Database Project II: Normalization

CSC 210 Database Fundamentals

Matthew Clayton Dunaway

**Database Normalization**

The first half of the class was dedicated to the syntax involved in SQL statements. The second part of the larger topic of database fundamentals concerns the architecture behind the creation of a database. The database architecture is based on rules set for object-oriented database entity relations, called normalization. **Normalization** is the process of reducing the number of repeating values in your database, thereby making database modifications and queries easier. (Wenzel, 2021) The purpose of normalization is to break the tables down so the data included in columns only supports a specific topic. Breaking tables down to one purpose reduces errors that can occur when inserting or deleting subsequent data to the database, querying, and reduces the duplicate entries. To break these unnormalized tables down, we need to apply rules, or forms, of normalization. (Wenzel, 2021) There are several forms or steps to normalization, but I will utilize the main three normal forms to organize my database. The 1NF, 2NF, and 3NF are all based on functional dependency of table fields. Field A of a table determines field B. Conversely, field B is functionally dependent on field A. (Unknown, 2021) We can determine whether the table is in a proper normal form by assigning a primary key as the **determinant field** and subsequent fields as **dependent** on the primary key.

**Data Collected for My Database**

The database I chose was the CSC210 database provided for Assignment 2 named *UNF*. The table contained project data for an unnamed enterprise or government organization. It included columns containing *project codes, project titles, project manager, project budget, employee number, employee name, department number, department name, and hourly rate*. It also contained 10 rows of data. The primary key was *Project Code*, and the foreign key was *Employee No*. and some of the columns had repeating entries. At the intersection of each column and row, there were atomic values present. Also, updating this one database in any database management system would lead to more duplication or inadvertently removing unintended data. While this information may have relational database ties, it needs to be separated into multiple tables having a singular purpose. FWith this, I can complete the assignment given and provide background information and insight into normalization procedures as I progress through normal forms.

**Normal Forms of the Project Database**

To begin my database schema, I drew an ER diagram with entities and relationships separated out from the unnormalized table, *UNF*. For the **1st normalization**, I separated out *Employee No.* and *Employee Name* from the *UNF* table because these two columns can rely on each other as an entity. I chose the primary key for the *Employee* table as the *Employee No*. because it is largely integer based and doesn’t repeat. This left me with the remaining data being placed in a table called *Projects*. This had a primary key and a foreign key of *Project Code* and *Employee No.* to make each entry unique. This approach, **constructing a separate relation**, is one of two ways to convert an unnormalized table to 1NF. The other method is **flattening**. I could not use this because there were no empty cells on repeating columns to fill. Flattening also injects more redundancy into your tables. This puts both tables having an atomic (single) value at the intersection of every column and row and meets the requirement of 1NF.

Because my *Project* entity still relies on multiple relationships and entities, I performed **2nd normalization** to the table. 2NF requires 1NF and that partial dependencies be removed from the primary key of the table. To perform 2NF, I created *Project Information* and *Department* entities. I had a many-to-many relationship in both tables with the parent table and each had one primary key each, *Project Code* and *Department No.,* respectively. This satisfied the requirement of my tables **fully functionally dependent** on the primary key and in 1NF. It also categorized my *Employee* table as 2NF.

I then saw that I could separate out more entities to make my database less error-prone and performed a **3rd normalization**. The requirements for 3NF are 1NF and 2NF must be met as well as **no transitive dependencies** can exist. Transitive dependencies are present when you must update two tuples to make one update. Again, the technique for this normalization involves creating more relations by my deconstructing tables into *Budgets* and *Salaries* with *Project Code* and *Hourly Rate* being my primary keys. After assessing all my deconstructed tables with the rules for 1NF, 2NF, and 3NF, I found all of them met the third normalized form and should not experience any anomalies when I construct my database.

**Design an Entity Relation Diagram**

This ER diagram represents how I needed to separate data tables to minimize errors when creating, deleting, or querying data in my schema. Next, I used a website, *www.diagrams.net*, to graphically represent my drawings on a graphed table. It took a few hours of exploring the shapes and tools but eventually I created my first Entity Relationship Diagram to represent the normalization of table *UNF*. (Draw.IO, 2021)

Diagram

Description automatically generated

**List SQL Commands to Create and Fill Tables**

These are the SQL commands I used to create my database in MySQL WorkBench. Initially, I created the *UNF* table to test my understanding of databases from the previous term. I also found it very useful in understanding how primary keys and foreign keys play a function in the architecture of a DBMS query. I had to play with parameters of SQL statements for setting integer length and understand why you would want to use character limits on fields. Finally, I learned the importance of setting values to null or not null, and what the difference is. In the end, I deleted the UNF table from my normalized list of tables to avoid query errors, but I spent the most time and learned the most about SQL in this section. For the sake of abbreviating the commands for creating and filling tables, I reduced the font size to 9.5 (same font size used in Visual Studio).

1. CREATE SCHEMA `project\_information` ;
2. CREATE TABLE `project\_information`.`unf` (`Project Code` VARCHAR(45) NOT NULL, `Project Title` VARCHAR(45) NULL, `Project Manager` VARCHAR(45) NULL, `Project Budget` VARCHAR(45) NULL, `Employee No.` VARCHAR(45) NOT NULL, `Employee Name` VARCHAR(45) NULL, `Department No.` VARCHAR(45) NULL, `Department Name` VARCHAR(45) NULL, `Hourly Rate` DECIMAL(2,2) NULL, PRIMARY KEY (`Project Code`, `Employee No.`));
3. INSERT INTO `project\_information`.`unf` (`Project Code`, `Project Title`, `Project Manager`, `Project Budget`, `Employee No.`, `Employee Name`, `Department No.`, `Department Name`, `Hourly Rate`) VALUES ('PC010', 'Pension Systems', 'M Phillips', '24500', 'S10001', 'A Smith', 'L004', 'IT', '22.00');

INSERT INTO `project\_information`.`unf` (`Project Code`, `Project Title`, `Project Manager`, `Project Budget`, `Employee No.`, `Employee Name`, `Department No.`, `Department Name`, `Hourly Rate`) VALUES ('PC010', 'Pension System', 'M Phillips', '24500', 'S10030', 'L Jones', 'L023', 'Pensions', '18.50');

INSERT INTO `project\_information`.`unf` (`Project Code`, `Project Title`, `Project Manager`, `Project Budget`, `Employee No.`, `Employee Name`, `Department No.`, `Department Name`, `Hourly Rate`) VALUES ('PC010', 'Pension Systems', 'M Phillips', '24500', 'S21010', 'P Lewis', 'L004', 'IT', '21.00');

INSERT INTO `project\_information`.`unf` (`Project Code`, `Project Title`, `Project Manager`, `Project Budget`, `Employee No.`, `Employee Name`, `Department No.`, `Department Name`, `Hourly Rate`) VALUES ('PC045', 'Salaries System', 'H Martin', '17400', 'S10010', 'B Jones', 'L004', 'IT', '21.75');

INSERT INTO `project\_information`.`unf` (`Project Code`, `Project Title`, `Project Manager`, `Project Budget`, `Employee No.`, `Employee Name`, `Department No.`, `Department Name`, `Hourly Rate`) VALUES ('PC045', 'Salaries System', 'H Martin', '17400', 'S10001', 'A Smith', 'L004', 'IT', '18.00');

INSERT INTO `project\_information`.`unf` (`Project Code`, `Project Title`, `Project Manager`, `Project Budget`, `Employee No.`, `Employee Name`, `Department No.`, `Department Name`, `Hourly Rate`) VALUES ('PC045', 'Salaries System', 'H Martin', '17400', 'S31002', 'T Gilbert', 'L028', 'Database', '25.50');

INSERT INTO `project\_information`.`unf` (`Project Code`, `Project Title`, `Project Manager`, `Project Budget`, `Employee No.`, `Employee Name`, `Department No.`, `Department Name`, `Hourly Rate`) VALUES ('PC045', 'Salaries System', 'H Martin', '17400', 'S13210', 'W Richards', 'L008', 'Salary', '17.00');

INSERT INTO `project\_information`.`unf` (`Project Code`, `Project Title`, `Project Manager`, `Project Budget`, `Employee No.`, `Employee Name`, `Department No.`, `Department Name`, `Hourly Rate`) VALUES ('PC064', 'HR System', 'K Lewis', '12250', 'S31002', 'T Gilbert', 'L028', 'Database', '23.25');

INSERT INTO `project\_information`.`unf` (`Project Code`, `Project Title`, `Project Manager`, `Project Budget`, `Employee No.`, `Employee Name`, `Department No.`, `Department Name`, `Hourly Rate`) VALUES ('PC064', 'HR System', 'K Lewis', '12250', 'S21010', 'P Lewis', 'L004', 'IT', '17.50');

INSERT INTO `project\_information`.`unf` (`Project Code`, `Project Title`, `Project Manager`, `Project Budget`, `Employee No.`, `Employee Name`, `Department No.`, `Department Name`, `Hourly Rate`) VALUES ('PC064', 'HR System', 'K Lewis', '12250', 'S10034', 'B James', 'L009', 'HR', '16.50');

1. CREATE TABLE `project\_information`.`employee` (`Employee No.` INT NOT NULL, `Employee Name` VARCHAR(45) NULL, PRIMARY KEY (`Employee No.`));
2. INSERT INTO `project\_information`.`employee` (`Employee No.`, `Employee Name`) VALUES ('S10001', 'A Smith'); INSERT INTO `project\_information`.`employee` (`Employee No.`, `Employee Name`) VALUES ('S10030', 'L Jones');INSERT INTO `project\_information`.`employee` (`Employee No.`, `Employee Name`) VALUES ('S21010', 'P Lewis');INSERT INTO `project\_information`.`employee` (`Employee No.`, `Employee Name`) VALUES ('S10010', 'B Jones');INSERT INTO `project\_information`.`employee` (`Employee No.`, `Employee Name`) VALUES ('S31002', 'T Gilbert');INSERT INTO `project\_information`.`employee` (`Employee No.`, `Employee Name`) VALUES ('S13210', 'W Richards');INSERT INTO `project\_information`.`employee` (`Employee No.`, `Employee Name`) VALUES ('S10034', 'B James');
3. CREATE TABLE `project\_information`.`projects` (`Project Code` VARCHAR(45) NOT NULL, `Project Title` VARCHAR(45) NULL, `Project Manager` VARCHAR(45) NULL, `Project Budget` VARCHAR(45) NULL, PRIMARY KEY (`Project Code`));
4. INSERT INTO `project\_information`.`projects` (`Project Code`, `Project Title`, `Project Manager`, `Project Budget`) VALUES ('PC010', 'Pensions System', 'M Phillips', '24500');

INSERT INTO `project\_information`.`projects` (`Project Code`, `Project Title`, `Project Manager`, `Project Budget`) VALUES ('PC045', 'Salaries System', 'H Martin', '17400');

INSERT INTO `project\_information`.`projects` (`Project Code`, `Project Title`, `Project Manager`, `Project Budget`) VALUES ('PC064', 'HR System', 'K Lewis', '12250');

1. CREATE TABLE `project\_information`.`departments` (`Department No.` VARCHAR(45) NOT NULL, `Department Name` VARCHAR(45) NULL, PRIMARY KEY (`Department No.`));
2. INSERT INTO `project\_information`.`departments` (`Department No.`, `Department Name`) VALUES ('L004', 'IT');INSERT INTO `project\_information`.`departments` (`Department No.`, `Department Name`) VALUES ('L023', 'Pensions');INSERT INTO `project\_information`.`departments` (`Department No.`, `Department Name`) VALUES ('L028', 'Database');INSERT INTO `project\_information`.`departments` (`Department No.`, `Department Name`) VALUES ('L008', 'Salary');INSERT INTO `project\_information`.`departments` (`Department No.`, `Department Name`) VALUES ('L009', 'HR');
3. CREATE TABLE `project\_information`.`budgets` (`Project Code` VARCHAR(45) NOT NULL,`Project Budget` VARCHAR(45) NULL, PRIMARY KEY (`Project Code`));
4. INSERT INTO `project\_information`.`budgets` (`Project Code`, `Project Budget`) VALUES ('PC010', '24500');INSERT INTO `project\_information`.`budgets` (`Project Code`, `Project Budget`) VALUES ('PC045', '17400');INSERT INTO `project\_information`.`budgets` (`Project Code`, `Project Budget`) VALUES ('PC064', '12250');
5. CREATE TABLE `project\_information`.`salaries` (`Employee No.` VARCHAR(45) NULL,`Project Code` VARCHAR(45) NULL, `Hourly Rate.` VARCHAR(45) NOT NULL, PRIMARY KEY (`Hourly Rate`));
6. INSERT INTO `project\_information`.`salaries` (`Employee No.`, `Hourly Rate`, `Project Code`) VALUES ('S10001', '22.00', 'PC010');INSERT INTO `project\_information`.`salaries` (`Employee No.`, `Hourly Rate`, `Project Code`) VALUES ('S10001', '18.00', 'PC045');INSERT INTO `project\_information`.`salaries` (`Employee No.`, `Hourly Rate`, `Project Code`) VALUES ('S10030', '18.50', 'PC010');

INSERT INTO `project\_information`.`salaries` (`Employee No.`, `Hourly Rate`, `Project Code`) VALUES ('S21010', '21.00', 'PC010');INSERT INTO `project\_information`.`salaries` (`Employee No.`, `Hourly Rate`, `Project Code`) VALUES ('S21010', '17.50', 'PC064');INSERT INTO `project\_information`.`salaries` (`Employee No.`, `Hourly Rate`, `Project Code`) VALUES ('S10010', '21.75', 'PC045');

INSERT INTO `project\_information`.`salaries` (`Employee No.`, `Hourly Rate`, `Project Code`) VALUES ('S31002', '25.50', 'PC045');INSERT INTO `project\_information`.`salaries` (`Employee No.`, `Hourly Rate`, `Project Code`) VALUES ('S13210', '17.00', 'PC045');INSERT INTO `project\_information`.`salaries` (`Employee No.`, `Hourly Rate`, `Project Code`) VALUES ('S31002', '23.25', 'PC064');

INSERT INTO `project\_information`.`salaries` (`Employee No.`, `Hourly Rate`, `Project Code`) VALUES ('S10034', '16.50', 'PC064');

**List SQL Queries for Browse, Search, and Lists**

To test the effectiveness of my normalization procedures in practice, I ran a few SQL queries to invoke any anomalies that would occur if done incorrectly. These anomalies are known as **modification anomalies** and include update, insert, and delete anomalies. **Update anomalies** occur when you must update more than one record in a table for a single update to be effective. **Insert anomaly** occurs either when you need two pieces of data to update one cell or the update causes redundancies. **Delete anomalies** occur when deleting data would cause an unintended removal of other records. I started by using a query to show a **list** of all the tables I created in the *project\_information* database. This returned a comprehensive list of my tables. Next, I wanted to run a simple **search query** for select information from my *project\_info* table. The returned table was complete and without anomalies. I then tested my ability to **order** my results based on ascending integer values and was successful. Finally, I wanted to try the more complex task of joining back together two decomposed tables using **inner join** statement. This rejoined the original table without missing or added data. This proved that my normalization removed one of the **join dependencies** present in the unnormalized table, *UNF*. Again, I displayed my code in 9.5 font size for abbreviation’s sake.

1. USE project\_information; SHOW TABLES; SELECT \* FROM employee;

USE project\_information;

1. SELECT `project\_info`.`Project Code`,`project\_info`.`Employee No.`, `project\_info`.`Hourly Rate` FROM project\_info;
2. ORDER BY `project\_info`.`Project Code`;
3. USE project\_information; SELECT`budgets`.`Project Code`, `projects`.`Project Budget` FROM budgets, projects WHERE `budgets`.`Project Code` = `projects`.`Project Code` ORDER BY `budgets`.`Project Code`;
4. SELECT `project\_info`.`Project Code`, `project\_info`.`Project Title`, `project\_info`.`Project Manager`, `project\_info`.`Project Budget`, `project\_info`.`Employee No.`, `project\_info`.`Department No.`, `project\_info`.`Department Name`, `project\_info`.`Hourly Rate`, `employee`.`Employee Name` FROM `project\_information`.`project\_info` INNER JOIN `project\_information`.`employee` ON `employee`.`Employee No.` = `project\_info`.`Employee No.`;

**References**

1. Wenzel, K. (2021). Database Normalization - in Easy-to-Understand English. [online] Essentialsql.com. Available at: <https://www.essentialsql.com/database-normalization>
2. Draw.IO. (2021). Flowchart Maker & Online Diagram Software. Retrieved 26 July 2021, from <https://app.diagrams.net/>
3. Sharpe, T. (2021). Entity Relationship Diagram (ERD) Tutorial - Part 1. Retrieved 26 July 2021, from <https://www.youtube.com/watch?v=QpdhBUYk7Kk>
4. Green, B. (2021). Logical Database Design and ER Diagrams. Retrieved 26 July 2021, from <https://www.youtube.com/watch?v=ZBgXb66Ckz>0
5. Joshi, K. (2021). How to do database normalization. Retrieved 26 July 2021, from <https://www.youtube.com/watch?v=UDFRhj_K508>
6. Unknown (2021). Normalization. Retrieved 28 July 2021, from SMU Moodle <https://moodle.stmartin.edu/mod/resource/view.php?id=642424>