CSCU9B3 Practicals 4 & 5: Normalisation

Computing Science, University of Stirling

You will need a pen and paper for this practical, so that you can make notes as you go. Things you do in this session will be useful later, so all notes will have further use.

PHPMyAdmin versus SQL

For the following exercises you can either mostly use the PHPMyAdmin interface or you can write your own SQL statements (storing them in a file for reuse, as you did for the previous practical). Which approach you use is up to you. Writing your own SQL will give you more practice in doing this.

Finding the Data

1. Log into PHPMyAdmin (http://wamp0.cs.stir.ac.uk/phpmyadmin/) just as you did in the previous practicals.
2. Get a copy the files called **normdata.csv** and **loaddata.csv** into your own file space by downloading them from the module's Canvas **Practicals** page.
3. Open normdata.csv (double-click on it) and have a look at its contents.

The data concerns employees and managers on different projects within a company. You should notice that:

- Each member of staff (employee) works on one or more projects.
- Each project has just one manager and each manager only manages one project.
- Not all staff are managers, but a manager must be a member of staff.

Answer following questions:

Is the data in first normal form?

Is it in second or third normal form?

Decomposing the data into third normal form

Your first job is to decompose the table into more than one table in third normal form (3NF). To do this, you will need to:

1. Identify all the entities, relations and attributes in the data and list them (on paper).
2. Identify primary and foreign keys amongst the attributes.
3. Draw the ER diagram for the data and check it is in third normal form. You should have examples of one to many and many to many relationships from this data, which should lead you to introduce extra relation(s) to obtain 3NF.

CHECKPOINT 4: show your ER diagram and explain why it achieves 3rd normal form.

4. Create tables (with different names from earlier practicals) in MyPHPAdmin ready for the newly decomposed data. How many tables do you need? To create the tables, you can either use the user interface or write suitable SQL statements directly. Note that you

can set a primary key in PHPMyAdmin by selecting the Primary Key option for a column in the Structure view of the table. (Foreign keys will be set later in this practical.)

Importing the data

Once you have done steps 1 to 4 above, you need to get the data from the badly organised CSV file into your new tables. To do this, carry out the following:

1. Create a new empty table called **loaddata** in PHPMyAdmin with a column for each of those in the normdata CSV file. Do not set any primary key.
2. Select this new table and go to the **Import** tab.
3. Click Browse and choose the file you wish to upload, in this case **loaddata.csv** (this is the same as normdata.csv but without the header information in the first row).
4. Make sure the Format is "CSV" (use down arrow to see options) and check the Format-Specific Options: comma (,) for column *separator* and " for *enclosed* and *escaped* are ok.
5. Click Go. It may take a little while to load… but once finished, your new table should be ready to browse. Take a look at the table.

Inserting data from one table into another

You now need to get the data from this table into the tables you have designed for your third normal form. At this point, you cannot use the PHPMyAdmin interface and must enter your own MySQL statements as there is no facility to copy part of one table into another.

The syntax for selecting from one table and putting the result in another is:

INSERT … SELECT

The help page on this command is here:

http://dev.mysql.com/doc/refman/5.7/en/insert-select.html

You will need to read about the **SELECT** statement too:

http://dev.mysql.com/doc/refman/5.7/en/select.html

You will need to create a SQL statement that selects only the columns from the original data table which, for example, are in your employee table (and then the same for your other tables). Also, make sure that your rows are DISTINCT.

Choosing Primary and Foreign Keys

You need to choose the primary and foreign keys for each table. Two of the tables should have a single primary key, with any "linking" table having a composite primary key consisting of two columns. When you have chosen your keys, move on to the next step. If you are not sure you made the right choice, ask the practical demonstrator.

Setting the Keys

You may have set the primary key when you created each table, by selecting the Primary Key option for the columns(s) under the Structure tab, or by specifying the key in your SQL CREATE statement. If you did not, then you can set a primary key now by going to the **Structure** page for each table in turn, selecting the required column(s) and clicking the Primary Key icon below the list of table columns.

Foreign keys are slightly more complicated as you need to specify which column is a foreign key and which foreign column it relates to. You are likely using a table to link employees to projects because there is a many to many relationship between the employees and projects. In such cases it is the linking table that contains the foreign keys. In general, the foreign key goes into the table on the ***many*** side of a one-to-many relationship. This is done as follows:

1. Go to the **Structure** tab for the table where you want to set a foreign key.
2. Make sure that the required foreign keys are already primary keys.
3. EITHER, click Relation view button above (or maybe below) the column definitions:
   a. Select the relationship you want for each of the foreign keys defined (NOTE: add a new constraint for each foreign key). Do this by selecting from the drop-down lists, the first of which enables you to select the appropriate table, and the second the column name. Ignore the "Constraint name" (this will be set automatically), ON DELETE and ON UPDATE boxes that appear (for now).
   b. Click **Save** to make the changes.
4. OR write SQL statements to add the foreign keys, using ALTER TABLE statements.
   a. See the first SQL lecture for examples of how to do this.
   b. Run your SQL via the SQL tab in phpMyAdmin.

CHECKPOINT 5: show your final tables, populated with data and with keys set appropriately.

Checking Referential Integrity

With appropriate Primary and Foreign keys set, there should be a set of things that you cannot do to the database without violating referential integrity.

Deleting Records

You should not be able to delete a project or an employee from their respective tables.

1. Try to do so (use the Browse tab or write an appropriate SQL statement) and see what error you get.
2. However, you should be able to remove an entry from your table that links employees to projects (DO NOT do this unless you are prepared to re-enter the data!), effectively removing an employee from a project.

Make sure you understand the difference between the two cases above. In the first, you are removing the details of an employee or a project, while they are still linked in the joining table. This leads to a loss of referential integrity. In the second case, no loss of integrity occurs.

Dropping Tables

Similarly, try to drop your employee table. You will find that you cannot. However, if you try to drop the table that links employees to projects you will find that you can (DO NOT do this as you will need it later on, unless you a prepared to reconstruct it). Why can you delete it?

Inserting Records

Try to insert yourself as a new employee on a new project called University:

1. Go first to your table that links employees to projects and try to insert an entry: you will find that you cannot via phpMyAdmin as your new project and employee number are not available from the drop down lists of values that you can chose from.
    a. Why can't you enter the required data?
    b. What do you need to do before you can enter data in this table?
2. Once you have understood the problem, enter the needed data into other tables and then try again. This time, your insertion should be possible.

Foreign Key Constraints

You can tell the database engine what to do if an entity is deleted when it is referred to in another table. In this example, we will choose CASCADE.

1. Under the Structure tab, go to the Relation view for your linking table and select CASCADE for ON DELETE and ON UPDATE for both foreign key columns.
2. Now go to the Browse view of the projects table and delete the entry for project "University" that you just created.

Remember when you tried to delete a project before, it produced an error? Now there is no error, and if you browse the table that links employees to projects, you will see that all references to the University project have been deleted too. This is what is known as a cascaded delete. You can read more about foreign key constraints here:

http://dev.mysql.com/doc/refman/5.7/en/innodb-foreign-key-constraints.html

SQL queries on your Relations

To get more experience with writing SQL statements to query a database, implement the following queries by directly writing the appropriate SQL (as you did in the previous practical):

1. List the name and description of all the projects that Alan Stoner works on.
2. List the name and description of the project (if any) that Alan Stoner manages.
3. List the names of all employees working on project Zoom.
4. List the name and the total salary cost for the project managed by 974650.

CHECKPOINT 6: show your results for searches 2 and 3 and the SQL you used in an editor.

Now try making up your own queries or updates for even more practice.