**COMP2411**

**Database Systems Project Report**

**Group 27**

**Group Members:**

**GUNAWAN Aristo Sinatra 20070491D**

**MAZUMDER Arnab 20041744D**

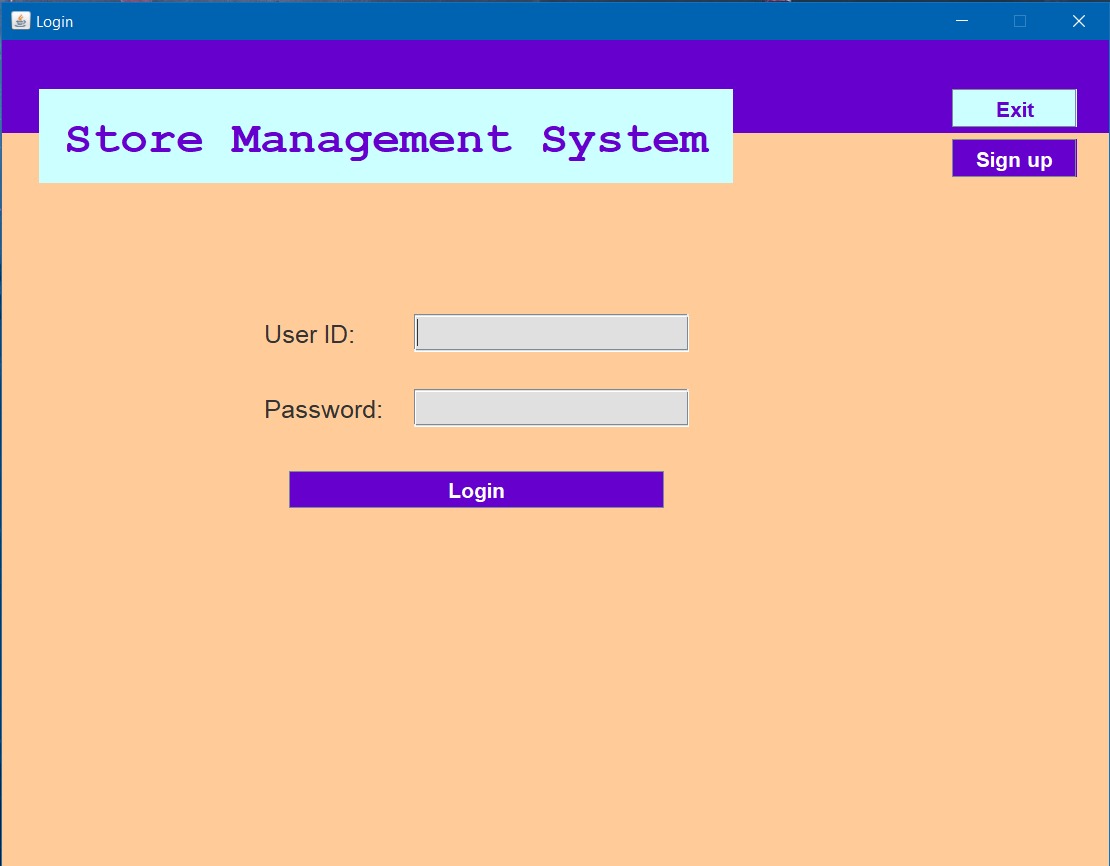
**TEXEIRA REIS Tiago 19085298D**

**WOEN Jerry 19085473D**

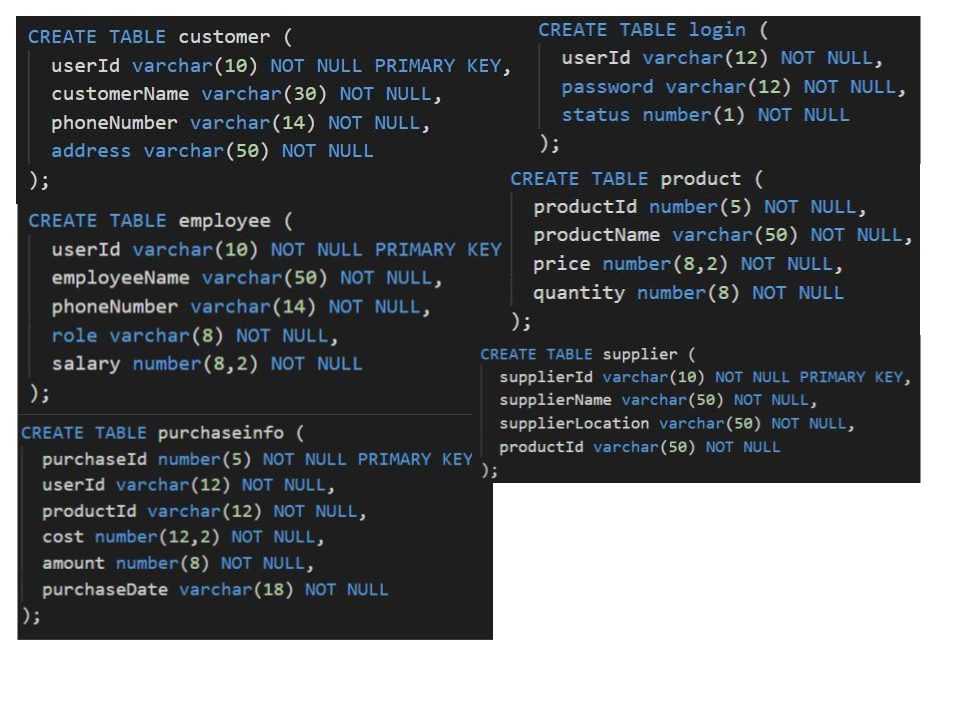
## 

## Abstract

Our goal is to provide retail businesses with a reliable and efficient data-base management application that provides a three-layer abstraction for company hierarchy. More specifically our application supports merchandise selling businesses such as supermarkets and retail stores, meaning that all software features are built around the software needs of such vendors. Hence our application supports employee/customer login and provides different functionalities for the differing levels of abstraction. It also supports functionalities for retail businesses, such as Managers having a higher level of access and ability to utilize all available functions to manage the business and database, as compared to the general employees. Our application allows for three levels of users to login, the customers who utilize the lowest level of the database such as viewing the database items and their personal information and the employee and manager who share access to the database but interact with the database differently. Additionally, the application produces a visual graphic where all of the functions can be interacted with and the database items can be seen.

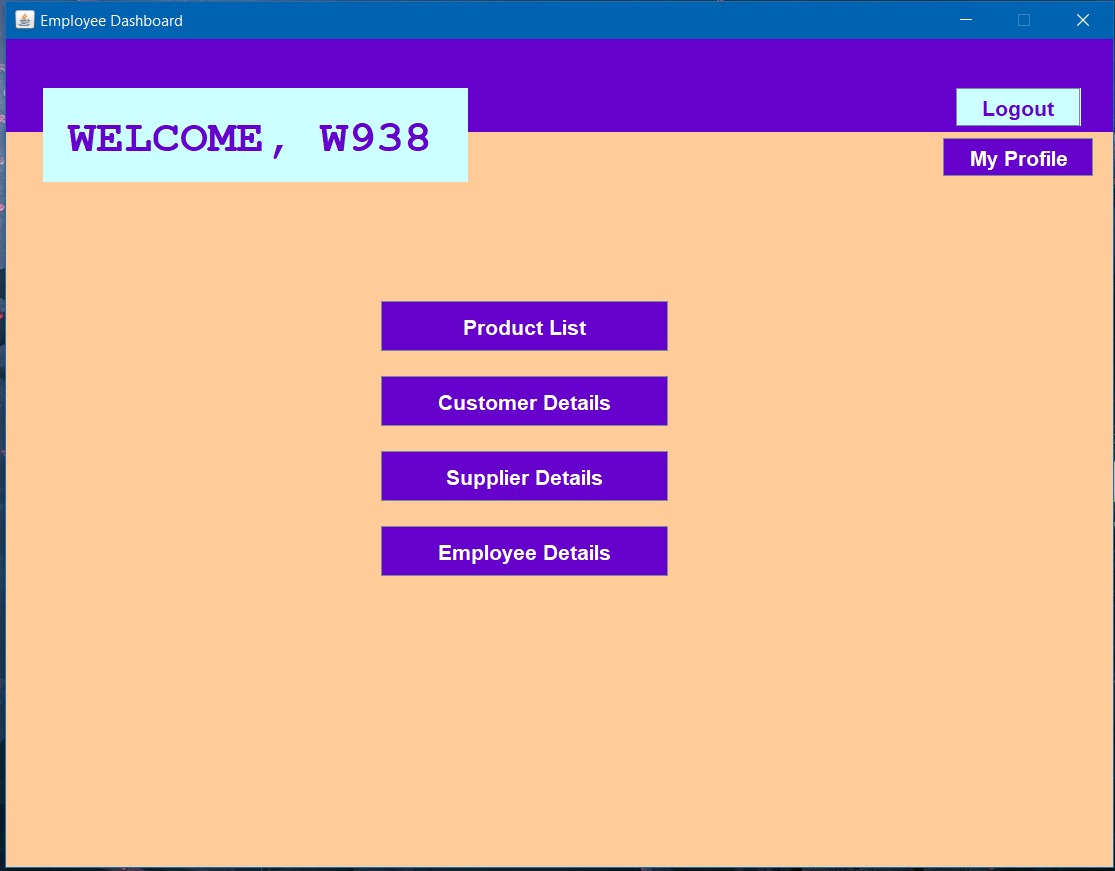


On the server-side of our application, our database runs through Oracle SQL and contains 5 tables that interact with each other and hold all relevant information. The tables all contain all relevant employee, customer, product and purchase information from the business.

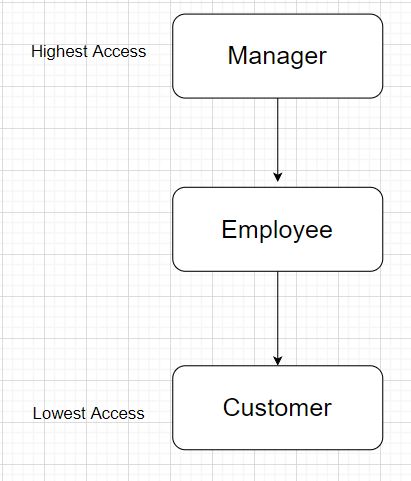


## **Main Content**

### **Design**



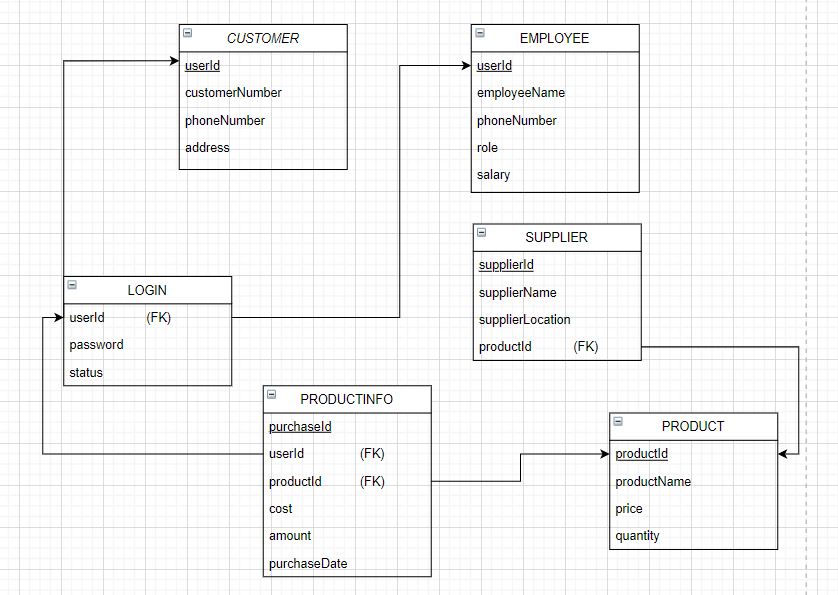
The application on the client side supports three levels of access abstraction, where access to functions and information depends on level. The manager level represents the highest level of access, and acts as the database administrator, managing user’s access and database information. Furthermore, Manager handles the cataloging of merchandise/products and can directly access the personal details of other employees of the store. Whereas, the general employees have more basic database access functions, such as viewing the product/merchandise and selling them to customers, but they are not able to view the details of the other employees in the store. That is why customers can only modify personal information and view items available for purchase, as they have the lowest level of access into the database; they are only allowed to view a fraction of the database’s information. This abstraction dictates what appears on the application for the users, with features being directly tied to database access needs of the user.



**Access Level Abstraction**

The database is built from five entity sets that contain all of the business’s important data. The database will hold employee, customer, product, login and purchasing information in individual tables that link together through keys and relationships. This means that tables that contain purchasing information will take primary key values from other tables to make each entry unique, as purchases of products/merchandise should not be equivalent to avoid errors and confusion among database users. These entity sets access is not equivalent among users, and depends on the access level of the user’s position.

**Relational Model for our Application:**



As we can see from this relational model, all of the relation between tables gives us a better idea on how a table with plenty of foreign keys contains information about various other tables, such as shopping purchase receipts having customer ID. Furthermore, for the sake of convenience and normalization, all tables contain a primary key, and some tables when necessary, take foreign keys.

## **Learning Experiences**

In general, we have mostly gained knowledge about the implementation of an Oracle Database and the SQL programming that has to be performed to create the tables; like dictating primary and foreign keys, inserting all of the available information into their respective tables and the different ways to access that information later on.

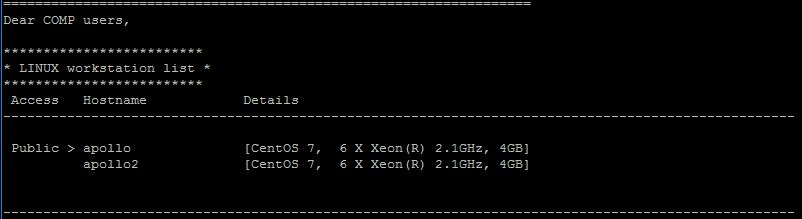
As the database must be hosted on PolyU COMP Oracle servers, we had to learn how to enter the PolyU COMP servers through the terminal, using our student ID’s to upload and host our databases. Furthermore, as the majority of our group cannot access PolyU’s labs directly, we had to share files through WinSCP and upload them into our respective student folders. Similarly, as this is the first proper implementation of an application utilizing a database and we had to learn how to program using the JDBC drivers to connect to the oracle server and database and then execute queries and receive the resulting information. Since we have developed a visual GUI application program, it required us to present data in tables, which was a new experience as well. Alongside this, we also had to put our prior programming experience and knowledge to practice, in order to utilize the Java frame classes for the development of the interactive GUI features of our application.

Example of the JDBC driver implementation on Java:

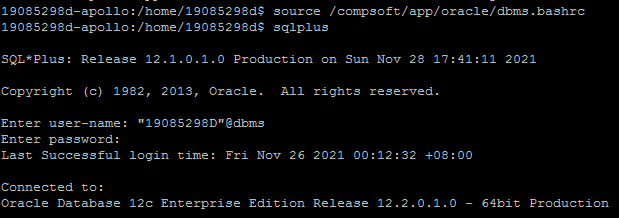


Additionally, as we were writing down the SQL code and commands, we bumped into countless difficulties along the way. A lot of changes had to be made to morph the database into one that we envisioned from the relational model. It took quite a while, as the relational model is also continuously changing as we move forward, and necessary changes needed to be made. Syntax errors were also persistent. Debugging was a hassle, as it took quite a while to understand all of the compilation errors and make the necessary adjustments. However, teamwork worked its charm as we were finally able to get the database up and running in the end.

Connection to COMP Server:



Connection to Oracle Server:



## 

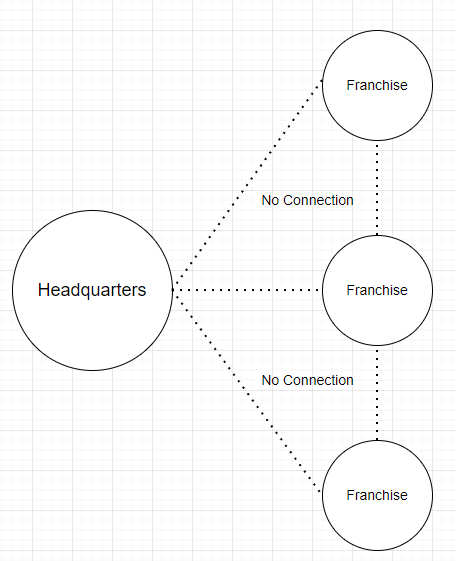
## 

## **Discussion and Conclusion**

### **Limitations:**

The DBMS application we have developed serves a specific retail purpose and its functionalities extend to serve the needs of those in the industry. Hence businesses that do not sell merchandise or products will have difficulties using our application as the database, alongside the functions, as they were built to support merchandise-based businesses. Furthermore, our database does not provide storage capabilities for invoices and only supports the storage of customer receipts. The application is also only built for individual franchise locations, meaning that the database is independent and has no connection to other chains of the same retail branch. This means that if this database is implemented in a business that hosts multiple franchised locations, the databases will have no data regarding the branches or overall company data such as headquarters location.

Example of Branch Independence in the Database:



### **Conclusion**

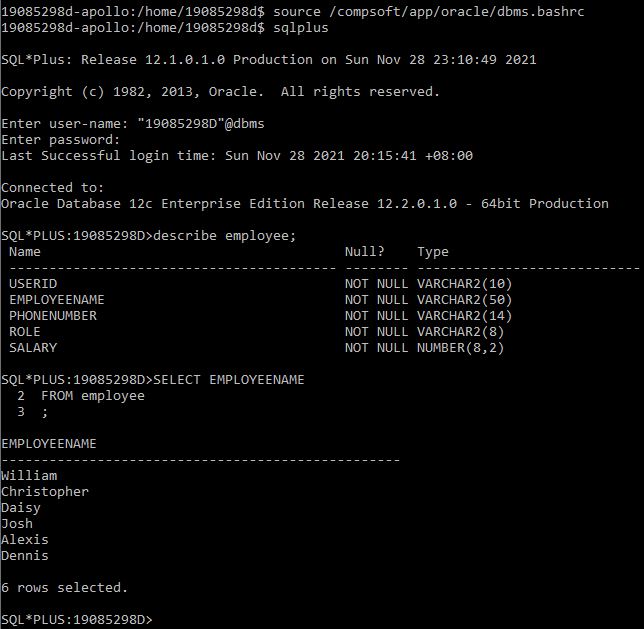
Our Store Management System Application serves a niche group of locally-owned businesses who wish to have a reliable database with access level abstractions. The database stores all essential business data while providing useful functions for data insertion, deletion and selection. Furthermore, the database provides a platform for further communication with customers, as additional functionalities can be added to the customers access level. Additionally, our Application supports retail based businesses who wish to sell merchandise and products to a customer, hence database access will be linked to typical job hierarchies in those businesses, being that Managers hold the most access.

### 

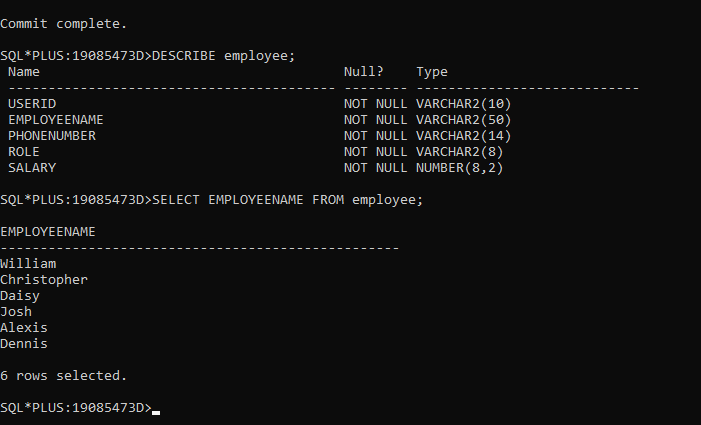
### 

### **Appendix:**

**Tiago Teixeira Reis 19085298D:**

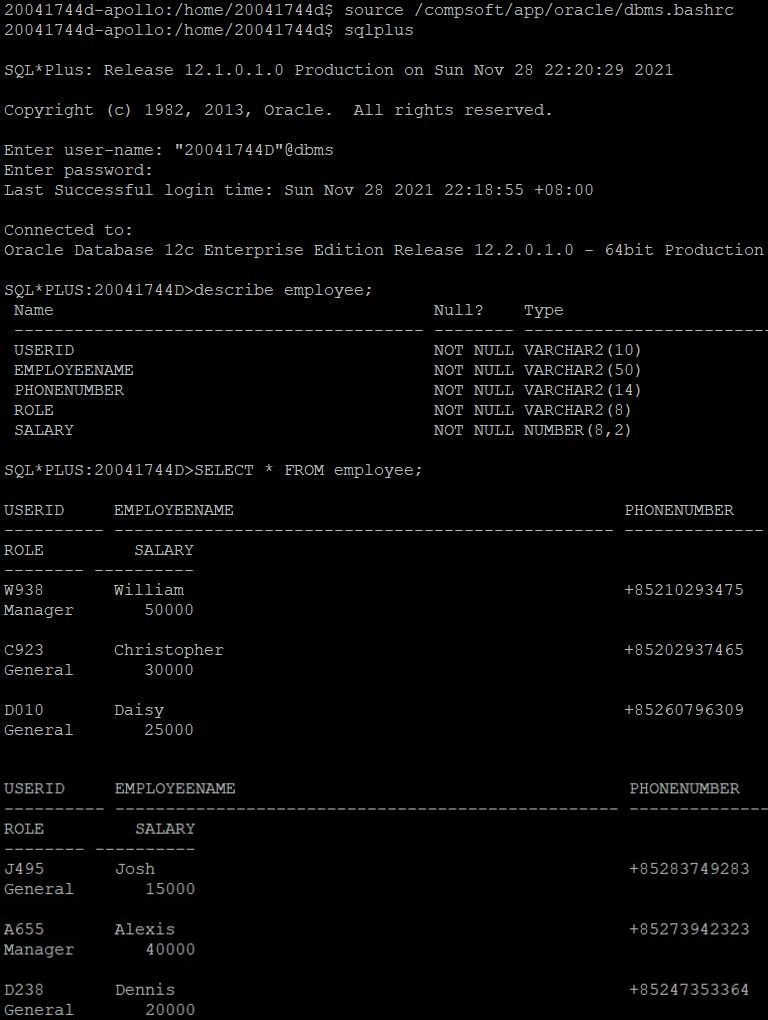


**Jerry WOEN 19085473D:**



References:

**Arnab MAZUMDER 20041744D:**



**GUNAWAN Aristo Sinatra 20070491D:**

