???)
Mickey Rooney (1920-2014)
Nancy Rooney (???)
Rooney Mara (1985-)
Sharon Rooney (1988-)
```

If two people have the same name and birth year, put them in order of their `Name.id` value.

Note that some people do not have death years or birth years.

If the name pattern, optionally combined with a year, matches a single person, then you should print their name and birth and death years, followed by a list of all the films they have been a principal in. Films should be in chronological order; if there are multiple films in a given year, order them by title within the year. For each film, show first any acting roles they had, including the role they played, and then any production crew roles they had.

Note that one person could be both and actor and director in the same movie. If a person has multiple roles as an actor in one movie, then show the records in order of the role name. Similarly for multiple roles as a crew member; order by role name.

```
grieg$ ./bio 'spike lee'
Filmography for Spike Lee (1957-)
===============
She's Gotta Have It (1986)
 playing Mars Blackmon     -- acting role
 as Director               -- crew role
 as Writer
School Daze (1988)
 as Director
 as Writer
Do the Right Thing (1989)
 as Director
 as Writer
Mo' Better Blues (1990)
 playing Giant
 as Director
 as Writer
... etc. etc. etc. ...
```

For more examples of how the script behaves, see [Sample Outputs].

Style (2 marks)

Your programming style will be marked according to the following criteria

- consistent indentation (somewhat forced by Python)
- meaningful variable/function names
- effective abstraction (use of functions)
- efficient use of the database

There is a performance requirement in each of the tasks. Any task that takes longer than 2 seconds on any of our test cases will be penalised for that case, even if it eventually produces the correct result.

Submission and Testing

We will test your submission as follows:

- create a testing subdirectory containing your `xtras.sql` and Python scripts
- create a new database `imdb` and initialise it with `imdb.dump.bz2`
- run the command: `psql imdb -f xtras.sql` (using your `xtras.sql`)
- load and run the tests in the `check` script

Your submitted code must be *complete* so that when we do the above, your `xtras.sql` will load without errors. You should throgouhly test your scripts before you submit them. If you Python or SQL scripts generate load-time errors and/or have missing definitions, you will be penalised by a 2 mark administrative penalty.

Before you submit, it would be useful to test out whether the files you submit will work by following a similar sequence of steps to those noted above.

Have fun, *jas*