Database Laboratory Project

2021-2022

Introduction

The project has two goals. First, familiarize yourself with the design, implementation, input and processing of data in a relational database (RD). Secondly, use JDBC technology to interface the DB with interfaces (Graphical User Interfaces) that you will implement in Java, in order to gain experience designing a more complete application.

A relational NW schema is available to you that covers **a portion** of the specifications listed in the description below. As part of the project, you should study the relational schema of the NW, expand it with additional fields and new tables and insert an appropriate number of records. Note that the database is similar to the one used in the lab exercises. In this way, a better understanding of the problem to be managed is ensured.

It is noted that the functional specifications may be incomplete or contain ambiguities and you may need to make some assumptions. Your assumptions should be realistic and clearly stated in your report.

Description

The BD concerns an **on-demand movie** service of a television content provider. The following functional requirements are covered by the relational schema you are given.

- For each **film ,** a unique code, the title, a description of the film, the year it was released, the language or languages in which it is available, its duration, the appropriateness mark and special features of the release are stored. • A code is stored for each **language** in which the movie is available in the system and her name.
- For each **actor** who participates in a film, a unique code, name and surname are registered in the system. • Each **actor** can participate in many **films** and one **film** can have many **actors.**

- Information about **movie categories** is maintained (unique code in system and genre, e.g., drama, comedy, etc.).
- No **category** belongs to another **category.** Also a **movie** can belong to several **categories** and obviously many movies can belong to the same **category .**

- The following information is kept for each **customer :** a unique code, first name, last name, email, address, whether they are an active customer or not and the date of registration.
- For design simplicity, **email** is his **username** customer in the service, while no password is used.

• For each **address** a unique code, street-number, region, city, postal code and telephone are stored. • A unique code, the name of the city and the country are kept for each **city** to which the city belongs.
• Correspondingly for the **country** a unique code and the name of the country.
• Each **customer** has an **address,** while the same **address** can belong to several **customers.** • Each **city** contains many **addresses** and each **country** many **cities.**

• The service offers its customers **a list** of movies available for viewing for a fee. The list **need not** be all inclusive
movies listed in the NW. From this customers can select and request to watch any movie. • For a **movie rental request,** a unique code, the rental date, the code that the movie has in the catalog and the customer's code are stored.

• Each **customer** can rent any **movie** (same or different) they want and each **movie** is **available** to all **customers** regardless of whether someone is watching it or not. The customer pays for each rental request. • A unique code, customer code, rental request code, payment amount and payment date are stored for each payment .

• One **payment** corresponds to one **movie rental request**

Part A: NW and SQL Planning

The TV content provider wants to upgrade the service so that, in addition to movies, it also offers series to registered users. In the first phase you will need to extend the DB to hold information about:

1. **The three categories of users who will connect to the NW:**
   i. **Customers** who subscribe to the service and have the ability to watch the available pay TV content. Information for this category already exists in the relational schema given to you.

   ii. **Employees of the provider (employees)** who update the available television content of the service and monitor the operation of the service.

   iii. **Administrators** of the system implementing the service.

   Each **customer** will be able to **subscribe** to watch only movies or only series or both. Depending on the **type of registration,** he will only have access to the corresponding content. The rental cost, depending on the type of registration, is as follows:

Recording Type	**Cost**
Movies Only	€0.4 per film
Series Only	€0.2 per series episode

Both	€0.3 per movie and €0.1 per series episode

The base shape should be modified to support the above requirement. For example, provision should be made as to how the customer will access the respective catalog (of films, series or both) depending on their subscription.

2. **Information about available series.** This information will be similar to the information of the films with additional elements that the series have, such as cycles, number of episodes per cycle, year of release of each cycle, etc.

   At least two tables should be added to store series, cycle and episode information. These tables should be appropriately linked to information that already exists in the NW, such as actors, languages in which the content is available and categories, e.g. drama, comedy. Tables, such as *inventory,* should also be created or modified to make the rows available to customers. Depending on your design, other tables involved in the content rental and payment process, such as *rental* and *payment, may need to be redesigned.*

3. **Maintaining a table of actions** (log). A table should be created to record the actions performed by any user (customer, employee, administrator) on specific tables through the application you will implement in part B. This table will be updated every time an input action is attempted, processing or deleting data and will record the username of the user, the date and time of the event, whether it was executed successfully or not, its type (insert, update, or delete) and the name of the table it concerned.

Wanted:

You will deliver **a report with all the documentation and all the SQL code.** The documentation pertains to Part A and Part B described below. Especially for part B you are required to explain its operation using screenshots. You will also submit a .zip file that includes the above, and all of the code for Part A and Part B, and all the files needed to demonstrate the code.

Instructions for submission can be found below. In detail, the requests are as follows:

1. Draw the revised NW. In your **chapter 1** report you will include:

A) The **relational diagram of the overall revised BD** (after the aforementioned extensions) and

B) description of all your design assumptions, new tables and modifications you made to existing tables.

2. Create the new tables, which resulted from the above design, in the NW by writing a set of **create and insert commands.** Register **a sufficient amount of data that will allow checking the correct operation of the DB** and the correct execution of queries, triggers and stored procedures that are subsequently requested. In your report in **chapter 2** you will include annotating the new tables you had to create to be able to implement the requirements below.

3. Create the following **stored procedures. In chapter 3 of your report you will provide the code and examples from the execution of the stored procedures:**

3.1 Stored procedure in which **a character ('mÿor 's'), a number (set) and two dates are given as inputs.** The result of the Procedure will be **the codes and full titles of movies** (if the first input argument is 'mÿ) or **the codes and full titles of series** (if the first input argument is 'sÿ) with the **most rentals in the time period specified by the two dates** in the input arguments. The **number** of movies or series that the result will contain will be determined by the number given as the second
input argument. The rentals for each series are calculated as the sum of the rentals of all episodes of all cycles.

For example if we give arguments ('m', 5, '2021-11-01', '2021-11-30') we should get the codes and titles of the first 5 movies with the most rentals in November 2021.

3.2 Stored procedure that is given a customer's email and a date. Returns the number of rentals it has made on this date. Rentals can be movies or series or both, depending on the type of subscription you have made to the service.

3.3 Stored procedure where it displays the income from payments per month. In each month the income from the films and the series will be calculated separately.

Hint: See the functions for handling MySQL time values
(https://dev.mysql.com/doc/refman/8.0/en/date-and-time functions.html#function_month)

3.4 a) Stored procedure in which two last names are given as input parameters and returns the first names, last names, and the number of actors whose last names are between the two input parameters. E.g. given 'Aco' and 'Alm' should return all actors whose last name
starts with Aco and all actors' last names up to those starting with Alm.

b) Stored procedure in which the last name of an actor is given as an input parameter and returns the first and last names of the actors with this last name. In case there are more than one, the number of actors is also returned.

Due to the large number of records you will need to create an additional index. Depending on the requested query you should choose the appropriate type of index and justify your choice in each case.

Hint: See creating its indexes and hash tables
MySQL (https://dev.mysql.com/doc/refman/5.7/en/create-index.html)

4. Create the following **triggers. In chapter 4 of your report you will provide the code and examples (screenshots) from the execution of the triggers :**

4.1. Triggers that will update the relevant action log table (log) for each **entry, update or deletion** action in the **rental** tables , **payment and any other table you create** that is involved in the process of watching a movie or series episode.

4.2. Trigger which for every third time the same user will request to see a movie or series episode within the same day will be 50% off. (Question 3.2 can be used)

4.3. Trigger which will prevent the change if any client tries to change items in their profile which they are not allowed to change.

Part B: GUIs

The DB will need a set of interfaces that will be used by the different categories of users who will have access to it. It is noted that the implementation of these interfaces may require some revisions to the design (eg adding some fields to tables).

1. Construct the three interfaces described next in Java using an IDE of your choice (Eclipse, NetBeans, etc.). In addition, you should create a home login page where each user will enter their email as username. The administrator should also be added to the DB as a user. Depending on the category of each user, the appropriate one of the following GUIs will be displayed:

The customer can:

- to see and edit his account details, but he will not be able to change his email.
- change its recording type (movies only, series only or both).
- to have access to the details of his rentals.
- access the list of available content by type of his registration.
- make a new rental and payment.

The employee can: • see the customer file and edit all their details, except the email.

- to have access to customer rental details.
- update the tables with information about actors, movies, series, languages, categories, addresses, cities, countries and the list of available movies and series.
- display the top 5 movies and series with the most rentals for the previous month from the date the query was run. (Question 3.1 can be used)

The **administrator** will be able to:

- create new accounts in the system for customers and employees.

- delete existing accounts for customers and employees. • be able to change an employee to an administrator and vice versa
- to display the total rental income per month. (Can be used question 3.3)
- be able to change the rental prices of movies and series

2. (1 unit). Where possible the GUI should limit user input to avoid errors and speed up processes (eg selecting from menus or lists). The options that will be available will come from the data that exists in the NW.

For example, if an employee wants to see movie and series rental prices by subscription type they should be displayed in a list. But since an employee is not allowed to change the values, they should appear as non-editable (locked) items. Unlike an administrator, who is allowed to make this change, these items will not be locked from being able to edit them.

In addition, the administrator can see the table of actions related to table or NW user. This will be done by entering (or selecting) a specific table or a specific user and time period.

Working in groups The project is designed for you to work in groups of 3 people. If you wish, you can work in groups of 2 with variation in the units of each subject compared to a group of 3. The variation exists because of the amount of work for a 2 person team. In addition, for a group of 2 people, there is a bonus if question B2 is implemented, while for a group of 3 people question B2 is mandatory and without bonus. Below is analyzed the differentiation of the score in the two cases of teams:

	THEME				
	A.1 A.2	*A.3*	*A.4*	*B.1*	*B 2*
Group of 3 people	0.5 0.5	3.5	1.5	3	1
Group of 2 people	0.5 1	3.5	1.5	3.5	**1 (bonus)**

A team of 3 people needs to answer all the questions correctly to achieve a score of 10. A team of 2 people if they answer the questions correctly from A1. up to B.1 achieves a score of 10, while subject B.2 gives 1 bonus unit and thus the maximum score for the project becomes 11.

The project is mandatory and constitutes 50% of your final grade in **the Database Lab.** The other 50% comes from exercises 1-5. It is not necessary to have a grade >5 in all of the above. As long as the overall lab grade is >4.7, as described in the original lab instructions.

It can be submitted to eclass until Monday **24/01/2022.** The oral exam will take place at the end of the exam on a date to be announced. The oral exam is MANDATORY for all team members. After the end of the oral exam, the final project grade will be calculated for each team member.

Delivery instructions The
submission of the project will be done through eclass in two phases:

A) **Submission of a .doc. or .pdf file** which will have **all the documentation and all the SQL code** and **will pass a similarity check.** The documentation concerns both A and B parts. Especially for part B you are required to explain its operation using screenshots.

B) **At the same time you will submit a .zip file** that includes the previous ones, and all the code for part A and B, and all the files required to demonstrate the code.

During the oral examination, each group **will demonstrate the project and its operation on their own computer** either remotely or up close. A relevant announcement will be made regarding the oral examination process, after the submission of the projects.