COMP7240 - Introduction to Database Concepts

Assignment 1: SQL

Due date: 11:55pm, 11 April 2021

This assignment will be marked out of 20. It will count for 20% of the final grade. Below you will find 2 questions to reach this score.

Instructions:

- This assignment should be done individually (**no group work**).
- A database namely `moviedb` is available on both the ANU VDI lab environment (Option 1) and the virtual machine (Option 2). You can open a command shell and then connect to the moviedb database by entering: "`psql moviedb`".
- You must submit two files: `mydatabase.sql` for Question 1 and `myqueries.sql` for Question 2 on Wattle before the due date. The template files are available in the folder "Assignment 1 " on Wattle. You need to enter your answers into the template files, and ensure that the answers are executable as described below:
  - For Question 1, it should be executable in your own database, e.g., "`u1234567=> \i mydatabse.sql`".
  - For Question 2, it should be executable in the database `moviedb`, i.e., "`moviedb=> \i myqueries.sql`".
- Late submission is not granted under any circumstance. You will be marked on whatever you have submitted at the time of the deadline. Please take careful note of deadlines and adhere to them. Of course, if you find yourself in a situation beyond your control that you believe significantly affects an assessment, you should follow the ANU's special consideration process. Note that, special consideration is the process by which we take extenuating circumstances into consideration during the marking of an assessment item, and it is not for requiring an extension.
- Plagiarism will attract academic penalties in accordance with the ANU guidelines. A student in this course is expected to be able to explain and defend any submitted assessment item. The course convener can conduct or initiate an additional interview about any submitted assessment item for any student. If there is a significant discrepancy between the two forms of assessment, it will be automatically treated as a case of suspected academic misconduct.

Good luck and enjoy the time you will spend on this assignment.

Question 1 **8 Marks**

A university manages car parking on its campus. The IT Service of the university has designed the following database schema and has implemented it in Postgres:

- USERS(UID, Name, Email)
- PERMITTYPE(PID, TypeOfPermit, Fee, Description)
- PARKINGPERMIT(PermitTypeID, UserID, IssueDate, ExpiryDate, VehicleRegNo, IsActive)

The table USERS contains the user information, where UID uniquely identifies a user. The table PERMITTYPE contains the information about different types of parking permits, which can be uniquely identified by PID. The table PARKINGPERMIT contains the parking permits issued to users, which has the primary key {PermitTypeID, UserID}.

The template file `mydatabase.sql` on Wattle contains the SQL statements which were used by the IT Service to create this database (no records being added into this database yet). However, there are some mistakes and issues in the design and implementation of this database. Your task is to answer the following questions by *adding SQL statements* into `mydatabase.sql` to fix the mistakes and issues.

Note that: (1) *Do not delete or change any existing SQL statements* that were written in `mydatabase.sql`, and (2) *Do not add any DROP TABLE statements* into `mydatabase.sql` (i.e., don't drop the original tables and create new tables).

1.1 For the table PARKINGPERMIT, can you ensure that the expiry date is after the issue date for any record to be inserted? Further, can you ensure that the default value of the attribute IssueDate is set to the date when a record is inserted? If you can, add your SQL statements into `mydatabase.sql`.

(1.5 Mark)

1.2 We consider that a record in the table PARKINGPERMIT is *valid* if and only if the record associates with a record in USERS via the attribute UserID and a record in PERMITTYPE via the attribute PermitTypeID. How can you ensure that every record to be inserted into PARKINGPERMIT must be valid? Add your SQL statements into `mydatabase.sql`.

(1.5 Mark)

1.3 The IT Service is required to ensure that the database must meet the requirement "each VehicleRegNo is associated with at most one active parking permit, regardless of its corresponding user(s)", where the attribute IsActive is used to indicate which vehicleRegNo is active for a user. Can you add your SQL statements into `mydatabase.sql` to help the IT Services achieve this?

(1 Mark)

1.4 The attribute Fee in PERMITTYPE is initially implemented as `INT`. Now the IT Service realises that this is not well designed because it does not allow to store fee that is a decimal number (e.g. 178.95) but does allow to store "negative fee" (e.g. -100). Can you

fix these issues by ensuring that Fee can only contain values between 0 and 100,000 which may have two decimal places? Add your SQL statements into `mydatabase.sql`.

(1.5 Mark)

1.5 It turns out to be essential to have a phone number for each user so that they can be contacted promptly. A valid phone number should be 10-digit numbers. Can you add an additional attribute PhoneNumber into USERS and also ensure that only a valid phone number (i.e., 10-digit numbers) can be entered into PhoneNumer for a user? Add your SQL statements into `mydatabase.sql`.

(1.5 Mark)

1.6 Suppose that there is a record in the table PERMITTYPE with 'Pay As You Go' being the value of TypeOfPermit. You want to decrease the parking fee of 'Pay As You Go' by 10%, which is effective from the time you make the change. How can you achieve this using a SQL statement? Add your SQL statement into `mydatabase.sql`.

(1 Mark)

Question 2 **12 Marks**

Consider the following database schema for a relational database `moviedb`:

MOVIE(title, production_year, country, run_time, major_genre)
`primary key`: {title, production_year}

PERSON(id, first_name, last_name, year_born)
`primary key`: {id}

DIRECTOR(id, title, production_year)
`primary key`: {title, production_year}
`foreign keys`: [title, production_year] $\subseteq$ MOVIE[title, production_year]
[id] $\subseteq$ PERSON[id]

WRITER(id, title, production_year, credits)
`primary key`: {id, title, production_year}
`foreign keys`: [title, production_year] $\subseteq$ MOVIE[title, production_year]
[id] $\subseteq$ PERSON[id]

SCENE(title, production_year, scene_no, description)
`primary key`: {title, production_year, scene_no}
`foreign keys`: [title, production_year] $\subseteq$ MOVIE[title, production_year]

ROLE(id, title, production_year, description, credits)
`primary key`: {title, production_year, description}
`foreign keys`: [title, production_year] $\subseteq$ MOVIE[title, production_year]
[id] $\subseteq$ PERSON[id]

APPEARANCE(title, production_year, description, scene_no)
primary key: {title, production_year, description, scene_no}
foreign keys: [title, production_year, scene_no]⊆SCENE[title, production_year, scene_no]
[title, production_year, description]⊆ROLE[title, production_year, description]

AWARD(award_name, institution, country)
primary key: {award_name}

MOVIE_AWARD(title, production_year, award_name, year_of_award,category, result)
primary key: {title, production_year, award_name, year_of_award, category}
foreign keys: [title, production_year] ⊆ MOVIE[title, production_year]
[award_name] ⊆ AWARD[award_name]

DIRECTOR_AWARD(title, production_year, award_name, year_of_award, category, result)
primary key: {title, production_year, award_name, year_of_award, category}
foreign keys: [title, production_year] ⊆ DIRECTOR[title, production_year]
[award_name] ⊆ AWARD[award_name]

WRITER_AWARD(id, title, production_year, award_name, year_of_award, category, result)
primary key: {id, title, production_year, award_name, year_of_award, category}
foreign keys: [id, title, production_year] ⊆ WRITER[id, title, production_year]
[award_name] ⊆ AWARD[award_name]

ACTOR_AWARD(title, production_year, description, award_name, year_of_award, category, result)
primary key: {title, production_year, description, award_name, year_of_award, category}
foreign keys: [award_name] ⊆ AWARD[award_name]
[title,production_year,description]⊆ROLE[title,production_year,description]

Your task is to answer the following questions using SQL queries. For each question, your answer must be a *single SQL query* that may contain subqueries, and you must write your answers into the template file `myqueries.sql`.

2.1 List the titles, production years, and run times of all movies whose run times are shorter than 90 minutes or longer than 3 hours, and sort these movies by their run times in ascending order.

(1 Mark)

2.2 For the period 1995-2005 (inclusive), what is the percentage of movies produced in *Germany* among the total number of movies produced in that period? List the percentage as a decimal (round to two decimal places).

(1 Mark)

2.3 At the time when the movie "Titanic" was produced, what was the age of the director?

(1 Mark)

2.4 Which actors have starred in at least one movie, which has also been directed and written by themselves? We consider that a person has directed (resp. written) a movie if this person is one of the directors (resp. writers) of the movie. Return the ids, first names and last names of these actors.

(1 Mark)

2.5 List the titles and production years of all movies that have won an *Oscar movie award*, together with the first names and last names of their directors.

(1 Mark)

2.6 List the first and last names of actors who appeared in *exactly one* movie, together with the titles and production years of these movies.

(1 Mark)

2.7 Find the movie(s) with the largest number of distinct actors. If an actor played multiple roles in a movie, we count such an actor only once. You may *not* assume that only one movie has the largest number of distinct actors. Return the title, production year and the number of distinct actors for each of the movie(s).

(1.5 Mark)

2.8 Which actors have won *at least one actor award* for a movie where the actors have appeared in less than 5 distinct scenes? Return the ids, first names and last names of the actors.

(1.5 Mark)

2.9 Who have always written a movie with *at least another* writer, i.e., every movie written by such a writer has at least two distinct writers? Return their ids, first names and last names.

(1.5 Mark)

2.10 Which directors have starred in *every* movie directed by themselves? Return their ids, first names and last names.

(1.5 Mark)

+++++