CMPU3010 Databases 2 – Relational Modelling Assignment

Students:

- Martin Januska (C21327411)
- Dylan Hussain (C21331063)

Case Study:

- Bicycle Repair Shop
- Bike856BB Schema

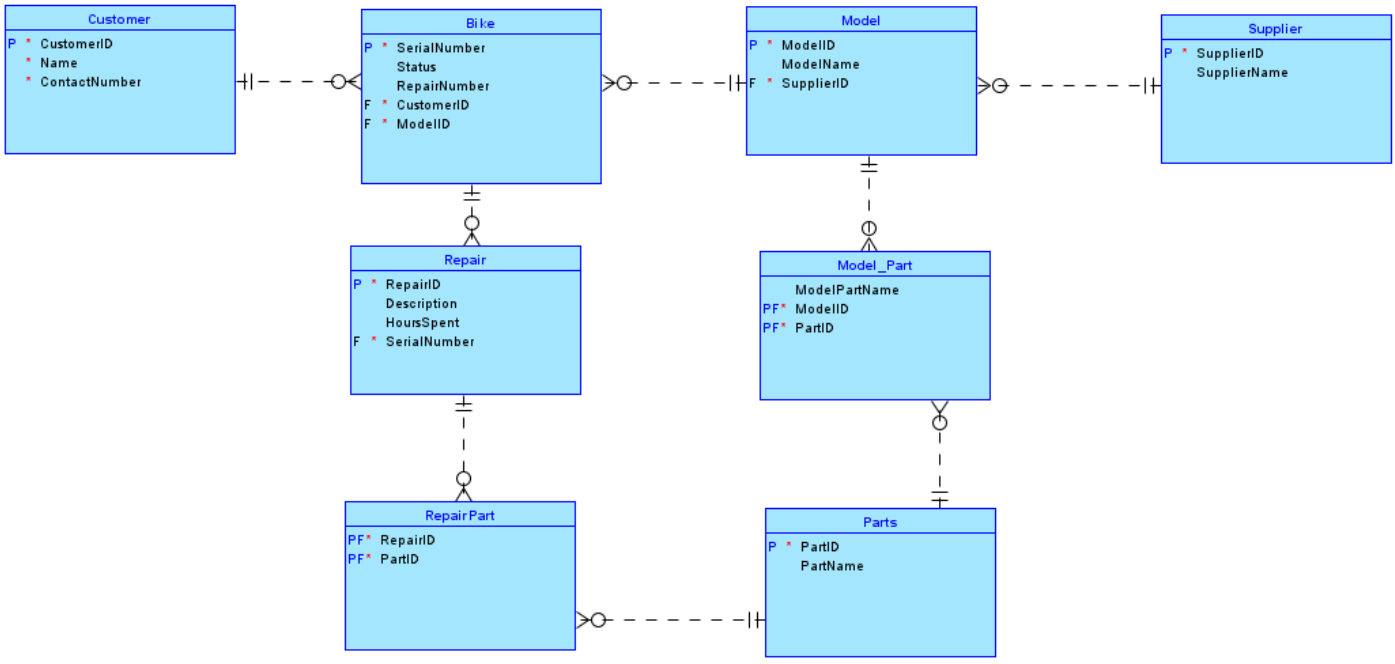
ERD Bicycle Repair Shop

To create the ERD for the bicycle repair shop based on the case study, we had to identify the entities necessary to create the diagram. We identified eight entities that we thought were appropriate based on the case study description.

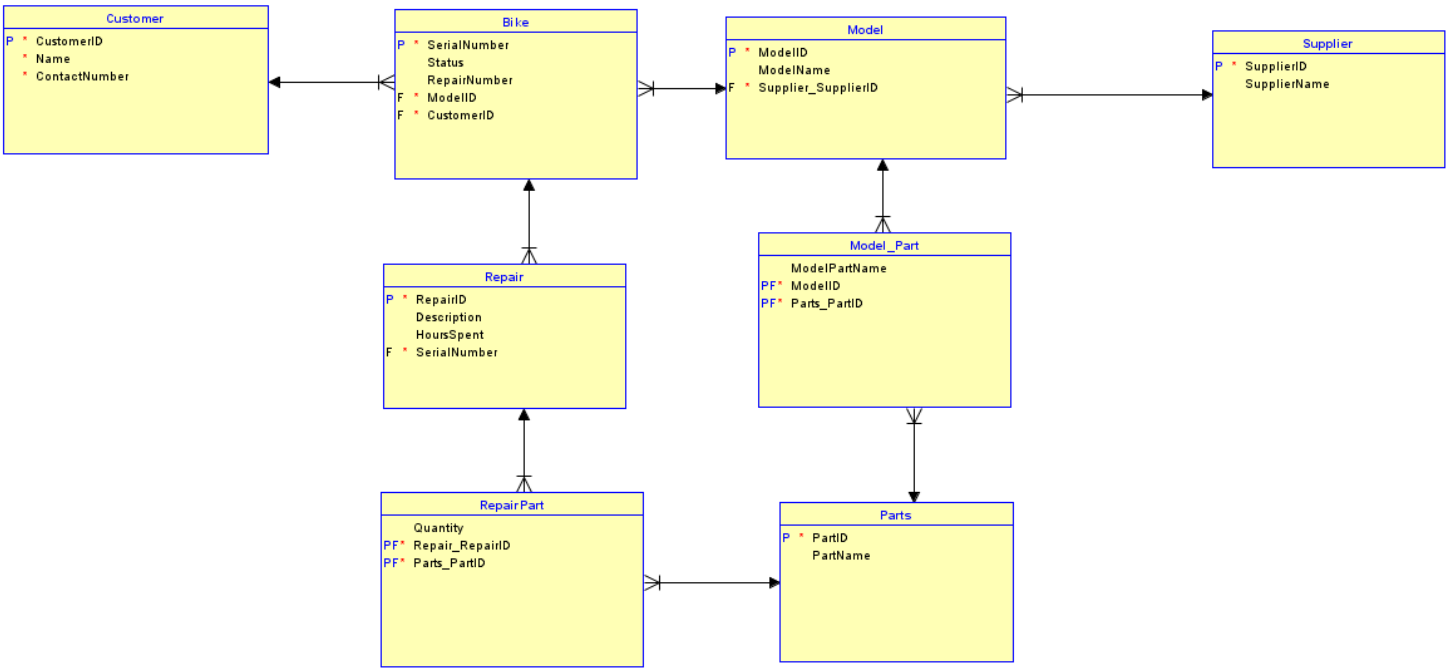
The eight entities we identified were:

1. **Customer**:
    - Case study mentions that the receptionist records the customer's name and contact number.
2. **Bike:**
    - Represents the bicycle brought in for repair, attributes including serial number, status, repair number, model id and a reference to the customer who owns it.
3. **Repair:**
    - Represents the repair job, including attributes for repair ID, description of the repair, hours spent on the repair. Associated with the bike serial number.
4. **Repair Part:**
    - This entity is to keep track of the parts replaced during a repair. Includes the quantity of parts used, relating to both Repair and Part entities. Using a compound primary key.
5. **Part:**
    - Responsible for managing various parts used in bike repairs. Name of the part is provided. Can be standalone or contain other parts.
6. **Model Part:**
    - This entity is used to make a relationship between bike models and the parts they require. Different bikes may use different parts.
7. **Model:**
    - Represents different bike models, also related to the Supplier entity since each model is supplied by a supplier.
8. **Supplier:**
    - Represents the suppliers providing bike models.

Logical Model



Relational Model



As part of designing the ERD, we had to identify the relationships and cardinality between the entities.

Primary and Foreign Keys were easily identified based on the case study.

All entities had an 'ID' attribute to uniquely identify them. In the Bike entity, the serial number was used as the primary key as it is uniquely identifiable.

Foreign Keys:

- CustomerID in Bike as a foreign key, to link the customer to their bike.
- SerialNumber in Repair as a foreign key, connecting repairs to specific bikes.
- RepairID and PartID in RepairPart form a compound primary-foreign key, relating repairs and parts.
- ModelID in Bike as a foreign key, associates bikes with their models.
- ModelID and PartID in ModelPart form a compound primary-foreign key, connects bike models and parts they use.
- SupplierID in Model as a foreign key, creates a relationship between suppliers and bike models they provide.

Cardinality:

- Customer-to-bike is a one-to-many relationship, since one customer can have multiple bikes in for repair.
- Bike-to-Repair is a one-to-many relationship, as one bike can have multiple repairs.
- Repair-to-RepairPart is a one-to-many relationship, one repair can have multiple parts.
- Model-to-ModelPart is a one-to-many relationship, as one bike model can use multiple parts, and each part can be used in multiple models.
- Model-to-Supplier is a one-to many-relationship as one supplier can provide multiple bike models.

Roles within the Bicycle Repair Shop System

We identified multiple roles from the Bicycle Repair Shop case study. These roles included the Owner, Receptionist, Shop Assistant and Mechanic.

Mechanic Role (Dylan Hussain)

I took on the role of the Mechanic in relation to the Bicycle Repair Shop System.

From the case study, when a bike is recorded, the Mechanic can select a bike to repair and check what's needed to be done, once they repair the bike, they can leave a description of the repair and hours spent on the repair, along with a list of the parts and quantity of each part needed for the repair.

Based on the case study, the Mechanic would need permissions to insert new records into the database, For the mechanic role, I received the following grants:

- To execute the following functions:
    - get_bikes_awaiting_repair
    - get_bike_description
    - log_repair
    - record_part_repaired
- Select and Insert on the table ‘bike’
- Select and Insert on the table ‘repair’
- Select and Insert on the table ‘repairpart’
- Select on the table ‘model_part’

PLpgSQL Function

As part of the Mechanic role, I created a few functions and one procedure to implement the full role.

get_bikes_awaiting_repair

This function returns a table of the bikes that are currently marked with the status ‘R’ for repair, if no bikes are found it will throw an exception stating it.

get_bike_description

This function returns a bikes repair description by taking a serial number as a parameter and checking for the serial number in the ‘repair’ table. If there is none found it will throw an exception stating it.

log_repair

This function inserts a repair record into the ‘repair’ table, it takes in a serial number, repair description and hours spent repairing the bike. The function returns the record ‘repair_id’ which can be used in the function ‘record_part_repaired’ for logging the records for parts required for the repair.

record_part_repaired

This procedure takes a quantity, repair id, and part id parameter. It will create a new record in the ‘repairpart’ table logging a part that would have been used in the repair.

PLpgSQL Trigger

I also created a constraint trigger to validate the part id given in the repairpart table. This constraint trigger checks the newly entered part id and queries the model_part table to check if the model part id exists. If the mechanic tries to insert a new part that does not exist in the model_part table, it will throw the following exception:

‘Invalid model part’

This trigger executes before an insert or an update.

Python

I created a simple python program that connects to my schema “C21331063” on the University Server provided the python has the correct Host IP, Port and Database. It includes a simple interface stating this is the mechanic role and prompts you to login.

Upon logging in, the user will be prompted with the bikes that are awaiting repair, if there are any. Then it will prompt the user to select a bike with the bikes serial number, if the given serial number is valid, it will give the description of the repair needed. The user will then be prompted to enter the details of the repair such as, the description of what was done, the hours spent, and then they will be prompted to enter the parts that were used, after each part is entered it will prompt the user if they would like to add another part.

Try except blocks are used to output any errors to the user, along with validation on both the SQL side and the python side. When the program starts a dictionary is made of the valid repair serials which is used when the user is selecting the bike that they would like to repair.

The program implements basic validation for every input such as type validation for integers and ensuring that a value was entered for strings. This basic validation on top of the dictionary validation

ensures that no incorrect data can be passed into the log_repair sql function or any other procedures or functions.

Finally, failure to login or to connect displays an appropriate error message.