Assignment 2

Advanced database and big data

Advanced Database and Big Data

U14440

Kieran Best

Reem Khider

Table of Contents

[Introduction 1](#_Toc134621706)

[Implementation 1](#_Toc134621707)

[PHP Database 1](#_Toc134621708)

[Data Preparation 1](#_Toc134621709)

[Technique 2](#_Toc134621710)

[SQL Queries 2](#_Toc134621711)

[MongoDB Database 3](#_Toc134621712)

[Data Preparation 3](#_Toc134621713)

[Technique 4](#_Toc134621714)

[MongoDB Queries 5](#_Toc134621715)

[Results 7](#_Toc134621716)

[Conclusion 7](#_Toc134621717)

[Splitting of Tasks 7](#_Toc134621718)

[Links 7](#_Toc134621719)

# Introduction

The purpose of this report is to explain the process in which we successfully created a PHP/MySQL application. The application was designed to handle data for a company’s private information such as:

* Employee Job Details
* Employee Salary
* Department Locations

This was implemented in MySQL to handle all data and was accessible from a PHP application that enabled the input of data.

# Implementation

## SQL Database

### Data Preparation

To prepare the data the first step was to normalise the data to be able to simplify accessibility of data. Creating links between databases using primary and foreign keys to access one another. Our initial normalised logical data model can be seen in Figure 1.

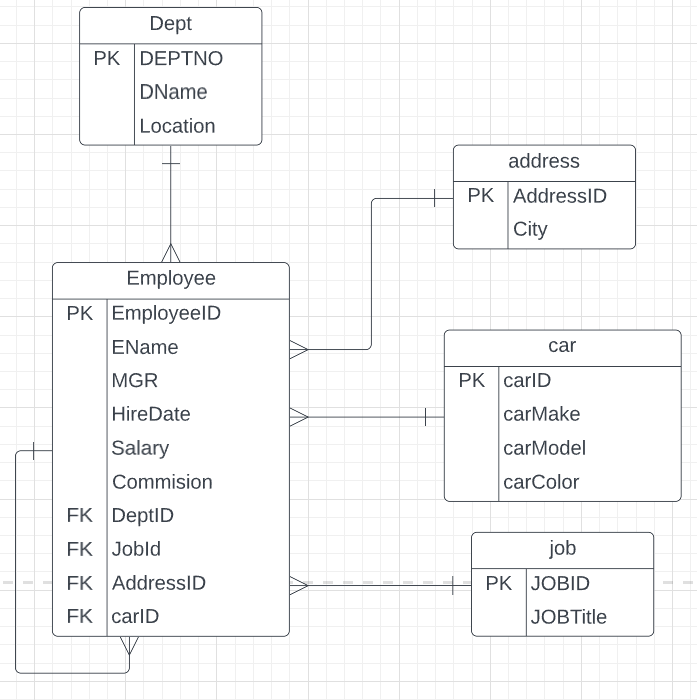


Figure : Normalised LDM

We decided that all connections from the “Employee” collection should be ‘many to 1’ as many employees may hold the same qualities as other employees and these should be referenced using foreign keys to save memory if the dataset was to become much larger.

* Many employees may work in the same department
* Many employees may live in the same city
* Many employees may have the same company car
* Many employees may have the same job

From there creating the database in MySQL and inserting data, this insertion script can be found here:

<https://github.com/Reem-313/Database_employee/blob/main/emloyeedb.sql>

### Technique

From here creation of the front-end using PHP to connect to the database as well as enabling user input to be able to add to the PHP database was implemented. This was done by firstly creating a connection to the MySQL database (Figure 2), then creating a form that handles user input in the form of PHP variables that can post to the connected database (Figure 3 & 4).



Figure : Connection TO SQL Database

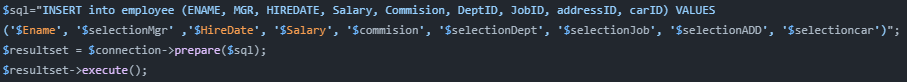


Figure : Insertion of Employee into SQL Database

***GET PICTURE OF FORM***

From here this developed to allow the user to input an employee with or without a commission or company car and the ability to add to the department table. We then created a table to display all entries from the employee and department table, this is refreshed once a new entry is inputted by the user.

### SQL Queries

The following queries have been created to match and display attributes.

#### Query 1

This query displays all data that is linked to an employee, it shows all attributes instead of ID numbers.



Figure : Input for Query 1

#### Query 2

This query captures how many people work in each department.

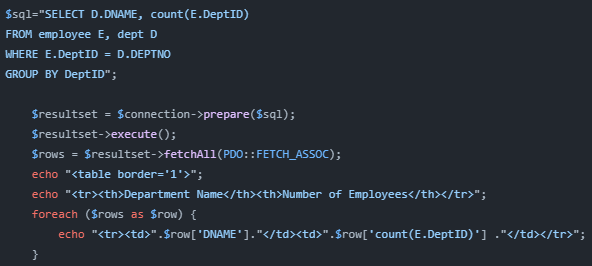


Figure : Input for Query 2

***OUTPUT FOR QUERY 2***

#### Query 3

This query captures which employees have a salary above 1000, a commission above 0 and have a company car. It outputs their name, salary, commission, and car ID.

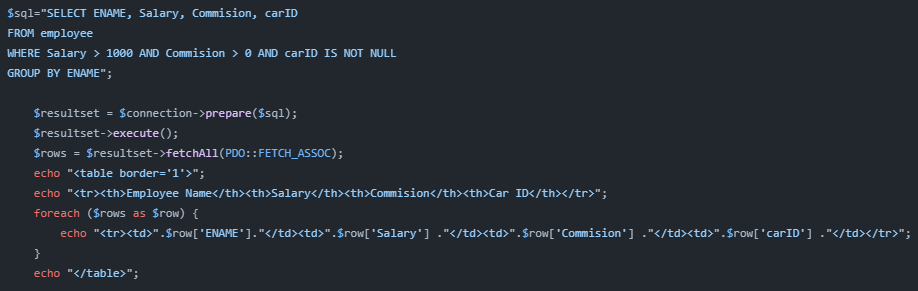


Figure : Input for Query 3

***OUTPUT FOR QUERY 3***

## MongoDB Database

### Data Preparation

When preparing the format for data for the MongoDB, it was important we remembered the output is in a JSON format. Whilst remembering this and remembering that it is usually outputted in a single format for all data, we denormalized the PHP data to obtain a logical data model that includes cardinalities as can be seen in Figure 4.

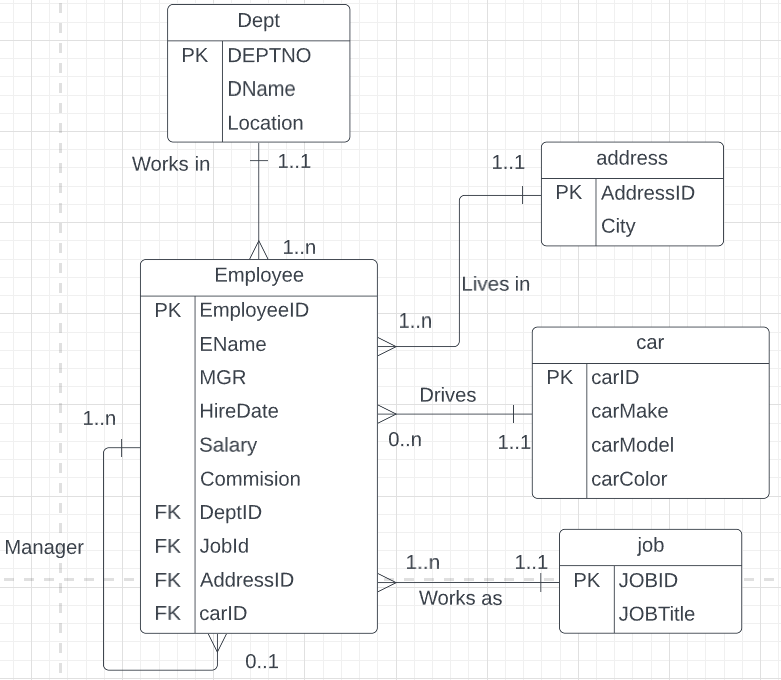


Figure : Logical Data Model with Cardinalities

The reason we included cardinalities into this diagram is to be able to fully understand what can and can’t be normalized. From Figure 2 we can see that everything can be migrated into the “Employee” collection:

* “DepNO” can be migrated into “Employee” as an employee will only work in 1 location, therefore we can remove the “DeptNo” attribute as we will no longer need to access a unique department ID
* “Address” can be migrated into “Employee” as an employee will only live in 1 location, therefore we can remove the “AddressID” attribute as we will no longer need to access a unique address ID
* “Car” can be migrated into “Employee” as an employee may or may not have access to a company car, and if they do, they only have access to 1. This means a singular object can be displayed for each employee, and therefore will no longer need the unique car ID
* “Job” can be migrated into “Employee” as it is only a single attributed collection, besides from the id, which once migrated can be removed as it is redundant data

This then resulted in the logical data model shown in Figure 3. The reason we combined it all data into 1 collection is because every employee has a 1 to 1 relationship with every attribute, they do not connect to multiple instances of said collections. Therefore, we felt it was appropriate to remove connections between collections and instead store each employees data in 1 collection.

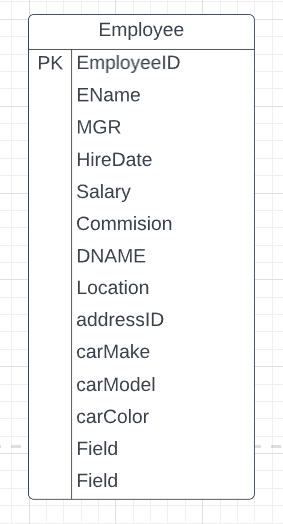


Figure : Denormalized LDM

Once Denormalized it allowed us to alter the existing data into a prefixed format that could then be used to create the template for all new data input. The existing data was manipulated to the following format in Figure 4:

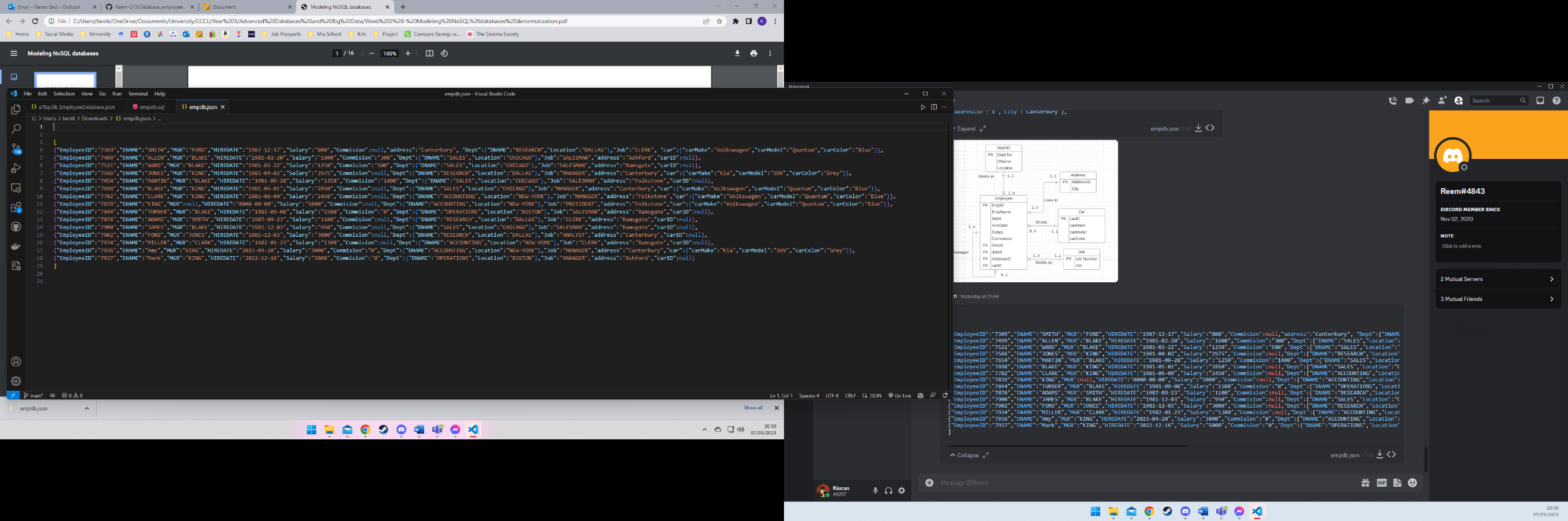


Figure : MongoDB JSON Data

### Technique

The same technique was used to create the front end when using MongoDB as was used for SQL. Once the logical data model was updated to suit MongoDB, to start the connection the following code was used in Figure 7.

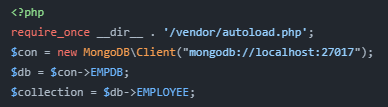


Figure : Connection to MongoDB

Once connected the format in Figure 6 was used to display the data inside the php page to allow users to view the contents of the database, this was done through fetching of the data and displaying it in a table as can be seen in Figure 9.

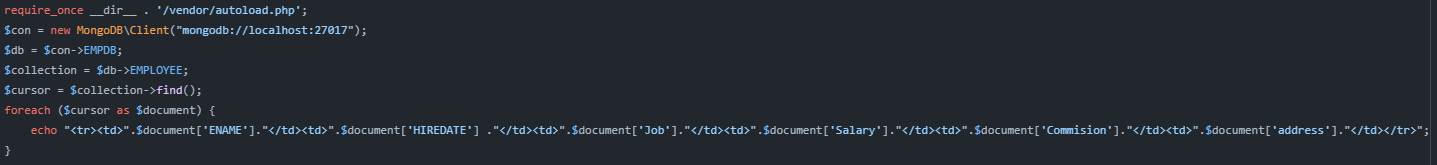


Figure : Retrieval of Employee Data

After that creating forms to enable the user to create their own additions into the employee and department tables.

### MongoDB Queries

The following queries have been created to match and display particular key values.

#### Query 1

This query captures all employees that work in accounting, and were hire between 1st June 1981 and 1st June 1982, it then displays their name and salary.

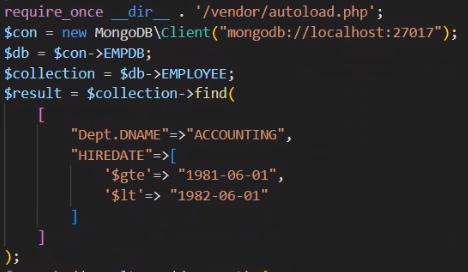


Figure : Input for Query 1

***OUTPUT FOR QUERY 1***

#### Query 2

This Query captures all employees who are employed as a manager, have a salary of over 2000 and do have a company car. It then displays their name, job, salary and address.

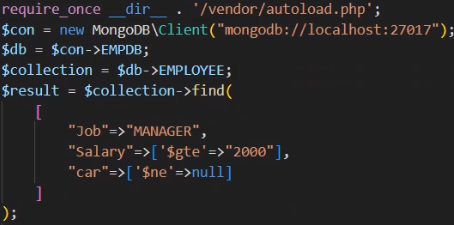


Figure : Input for Query 2

***OUTPUT FOR QUERY 2***

# Results

The final deliverable for this project allowed the user to specify which database they would like to work in and enabled them the choice to easily add to both employee and department tables in both SQL and MongoDB data tables.

# Conclusion

Creation of the SQL design and database was a simple implementation that could easily be implemented into the DBMS. However, implementing a front-end PHP page proved harder than anticipated due to the different attributesneeded.

Creation of the MongoDB was frustrating due to the logic behind it, this caused constant second-guessing and eventually we just had to pick our method and ensure valid justification. Putting it all into one collection due to each employee only having singular entries for each attribute seemed like the right way forward.

The issue we found with this project is that only 1 person can have full control of the DBMS, therefore it limits the other person’s ability to get involved with the project. For example, Reem had access to the MySQL database and therefore Kieran could not work on the front-end development due to constant testing needed which requires access to the DBMS. It wasn’t until reaching the creation of MongoDB that Kieran could become self-resilient and create queries that he could test on his own.

# Splitting of Tasks

Tasks were divided between the 2 of us based on our strengths and weaknesses, while there was some major overlapping in working individually, we did work well independently as well as coherently. Designing of the databases was a joint effort in proposal of ideas and discussing thoroughly to reach an outcome that was both agreed on.

Creating the PHP and database connectivity was primarily Reem’s task, however this was worked dually when obstacles occurred, such as connecting the MongoDB to PHP whilst keeping the SQL connection viable.

Creating the front end was a joint effort as we are both fully capable in doing this, therefore just speeding up the process whilst working on this.

Creating the queries was Kieran’s task with Reem’s input on testing to ensure full working queries.

Writing the report was also done by Kieran alongside creation of the project.

# Links

Github Repository - https://github.com/Reem-313/Database\_employee