CS1003 Practical3: Databases

This practical is worth **30%** of the overall coursework mark. It is due on **Friday 19th of March**.

This practical will be a bit different due to the current COVID19 pandemic. However, just like a normal practical you should prepare in advance by studying this specification and reviewing relevant course material and do any Exercise work that was set. You will be required to use **Java** for the programming parts of this assignment and a relational database for the database components. Databases are discussed below.

Skills and Competencies

- developing skills at ER modelling
- designing an appropriate relational schema
- creating a relational database
- developing robust software to manipulate data stored in a database
- choosing an appropriate way to store and manipulate data in your program and in the database
- choosing appropriate classes and methods from the JDBC API
- identifying and dealing with possible error conditions
- testing and debugging
- writing clear, tidy, consistent and understandable code

Requirements

Since there is not going to be an exam, this practical is a bit longer than usual. There are 7 requirements for this practical, listed below.

Requirement 1 – Create an ER diagram

You are required to create and submit an ER diagram for the following scenario.
Scenario:

You are required to create a relational database for Movies, like IMDB or the OMDB system you used in P2. Your database should be able to model at least: **Actors** who have a name, the awards they have won, what films they have been in, their birthday, what awards they have won; **Movies** which have a title, a release date, running time, genre, the director, actors, plot, some ratings and awards.

Requirement 2 – Create a Relational Schema

You are required to design a relational schema corresponding to your ER diagram above. You should hand in file containing commands that can be run in the sqlite3 browser. The file should contain a data definition language description (DDL) for your database which should contain SQL statements of the form:

```
CREATE TABLE `Movie` (
        `Name`, …
);
```

Running this DDL should result in the appropriate database schema being created. **The file should also contain the necessary sqlite3 commands to create the database file.**

Requirement 3 – Create a Java program to create the Relational Schema

Next you need to integrate Java with your database. The first requirement is to write a Java program with a main in a class called *InitialiseDB*() that:

- checks whether a file for the database already exists in the database, and if so, deletes it
- creates the tables (by reading the DDL file you created for requirement 2)
- Tests that the action was successful and prints out "OK" if the database was initialised.

Requirement 4 - Create a Java program to populate the database

Write another Java program with a main in a class called *PopulateDB*() which populates your tables with data (you might use some data from the online database you used in P2). You could also do queries like "list Oscar actors" into Google – gives IMDB results, or try https://en.wikipedia.org/wiki/List_of_highest-grossing_actors, or https://www.imdb.com/list/ls053501318/ - top grossing actors etc.

You should save the data from the website or elsewhere in files on your local machine and read from them. All the files must be accessible to the *populateDB* class **using relative path names** (that is your class must be able to find the files when we run it). You must also provide the files as part of your submission if you go down this route.

Requirement 5 – Create a Java program to query the database

Write another Java program with a main in a class called *queryDB*() which queries your tables. The program should take a single integer as a parameter and execute the query with the corresponding number from the list below. The queries that should be performed are as follows:

1. List the names of all the movies in the database.
2. List the names of the actors who perform in some specified movie.
3. List the synopses of a movie with a specified actor in it and directed by some particular director.
4. List the directors of the movies that have a particular actor in them.
5. A (complex) query of your own choosing.
6. Another (complex) query of your own choosing.

Your program should list the output of the queries on standard output. For the queries , you should use JDBC to get the results from the database, i.e. your query program should not access files in the file system. You can use example code from StudRes as a starting point. You should clearly identify using in-line comments in your source code and in your report, which files or code fragments you have modified from that which was given out in class or which you have sourced from elsewhere. All the classes and methods should have **proper Javadoc comments** documenting what the classes and methods do.

Requirement 6 – Project report

You are required to hand in a PDF report containing the usual sections for Overview, Design & Implementation, Testing, Examples (example program runs), Evaluation (against requirements), Conclusions (your achievements, difficulties and solutions, and what you would have done given more time). Your report should also contain an accurate summary stating which files or code fragments you have written and any code you have modified from that which was given out in class or which you have sourced from elsewhere.

Requirement 7 – A short comparative essay

You are required to write a short essay (no more than 3 pages) contrasting the use of different storage technologies that you could have used to this practical. In addition to relational databases you might discuss

JSON, XML, CSV files, spreadsheets although the choice is yours. Your essay should be a critical appraisal of these different technologies and the pros and cons of each.

Database usage

Due to the COVID19 pandemic, this assignment will have to be done remotely. **If any of you do not have equipment to do this practical please let first-coord-cs@st-andrews.ac.uk as soon as possible**. You are required to use the SQLite database for this project. There are instructions about using SQLite in the online Lecture and Exercise materials so I will not repeat them here.

Setting Up

Unlike previous assignments, due to its nature we will not using the automated checker *stacscheck*. Also unlike previous assignments, you will be required to create more than one program with a main. All the java code developed should be in a CS1003-P3 directory. You will need to set up the appropriate CLASSPATHs so that the necessary jar files can be found. Please ensure that the jars that you use are included with your submission along with instructions on how to compile and execute the programs in your submission.

Testing

You are encouraged to create and document your own tests to test how your program deals gracefully with possible errors such as the input file being unavailable, or the file containing data in an unexpected format. To prove that you have tested the above scenarios, paste terminal output from each of your tests into your report in the Testing section. Be clear what each test demonstrates.

Ideally your testing report should include a table of the following form:

What is being tested	**name of test method**	**pre conditions**	**expected outcome**	**actual outcome**	**evidence**
Database initialisation	db_init()	Database is empty	Database should contain tables	Tables are initialised correctly	Print out of schema using Dbeaver is correct – see Fig X
Load actors	…	…	…	…	Etc.

Deliverables

Hand in via MMS to the *P3* coursework slot, a .zip file containing your entire CS1003-P3 directory which should contain:

- All your Java source files in the src directory as specified above.
- Instructions on how to run your program including the necessary classpath parameters (you may assume we have the necessary jars for MariaDB and sqlite).
- All data files that your program needs to run *InitialiseDB* these must be in an appropriate directory and accessible using relative paths when running the program.
- An ER diagram contained in a pdf, jpeg or png file.

- A text file containing your DDL definition and the sqlite database creation operations.
- A **PDF** report containing the usual sections for *Overview*, *Design & Implementation*, *Testing*, *Examples* (example program runs), *Evaluation* (against requirements), *Conclusions* (your achievements, difficulties and solutions, and what you would have done given more time). Your report should also contain an accurate summary stating which files or code fragments you have written and any code you have modified from that which was given out in class or which you have sourced from elsewhere.
- A PDF containing a short essay as described above.

Dependencies

Your program must only depend on the sqlite jar used in the Exercises **and no other jars; the use of other jars will be penalised by 4 grade points.**

Hints

Here are some hints to help you to develop your solution.

1. Use the *Dbeaver* interface to design and test your database and to create tables with the appropriate structure (it is free to download).
2. Examine the examples on StudRes. These example programs show how you can connect to a database, create a table, add data and run a simple query.
3. You could write a method String makeRowInsertStatement(String[] elements) that takes the required elements from one line of a data file, and creates an appropriate SQL command to insert these elements into the table.
4. You might call this method within a loop that iterates through each data file, so that an SQL insertion is executed for each row in the file.
5. Test your program on the small data files, and check that the tables have been appropriately created using Dbeaver.
6. Write a method executeQueryX which can execute an SQL query and pass the resulting ResultSet object to a void printResultSet(ResultSet rs) method which can print the results to the console. For the latter method, you will need two nested loops: an outer loop that iterates through the rows of the query result, and an inner loop that iterates through each column value for that row.

Marking

Your submission will be marked using the standard mark descriptors in the School Student Handbook:

https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors

However, more specifically, for this practical:

- 1-6: Very little evidence of work, software does not compile or run, or crashes before doing any useful work. You should seek help from your tutor.
- 7-10: A decent attempt which gets part of the way toward a solution, but has serious problems such as not compiling (at the low end), or lacking in robustness and maybe sometimes crashing during execution (at the high end).
- 11-13: A solution which is mostly correct and goes some way towards fulfilling basic requirements, but has issues such as not always printing out the correct values.

- 14-15: A correct solution accompanied by a good report, but which can be improved in terms of code quality, for example: poor method/OO structure, lack of comments, or lack of reuse.
- 16-17: A very good, correct solution to the main problem containing clear and well-structured code achieving all required basic functionality, demonstrating very good design and code quality, good method and class decomposition, elegant implementation of methods with reuse where appropriate, and accompanied by a well-written report and essay.
- 18-19: As above, but accompanied by an excellent report and essay.
- 20: Outstanding code decomposition and design and accompanied by an exceptional report with beautiful English and excellent comparative evaluation.

Lateness

The standard penalty for late submission applies (Scheme B: 1 mark per 8 hour period, or part thereof):

https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#lateness-penalties

Good Academic Practice

The University policy on Good Academic Practice applies:

http://www.st-andrews.ac.uk/students/rules/academicpractice/