Programming Assignment 2 Report

Student(s):	Fabian Dacic - fd222fr@student.lnu.se Yuyao Duan - yd222br@student.lnu.se

1. Project Idea

Scene simulation:
Dacic & Duan Däckia Experten AB is one of the biggest tire workshops in Växjö. Due to the good reputation, the business of this workshop continues to grow, and the traditional information management method is not sufficient to support management demand which needs to be substituted. The business model of Dacic & Duan Däckia Experten AB includes the following aspects: tire sales, the company sells both summer tires and winter tires consisting of various well-known brands to fulfill different customers' demands; tire repair, the workshop offers comprehensive repair services; tire storage, customers can store their tires for changing of the seasons; tire order, Dacic & Duan can directly order for customers when the required products out of stock or typical tires in the warehouse need to be restocked.

Model analysis:
According to the above case description, we deconstruct the business model into four different aspects which will, later on, compose the new database schema. In order to fulfill the business needs, there are four relations needed to be included in this new design: Customer Contact Information relation, which is used to store all the contact information including name, telephone number, email, address, and vehicle registration number. Customer Car Information relation is designed for collecting detailed information of the focal vehicle aiming for accurate service delivery. Two individual relations – Summer Tire Offers and Winter Tire Offers are used for storing tire information which can be considered as a product list, in the case to handle price consulting. Order relation is used to save all the orders from customers. In reality, this relation can be the interface accessing the supply chain management system i.e. suppliers will directly receive the orders from the workshop which will contribute to a higher level of business integration.

Design implementation:
This new application enables the workshop:

- Query customer's contact information
- Query customer's car information
- Query tire information of the car
- Query winter tire options for the car
- Query summer tire options for the car
- Query average price, maximum price, and minimum price of a typical tire size with brands
- Query existing inventory
- Query current inventory cost of the workshop
- Generate views for workshop employees/workers
- Generate (Insert) an order for customers/workshop's warehouse

The advantage of this design is that it can help the workshop's service desk quickly locate the customer's information which links to the car's information that will identify what tires fitting

with the focal car. The system can present all available tire options to the car in terms of different brands and prices, aiming for a higher level of customer satisfaction. The generated Views of this system will sufficiently support the works of employees/workers of the workshop, and at the same time hiding some sensitive information in the database.

Data preparation:
For this assignment, we utilized our knowledge in the car industry as well as applying a web-based data generator for the raw data of the designed relations. We generated and downloaded the data from this source: https://www.mockaroo.com/, which are stored in .csv format.

2. Schema Design

For Dacic & Duan Däckia Experten AB, we applied the following E/R model for its new database system:

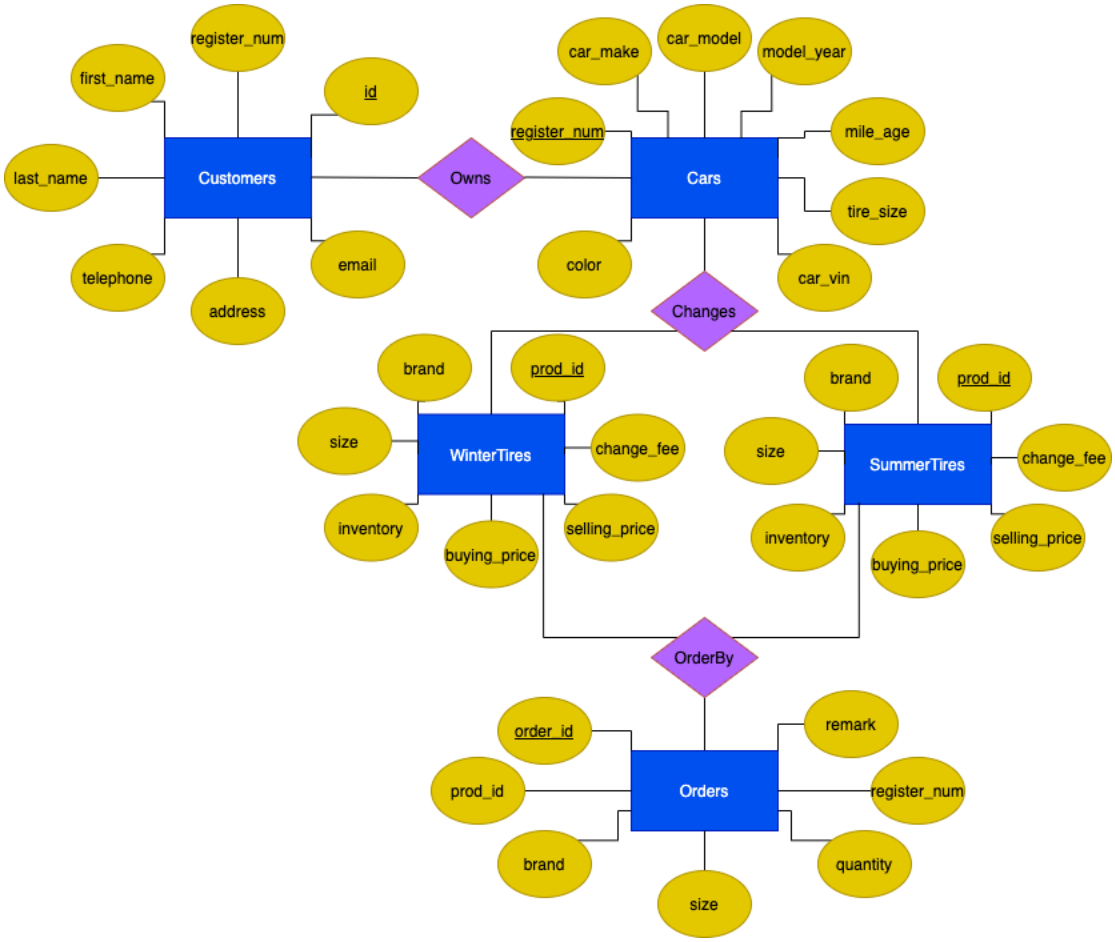

Figure 1. Proposed E/R model design for Dacic & Duan database system

There are five tables/relations in this schema:

- Customers (customer contact info) relation – includes all the detailed contact information of customers, there are seven attributes of this entity set: customer's id in this workshop, customer's car's register number, customer's first name and last name, telephone number, address, and email. The customer's id is the key to this relation.
- Cars (customer car info) relation – consists of important aspects of customer's car. To achieve a higher level of service satisfaction, this table collects the car's register number, car make, car model, model year, color, mileage, car's VIN code, and tire size. The register number of the car is the key to this relation.

- WinterTire (car_tire_winter) relation – can be considered as a product list (product offers) of winter tires. It includes tire brand, size, current inventory quantity, selling price, buying price, and change fee of a typical tire. The workshop gives each product a unique product id (product_id) which can be considered as the key to this table. The buying price attribute is used by workshop administrators e.g. calculating current inventory cost which should be hidden for normal workers and employees.
- SummerTire (car_tire_summer) relation – has a similar schema as WinterTire relation, product id (product_id) is the key to this table.
- Orders (orders) relation – is used for ordering tires either for customers or workshop's warehouse. In case, when the customer's requirements are out of stock, the workshop can directly order for customers, therefore, the register number needs to be filled in to link the orders to the cars. By comparison, when workshop orders for its warehouse, the register number can be empty (or NULL). Other attributes are tire's prod_id, brand, size, quantity, remark, and order_id. Order_id is key and prod_id is foreign key to this table.

As we can see, there are three relationships in this E/R model design. The relationship between the Customers entity set and the Cars entity set is Owns. Different customers own different cars. The Cars entity set and the two Tires entity sets are linked by the relationship "Changes" which means that car changes tires. SummerTires and WinterTires relations link Orders relation by the relationship "OrderBy".

3. SQL Queries

In this design, we plan to assign different entry accesses to users. The admin access has the highest authority and can query all information in the database, while the worker access is mainly used for supporting workshop's workers which provides essential information for their work. In the worker access, the data related to trade secrets, management and customer's personal information will be hided from this access.

Admin access queries:

There are six SQL queries in the admin access including query customer's all personal information, query customer's all car information, query all information of summer tire and winter tire relations, query current inventory cost, and query all created orders. The first four SQL queries are designed to make this system with complete function that the queries themselves are relatively simple – by utilizing "SELECT * FROM focal_table". Administrator can access all columns of all tables in the database. The most interesting queries in this part are "query current inventory cost" and "query all created orders".

Q: **Get current inventory cost (List tire brands, inventory quantity, total buying price i.e. cost from both summer tire relation and winter tire relation filtering the tire brand with 0 inventory quantity).**
The following query is multi-relational query which uses *UNION ALL*. We use *SUM (inventory)* to calculate the total number of both summer tires and winter tires of certain brand, and then using *SUM (buying_price)* for getting current total inventory costs of certain tire brand. We use *UNION ALL* to link summer tire relation and winter tire relation. We use *HAVING Inventory > 0* to filter the brands which are already sold out.

```
SELECT brand AS TireBrand, SUM(inventory) AS Inventory, SUM(buying_price) AS
cost

FEOM ((SELECT brand, inventory, buying_price FROM Dacic_Duan.summer_tires )

UNION ALL (SELECT brand, inventory, buying_price FROM Dacic_Duan.winter_tires )) sl

GROUP BY TireBrand

HAVING Inventory > 0;
```

Q: **Query all orders (List all columns of order relation).**
The query itself is very simple, however, in this database system this query design makes the information between administrators and workers realizing interactive i.e. the workers inserting orders can be checked by administrators through this query.

```
SELECT *

FROM Dacic_Duan.orders

ORDER BY order_id;
```

Worker access queries:

There are seven SQL queries in the worker access including querying customer's ID and its car register number, listing all available tires for a car, finding out the average, maximum and minimum price of all tire sizes (for answering customer's questions), creating a view for workers regarding customer's information, creating two views for workers regarding winter tire information and summer tire information respectively, inserting a new order in the order relation.

Q: **List all available tires for the car (List the car's register number, tire size, the available tire brand, and price).**
This query is designed as including two sub-queries – querying for winter tires or summer tires. The two sub-queries have same SQL structure but only searching from different tables (winter or summer tires). Here we focus on searching for available summer tires for example. The following query is a multi-relational query and uses *JOIN.* We pass the argument of the register number (marked with ? in the query) and the query should give us all available summer tires for this car. We join table *Cars* on table *Summer_Tire* by matching the *Cars.Tire_Size* to the *Summer_Tire.Size*. In order to have a better presentation, we order selling price by *ASC.*

```
SELECT a.register_number, a.tire_size, b.size, b.brand, b.selling_price

FROM Dacic_Duan.cars AS a

JOIN Dacic_Duan.summer_tires AS b ON a.tire_size = b.size

WHERE a.register_number = ?
```

```
ORDER BY b.selling_price ASC;
```

Q: **List the average, maximum and minimum price of all tire sizes.**
The following query utilizes aggregation (*AVG*, *MAX, MIN*), grouping and union operators. The argument is passed to retrieve the tire size, average price of each tire, maximum and minimum prices along with that from both tables, combining the results through the usage of *UNION ALL* and after it grouping them by the tire size thus finally, providing the results.

```
SELECT AVG (SellingPrice) AS AveragePrice, MAX (SellingPrice) AS MaxPrice,

MIN (SellingPrice) AS MinPrice, Size AS TireSize

FROM ((SELECT Size, SellingPrice FROM Dacic_Duan.summer_tires ) UNION ALL (SELECT
Size, SellingPrice FROM Dacic_Duan.winter_tires )) sl

GROUP BY TireSize;
```

Q: **Create a view for workers regarding customers' information.**
The following query is regarding creating a view. We use *CREATE OR REPLEACE VIEW* to create and only select customer's ID, car register number, car make, car model, and tire size as the view's columns for workers. Other personal information will be filtered. We use *USING (Register_Number)* to *JOIN* the table Customers and the table Cars.

```
CREATE OR REPLACE VIEW Workshop_Employees AS

SELECT a.ID, a.register_number AS c_r_num, b.register_number AS cc_r_num,
b.car_make, b.car_model, b.tire_size

FROM dacic_duan.customers AS a JOIN dacic_duan.cars AS b

USING (register_number);
```

Q: **Create a view for workers regarding summer tire information.**
The aim of this view is to hide the buying price of each due to the fact that this may related to trade secrets which is not suitable for workers to know. Furthermore, we use aggregation (*AVG*, *SUM*) for finding out current inventory quantity, average selling price of a typical tire, and average change fee regarding tires in order to better answer customer's question during service. The view for winter tires follows similar structure.

```
CREATE OR REPLACE VIEW wt_employees AS

SELECT prod_id AS st_prod_id brand AS wt_brand, size AS wt_size, SUM (inventory) AS
wt_inventory, AVG (selling_price) AS wt_sell_price, AVG (change_fee) AS wt_changing_fee

FROM (SELECT prod_id, brand, size, inventory, selling_price, change_fee FROM
Dacic_Duan.winter_tires) sl

GROUP BY prod_id
```

```
ORDER BY wt_size;
```

Q: **Insert a new order to the orders relation.**
The worker can generate orders for either customers or workshop's warehouse. For customers, the workers need to fill the register number which can leave as empty when ordering for workshop's warehouse. *INSERT INTO* and *VALUES* are used for this query. We pass the argument of *order_id*, *prod_id, brand, size, quantity, register_number, remark* (marked with ? in the query)

```
INSERT INTO orders

VALUES ('?', '?', '?', '?', '?', '?', '?', '?');
```

4. Discussion and Resources

In this assignment, we successfully implement a simple database system for tire workshop. The database schema includes five relations in order to cover its service and management demands. In reality, different system users have different accesses to system, therefore, we design two entry accesses for admin users and normal workers respectively. To access the related queries and data, the users have to input correct passwords. The system uses function *getpass.getpass()* to receive user's input in order to hide the passwords during inputting process. In addition, based on the password feature, we simulate password validation that when the user input wrong passwords three times will automatically exit from the database system which can be considered as a valuable feature. We design this system to maximize simulation of reality instead of solely fulfilling the assignment's requirements which can be considered as the most satisfying aspect.

It is important to note a few aspects of the program. When setting up the MySQL connection, the standard root and root procedure is applied to it however there are cases in which that is not the case for some therefore it is advised to change the credentials so that they are in accordance with what the user in question has.

When the program is prompted, the user will be greeted upon a menu and depending on the user's authentication, different submenus are offered. For example, if the user is an admin, said admin logs in with the password "admin" and is taken to the menus that are of relevance to the position of administrator. If the user is a worker, said worker logs in with the password "worker" and is taken to the options which are available to that position. It is worth noting that in case the user fails to insert the correct password, the program exits.

The table "Orders" is at first empty due to not being any orders present at the moment. So to request an order and issue it, the user has to log into the workers' menu, query an order (insertion of the ID of the order itself, registration number for the car which the order is for, the tires for it and any remarks that there can be issued along with the order). After the order has been requested, it is visible in the admins' menu afterwards.

It is of most importance that whenever inserting an input, that the input format is followed correctly and not insert invalid characters.

We consider that personal interests and knowledge reservation are very important for this project since both of us are very familiar with car industry and both of us have experience of changing car tires. The groupwork working style builds our advantages since we can pool our benefits and learn each other's advantages during work. Under this working manner, we improved ourselves in terms of method encapsulation, handling exceptions, writing SQLs, and schema design. The important lesson we learned from this assignment is that in the initial stage of this group, we did not have sufficient discuss and have agreement on how to name the attributes of the tables. This brought our troubles during cooperation since the names of attributes are different. This was later on corrected in time after both of agreeing on using the same attribute names as the tables' original ones.

The project uses Python library : csv, mysql.connector, getpass, errorcode.
Source code: [https://gitlab.lnu.se/yd222br/database-technology-programming-assignment2]
Video demonstration: [https://www.youtube.com/watch?v=8KXjmg8_J-4&t]