**Applied Databases Project 2020**

**4.3 Normalisation**

**Written and submitted by: John Dunne G00273895**

I have written this answer myself drawing on the lecture slides presented in the course and my own notes made at the time of viewing the lecture slides on normalisation.

--------------------------------------------------------------------------------------------------------------------------------

**What is normalisation?**

Normalisation is the process of organising data in a relational database such as MySQL in order to minimise data redundancy. Normalisation can also serve as a good way to preserve the integrity of data entered in a database by ensuring data entered in one table is consistent with data contained in another table.

**Foreign key reference:**

A foreign key reference is when there are multiple tables in a database and one column of data in a table is referencing a column of data in another table in the database. Foreign key references can be seen by running the showcreatetable command in MySQL. When the showcreatetable command is ran it will show if there is a foreign key reference from the table in question to another table in the same database. An example would be one table containing data about students in a third level institution and a second table containing the names and details of lecturers. If there is a column of data in the student table that has a foreign key reference to a column of data in the lecturer table, this would place a constraint on data being entered in the student table. For example, a lecturer ID number, then this column could be referenced as a foreign key. Only lecturer ID’s that already exist in the lecturer table can be entered in the student table. When working in MySQL Workbench a graphical look at foreign key references can also be seen by reverse engineering the database.

**Joining tables:**

In order to access data that is present in two or more tables in a relational database the tables must be joined. Tables must only ever be joined on the columns of data that are reference by a foreign key.

**My proposal for the table in this project question:**

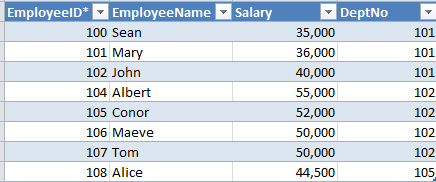
The proposed database contained in the project specification document consists of one table which in my opinion is not the most efficient way to design this database. I think a better design would be to have two tables – one table called Employees storing the details of employees and another table called Department storing the details of departments. I would propose adding a column containing a unique department number to both tables where each separate department would have a unique department number. This department number would serve as the foreign key reference between the tables. This would serve to preserve the integrity of the database and minimise the potential for errors in the database. It would also minimise duplication of data as instead of entering department data for each employee in the employee table, the department data would be stored in the department table as a unique entry. In my proposal, the department number column in the employee table would reference the department number table in the department table. This would help to reduce data redundancy as when a user wants to enter some data about an employee in the employee table they won’t need to enter extra details about the department such as location and budget which may not be relevant in a lot of cases.

Instead they would only be required enter the corresponding department number and anyone who would need to find more information about the department would be able to find this by joining the tables on the foreign key reference.

For visualisation purposes I have drafted what both tables would look like in my proposal:

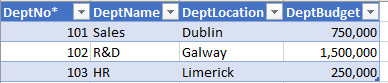
**Table 1- Employee Table**

* Primary key – Employee ID
* Foreign key reference – Dept no. (Department table)



**Table 2 – Department Table**

* Primary key – Dept no.



In my proposal the department number column in the employee table would reference the department number column in the department table. Therefore, data being entered in the employee table would be constrained by this foreign key reference. Only department numbers that exist in the department table may be referenced in the employee table. Important to note also that I would make the department number column the primary key of the department table. A table can only have one primary key and all entries in the primary key column must be unique and no null values are allowed. This would further serve to maintain the integrity of the database as all new entries in the department table must have a unique department number entered.

**Issues with the current database shown in question:**

Looking at the current database there is potential for incorrect data to be entered in the table. The only restriction on the current database is that EmployeeID is the primary key and so a unique value must be entered. There is no restriction on any of the department data. This could cause a potential issue with incorrect data being entered. It also leads to duplication of data as department data must be entered for each employee instead of being entered just once and referencing this entry in the employee table.

**Example entering incorrect department data into table:**

If I pass the following command to insert a person into the table in the current structure, and I have erroneously entered Finance as the department name instead of R&D, there will be an incorrect record added to the table:

Insert into Employee (EmployeeID, Employeename, Salary, **DeptName**, DeptLocation, DeptBudget)

values ("109", "John", "Dunne", “30,000”, **“Finance”**, “Galway” “1,500,000”);

This won’t cause any error as the only constraint on the current database is that a unique employee Id is entered which I have done.

However, with my proposed structure of two tables with a foreign key constraint on the department number it would not be possible to enter data in the employee table for a department that doesn’t already exist in the department table. This ensures the integrity of the database is maintained.