Database Design
COMP23111

Jaeyoung Kim
ID:10739093

November 2023

Contents

1 ERD

In Task 1, I opted to categorize entities such as 'Packagers,' 'Drivers,' 'HR Employees,' and 'Management Employees' under a broader category of 'Employees,' considering them as possible employee classifications. Regarding the 'Department' entity, the department number wasn't specified as a unique identifier. Therefore, I assumed it to be a numerical designation for each department. I established 'Head Office Location' and 'Department Name' as a composite key for the 'Departments' entity. Additionally, it includes 'Manager's Employee Number' as an attribute, reflecting the one-to-one relationship between managers and departments.

Regarding the 'Area' entity, an 'Area' can encompass either 'Warehouses' or 'Offices.' Employees from the 'HR' and 'Management' departments operate in 'Offices,' while 'Drivers' and 'Packagers' are associated with 'Warehouses.' Given that 'Drivers' and 'Packagers' report to 'Warehouses,' their relationship is characterized as many-to-many. Consequently, I introduced mapping tables for each entity. 'Drivers' can be assigned to multiple routes, and some 'Drivers' might not be assigned any routes, resulting in a many-to-many relationship with the 'Routes' entity. I created a mapping table called 'Allocate,' with an added optionality feature. Since the estimated time of arrival depends on each route allocation, I separated it from the 'Routes' entity and included it in the 'Allocate' entity. Moreover, 'Drivers' can be assigned to 'Vehicles,' establishing another many-to-many relationship. A mapping table named 'Assign' was introduced to handle this relationship.

Concerning 'Warehouses', as they store products, the relationship between 'Warehouses' and 'Products' entities is deemed many-to-many, as is the case with 'Products' and 'Orders' entities. Regarding keys, I utilized unique IDs or names as specified in the instructions for entities such as 'Employees,' 'Vehicles,' 'Routes,' 'Area,' 'Products,' and 'Complaints.' For entities using an employee number as the primary key and for mapping tables, I assigned unique IDs as their primary keys. After addressing the relationships between entities, appropriate foreign keys were added to the relevant entities.

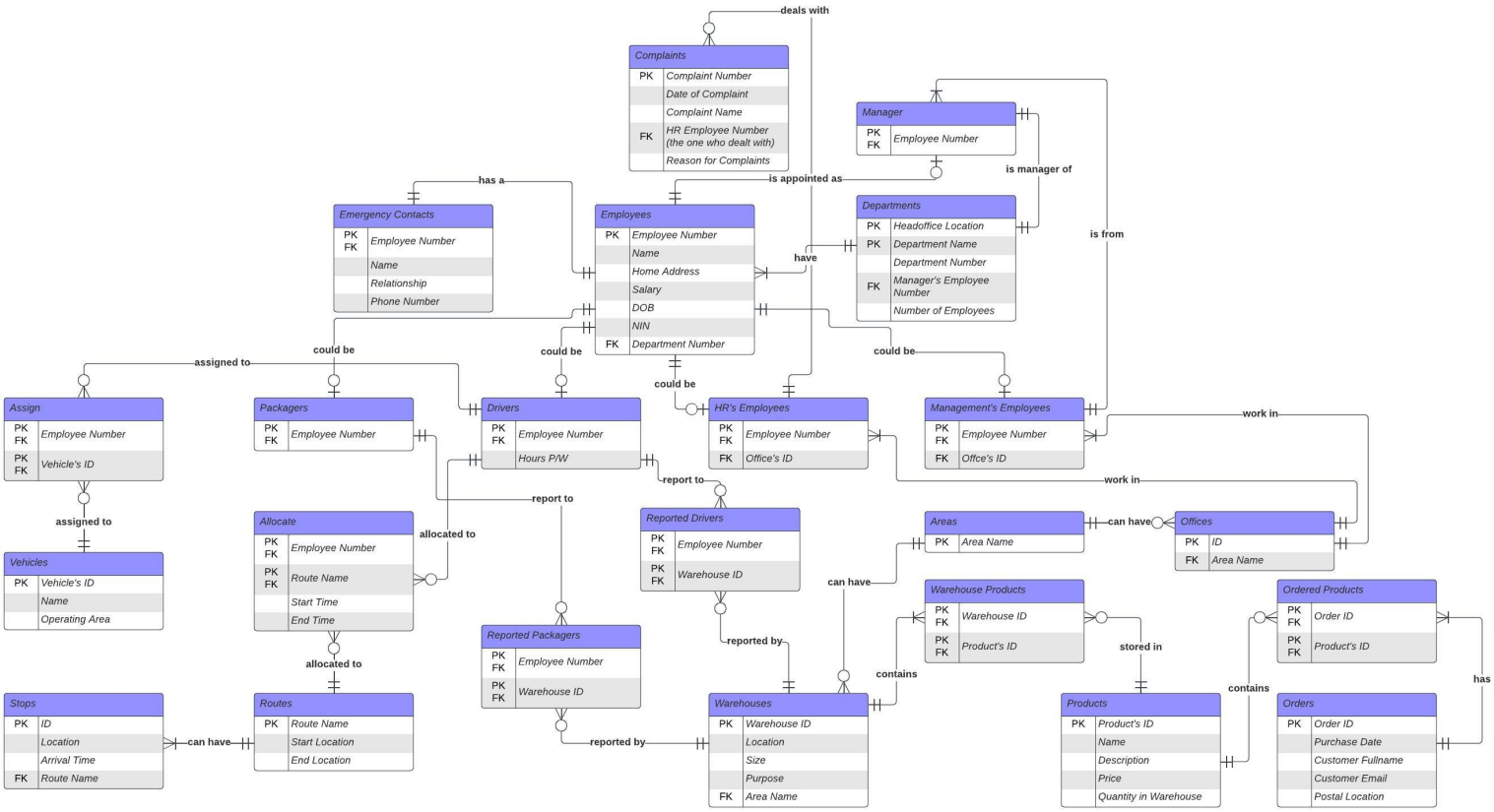

Figure 1: task 1 diagram has to be changed

2 Normalisation

In the first normalization phase of Task 2, adhering to the 1NF requirement, the table must lack repeating groups, and values in each column should be atomic. Recognizing the possibility of multiple operating areas for a single vehicle, I introduced an entity called 'Vehicles Operating Areas' to enable the table to accommodate several operating areas under one vehicle ID while maintaining atomicity. Additionally, I identified the potential for multiple reasons for complaints. To address this, I established a separate table named 'Reasons for Complaints.'

Moving to the second normalization phase (2NF), a table must satisfy 1NF, and there should be no partial dependencies. Upon inspection of the 'Departments' table, it became apparent that 'Department Number' was partially dependent on 'Department Name,' despite having 'Head Office Location' and 'Department Name' as a composite key. To eliminate partial dependencies, I created a table named 'Department Name Number' and included 'Department Name' and 'Department Number' as attributes.

Advancing to the third normalization phase (3NF), a table must meet the 2NF criteria, and there should be no transitive dependencies for non-key attributes. An analysis of the 'Warehouses' table revealed transitive dependencies, where knowledge of 'Warehouse ID' allowed retrieval of 'Location,' 'Size,' 'Purpose,' and 'Area Name.' Recognizing that 'Location' is a non-key attribute, I addressed this by creating a distinct table called 'Warehouses Located Area,' incorporating 'Location' and 'Area' as attributes. This adjustment ensures the table adheres to 3NF standards.

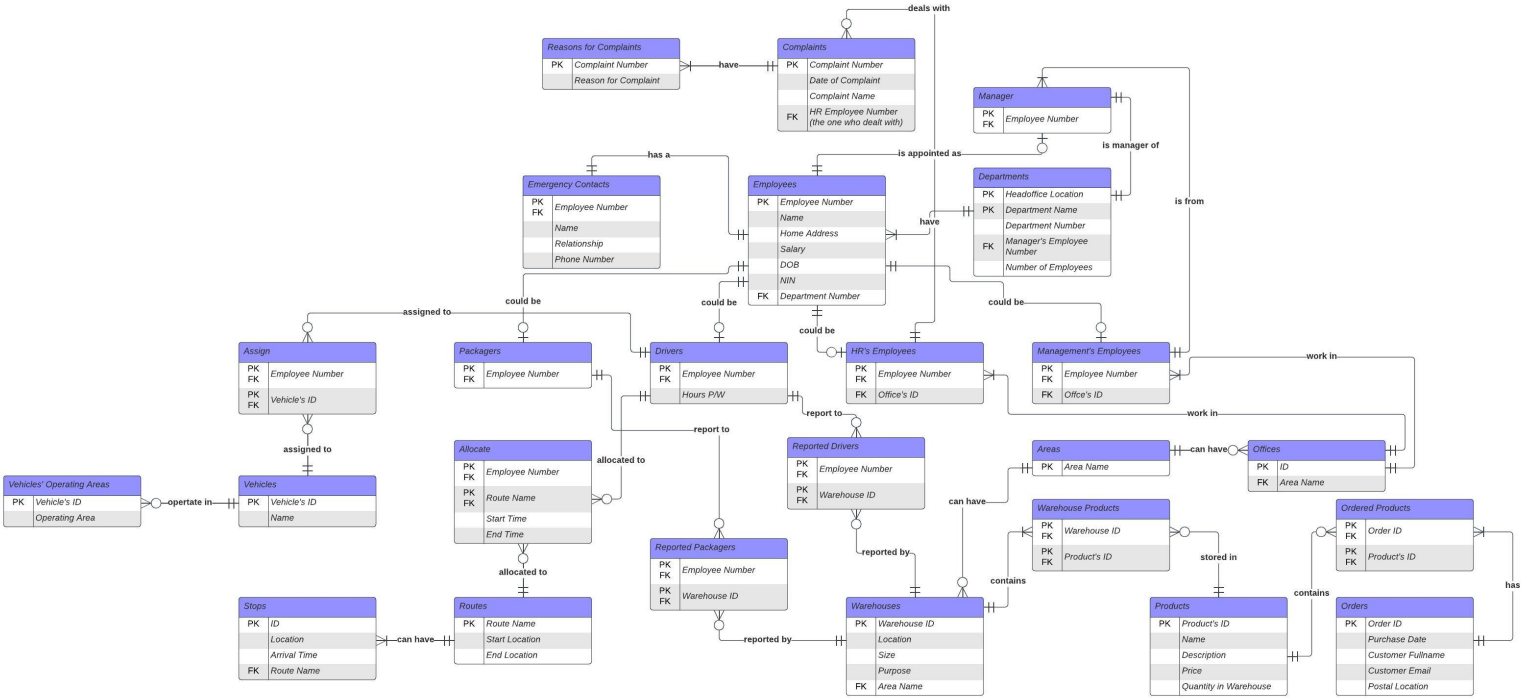

Figure 2: Tables in 1NF

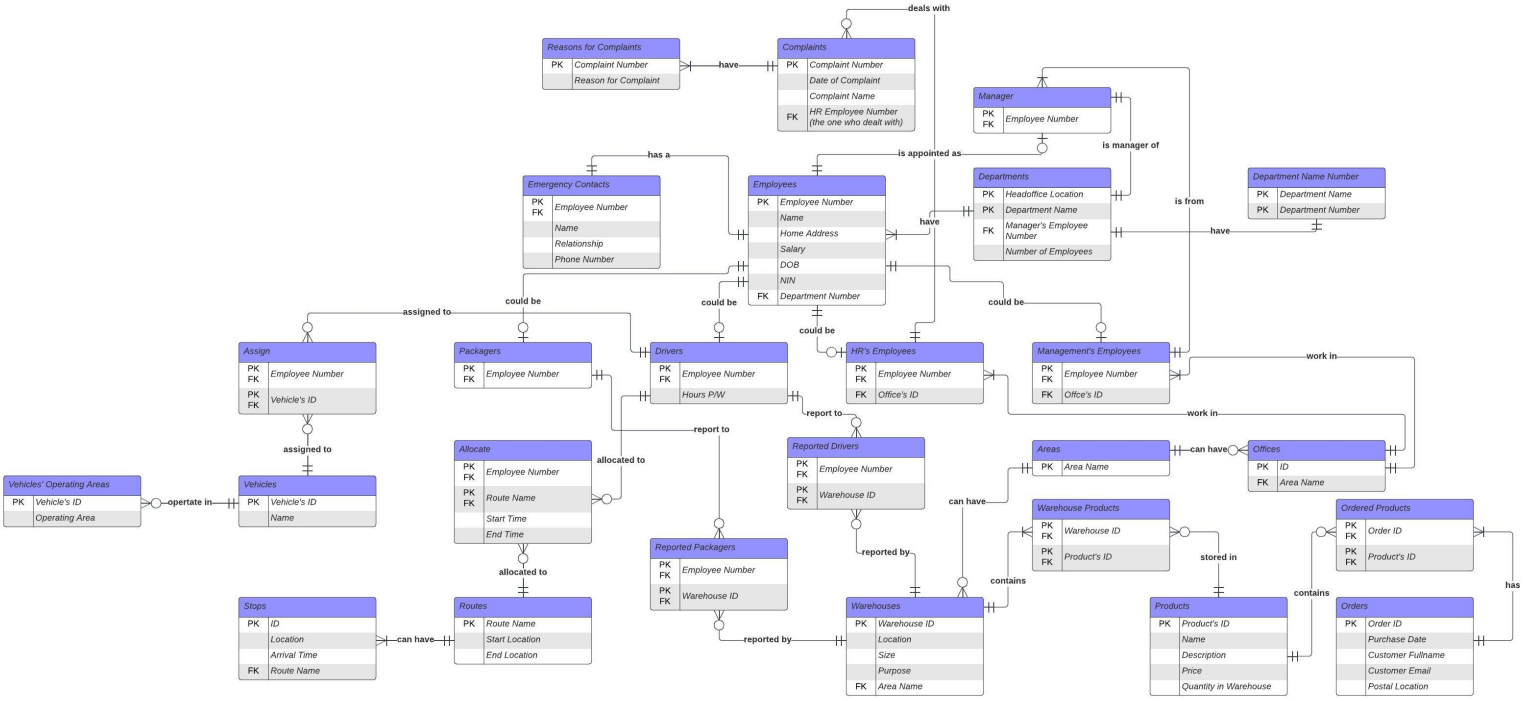

Figure 3: Tables in 2NF

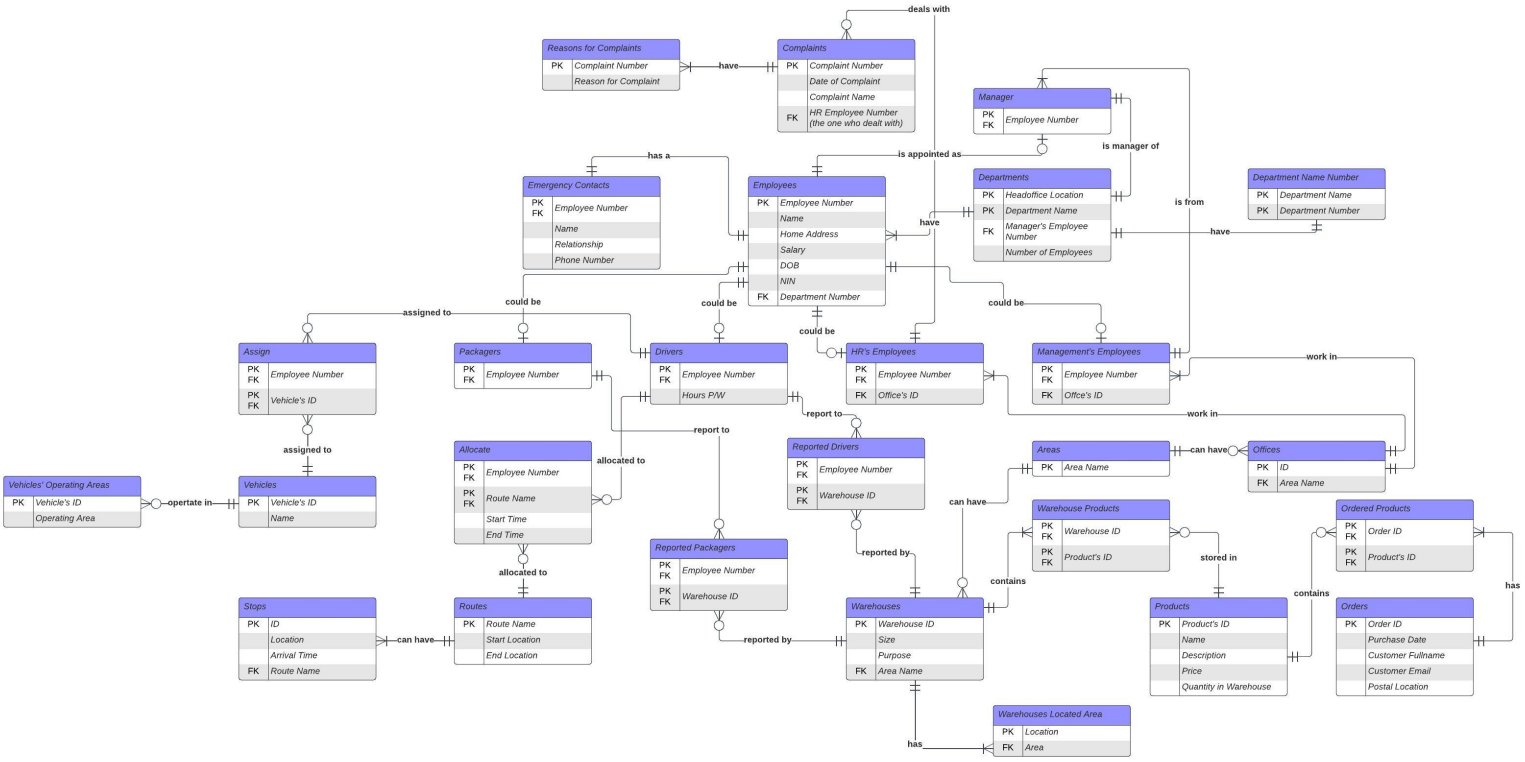

Figure 4: Tables in 3NF

3 Physical Data Model

During the physical data modeling phase, I considered four essential steps: following naming conventions, avoiding reserved keywords, specifying data types, and adding constraints.

In adhering to naming conventions, I began by removing all empty spaces between table and attribute names. Since none of my table names contained reserved keywords, I proceeded to the next phase without any adjustments.

For specifying data types, I assigned the 'int' type to attributes handling numerical values, 'varchar(255)' to those managing mixed strings, integers, and potentially lengthy non-alphanumeric words. Attributes containing brief information were designated as 'varchar(20)', while attributes managing date and time information were assigned the 'DATETIME' data type.

In the final phase of adding constraints, I applied 'NOT NULL' to attributes requiring values from the initial stage. This ensures that those attributes must have a value and cannot remain empty.

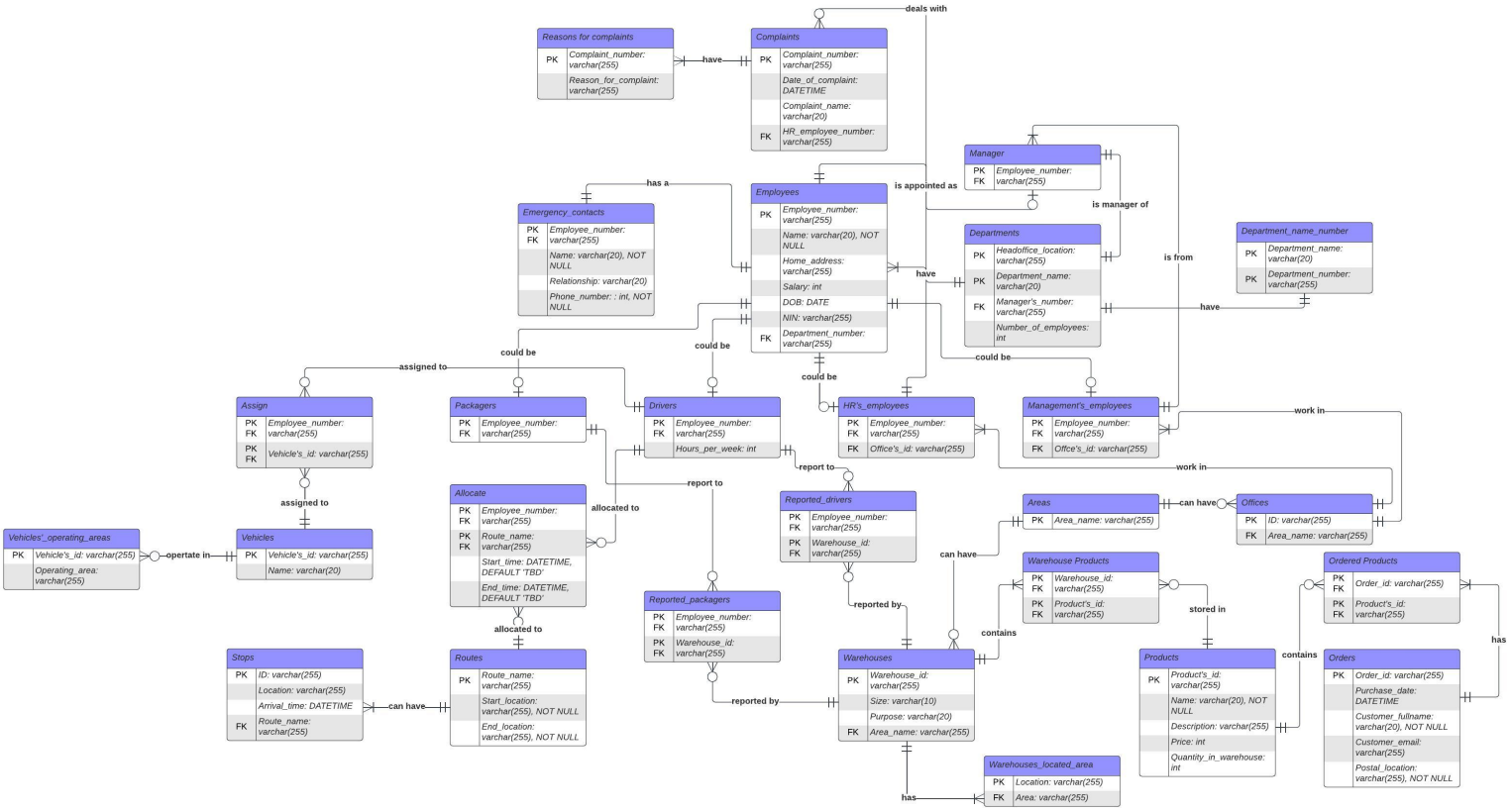

Figure 5: Physical Data Model