Student: Iva Jorgusheska **11/10/2023**

Student ID: 11114620

<u>Coursework: Database Design and Implementation</u>

<u>Milestone 1: Implementation</u>

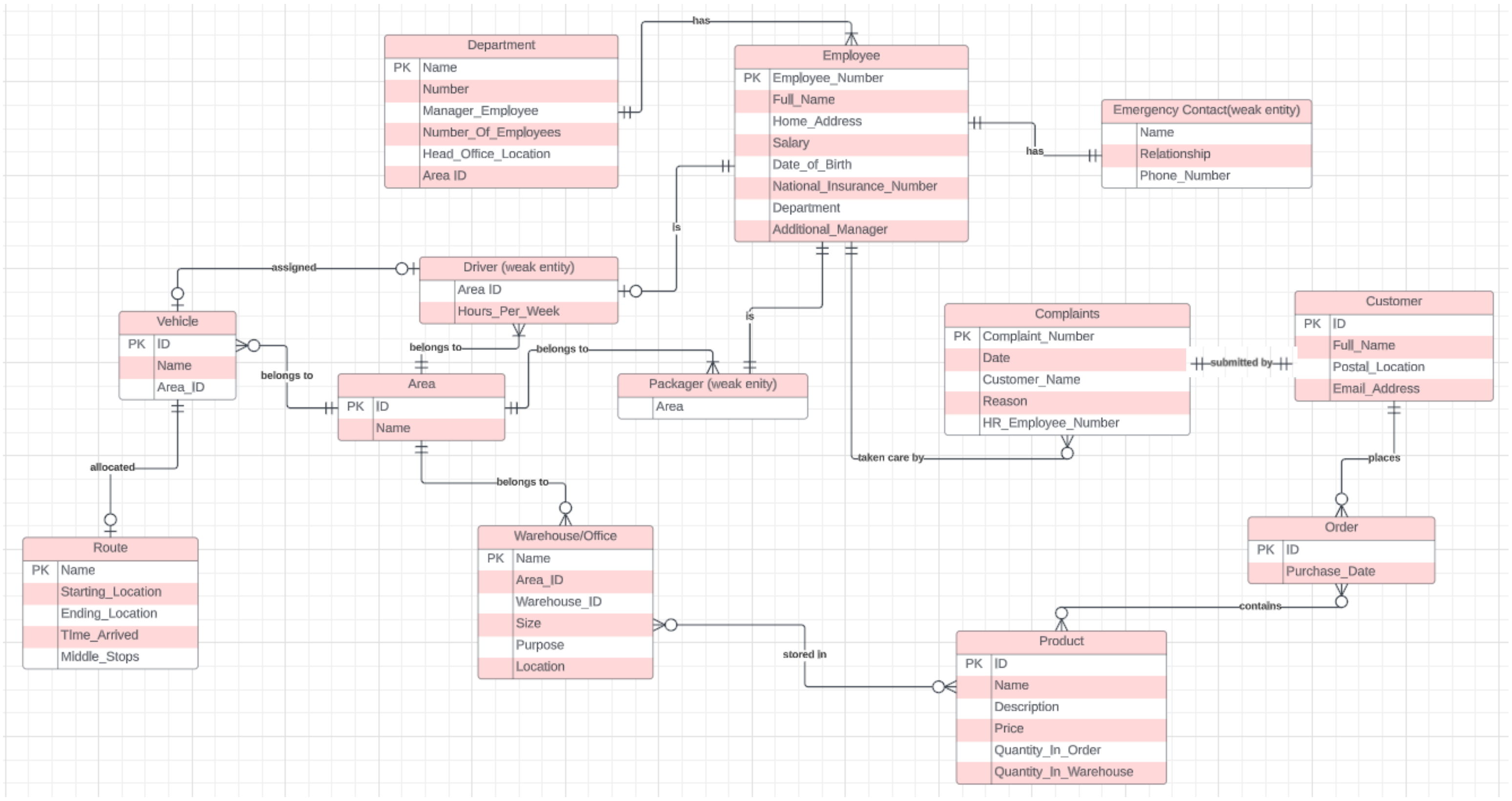
Course Code: COMP23111

Databases Systems

Table of Contents

Task 1: ERD

In this section, I provide an Entity Relationship (ER) Model based on the specification the customer has given. I focus on the conceptual data model meeting the requirements of the client, in this case, Kilburnazon. The ER diagram presented below is abstract enough in order that the client can understand the entities, attributes, and relationships, while still having the needed complexity for the future design phases.



I designed a database with 13 entities: Department, Employee, Emergency Contact, Driver, Packager, Area, Vehicle, Route, Warehouse, Product, Order, Customer, and Complaint. These entities are enough to represent all the necessary attributes and the connections between the entities, both the obvious and the more complex ones. As a primary key for each entity, I put the attribute that is unique for each instance of the entity. For example, each employee has its Employee Number, which is a great example of a unique attribute, and one which is an integer, which makes it the perfect primary key. In the case of Department, Route, and Warehouse entities I decided the primary key to be the name of the corresponding entity. I believe it is the right choice for these entities since in the specification it was explicitly mentioned that these will be unique. Moreover, it is more natural and convenient for people (the users of the database) to distinguish them by name. In some other entities like Product, Order, Area, Vehicle, and Customer which did not have any unique attribute I added an ID attribute which will be unique and serve as a primary key. There are also 3 weak entities: Emergency Contact, Packager, and

Driver. All of them would not exist if the Employee entity did not exist. That is why as a primary key they will use the primary key of the Employee entity and their additional attribute respectively.

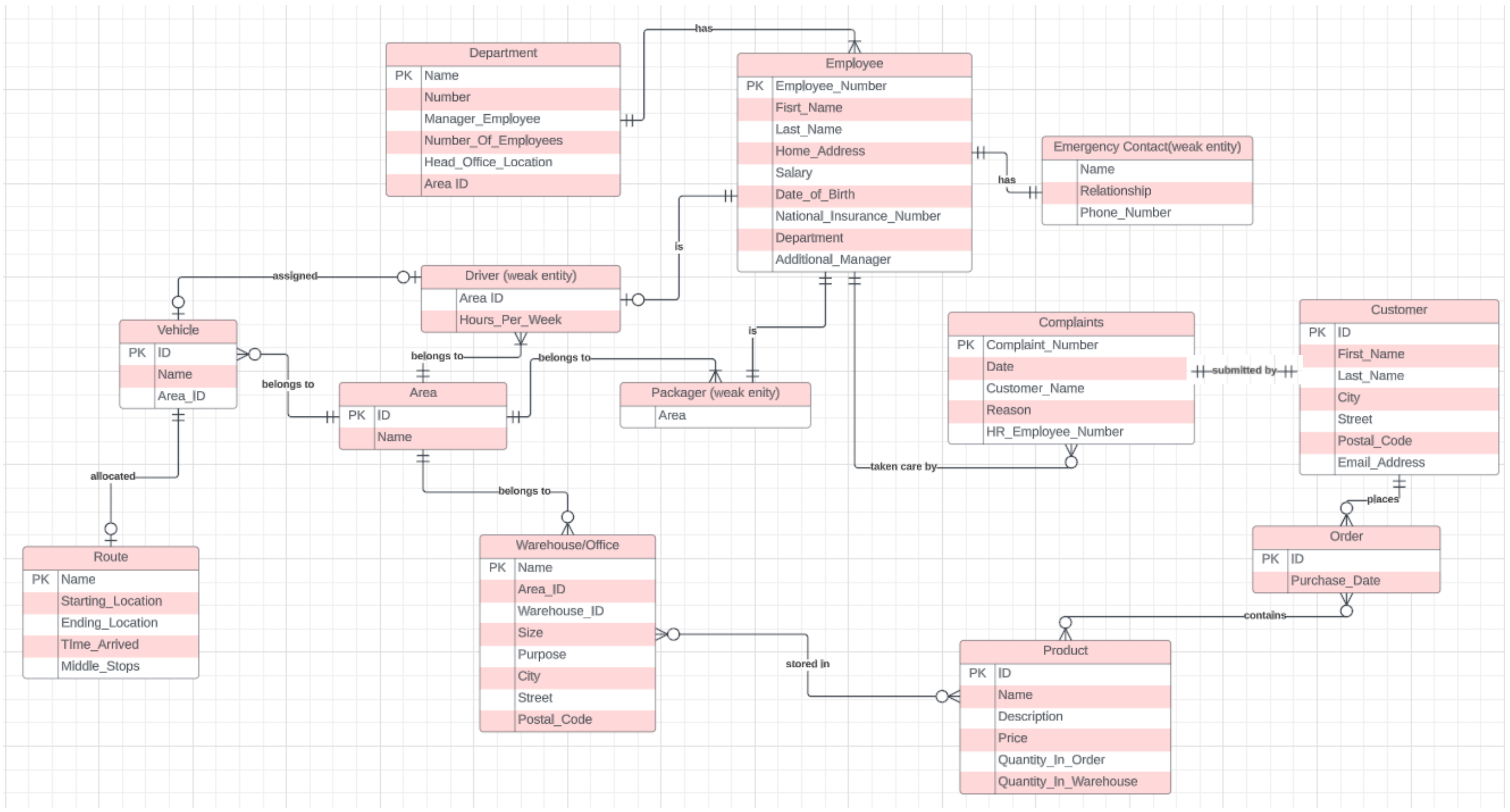
You can see that the entities contain all the necessary attributes. For example, in the Employee entity, I have only put the 'Additional_Manager' attribute, and not the department one. This is the case since the department manager will be the same for all the employees of that department and we can query it from the department entity the employee is connected to. However, later in the specifications it was written that managers can supervise workers from multiple departments, hence this is the one we need to make specific record for each employee.

The relationships represent the connection between the entities. Every department contains one or more employees. Each employee has an emergency number. Driver and packagers are employees. Driver is assigned a vehicle or not, which is then allocated a route or not. Both vehicles and drivers belong to a certain area and are assigned depending on it. Each customer can place zero or more orders, which contain zero or more products. These products are allocated in warehouses, in 0 if there is no left in stock, or in more of them. Each complaint is submitted by one customer and taken care of by one employee.

Task 2 – Normalisation

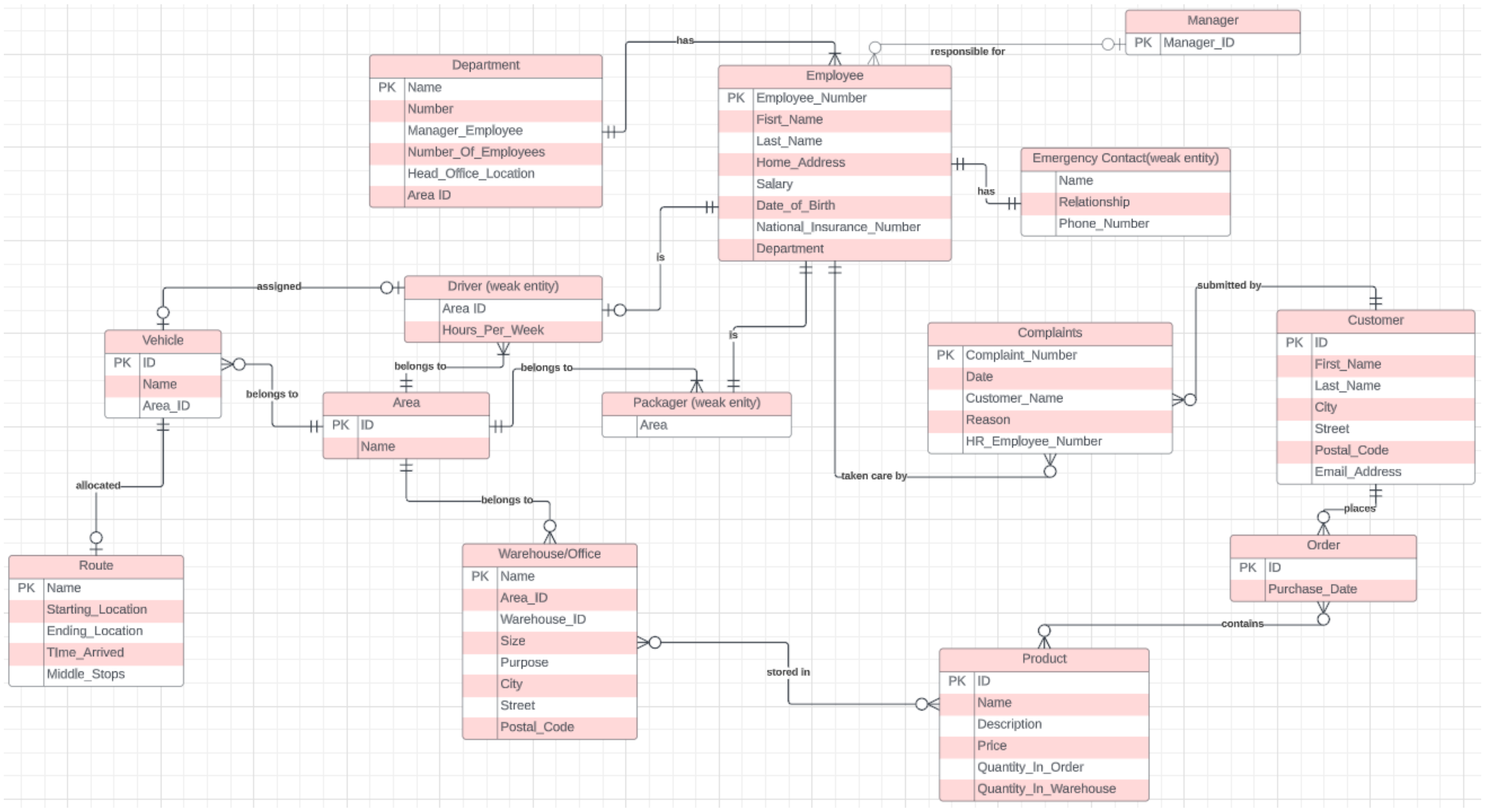
We follow the first 3 Normal Forms to make sure we prevent data redundancy, offer consistency and flexibility, and enforce relation integrity.

First Normal Form – 1NF



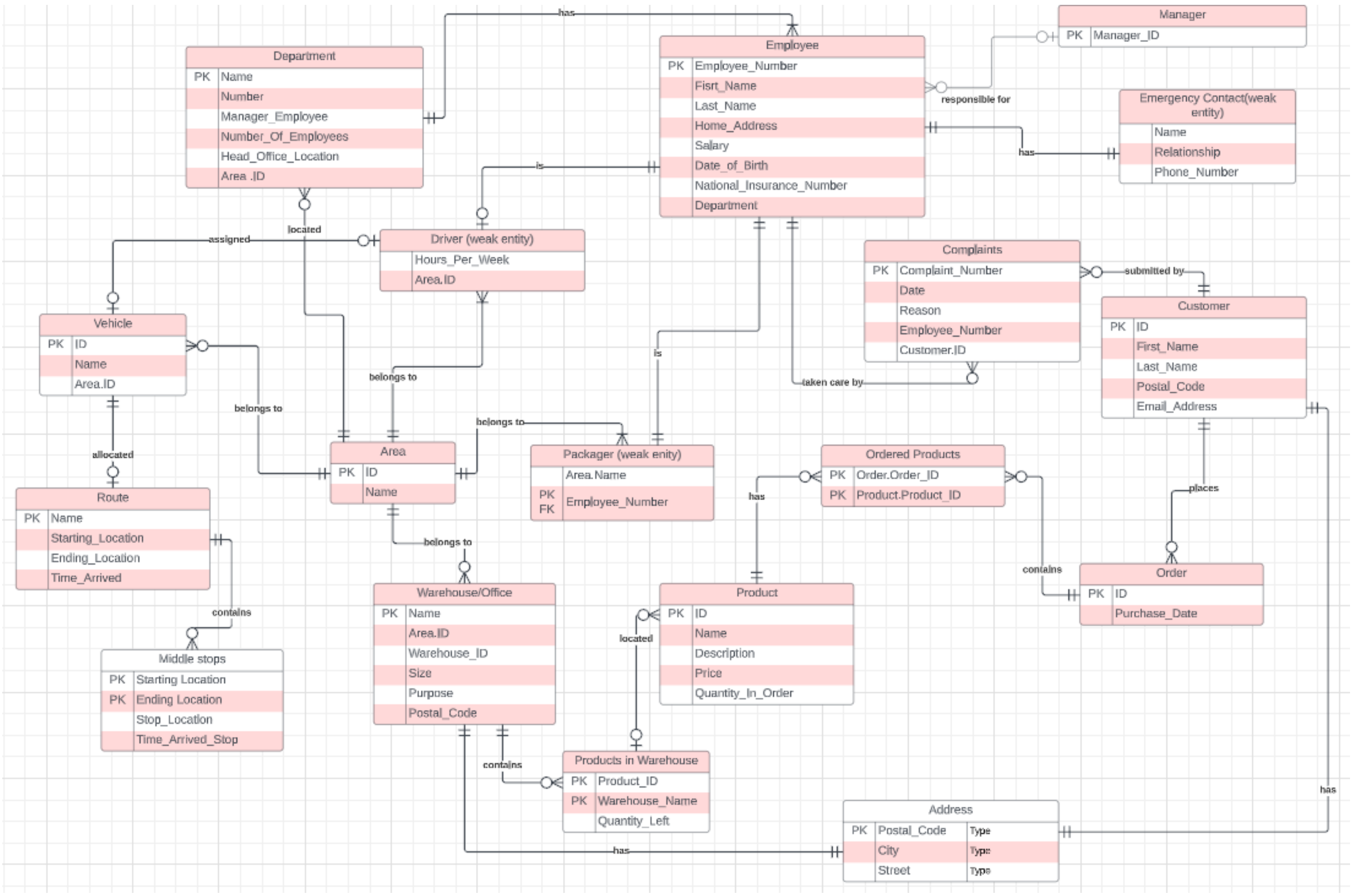
In order to make sure there are no repeating groups and that the values in each table are atomic I make small changes. In the Customer and Employee tables, I divide the Name attribute into First Name and Last Name. In this way, if we need to query a customer or employee by only one of them, we can be sure that the values will be atomic. I leave the full name for the Emergency Contact table. If we query it like a one value like in this case, we consider it atomic. I made the same separation from location to: city, street, and postal code in Customer, Warehouse/Office tables for the same reasons.

Second Normal Form – 2NF



In the Second Normal Form, all the non-key attributes must be functionally dependent on the primary key. To make sure that I do not violate this rule, I created another table called Manager which is connected to the Employee table it is responsible for. In this way, I removed the ‘Additional_Manager’ attribute from the Employee table which was not dependent on the Employee Number, since an employee can be supervised by any manager.

Third Normal Form – 3NF

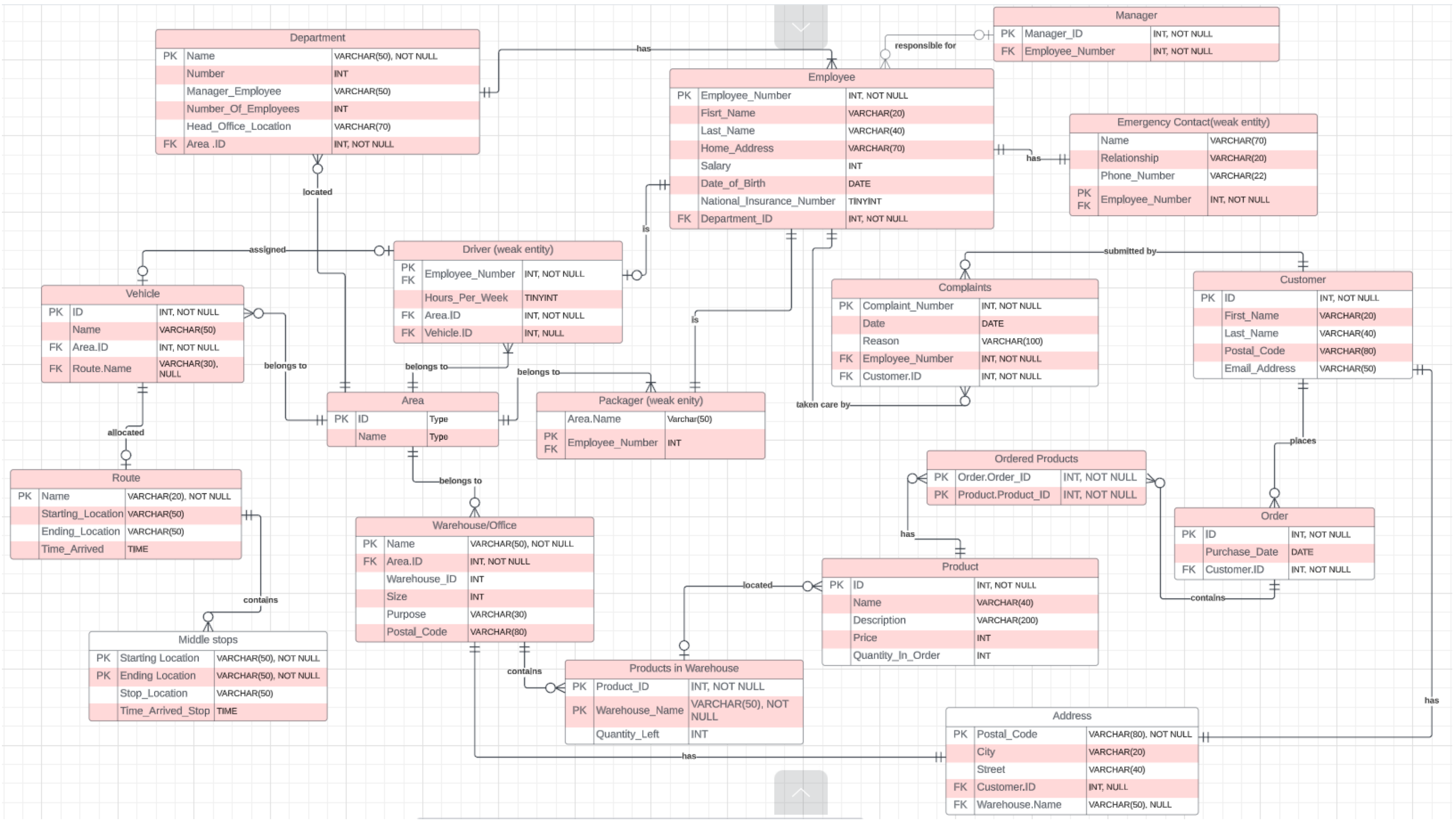


In the Third Normal Form, we make sure there are no transitive dependencies. Since City and Street can be determined by postal code, we should not put all the three as attributes in one table. I rather created another table called Address which takes the postal code as a primary key and gives us the street and the city. In this way, I removed the city and street attributes from the customer and warehouse/office tables. In the address table, if we look for the address of warehoue, customer id will be null and vice versa. The same logic follows the creation of the Middle Stops table which can be determined by the starting and ending location of each route, and therefore there is transitive dependency. I created another table that given the starting and ending locations as primary keys, has the location of each stop and the arrival time to each of these stops.

Finally, in order to make sure our database is normalized, we can not have many to many relationships since we can end up in a loop and violate the atomicity. That is why I created a joining table for the connection between Order and Product called Ordered Products, and between Product and Warehouse/Office called Products in Warehouse. The primary keys of both of the tables we are connected are the primary keys of the joining table, which determine the values of the other entities of those tables. In the Product table we keep track of the quantity of each product in the specific order, and in the Products in Warehouse table we keep track of the amount of products left in each warehouse. In the Ordered Products table, we join the Order and Product table making it easier to record which products are requested in a specific order.

Task 3 – Physical Data Model

The physical data model consists of a lot of physical details about the database management system. It is the step before implementing the database. It is technical and its purpose is not for the customer to understand it, but to make sure that the implementation goes smoothly and all the developers are on the same page on how the database should look like.



In this phase, I have added Foreign Keys which are a way of connecting two tables. In the case where we have weak entities, the primary key of the parent is the foreign key of the child. At the same time, the primary key of the parent combined with an attribute from the child is also a primary key for the child table. For example, each employee belongs to a department, so I have added Department_ID as a foreign key in the employee table so we can query in which department is a certain employee, or all the employees that work in a certain department. Next, we know that first a driver is allocated a vehicle that belongs to its area, and then the vehicle is allocated a route. That is why we need Area.ID foreign key in the Driver and Vehicle tables, so we can make sure they belong to the same area (they are connected with the same area). Then I put Route_Name as a foreign key in the Vehicle table so we can keep track of which route was assigned to each vehicle. For all the attributes in the tables, I have put the expected type. The Route_Name entity in the Vehicle table can have a NULL value when no route has been allocated to the vehicle. In the same way, the Vehicle_ID can be NULL in the Driver table when a driver has not been allocated a vehicle. As I have already mentioned, the Customer_ID and Warehouse_Name will be NULL interchangeably depending on which address we are looking for. The primary keys in the tables can not have NULL values since they contain the crucial information in order for that table to exist. Consequently, those keys when have the role of foreign keys in other tables also cannot have NULL values.

I have followed the naming conventions needed for the Physical Data Model since creating the ERD in the beginning so there was nothing to change in that sense.

Finally, we have successfully designed a database model that satisfies all the requirements that can be inferred from the description of the customer, contains all the necessary technical details, and is ready for implementation.