4CCS1DBS - Database Systems
Coursework assignment
March 2020
(deadlines as advertised on KEATS)

Overview

The purpose of this coursework is to create the **conceptual schema for a database using ER diagram notation** for a specific domain based on the provided requirements. The coursework also includes the **implementation of a relational schema in SQL and writing queries in SQL Data Manipulation Language**.

The entire coursework is formally assessed and is worth **15%** of your final grade. You will receive some feedback as part of the marking of the coursework. You will be graded on the quality of your submission, including overall written presentation, legibility and proper use of language (where applicable).

The coursework is comprised of two constituent parts, as below.

Part 1: Design

(1.1) Identification and representation of entities, relationships and attributes.

(1.2) Identification and representation of relationship cardinalities based on the coursework requirements, or by making proper use assumptions if the coursework requirements are vague. Any assumptions need to be reasonable and realistic.

(1.3) Identify domain constraints specific to the database requirements.

Part 2: Implementation

(2.1) Create and implement an ER model in appropriate SQL schema and table creation queries, including entities, relationships and constraints.

(2.2) Insert appropriate sample data using INSERT queries.

(2.3) Create and output the appropriate SELECT queries.

(2.4) Update data using the appropriate UPDATE query.

(2.5) Remove data using a DELETE query.

Note: Part 2 will likely require more time to complete than Part 1, so please plan your time accordingly, so that you are able to finish all sections within the given coursework timescales.

Setup. Based on a given ER diagram, and a set of requirements, you will implement a relational model for a television talent show named **TheVoiceLondon**. Then, you will have to perform different data manipulation and retrieval operations on the database. On KEATS you will find a .zip containing template SQL files to edit for Part 2 of this coursework.

Requirements. A television channel has decided to create a simple database to register payment information about its most successful show 'TheVoiceLondon'.

Contenders <many to one> Coaches

In this show, there are contenders that compete to represent the UK in Eurovision. These contenders are coached by famous artists (namely the coaches). Contenders can be formed by a group of participants or a single participant. Both coaches and participants are paid based on the number of shows they attend.

For each coach and participant, the database stores their id, name, surname, date of birth, phone, gender and daily salary.

contender = group of participants

For each contender, the database stores its id, type (group or individual), stage name, its coach and the participants forming that contender. Each contender should have at least one participant.

multivalue and composite

1 to n

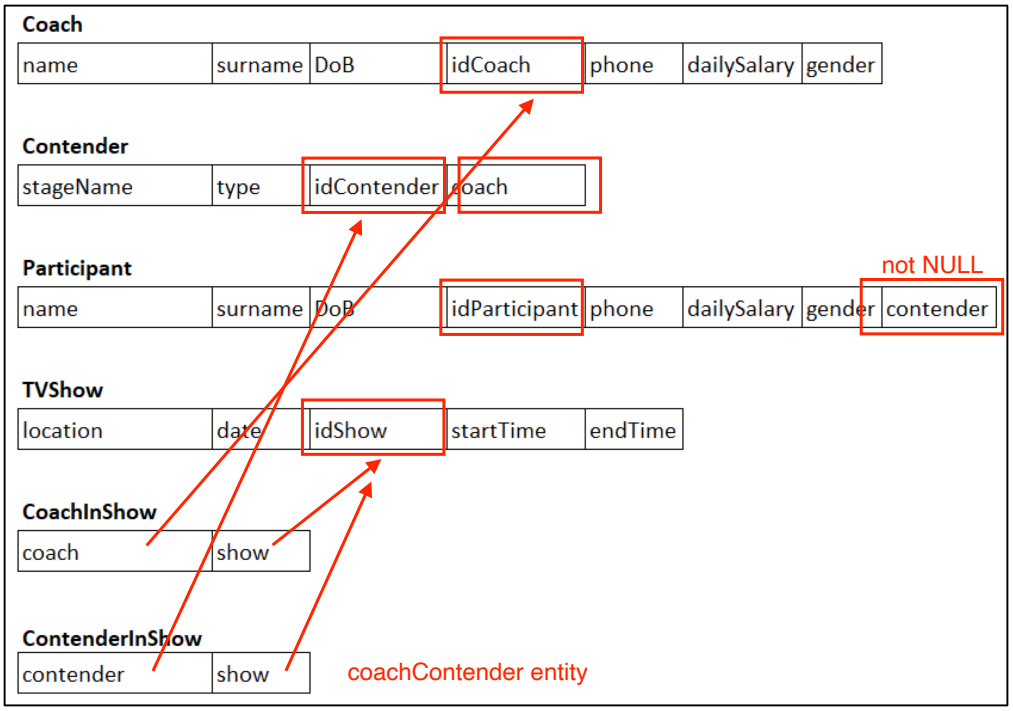
For each show, the database stores its date, start time, end time and a location if the show does not take place in the television studio.

if null -> studio

Finally, the database also registers which coaches and contenders attended each show.

If a coach decides to leave the program, then their personal and attendance information must be deleted from the database and any contenders they coach need to be assigned a replacement coach.

The following relational schema contains the database model for this TV show. In the relational schema below, primary and foreign keys are **not** given to you.


Coach
name
surname
DoB
idCoach
phone
dailySalary
gender
Contender
stageName
type
idContender
coach
Participant
not NULL
name
surname
DoB
idParticipant
phone
dailySalary
gender
contender
TVShow
location
date
idShow
startTime
endTime
CoachInShow
coach
show
ContenderInShow
contender
show
coachContender entity

Part 1: Design

1.1 ER Diagram. Draw the ER Diagram that corresponds to the above requirements. Include all entities, attributes, and relationships.

If necessary, please explain/justify your design choices.

1.2 Cardinalities and Assumptions. On the ER Diagram include all cardinalities of the relationships using the ERD (min, max) notation from lecture. State assumptions made and ensure that those assumptions do not contradict with the coursework requirements.

Please list your assumptions as bullet points.

1.3 Constraints.
For each relation, identify their **primary** and **foreign keys**. You may write (PK) and (FK) to indicate the attribute(s) that serve as primary and foreign keys, respectively. For example, 'ProjectNumber (PK), EmployeeNumber (FK)'.

What to turn in for Part 1

Organise your design into a single **PDF** document for Part 1 and submit it on KEATs before the deadline. Submit to the KEATs section named:

"PART 1 COURSEWORK UPLOAD".

Other documents and extra materials will be ignored.

Be sure to include your **Name** and **Student number** on your submission.
Your design will be assessed not only on the **quality of the design**, but also on the overall **presentation** including **legibility** and proper use of language in your explanations. Utilise a diagramming / sketching program (as practiced in the Lab Practicals) to create an ER Diagram and relational schema. **It will be hard for the marker to properly access your work if it is drawn by hand.**

Part 2: Implementation

2.1 Schema Definition. Based on the both the requirements above, and the keys you have identified, write the required SQL DDL (Data Definition Language) statements (i.e. CREATE TABLE…) to create the schema and corresponding tables.

Ensure that:

- table and attribute names **do not** conflict with SQL reserved words
- attribute data types are **core primitive SQL data types** as described in the lectures (i.e. **do not** use the ENUM type for example)
- table columns have appropriate **key** and **entity** constraints properties
- every table has a **primary key** specified as it corresponds to your relational model
- **all foreign keys** are properly declared, and explicitly describe how they handle potential **referential integrity** constraint violations (i.e. it is up to you to decide the triggered action to the foreign key constraints)
- your schema enforces the **domain and semantic constraints** stated in the requirements.

Note that you may not be able to enforce all the semantic domain constraints in the `CREATE TABLE` statements and MySQL does **not** have Assertions in the manner that we discussed in lecture (i.e. using `CREATE ASSERTION`). If you are unable to enforce a semantic domain constraint **include a comment** in your schema explaining your constraint and the reason it is not implemented.

Write your schema in the provided template file: `schema.sql`

Assume that the database schema will already be created for you (i.e. **do not** include a `CREATE SCHEMA` statement in your file, **it will result in an error**). Also assume that your script will already be run within your database schema (i.e. do not include a `USE…;` statement in your file, **it will result in an error**).

2.2 Populate Database with data. Time to get creative! Populate your database with some data that you will come up with on your own. Since you only require a small test sample of data, you will use SQL INSERT statements to populate your database.

More precisely:

- Pick at least 3 of your favourite celebrities to include as **Coaches**. Make up their personal data. If you don't really care for celebrity culture, then pick 3 random people to include as Coaches.
- Create **at least 10 participants**.
- Create at **least 5 contenders** and assign them participants, so that there is **one group contender**.
- **Assign these contenders to the different coaches**, making sure that there is at **least one coach without contenders**.

- Create **shows** taking place all **Saturdays and Sundays in March and April 2019**. The shows can start at any time you want but **must have a duration of 2 hours**. Note that not all shows can have the same start and end times.
- For each show create **at least 3 attendances of contenders and 2 attendances of coaches**.

Write INSERT statements in the provided template file: `insert.sql`

You **may only** use the DML (Data Manipulation Language) commands covered in lecture to help you populate your database.

All data **must** be contained within the `insert.sql` file, do **not** load the data from separate data files (i.e. using a CSV file).

Do **not** use other SQL statements, such as FUNCTIONs, PROCEDURESs or other programmatic MySQL-specific commands.

Assume that your script will already be run within your database schema (i.e. do not include a `USE`…; statement in your file, **it will result in an error**).

2.3 Query the Data. Write the SELECT statements that to obtain the following queries:

- **Average Female Salary.** TheVoiceLondon would like to know the average daily salary for female participants. Write a SELECT query that gives the average daily salary for female participants. Have your result return a **single scalar value** (i.e. *in total GBP*).

- **Coaching Report.** For each coach, list the total number of contenders they are coaching. In the listing, include the information about the coaches without any contender.

- **Coach Monthly Attendance Report.** For each coach, list the total number of shows attended in each month.

- **Most Expensive Contender.** TheVoiceLondon would like to know which contender has the highest total daily salary (i.e., sum of the daily salaries of the participants forming that contender). Write a SELECT query that lists the stage name of the contender with the highest total daily salary.

- **March Payment Report.** Create an itemized payment report for March corresponding to the shows attended by each coach and participant in March. Write a SELECT statement(s) that retrieves:

    - For each coach, show their name, the number of shows attended in March, their daily salary and their total salary for March

(calculated as the number of shows attended multiplied by their daily salary).

- For each participant, show their name, the number of shows attended in March, their daily salary and their total salary for March.
- The last line of the report should just contain the total amount to be paid in March.

Hint: You may use the string concatenation function `CONCAT` (https://dev.mysql.com/doc/refman/5.5/en/string-functions.html#function_concat) and UNIONS to help to build the payment report.

- **Well Formed Groups!** Note group contenders should be formed by more than one participant (otherwise they are individual contenders).

  Since MySQL does not support an assertion to check this constraint, write a SELECT statement that returns only a scalar Boolean value (i.e. either True or False). It should return True if there are **no violations** in the database of this regulation. If there is a violation, then the SELECT statement should return False.

  There is a violation if there is a group contender formed by less than 2 participants.

  Show that your SELECT statement works by creating a group contender that violates this rule and then running your SELECT statement.

Write all of these SELECT statements in the above order in the provided template file: `select.sql`

Assume that your script will already be run within your database schema (i.e. do not include a `USE...;` statement in your file, **it will result in an error**).

2.4 One more thing... To avoid that coaches and contenders arrive late to the shows, TheVoiceLondon has decided to change to hourly payments instead of daily payments:

remove daily column and add hourly

1. Update the coach and participant information to only contain the hourly payment. Given that the shows have a duration of 2 hours and that coaches and participants were required to arrive one hour before the show and to leave one hour after the show, the hourly payment should be calculated as the daily payment divided by 4.
2. Add new fields to the attendance table to register when coaches and contenders arrive to and leave the shows.
3. UPDATE the attendance information to include the arrival and departure times for the past shows. Your query should set the arrival time to one

~~hour before the show started and the departure time to one hour after the end time.~~

Write all these statements in the provided template file: `update.sql`

Assume that your script will already be run within your database schema (i.e. do not include a `USE`…; statement in your file, **it will result in an error**).

2.5 Fair payment! The contender with the lower total salary became upset and wants to leave TheVoiceLondon. The participants forming that contender have demanded to **have all their contender and personal data** removed from TheVoiceLondon database.

Using this contender stage name as its identifying attribute in the query, write the DELETE statement(s) that removes this contender and all their related data from the database. To avoid any future embarrassment in case of a data leak, make sure you also remove all trace of the participants forming that contender from the database.

Write all of these DELETE statements in the provided template file: `delete.sql`

Assume that your script will already be run within your database schema (i.e. do not include a `USE`…; statement in your file, **it will result in an error**).

What to turn in

For each SQL file that you turn in:
1. Include your **NAME** and **STUDENT NUMBER** at the top of every SQL file in a SQL line comment.
2. Edit these files as **text files, not Word files or propriety SQL software.**
3. Do **NOT** rename the files.
4. **ONLY** use the SQL line comment character (i.e. lines beginning with --) for comments.

Comment your SQL.

Just like any program code, comments will help outline, structure, and make clear what is written. You will be evaluated on your ability to provide comments to help provide structure, organization, and clarity to your SQL code. Make your comments useful, concise, and clear.

Submission

Put all SQL files (and your optional updated database design PDF) in a **ZIP** file (i.e. with a .zip file extension) and submit it on KEATs before the deadline.
Do **not** put the files in a RAR (i.e. rar file).
Do **not** put the files in a tar-gzipped file (i.e. tar.gz.).
Submit your files in a ZIP file (i.e. with a .zip file extension).

Submit to the KEATs section named:
"PART 2 COURSEWORK UPLOAD".

Any SQL file that is missing or renamed will result in 0 marks for that sub-part.

Evaluation

The SQL files you create will be evaluated using the NMS database server that you utilized in lab**. Test all your database SQL files** on the NMS's database server in your own personal database to be absolutely sure that they work and do not have any errors.

Your files will be executed in following order:

1. `schema.sql`
2. `insert.sql`
3. `select.sql`
4. `update.sql`
5. `delete.sql`

Each file will be tested from the NMS UNIX command line, with this command:

```
mysql -u k123456 –p -h mysql2.nms.kcl.ac.uk -P 33306 your_db < file.sql
```

Where `k123456` is the database user name (i.e. your k-number), `your_db` is the database name, and `file.sql` is the SQL file to be executed. Using your NMS databases, test your files, in this order, to make sure that they run, before you submit them. Test your files, even if you are "just adding comments".

If any of your submitted files do not run on the NMS database server, your Part 2 coursework will be capped to 60% of the Part 2 total marks.