MySQL practicals

N.B.: before you leave the practical sessions please show the demonstrators that you have completed the proposed exercise

Exercise TWO

OBJECTIVES: learn how to reduce duplications by separating information that relates to different concepts; create **relationships** between tables; use foreign keys; simple queries.

In exercise 1 we built a table with information on students and courses. The table contained duplications (e.g. the same course information repeated for all students that were taking that course). We also encountered difficulties if we wanted to delete information about a student, i.e. all the information relating to a course could be lost if the student we were deleting was the only one taking that course (deletion “anomaly”). These problems are due to the fact that a lot of information is included in the same table some of which relates to students and some of which relates to courses.

In this exercise we want to remove unnecessary repetitions of course detail and avoid the deletion “anomaly”. This is done by splitting the table created in Exercise 1 into two tables to separate the information about students from the information about courses (**Task 1**). However, the table of Exercise 1 also contained information about what course a student was taking (i.e., it encoded a “relationship” between students and courses). This information has to be preserved (**Task 2**).

A **relationship** exists between two tables when rows in the first table are related to rows in the second table. There are several types of relationships (one-to-one, one-to-many, many-to-many).

In this exercise we build a **one-to-many relationship**. A one-to-many relationship is a relationship where a row in the first table is associated with at most one row in the second table but a row in the second table can be associated with more than one row in the first table. For example, the relationship between *persons* and *birthplaces* is one-to-many: many persons can be born in a given place but each person only has one birthplace.

In our example we build a one-to-many relationship between students and courses to represent the information that was contained in the table of Exercise 1. Remember that in such a table, each student was associated with only one course but a course could be associated with several students.

In order to define a one-to-many relationship you need to include a **foreign key** in the table **students**.

Task 1 - Create a new database containing two tables called **students** and **courses**, respectively, with the following schemas (and the same properties as in Exercise 1):

students (studentid, firstname, lastname, telephone, email)
courses (coursecode, coursename, duration, coursecost)
where the attributes underlined in each table form the primary key (PK) for that table.

Populate **students** with the following data:

- 12345, Mary, Murphy, 2888888, MMurphy@ucd.ie,
- 23456, Brian, Smith, 6498888, Bsmith@ucd.ie,
- 34567, Cora, Williams, 1234567, Cwilliams@ucd.ie,
- 12222, David, Honan, 2888888, Dhonan@ucd.ie,
- 11111, Frank, Murphy, 4568777,
- 23000, Aoife , Byrne, 987789, Abyrne@ucd.ie,

Populate **courses** with the following data:

- REL20, Relational databases, 10, 300
- WEB20, Web design, 10, 200
- EXL20, Excel, 9, 200

Task 2 – Create a **one-to-many** relationship between **students** and **courses** (i.e., each student can take only **one** course, a course can be taken by **many** students).

In order to implement this relationship, you need to add an extra column *takingcourse* in the table **students** (edit the table **students**). Then you need to add a **foreign key** to table **students** by specifying that *takingcourse* is a foreign key referencing the primary key of **courses** (*coursecode*). To add a foreign key use the Table Editor from the Navigator panel (left) and then the Foreign key tab (right). This way you have created a relationship between **students** and **courses**.

This relationship is currently empty, i.e. there are no associations between courses and students (as the column *takingcourse* has just been added). To create these associations, you need to enter valid *coursecodes* in the *takingcourse* column (**referential integrity** constraint): valid *coursecodes* are those that exist in the table **courses**. Insert the information about courses taken by students (as from Exercise 1):

- 12345, Mary, Murphy, 2888888, MMurphy@ucd.ie, REL20
- 23456, Brian, Smith, 6498888, Bsmith@ucd.ie, REL20
- 34567, Cora, Williams, 1234567, Cwilliams@ucd.ie, REL20
- 12222, David, Honan, 2888888, Dhonan@ucd.ie, WEB20
- 11111, Frank, Murphy, 4568777,   , WEB20
- 23000, Aoife , Byrne, 987789, Abyrne@ucd.ie, EXL20

Task 3 – Create a table called **tutors** with first name, last name, telephone number, and email. Create a suitable primary key (PK). A PK must be unique and cannot change over time. This is why a PK is often an abstract code created on purpose (e.g. PPS number). Populate the table. Each tutor can only teach one course and a course can be taught by many tutors. Create a **one-to-many relationship** between courses and tutors, using an appropriate foreign key. Populate the relationship.

Task 4 – Write the following queries in **Relational Algebra (solutions to these will be posted to Brightspace at the end of this week but you should try by yourself today)**:

- Find the *studentid* of all students. This query can be done in Workbench by right-clicking on the *studentid* column in the Navigator panel and clicking on "Select Rows".
- Find the *lastname* of all students. Do the same query in Workbench.
- Find the *firstname* of all students. Do the same query in Workbench.
- Find the *firstname* and *lastname* of the student with *studentid* = 11111
- Find the *coursename* of courses taken by students with *lastname*='Murphy'