Assignment 2

Advanced database and big data

Advanced Database and Big Data

U14440

Kieran Best

Reem Khider

Table of Contents

[Introduction 1](#_Toc134302490)

[Implementation 1](#_Toc134302491)

[PHP Database 1](#_Toc134302492)

[Data Preparation 1](#_Toc134302493)

[Technique 2](#_Toc134302494)

[MongoDB Database 2](#_Toc134302495)

[Data Preparation 2](#_Toc134302496)

[Technique 2](#_Toc134302497)

[Results 2](#_Toc134302498)

[Conclusion 2](#_Toc134302499)

[Splitting of Tasks 3](#_Toc134302500)

[Links 3](#_Toc134302501)

# 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

## PHP 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.

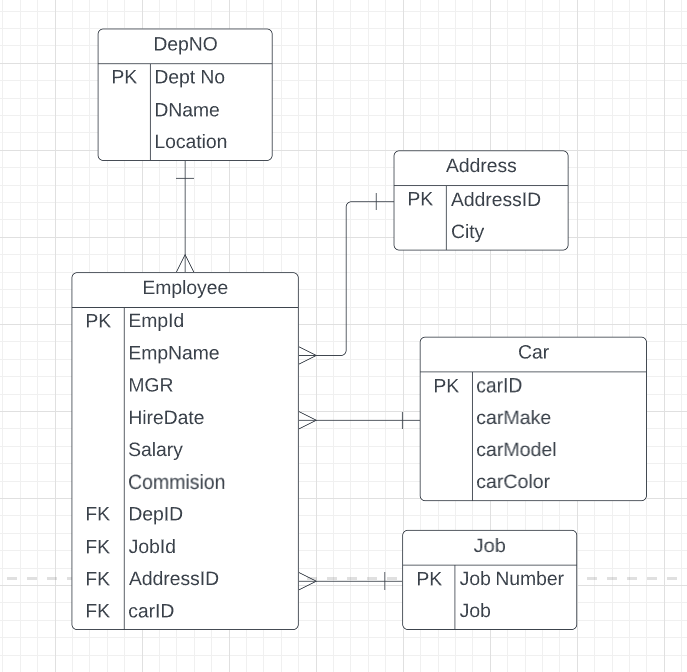


Figure 1: Normalised LDM

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.

* Many employees may work in the same department
* Many employees may live in the same city
* Many employees may have the same company vehicle
* 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, before creating a form that handles user input in the form of PHP variables that are able to post to the connected

## 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 2.

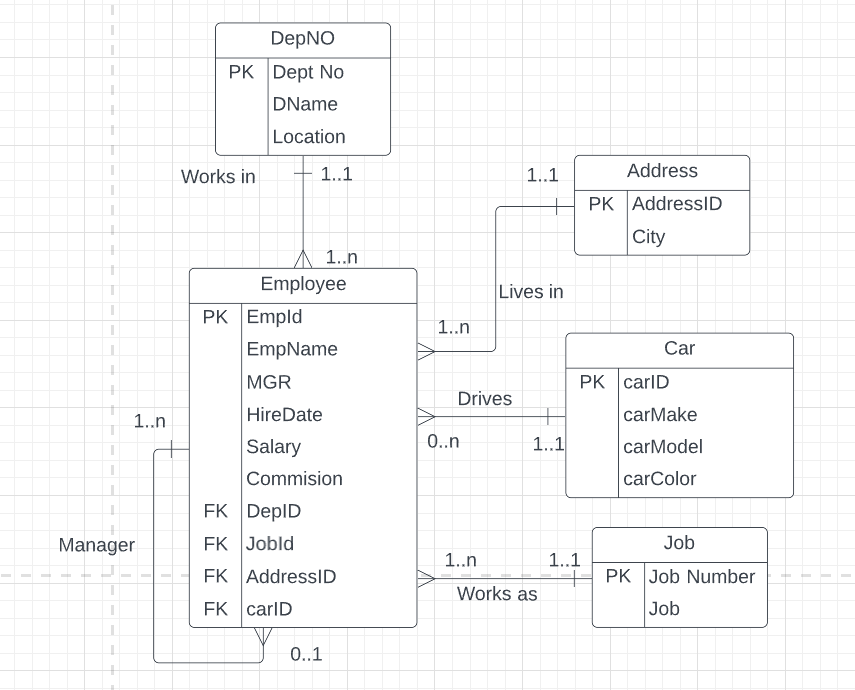


Figure 2: Denormalized LDM

### 

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

### Technique

## Results

## Conclusion

# Splitting of Tasks

# Links

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