Organizing a Beauty Salon

Project Report

Attieh Joseph, Akiki Clara

Abstract—Every beauty salon needs its own database. Even though some might find it easier to use file-based systems to manage their receords, this approach might get complicated on the long run, especially when the amount of data increases heavily.; papers can be lost, data may be duplicated which can make the life of the staff more difficult. Our project consists of developing and designing a database system that would make everyone's life easier and that would lead to an easier handling of the data.

Index Terms— beauty salon, customer, database, mySQL, order, staff

CONTENTS

1 INTRODUCTION

This project presents the work done on the BeautySalon Database that allows non-expert users to store, access, and manipulate the database. The purpose of our project is to use JavaFx in creating GUIs in order to allow user to interact with the database using a mysql connector that links the graphical interface with the database in MySQL. This beauty salon is run by one and only one manager who supervises the staff. The staff can add, modify and delete customers. They can also book appointments for services to the customers just as they can also take orders for customers for the products present in the shop. But the staff are not allowed to see the password of each others, only the manager can and is omnipresent. Also, they are not allowed to insert, edit or remove any data that concerns the staff, suppliers and product. Only the manager has this priviledge.

2 BACKGROUND

2.1 Problem Analysis

Most of the regular Beauty Salon relies only on manual effort when it comes to storing data such as the staff, customers, appointments taken, services provided. Such Beauty Salon do not have a DataBase that can hold all their records and data. Since everything is being done on a file-based system, many errors have been occurring such as data duplication, limited indexing capabilities and the need for the physical location of every single file.
As a solution to this, we created this database to automate the entire documentation process from inserting staff, to adding client, to taking appointments, to deleting staffs who no longer work at the salon…

2.2 Functionality

This Beauty Salon Database system is designed to:

- Automate the processes
- Keep track of all the records easily
- Consolidate / Verify all inputs

This system uses JAVA as its core language, MySQL for the database storage and JAVAFX for the GUI design.
The main functionalities of the system are the following:

- Adding, editing, deleting data
- User authentication for data access control

2.3 Information Needs

This database that we created can be used in any beauty salon. It consists of 7 entities, which are the following:

1 STAFF: this table contains all the relevant data concerning the staff such as his/her ID, name, salary, password…
2 CUSTOMER: This table contains all the relevant data concerning the customers such as their ID, name, phone number…
3 SERVICE: This table contains all the relevant data concerning the services provided by the staff such as the service ID, type, staff who performs it…
4 APPOINTMENT: This table contains all the relevant data concerning the appointments made by the staff for the customers such as the booking ID, date, time, service being provided, customer who booked the appointment and the staff who made the booking…
5 ORDER_DATA: This table contains all the relevant data concerning the orders made by the customers such as the order ID, date, time and customer who placed the order…
6 SUPPLIER: This table contains all the relevant data containing the information on the supplier such as his/her ID, the company Name…
7 PRODUCT_LIST: This table contains all the relevant data concerning the products ordered by the customers such as the prod ID, name, price, supplier who provided it…

3 PROPOSAL

As explained earlier, we have 8 tables as know as entities or relations later on. These tables will be connected to one another in a logical way, the way we want our data to be displayed.

3.1 CDM

Let us start with the Conceptual Data Model(CDM) which is the following:

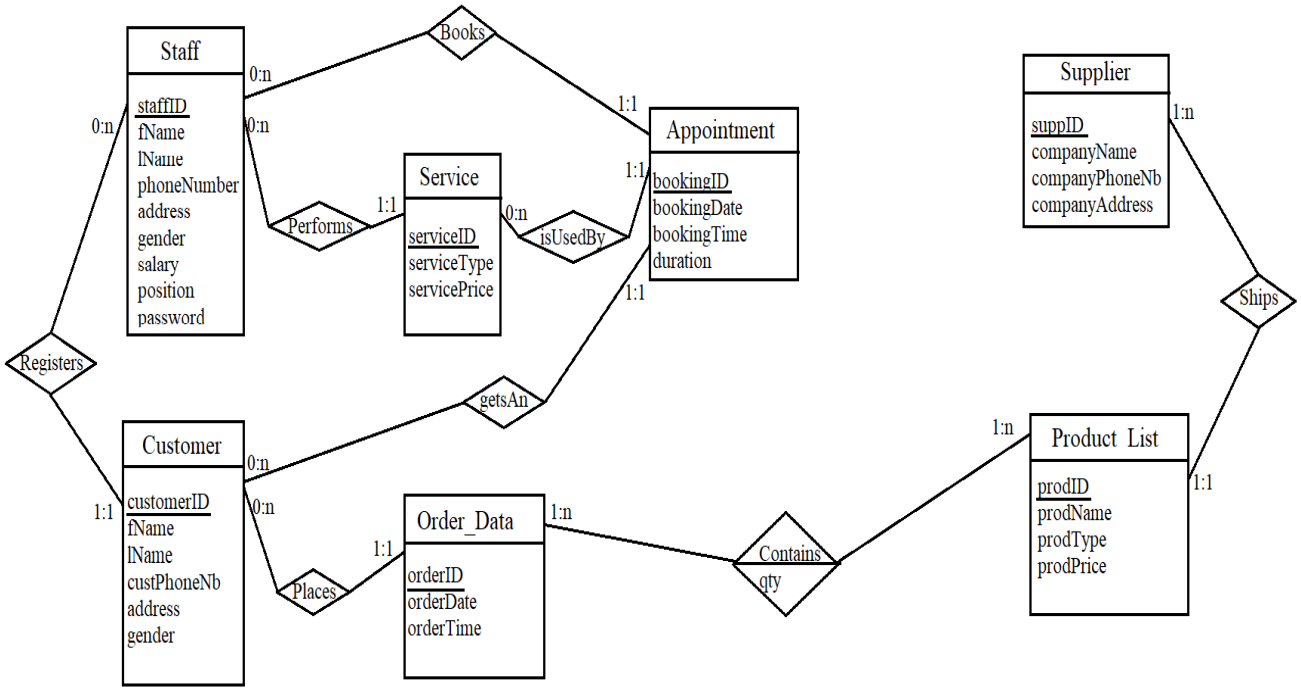


Figure 1: CDM of the BeautySalon

Let's explain and describe the CDM:

- A **staff** has attributes staffID, first name, last name, phone number, address, gender, salary, position and password. To uniquely identify between each staff, we use the staffID as a primary key since each staff has a unique ID.
- A **customer** has attributes customerID, first name, last name, phone number, address and gender. To uniquely identify between each customer, we use the customerID as a primary key since each customer has a unique ID.
- A **service** has attributes serviceID, type and price. To uniquely identify between each service, we use the serviceID as a primary key since each service has a unique ID.
- **Order_Data** has attributes orderID, date and time. To uniquely identify between each order data, we use the orderID as a primary key since each order data has a unique ID.
- An **appointment** has attributes bookingID, date, time and duration. To uniquely identify between each appointmnet, we use the bookingID as a primary key since each booking has a unique ID.
- A **product_list** has attributes prodID, name, type and price. To uniquely identify between each product, we use the prodID as a primary key since each product has a unique ID.
- A **supplier** has attributes suppID, company name, company phone number and company address. To uniquely identify between each supplier, we use the suppID as a primary key since each supplier has a unique ID.

Now we must explain how the relationships occur with the cardinalities:

- A staff can register many customers while customers are registered by only one staff, he/she can also book an appointment which can only be booked by one staff, he/she can also perform a service which is performed by only one staff at the moment.
- An appointment uses one service, but a service can be used by many appointments.
- A customer may get many appointments, but an appointment is for only one customer.
- A customer can also place orders for products.
- The Order_Data contains from the Product_List which can be many products. The supplier ships many products and a product is shipped by only one supplier

3.2 LDM

After having done the CDM, we can now switch to the LDM. The LDM is the following:

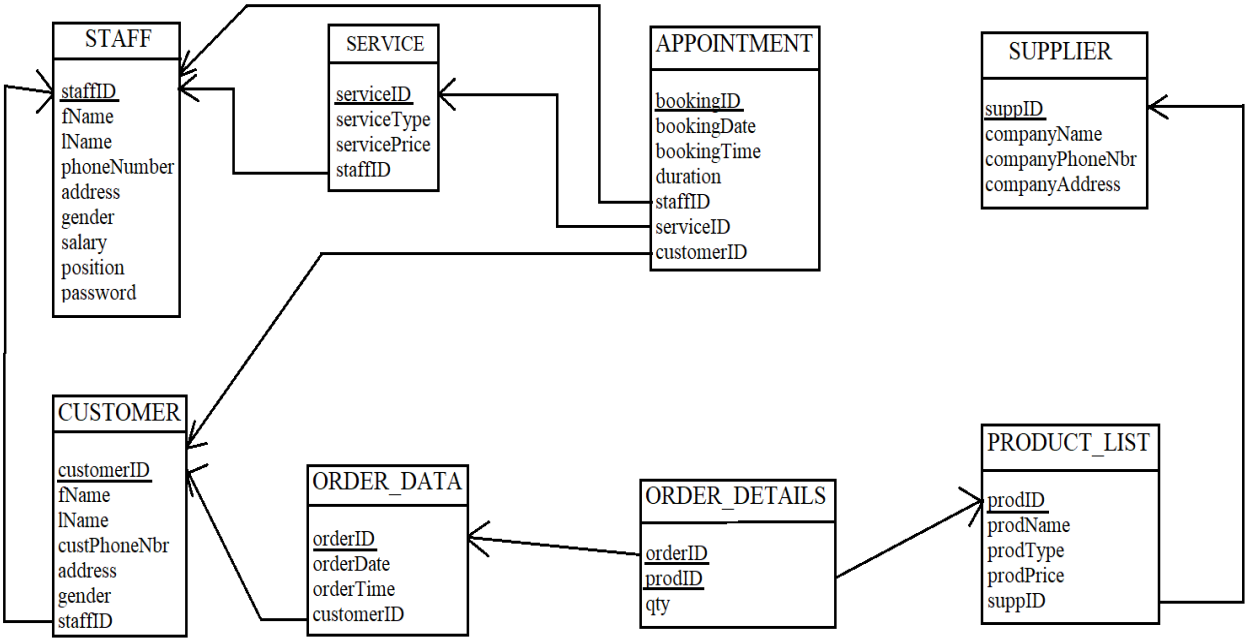


Figure 2: LDM of the BeautySalon

Relations: Staff, Customer, Service, Order_Data, Order_Details, Appointment, Supplier, Product_List are relations based on the entities we had in our CDM. Looking at the cardinality constraints we have in our project, we had to create one extra relation due to the (1:n) - (1:n) cardinalities. This new relation is the ORDER_DETAILS which contains orderID and prodID as primary keys and an extra attribute from the relationship contains which is quantity.

Following the creation of the LDM, we can easily create a database in **MySQL** called BeautySalon, build its tables (which are defined by the relations above), add the appropriate constraints and insert values if we want.

3.3 GUI

After building the relational data model of the database, we need to think about an interface that allows the non-expert user to create and manipulate the data. We will build this user friendly interface using javafx and with the help of css. First, we need to connect MySQL, which contains the tables of our database and the data, to java, which will allow us to create the interface. This would be achievable with the help of java database connectivity (jdbc). We will add MySQL connector to our package application. Second, we need to populate the database data in GUI. In order to do the following, we need to add dynamically the table columns to the TableView then we add the data to the ObservableList by iterating the rows and the columns. Also, we preserved some constraints using this interface. Since MYSQL was not allowing us to SET DEFAULT on delete and on update, to preserve the data, we updated the foreign key to a default one (0000) each time a primary key is deleted (the foreign keys depending on this primary key were set to the default value already inserted into each corresponding table). Also, to enforce the Referential Integrity constraint, we

used ComboBoxes that contains the values that we are allowed to use in case we are doing an insert. Also, we made sure to update those ComboBox in case any tuple was added or deleted from the table attribute/column that it is referring to. Moreover, we added a nice feature that allows the user to do a customized search by inserting in certain TextFields data, then we displayed the resulting query as it is the TableView. This enables the user to see the tuples he/she is targeting.In this project, we used from the simple queries to more complex ones and we executed them using getConnection().prepareStatement().executeQuery(SQL) with having getConnection() as a method that returns the connection and SQL as a string that contains the query.

The Classes Scene1 ,Scene 2 , Scene 3.., Scene 4.., Scene 5.. , Scene 6.. deals respectively with login, the choice of table, the view of the data, the modification of the data, the selection of the data,…

4 EXPERIMENTAL EVALUATION

After creating our tables, we now need to test our work. This is done by creating queries. A query is a question we ask the database and it responds accordingly by presenting the relevant records.
Many queries were implemented that dealth with searching, adding, updating, deleting, ordering, grouping, projecting,… The queries we chose to highlight in this report are the following:

★ Query 1 : compare the credentials entered by the user with the ones in the table to be able to log in

- For the expert user, the following query returns an empty table if the login information is incorrect:

SELECT password FROM STAFF WHERE staffID = **getStaffID()** AND password = **getPassword()**;

```
mysql> SELECT password FROM STAFF WHERE staffID = "ST000" AND password = "Boss";
+----------+
| password |
+----------+
| Boss     |
+----------+
1 row in set (0.00 sec)

mysql> SELECT password FROM STAFF WHERE staffID = "ST000" AND password = "Bos";
Empty set (0.00 sec)
```

Figure 3 - Login process

- For the non-expert user, he/she should basically enter the credentials and either press on enter or on the LOGIN button.

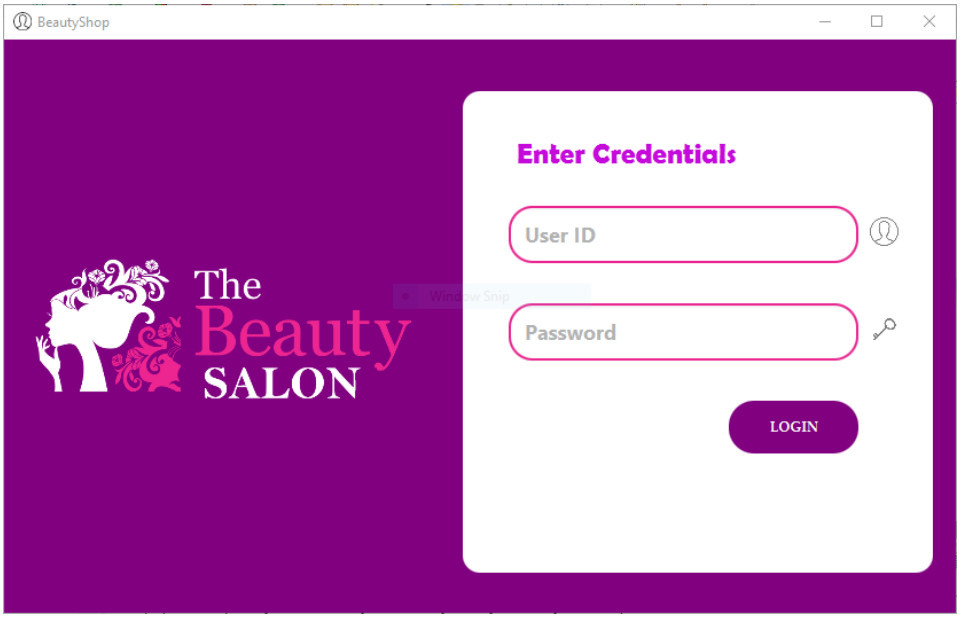


Figure 4 - LogIn Interface

★ Query 2: Select the details of the orders and their products on a certain date:

- The expert user should be able to see those data by using:

SELECT A.bookingID, A.bookingDate,A.bookingTime, A.duration, A.custId , C.fName ,c.lName ,A.staffID, S.serviceId, S.servicetype, ST.fname , st.lname FROM APPOINTMENT A, SERVICE S, STAFF ST, CUSTOMER C WHERE A.serviceID=S.serviceID AND A.custID=C.custID AND S.staffID=ST.staffID ;

```
mysql> SELECT A.bookingID, A.bookingDate,A.bookingTime, A.duration, A.custId , C.fName ,c.lName ,A.staffID, S.serviceId, S.servicetype, ST.fname , st.lname   FROM APPOI
NTMENT A, SERVICE S, STAFF ST, CUSTOMER C WHERE A.serviceID=S.serviceID AND A.custID=C.custID AND S.staffID=ST.staffID ;
+-----------+-------------+-------------+----------+--------+-----------+----------+---------+-----------+-------------------+--------+--------+
| bookingID | bookingDate | bookingTime | duration | custId | fName     | lName    | staffID | serviceId | servicetype       | fname  | lname  |
+-----------+-------------+-------------+----------+--------+-----------+----------+---------+-----------+-------------------+--------+--------+
| B001      | 2017-12-08  | 10:00:00    |       30 | C001   | Rachelle  | Rachid   | ST002   | SE001     | Make up           | Clara  | Akiki  |
| B002      | 2017-12-08  | 10:00:00    |       10 | C002   | Gaelle    | Abboud   | ST003   | SE002     | Manicure          | Lea    | Sawaya |
| B003      | 2017-12-08  | 10:30:00    |       10 | C003   | Maria     | Hitti    | ST003   | SE002     | Manicure          | Lea    | Sawaya |
| B007      | 2017-12-09  | 11:00:00    |       10 | C005   | Marinelle | Attieh   | ST003   | SE002     | Manicure          | Lea    | Sawaya |
| B008      | 2017-12-09  | 11:10:00    |       10 | C005   | Marinelle | Attieh   | ST003   | SE003     | Pedicure          | Lea    | Sawaya |
| B011      | 2017-12-09  | 14:00:00    |       30 | C002   | Gaelle    | Abboud   | ST003   | SE004     | Waxing            | Lea    | Sawaya |
| B005      | 2017-12-08  | 13:00:00    |       30 | C005   | Marinelle | Attieh   | ST001   | SE005     | Facial Massage    | Joseph | Attieh |
| B004      | 2017-12-08  | 11:00:00    |       60 | C004   | Agapius   | Bou ghosn | ST001   | SE006     | Body Massage      | Joseph | Attieh |
| B009      | 2017-12-09  | 12:00:00    |       60 | C006   | Christelle | Saliba   | ST001   | SE006     | Body Massage      | Joseph | Attieh |
| B010      | 2017-12-09  | 14:00:00    |       60 | C001   | Rachelle  | Rachid   | ST002   | SE007     | Laser epilation   | Clara  | Akiki  |
| B006      | 2017-12-09  | 10:00:00    |       30 | C007   | Grace     | Akiki    | ST002   | SE008     | Electric epilation | Clara  | Akiki  |
+-----------+-------------+-------------+----------+--------+-----------+----------+---------+-----------+-------------------+--------+--------+
11 rows in set (0.01 sec)
```

Figure 5 – Refresh Demo

- The non-expert user has to press Refresh and the table will be loaded in the tableView.

Figure 6 - Refresh Demo

★ Query 3: modify the attributes of a product in PRODUCT_LIST

- The expert user should execute this query:

UPDATE PRODUCT_LIST SET prodId = **getProdId()** , prodName = **getProdName()** , prodType = **getprodType()** , prodPrice = **getprodPrice(),** suppID = **getSuppId()** WHERE prodId =**getProdId();**

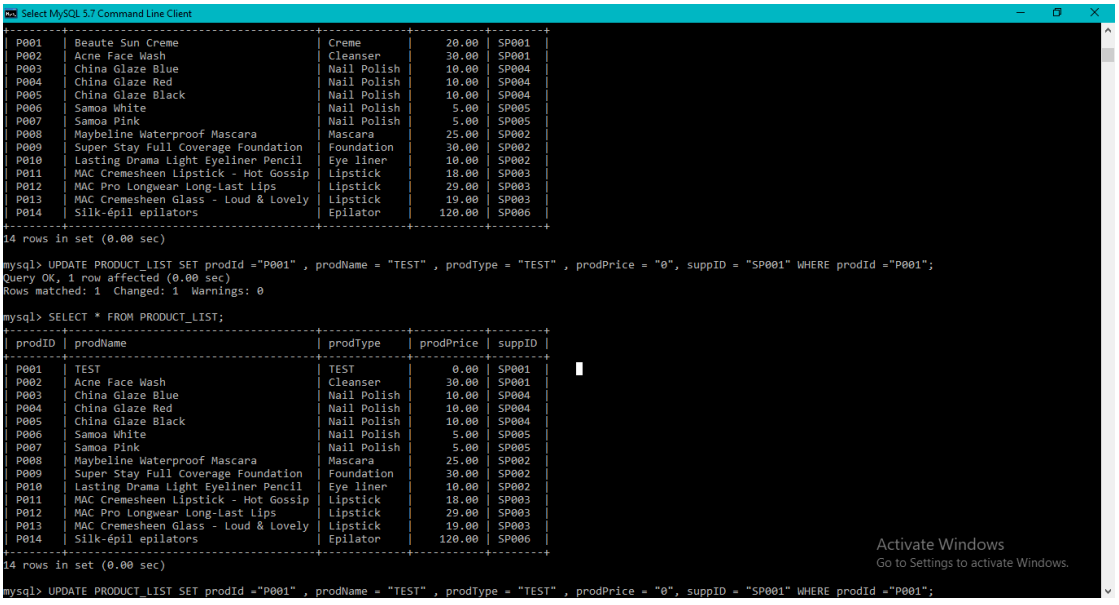


Figure 7 - Demo of an update

- The non-expert user press on the load button which loads the specific tuples using its prodId and is to be made modifiable by changing the contents of the TextFields/ComboBox and by pressing update. The ComboBox was provided to restrict the user with certain values to reinforce referential integrity. Also, we made sure to update the ComboBox each time the data it represents was modified.

Figure 8 - Attributes Ready to be updated

★ Query 4: delete a supplier from the database having its supplier ID

- The expert user can do the following using this query:

  DELETE FROM SUPPLIER WHERE suppId = **getSuppId()**;

  This query would result into null fields in the tables where customerId is a referential key. So, the user could perform additional queries to avoid this by setting the other elements into a default one.

  One of these queries are:

  UPDATE PRODUCT_LIST SET suppId = "SP000" WHERE suppId =**getSuppId();**

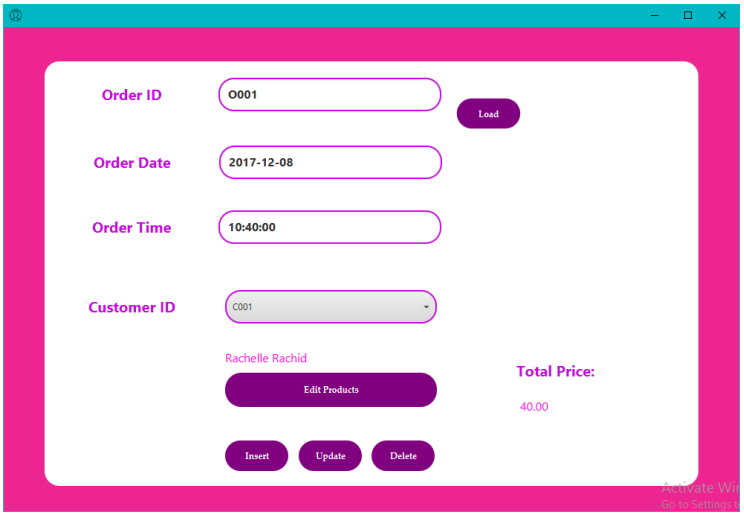
- The non-expert user can simply put the SupplierId in the corresponding box and press the delete button. The delete feature was done in a way that to prevent showing up empty attributes and null fields in the tables which have the primary key of Supplier as their foreign key, we simply updated those tuples in a way that it sets the foreign key to a default one then perform the deletion (as if we put in a foreign key contraint SET DEFAULT).

★ Query 5: get the total price of an order

- The expert user uses this query to get the total price:

  Select SUM(qty*prodPrice) FROM ORDER_DATA ODA, ORDER_DETAILS ODE, PRODUCT_LIST PL WHERE ODA.orderID=ODE.orderID AND ODE.prodID=PL.prodID AND ODA.orderID = **getorderID()** GROUP BY ODA.orderID;

- The non-expert user enters the orderId in the TextField and press load to be able to view the total price:



More queries can be found in the project, but those were worth highlighting.

5 CONCLUSION

As explained above, our database represents a Beauty Salon and includes the basic uses of it. It can be implemented in any Beauty Salon though some minimal changes would have to be made. Also, to improve our project, we could add more functionalities and better graphical interface so that the user could have better use of the system. Also, at this level the system fully depends on the user and cannot fully perceive if he make an error and cannot undo it. All in all, this project can work for a small shop but if we were to have more than 100-200 customers MySQL would not be the ideal DBMS to be used and we would have to go to another one such as Oracle DB for example.