Namal Tutorial Groups Management System

Design Report

This Report is about how I Design Database.

Nisar Ali **14031279**

8/9/2017

Contents

1. Abstract

This report addresses the database design of my SEGP project. The database design was to incubate the different logical requirements of a Group Assignment and Updating system for Namal College Mianwali. A logical design is created and demonstrated in the report which aims to address all the different requirements of the database design. Justifications and visual diagrams are provided to help understand the solution.

2. Introduction

In this report, I have discussed the different aspects of choosing and creating a database design for a tutorial groups system. The database design demands a logical solution to accommodate the different aspects indicated in the problem statement. So, in order to create the database design I have shown further in the report the different attributes and entities related amongst each other and how they try to solve the problem successfully. Along the path, I have also provided different justifications for the approach I have used and some demonstrations to make the solution more vivid.

3. Problem Review

The problem requires developing a database for a tutorial groups system. The system should be developed to provide functions for uploading students and staff members, and allocating/removing staff and students to/from tutorial groups and every tutorial group will have no more than 6 students. Each group has a tutor, a member of staff and is identified by a numeric code of three digits of which the first one indicates the year – for MSc students this will be 5. A staff member might be allocated to more than a tutorial group. In the same group there are only students from the same year.

4. Data Analysis (Top-Down Analysis)

According to the requirements provided in the problem statement I deduce that there should be three tables for simply. First table will be used to record the student's anomalies, secondly for recording the staff there should be a separate table and lastly for the group record maintenance I would recommend a single table to hold all of its anomalies as there is a fixed number of students to be added i.e. six students per group. The tables for student and staff are pretty straightforward and simple however for the group table there will be foreign keys to attend to students and staff members but since there is a restriction on groups that there should be no more than six students therefore I have also restricted the groups table to have six individual columns for accommodating student however for groups that have less than six students some of the columns will be left empty but if we look at the problem logically we see that in real world there is usually only one group per

class that will have some of the columns empty i.e. the last group (because the number of students in the class is not always in multiples of six) to be allocated otherwise for every six students there is a group and hence all the six entries are filled and used.

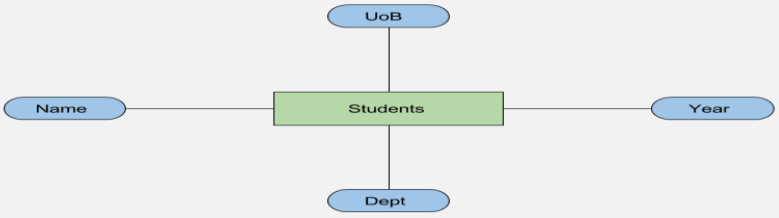
5. Design Normalization

In the database designed above, we can see that there are no multi-valued attributes in staff and students tables however the condition for group table to have multivalued attributes is discussed above. Except Group table students and staff, both of them have its unique set of attributes identified by a primary key. From this we can conclude that the database design inhibits the first normal form. Secondly, the design of the database shows that each table is identified entirely by the primary key. We can pinpoint any row of the table using their primary keys. So, the database design is also following second normal form. Thirdly, There might seem a need for transitive functional dependency in the groups table but since each problem has its own specific problems and in this problem there is a specification that there will be no more than six student per group therefore we have to look into each problem specifically and then decide whether the transitive functional dependency needs to be removed or not. In this case I have decided not to remove the seemingly obvious transitive functional dependency in accordance to the specifications of the problem.

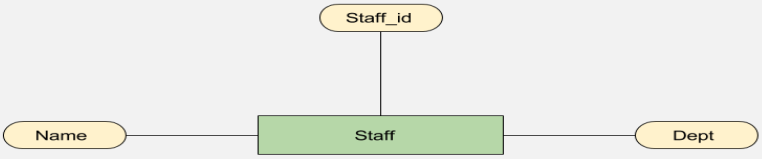
6. Entity Diagrams

These entities can be better visualized in these diagrams:-

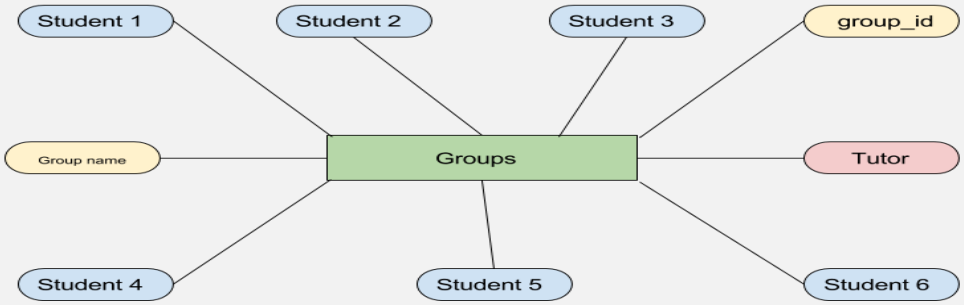
6.1 Students:



6.2 Staff:



6.3 Groups:



7. Data Design

In order to incubate the above mentioned stories of the problem, I have designed a database with the following relations:-
The usage justification of each is also enlisted.

7.1 "Students"

Attributes: UoB, Name, Year, Dept.
Primary Key : UoB
Value Sets:
UoB – {Numeric value}
Name – {Alphabetic Character Values}

Year – {Numeric value in a specific format}
Dept – {Alphabetic Character Values}

7.1.1 Description:-

In our database, each student is recorded in the Students Relation using his/her Uob, name, year, and department values. The primary key for Students is the UoB because each student is assigned a unique UoB and can serve the purpose of primary key rather than creating another primary while having one unique attribute already. Name column records the name of the student and respectively year and department columns record the values related to a particular entry of a student to the Students table.

7.2 "Staff"

Attributes: Staff_id, Name, Dept.
Primary Key : Staff_id
Value Sets:
Staff_id – {Numeric value}
Name – {Alphabetic Character Values}
Dept – {Alphabetic Character Values}

7.2.1 Description:-

In our database, each staff member is recorded in the Staff Relation using a unique identification named staff_id along with his/her name and department values. The primary key for Staff is the staff_id having a unique entry for every staff member. Name column records the name of the student and respectively department columns record the values related to a particular entry of a staff member to the Staff table.

7.3 "Groups"

Attributes: Group_id, Group_Name, tutor, Student_1, Student_2, Student_3, Student_4, Student_5, Student_6.
Primary Key : Group_id
Value Sets:
Group_id – {Numeric value}
Group_Name – {Alphabetic Character Values}
Tutor – {Alphabetic Character Values}
Student_1 to Student_6 – {Alphabetic Character Values}

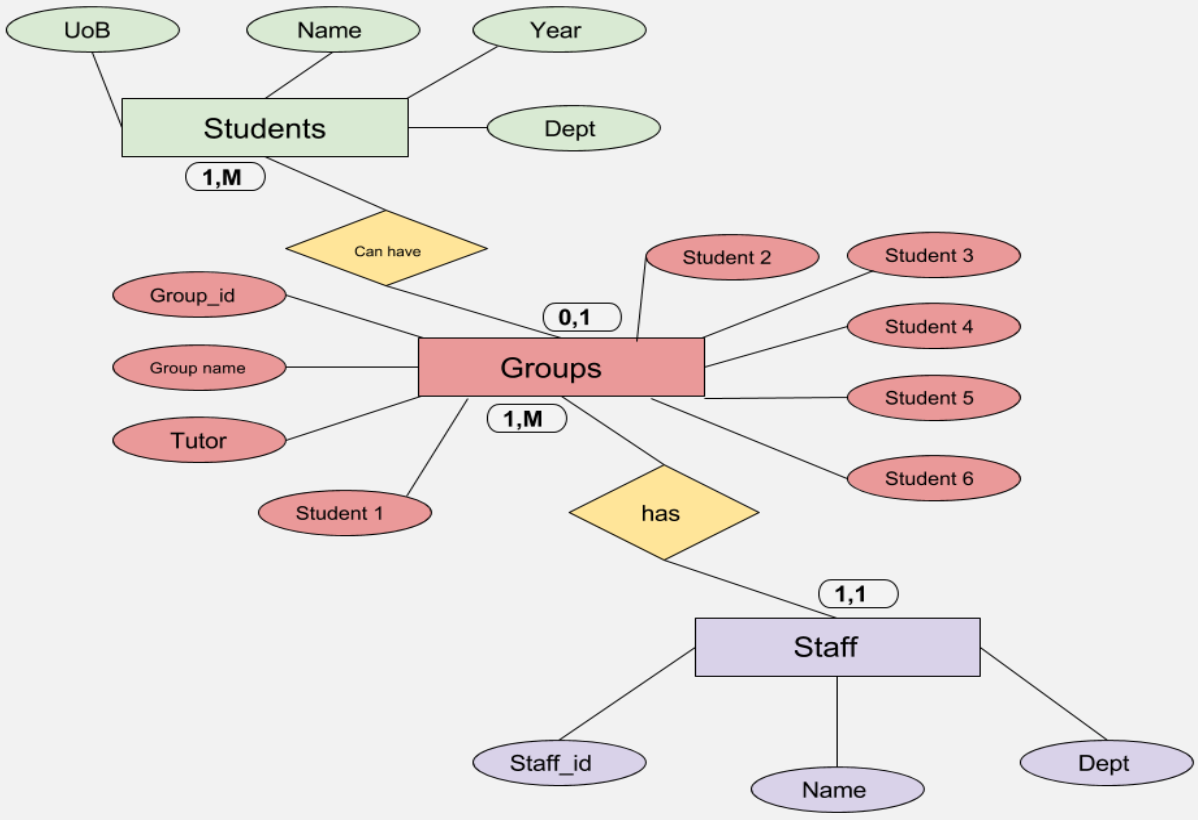
7.3.1 Description:-

In our database, group relation is the main relation connecting the staff and students to form and assign a group. Each Group is recorded in the Group Relation using its Group_id, group_name, Tutor, and six columns for students. The primary key for Group is the Group_id recoding a unique number to each group. Group_Name column records the name of the group, Tutor represents the staff member allocated to the group and respectively six individual columns are allocated to record the six students to be assigned to that particular group.

7.4 Foreign Keys:-

In this table, the tutor and the six student columns hold foreign key values to the actual entries recorded in the student and staff table.

8. Entity Relation Diagram



9. Final Database Design

9.1 Student:-

```
mysql> use seip;
Database changed
mysql> desc students;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| UoB   | int(11)     | NO   | PRI | NULL    |       |
| name  | varchar(30) | YES  |     | NULL    |       |
| year  | int(11)     | YES  |     | NULL    |       |
| dept  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

9.2 Staff:-

```
mysql> desc staff;
+----------+---------------------+------+-----+---------+----------------+
| Field    | Type                | Null | Key | Default | Extra          |
+----------+---------------------+------+-----+---------+----------------+
| staff_id | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| name     | varchar(30)         | YES  |     | NULL    |                |
| dept     | varchar(20)         | YES  |     | NULL    |                |
+----------+---------------------+------+-----+---------+----------------+
```

9.3 Group:-

```
mysql> desc groups;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| group_id   | int(11)     | NO   | PRI | NULL    |       |
| group_name | varchar(30) | YES  |     | NULL    |       |
| tutor      | varchar(30) | YES  |     | NULL    |       |
| student1   | varchar(30) | YES  |     | NULL    |       |
| student2   | varchar(30) | YES  |     | NULL    |       |
| student3   | varchar(30) | YES  |     | NULL    |       |
| student4   | varchar(30) | YES  |     | NULL    |       |
| student5   | varchar(30) | YES  |     | NULL    |       |
| student6   | varchar(30) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

10. Conclusion

A well-designed database will ensure that your data is accurately entered, that you can extract the information you need, and that your database can evolve as your requirements change. If you want to update to update or delete an aspect of the database, a well-designed database will not keep you in secrecy. In the end I would like to conclude this report. In conclusion, a database is a very efficient mechanism to store and organize data than spreadsheets or any other record system. Databases allow multiple users to share a central resource. With the help of a proper front end understanding database is no more a requirement.

11. Bibliography

Elmasri, R., Navathe, S.B. (2010) Fundamentals of database systems, 6th edn., : Addison-Wesley.