The University of Manchester

Coursework - Database Design & Implementation

Milestone 1 – Database Design

Samuel Belkadi

n61655sb

Table of Contents

Entity-Relationship Diagram (ERD)

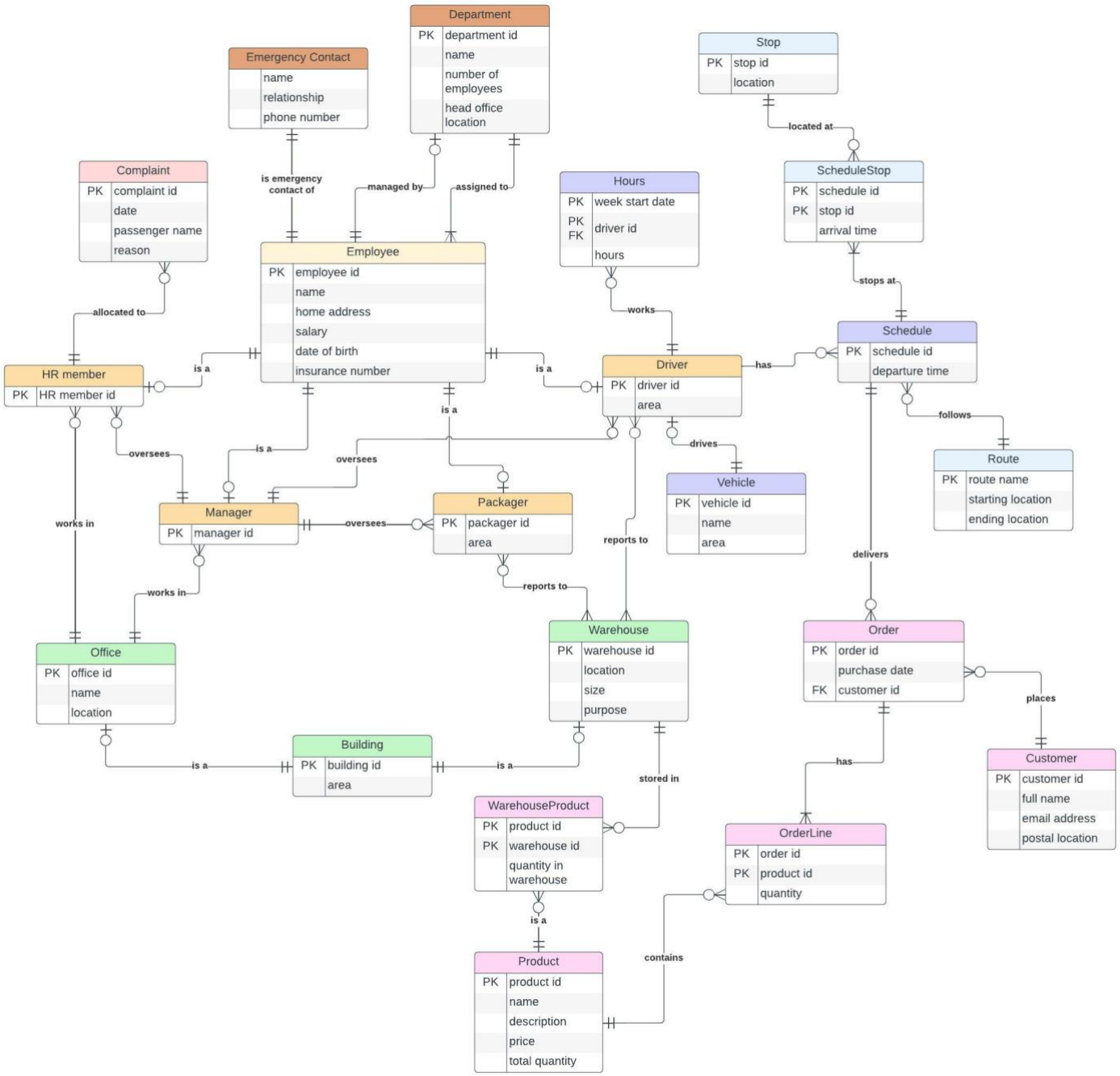
Introduction

In this first section we are going to draw an ERD capturing all the information provided about the new delivery company, “Kilburnazon”.

We will follow Crow’s Foot notation, and since the language is not precise, assumptions and decisions will be taken as we go.

We will produce entities, attributes, relationships between entities and cardinality to each relationship.

Entity-Relationship Diagram



Report

In order to draw our ERD, we produce the following:

A "Department" entity. Each employee is assigned to a single department, and we assume that each department has at least one employee: the employee in charge. We aknowledge that an employee can manage only up to one department, for workloads purposes.

An "Employee" entity which is also Parent of an entity recording emergency contact details in a one-to-one relationship (we assume such a contact can only support one employee). Since each department has different requirements for each role, we create four entities inheriting the "Employee" entity in an "IS-A" relationship. We obtain the following: HR member, Manager, Packager, and Driver.
Each employee is a single one of these.

Furthermore, we note that every employee will be supervised by a Manager. However, we assume that this does not include managers themselves and that some managers may have no employees to oversee.

Additionally, we know that HR members handle Complaints. In that situation we presuppose that while such a member can deal with multiple complaints, a complaint can only be assigned to a single HR member.

We also create a "Building" entity for which we create the attribute "building id" as PK. Assuming there might be multiple buildings in the same area, it prevents us from using the area as PK.

We note that a building is either an "Office" or a "Warehouse", therefore we represent both entities inheriting "Building" in an "IS-A" relationship:

1. "Office" has the primary key "office id". We decide to create a new PK attribute because we suppose that its "name" may not be permanent. Furthermore, we keep both attributes "name" and "location" as we assume that there might be multiple buildings per location, and "building name" being unique, it is therefore not always named after its location. Finally, we assume Managers and HR members will always have a building allocated from them to work in.

2. "Warehouse" simply has the PK "warehouse id". Furthermore, we deduced that Warehouses store Products, and use a bridge entity "WarehouseProduct" to handle the many-to-many relationship between Products and Warehouses.

The "Driver" entity takes care of deliveries by being assigned to a "Schedule". A schedule follows a single route (a starting and ending location), while going through one or more "ScheduleStop" to deliver some orders, being a bridge entity between "Schedule" and possibly multiple "Stop". The bridge entity allows us to handle the many-to-many relationships between schedules and stops, while recording the arrival time at each location. The latter will have the PKs: Schedule.id, Stop.id.

To end with, we keep track of the working hours of a Driver by creating a weak entity "Hours", child of Driver entity. The latter will record the number of hours worked in a week, and has a composite key (Driver.id, Hours.week_start_date).

Finally, an Order will contain one or more products, therefore we create a bridge entity "OrderLine" to handle the many-to-many relationship between "Order" and "Product" while recording the quantity and price of each ordered product into "OrderLine" attributes.

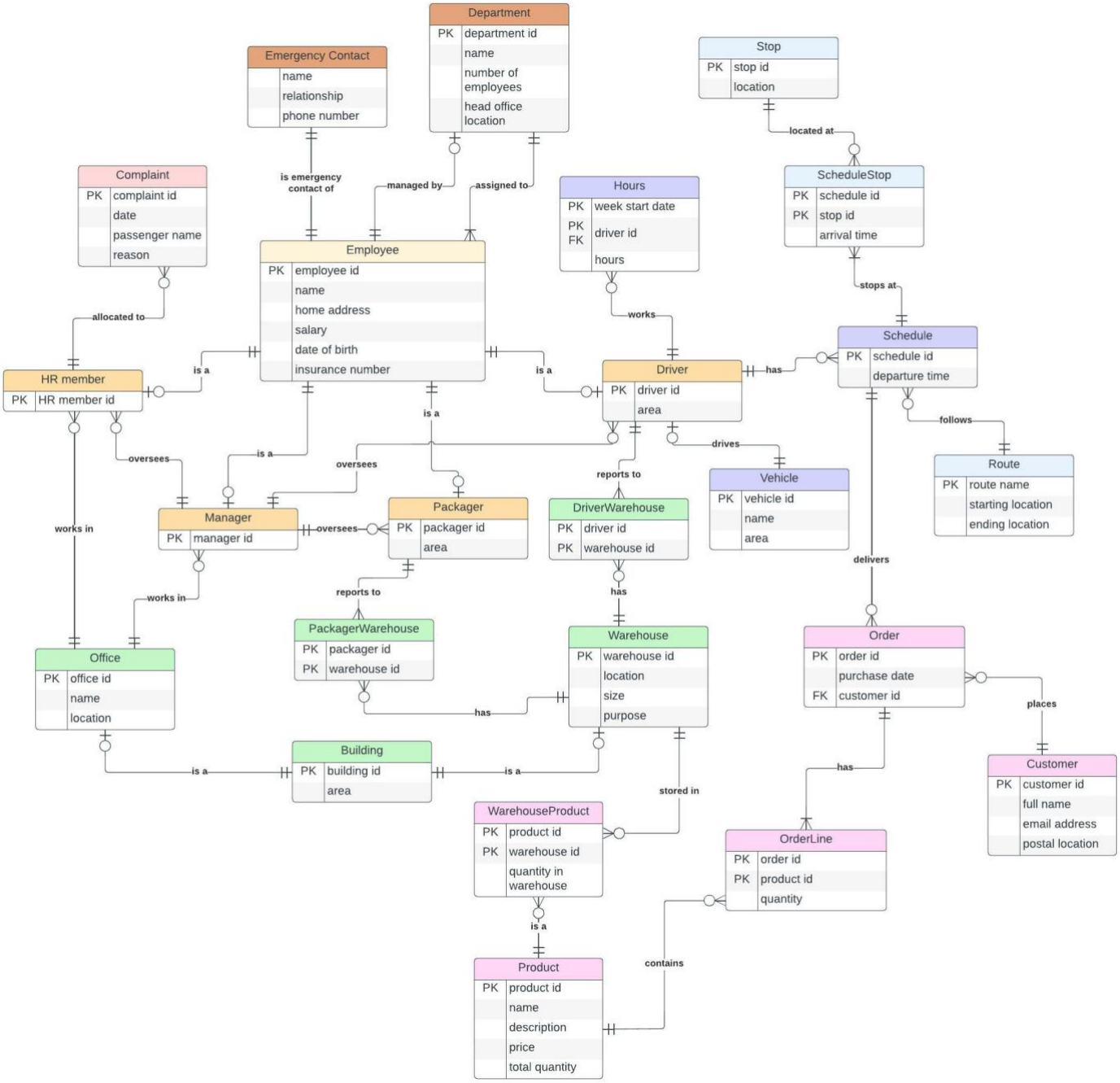
Normalisation

Introduction

In this second step we are going to normalize our ERD by reorganizing its design in order to offer data consistency and minimize redundancy in relations.

We will address the three data modification anomalies: insertion anomally, deletion anomally and update anomally, through three different normal forms namely: 1NF, 2NF and 3NF.

ERD Diagram Normalized in 3NF


Department
PK department id
name
number of employees
head office location
Emergency Contact
name
relationship
phone number
Stop
PK stop id
location
located at
Complaint
PK complaint id
date
passenger name
reason
is emergency contact of
managed by
assigned to
Hours
PK week start date
PK FK driver id
hours
ScheduleStop
PK schedule id
PK stop id
arrival time
stops at
Employee
PK employee id
name
home address
salary
date of birth
insurance number
allocated to
works
Schedule
PK schedule id
departure time
HR member
PK HR member id
is a
Driver
PK driver id
area
has
follows
drives
oversees
Vehicle
PK vehicle id
name
area
Route
PK route name
starting location
ending location
Manager
PK manager id
Packager
PK packager id
area
DriverWarehouse
PK driver id
PK warehouse id
reports to
works in
delivers
has
PackagerWarehouse
PK packager id
PK warehouse id
Warehouse
PK warehouse id
location
size
purpose
Order
PK order id
purchase date
FK customer id
places
Office
PK office id
name
location
Building
PK building id
area
Customer
PK customer id
full name
email address
postal location
stored in
WarehouseProduct
PK product id
PK warehouse id
quantity in warehouse
OrderLine
PK order id
PK product id
quantity
Product
PK product id
name
description
price
total quantity
contains

Report

First Normal Form (1NF):

The following conditions must hold:
1. We need to make sure that each column of our tables will contain atomic values.
To accept this, we create one bridge entity for each relationship of driver-warehouse and packager-warehouse, named DriverWarehouse and PackagerWarehouse respectively, to maintain the atomicity of values.
Now, each column of our tables will contain atomic values since we use bridge entities to handle the many-to-many relationships and avoid multi-valued or composite attributes.
2. None of our entities contain repeating attributes.
3. Finally, values in every column will be of the same type and the order in which the data will be saved does not matter.

Therefore, our ERD is now in 1NF.

Second Normal Form (2NF):

The following conditions must hold:
1.We are in 1NF.
2. Partial dependencies: an entity is in Partial Dependency whenever there is a non-key attribute that is not dependent on every of its entity's PKs. We can assume that an entity with only one PK and no composite key is never in Partial Dependency.
- For "ScheduleStop", the non-key attribute "arrival time" depends on both PKs "schedule id" and "stop id". Not in Partial Dependency.
- For "Hours", the non-key attribute "hours" depends on both parts of its Composite Key "driver id" and "week start date".
- For "WarehouseProduct", the non-key attribute "quantity in warehouse" depends on both PKs "product id" and "warehouse id".
- For "OrderLine", the non-key attribute "quantity" depends on both PKs "product id" and "order id".

Since we are already in 1NF and every entity has all its non-key attributes dependent on each part of its PKs, we can assume no Partial Dependency and therefore that we are in 2NF.

Third Normal Form (3NF):

The following conditions must hold:
1. We are in 2NF.
2. Transitive dependencies: an entity is in Transitive Dependency whenever one of its non-key attributes is dependent on another of the entity's non-key attributes. We can assume an entity with one or fewer non-key attributes to not be in Transitive Dependency.

- We have assumed earlier that an Office name does not depend on its location by definition. Therefore no non-key attribute in the Office entity depends on another such attribute. Warehouse is also not in Transitive Dependency.
- Customer and Product entities have all their non-key attributes uniquely dependent on their PK.
- The same holds for Department and Employee entities.
- Finally, it may not be obvious, but the Emergency Contact entity, with no PK, has no non-key attributes dependent on another since name doesn't determine phone number and vice versa.
- Further entities with more than one non-key attribute clearly have no transitive dependencies: Vehicle, Route, and Complaint.

Since we are already in 2NF and every entity has none of its non-key attributes dependent on another of its non-prime attribute, we can assume no Transitive Dependency and therefore that we are in 3NF.

We can observe a similar Diagram after Normalisation due to an ERD near to being in Normalisation form.

Relational Schema

Introduction

To end with, we are going to make sure every relationships are linked using Primary Keys and Foreign Keys.
We must make sure that these relationships do not break or that deletion and update anomalies do no occur when working on relations. To make sure this does not happen, we are giving each entity some delete and update constraints.

Relational Schema

Department (department_id, name, number_of_employees, head_office_location, employee_in_charge_id)
FK employee_in_charge_id → Employee (employee_id)
ON DELETE RESTRICT, ON UPDATE CASCADE

EmergencyContact (name, relationship, phone_number, employee_id)
FK employee_id → Employee (employee_id)
ON DELETE NO ACTION, ON UPDATE NO ACTION

Employee (employee_id, name, home_address, salary, date_of_birth, insurance_number, department_id)
FK department_id → Department (department_id)
ON DELETE RESTRICT, ON UPDATE CASCADE

Manager (manager_id, employee_id, office_id)
FK employee_id → Employee (eployee_id)
FK office_id → Office (office_id)
ON DELETE RESTRICT, ON UPDATE CASCADE

HRMember (hr_member_id, employee_id, office_id, manager_id)
FK employee_id → Employee (eployee_id)
FK office_id → Office (office_id)
FK manager_id → Manager (manager_id)
ON DELETE RESTRICT, ON UPDATE CASCADE

Packager (packager_id, area, employee_id, manager_id)
FK employee_id → Employee (eployee_id)
FK manager_id → Manager (manager_id)
ON DELETE RESTRICT, ON UPDATE CASCADE

Driver (driver_id, area, employee_id, manager_id, vehicle_id)
FK employee_id → Employee (eployee_id)
FK manager_id → Manager (manager_id)
FK vehicle_id → Vehicle (vehicle_id)
ON DELETE RESTRICT, ON UPDATE CASCADE

DriverWarehouse (driver_id, warehouse_id)
FK driver_id → Driver (driver_id)
FK warehouse_id → Warehouse (warehouse_id)
ON DELETE NO ACTION, ON UPDATE NO ACTION

PackagerWarehouse (packager_id, warehouse_id)
FK packager_id → Packager (packager_id)
FK warehouse_id → Warehouse (warehouse_id)
ON DELETE NO ACTION, ON UPDATE NO ACTION

Warehouse (warehouse_id, location, size, purpose, building_id)
FK building_id → Building (building_id)
ON DELETE RESTRICT, ON UPDATE CASCADE

Office (office_id, name, location, building_id)
FK building_id → Building (building_id)
ON DELETE RESTRICT, ON UPDATE CASCADE

Building (building_id, area)
ON DELETE RESTRICT, ON UPDATE CASCADE

Complaint (complaint_id, date, passenger_name, reason, hr_member_id)
FK hr_member_id → HRMember (hr_member_id)
ON DELETE NO ACTION, ON UPDATE NO ACTION

Hours (week_start_date, driver_id, hours)
FK driver_id → Driver (driver_id)
ON DELETE NO ACTION, ON UPDATE NO ACTION

Vehicle (vehicle_id, name, area)
ON DELETE RESTRICT, ON UPDATE CASCADE

Schedule (schedule_id, departure_time, driver_id, route_name)
FK driver_id → Driver (driver_id)
FK route_name → Route (route_name)
ON DELETE RESTRICT, ON UPDATE CASCADE

Route (route_name, starting_location, ending_location)
ON DELETE RESTRICT, ON UPDATE CASCADE

ScheduleStop (schedule_id, stop_id, arrival_time)
FK schedule_id → Schedule (schedule_id)
FK stop_id → Stop (stop_id)
ON DELETE NO ACTION, ON UPDATE NO ACTION

Stop (stop_id, location)
ON DELETE RESTRICT, ON UPDATE CASCADE

Customer (customer_id, full_name, email_address, postal_location)
ON DELETE CASCADE, ON UPDATE CASCADE

Order (order_id, purchase_date, customer_id, schedule_id)
FK customer_id → Customer (customer_id)
FK schedule_id → Schedule (schedule_id)
ON DELETE CASCADE, ON UPDATE CASCADE

OrderLine (order_id, product_id, quantity)
FK order_id → Order (order_id)
FK product_id → Product (product_id)
ON DELETE RESTRICT, ON UPDATE NO ACTION

Product (product_id, name, description, price, total_quantity)
ON DELETE CASCADE, ON UPDATE CASCADE

WarehouseProduct (product_id, warehouse_id, quantity_in_warehouse)
FK product_id → Product (product_id)
FK warehouse_id → Warehouse (warehouse_id)
ON DELETE NO ACTION, ON UPDATE NO ACTION

Report

Here, we describe the way we assign constraints to deletion and update of our entities in order to avoid deletion and update anomalies based on their relationships:

When updating a Department, Employee or any inheritance of Employee (Manager, HRMember, Packager, Driver), we want it to be updated accordingly on all row of tables referencing it. Therefore we update as Cascade.
On the other hand, we want to restrict its deletion to not create deletion cascade or null-values.

The same happens with Warehouse, Office and Buildings. As they are Parent entities, we want to update them in cascade and restrict their deletion to not create deletion cascade or null-values.

Vehicle, Schedule, Route and Stop all four are entities used by others as FK or child of entity. Therefore, we want to update them in cascade and restrict their deletion. The entities having them as foreign keys do not accept null-values, therefore, we do no delete them with null-value replacement.

However, some exception exist. Our bridge entities allow update or deletion since they are not FK of any other entities and do not have any child. Therefore it doesn't matter for them to be updated or deleted.

Some others as Customer, Order and Product are entities that can be deleted but must be deleted in cascade as they are used in our ordering and storing system. Therefore, their update must also be done in cascade.

Finally, OrderLine cannot be deleted as a child of Order without deleting Order itself. However, we can update an OrderLine with no particular issue since it is a bridge entity.