CSC3170 Database system Report

Group 11

Xu Xiangyu
Zhang Fengyu
Huang Junlin
Xu Boshi
Guo Chaojin
Zhang Sheng
Zhao Hongyi

Contents

1 Introduction **3**

1.1 Background . . . 3

1.2 Motivation . . . 3

2 Design **3**

2.1 design Overview . . . 3

2.2 Entity list . . . 4

2.3 E-R diagram . . . 5

2.4 Schemas . . . 6

3 Implementation **6**

3.1 Overall implement . . . 6

3.2 Creating tables via SQL . . . 7

3.3 User Interface and Sample Queries . . . 7

3.3.1 Login . . . 7

3.3.2 selecting goods from price range . . . 8

3.3.3 identifying best-sellers . . . 8

3.3.4 advanced queries for admins . . . 9

4 Conclusion **11**

5 Self-evaluation **11**

6 Reference and Appendices **12**

1 INTRODUCTION

1.1 BACKGROUND

In today's world, shopping malls are becoming increasingly popular due to their ability to offer a wide variety of products and services in one place. As the number of visitors to shopping malls continues to grow, it has become crucial for mall owners and managers to have a comprehensive database management system (DBMS) that can handle the increasing volume of data and transactions.

The purpose of this project is to put forward a DBMS for a shopping mall **"WeMall"**. WeMall can effectively manage the various data elements, such as customer and employee information, inventory, sales, and transactions. This report will outline the key features and functionalities of the proposed system, including data modeling, database design, and user interface design.

1.2 MOTIVATION

As the amount of data burgeons, keeping track of everything in a shopping mall can be a daunting ordeal. The motivation behind developing this DBMS system is to help mall owners and managers efficiently manage their operations and provide a better shopping experience for their customers. With a well-designed DBMS system, mall owners can track their inventory levels, record transactions, analyze sales trends, and manage promotions in real-time. A well-designed DBMS system can also help mall owners and managers to better understand their customers by tracking their purchasing behavior, preferences, and demographics, meanwhile benefiting customers by providing them a seamless shopping experience.

2 DESIGN

2.1 DESIGN OVERVIEW

The database design includes two parts: Consumer and Supplier. The Consumer part has four entities: customers, goods, sales, and sales_items, while the Supplier part has six entities: employees, purchase_order, inventory, departments, jobs, and supplier. The customers entity records user information, goods entity records product information, sales entity records bill information, and sales_items entity records customer shopping bills. The employees entity records employee personal information, purchase_order entity records purchase information, inventory entity records the inventory of goods, departments entity records department information, jobs entity records job types and salary ranges, and supplier entity records supplier information.

2.2 ENTITY LIST

customers(customer_ID, password, reward_points, phone_number, registration_time)
Description: The *customers* entity is used for collecting user information. Each customer account has an unique Customer ID, together with password, reward_points, phone_number, and registration_time. Each account is bounded to a telephone number. Note that a person can register multiple accounts.

goods(goods_ID, goods_name, brand, category, price, discount, size_range)
Description: The *goods* entity is designed for collecting relevant information about products sold in the mall. Its attributes include the unique goods_ID, goods_name, brand, category, price.

sales(sale_ID, customer_ID, employee_ID, sales_date, sales_time, total_price)
Description: The *sales* entity records information such as bill number and date, combined with sales items to form a complete bill. Its attributes include the unique sale_ID, customer_ID, employee_ID, sales_date, sales_time, and total_price.

sales_items(sale_ID, goods_ID, amount, price)
Description: The *sales_item* entity works as a weak entity, combining sales and customer information to record customers' shopping bills. Its attributes include the composite primary key sale_ID, goods_ID and amount.

employees(employee_ID, employee_name, email, phone_number, hire_date, job_ID, salary, department_ID, manager_ID)
Description: The *employees* entity is designed to collect employees' personal information, such as their name, phone_number; Other attributes include key data related to their job – the unique employee_ID, employee_name, email, phone_number, hire_date, job_ID, salary, department_ID, and manager_ID.

purchase_order(order_ID, supplier_ID, employee_ID, purchase_date, goods_ID, purchase_amount, total_cost)
Description: The *purchase_order* entity collects purchase information of shopping malls. Its attributes include the unique order_ID, supplier_ID, employee_ID, purchase_date, goods_ID, purchase_amount, and total_cost.

inventory(goods_ID, inventory, cost, inventory_location)
Description: The *inventory* entity keeps record of the goods inventory in the mall, including the quantity of goods purchased and sales volume. Its attributes include the unique goods_ID, inventory, cost, and inventory_location. This entity plays a key role in recognizing in a timely

fashion any shortage of goods, meanwhile analyzing best-sellers so that the amount of goods purchased in the mall can be adjusted based on different sales volumes.

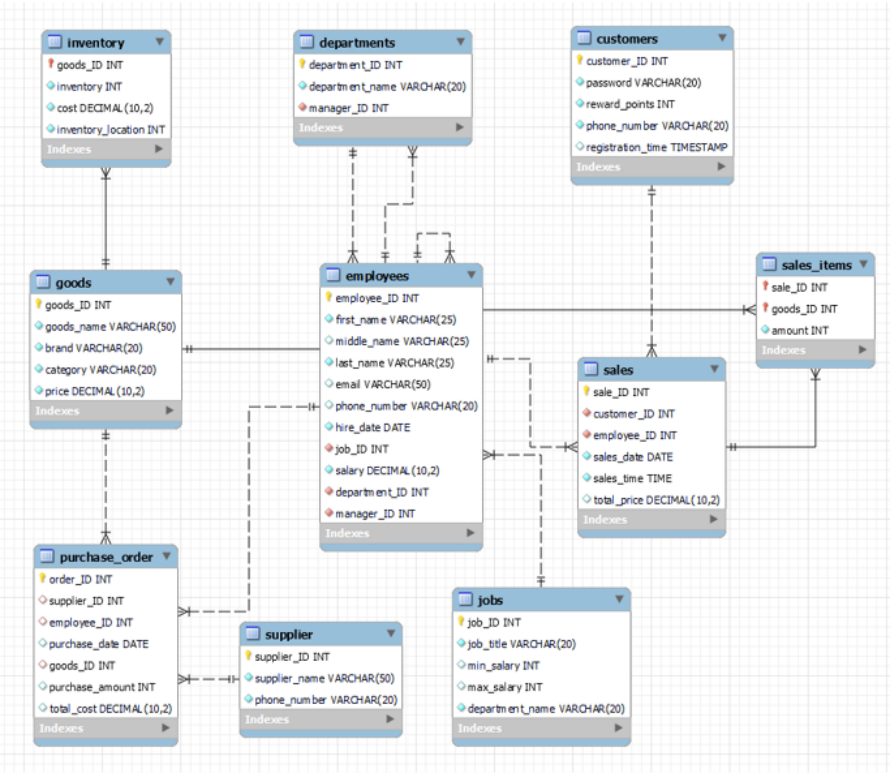
departments(department_ID, department_name, manager_ID)
Description: The *departments* entity stores information about the department to which each employee belongs, as well as the management personnel of each department. Its attributes contain departments_ID, department_name, and manager_ID.

jobs(job_ID, job_title, min_salary, max_salary, department_name)
Description: The *jobs* entity stores information about the types of jobs in the mall, as well as information about each job. Its attributes include the unique job_ID, job_title, min_salary, max_salary, and department_name.

supplier(supplier_ID, supplier_name, phone_number)
Description: The *supplier* entity stores the suppliers' information. Its attributes include a unique supplier_ID, supplier_name, and phone_number.

2.3 E-R DIAGRAM

After introducing the entities, we need to design the Entity-Relationship model. The E-R diagram that depicts the above entities and the relationships among them is shown as follows:



2.4 SCHEMAS

All the schemas in the E-R diagram are in BCNF. The properties of the schemas are shown as follows:

Schema	Primary keys	Foreign keys	Function dependencies
Customer	customer_ID		customer_ID -> password, reward_points, phone_number, registration_time
Goods	goods_ID		goods_ID -> goods_name, brand, category, price
Inventory	goods_ID	goods_ID	goods_ID -> inventory, cost, inventory_location
Purchase_order	order_ID	goods_ID, supplier_ID, employee_ID	order_ID -> supplier_ID, employee_ID, purchase_date, goods_ID, purchase_amount, total_cost
Supplier	supplier_ID		supplier_ID -> supplier_name, phone_number
Sales	sale_ID	customer_ID, employee_ID	sale_ID -> customer_ID, employee_ID, sales_date, sales_time, total_price
Sales_items	sale_ID, goods_ID	sale_ID, goods_ID	sale_ID, goods_ID -> amount
Employees	employee_ID	job_ID, department_ID, manager_ID	employee_ID -> first_name, middle_name, last_name, phone_number, email, hire_date, job_ID, salary, department_ID, manager_ID
Departments	department_ID	manager_ID	department_ID -> department_name, manager_ID
Jobs	job_ID	department_ID	job_ID -> job_title, min_salary, max_salary, department_name

3 IMPLEMENTATION

3.1 OVERALL IMPLEMENT

With the schemas ready, we implemented the design to utilize our database design in the context of our mall management system. We chose Mysql as our DBMS, and used Vue.js and Express to incorporate the database into our system.

3.2 CREATING TABLES VIA SQL

According to the schemas we derived above, we use SQL syntax to create a table for each entity. There are ten tables in total with the sample code of the ”customers” table shown below. The complete code of the table design is included in the appendix.

```
CREATE TABLE Customers (
  customer_ID INT PRIMARY KEY,
  password VARCHAR(255),
  reward_points INT,
  phone_number VARCHAR(20),
  registration_time DATE
);
```

3.3 USER INTERFACE AND SAMPLE QUERIES

This section introduces the implementation of the user interface of our **WeMall** mall management system. We will also show some sample SQL queries and the corresponding SQL code behind them. The complete SQL codes are attached in the appendix.

3.3.1 LOGIN

The client needs to login the system before performing any operations. The login page is shown below. It contains two modes: the user mode and the administrator mode. The client shall type in her/his username and password, and click the “Login” button.

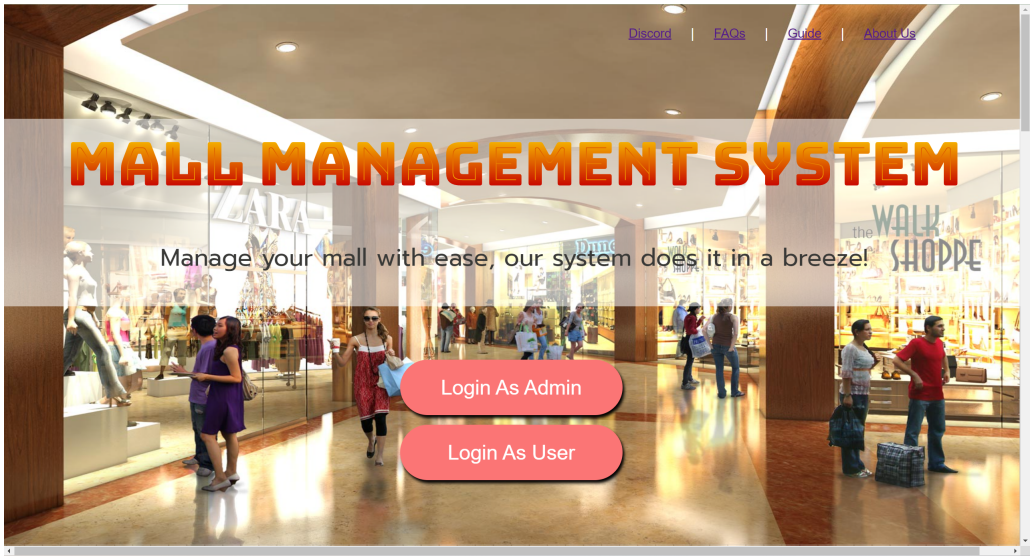


FIGURE 1: LOGIN

After logging in, the client can perform various queries on the website. The queries may vary depending on whether the client is a user or administrator.

3.3.2 SELECTING GOODS FROM PRICE RANGE

Firstly, we can query for products within a specific price range. By simply inputting a minimum and maximum price, the website will return all food items that meet the criteria. The sample UI and the corresponding SQL code are shown below.

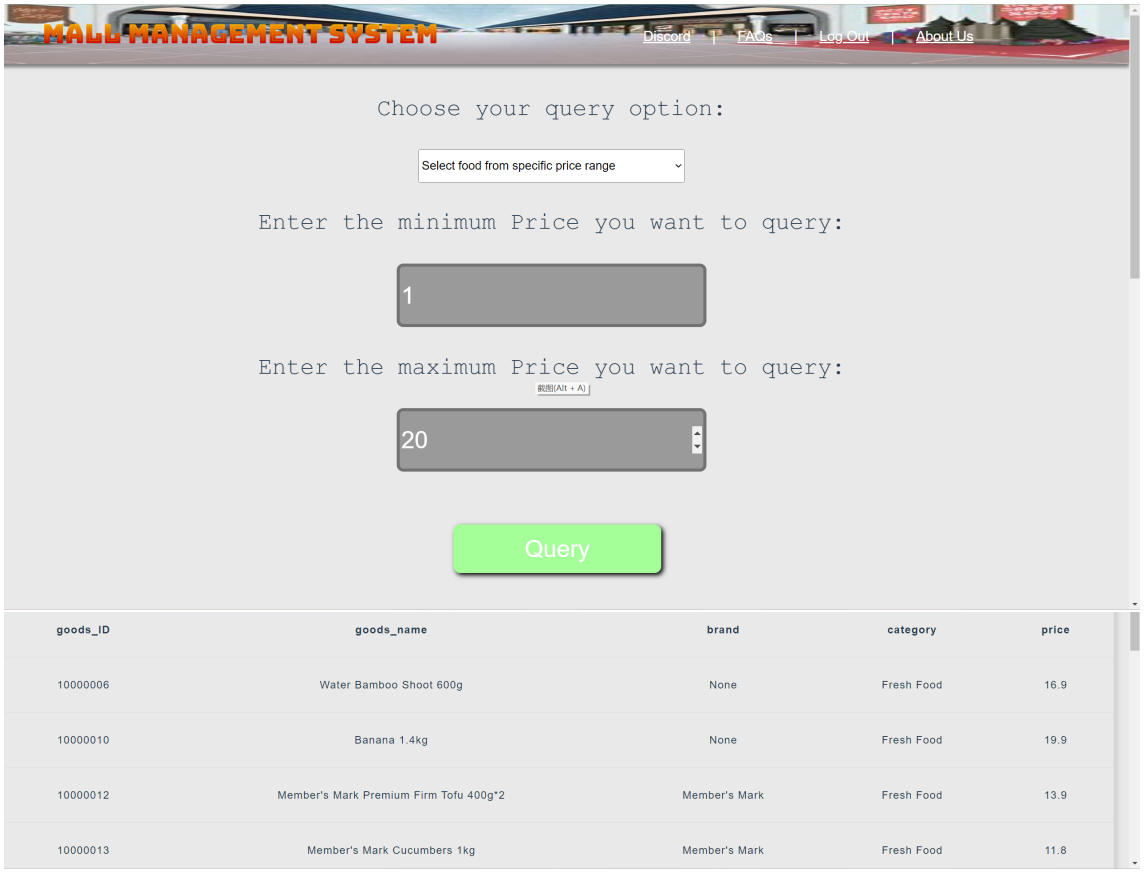


FIGURE 2: SELECTING GOODS FROM PRICE RANGE

```
SELECT * FROM Goods WHERE price BETWEEN min_price AND max_price;
```

FIGURE 3: CORRESPONDING SQL CODE

3.3.3 IDENTIFYING BEST-SELLERS

We can also query for products that sells well. By inputting the number of food items to query for, the website will return the best-selling food items. In the sample UI and the corresponding SQL code below, we select the top 10 best-sellers of this mall. The parameter can be changed.

FIGURE 4: IDENTIFYING BEST-SELLERS

```sql
SELECT g.goods_name, SUM(si.quantity) as total_quantity, SUM(si.total_price) as total_sales
FROM Sales s
JOIN Sales_Items si ON s.sale_ID = si.sale_ID
JOIN Goods g ON si.goods_ID = g.goods_ID
GROUP BY g.goods_name
ORDER BY total_sales DESC
LIMIT 10;
```

FIGURE 5: CORRESPONDING SQL CODE

3.3.4 ADVANCED QUERIES FOR ADMINS

Aside from the basic functions for users listed above, we have also implemented many advanced features for administrators. We can query for customers whose reward points are greater than a certain value, retrieve detailed information about a customer with a specific ID, retrieve the names of customers who registered on or after a certain date, retrieve all customer phone numbers, retrieve the average salary of employees in each department along with the department name, retrieve the number of employees in each department, and retrieve employees in ascending order of salary and retrieve the names and salaries of all employees whose salary is greater than a specific value.

Here is the sample UI and the corresponding SQL code for retrieving the customer ID of those who registered on a specific date. Note that the ”2019-01-01” is just a placeholder and its actual value depends on what the administrator types in the query:

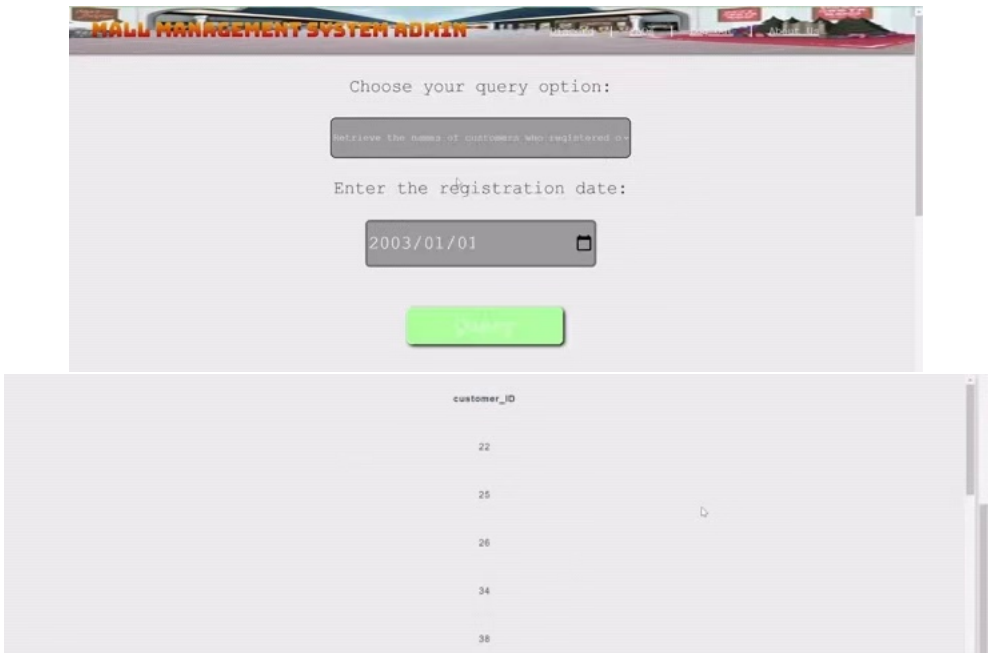


FIGURE 6: SELECTING CUSTOMERS BY DATE

```
SELECT customer_ID FROM Customers WHERE registration_time >= '2019-01-01';
```

FIGURE 7: CORRESPONDING SQL CODE

Here is the sample UI and the corresponding SQL code for retrieving the average salary of employees in each department along with the department name:

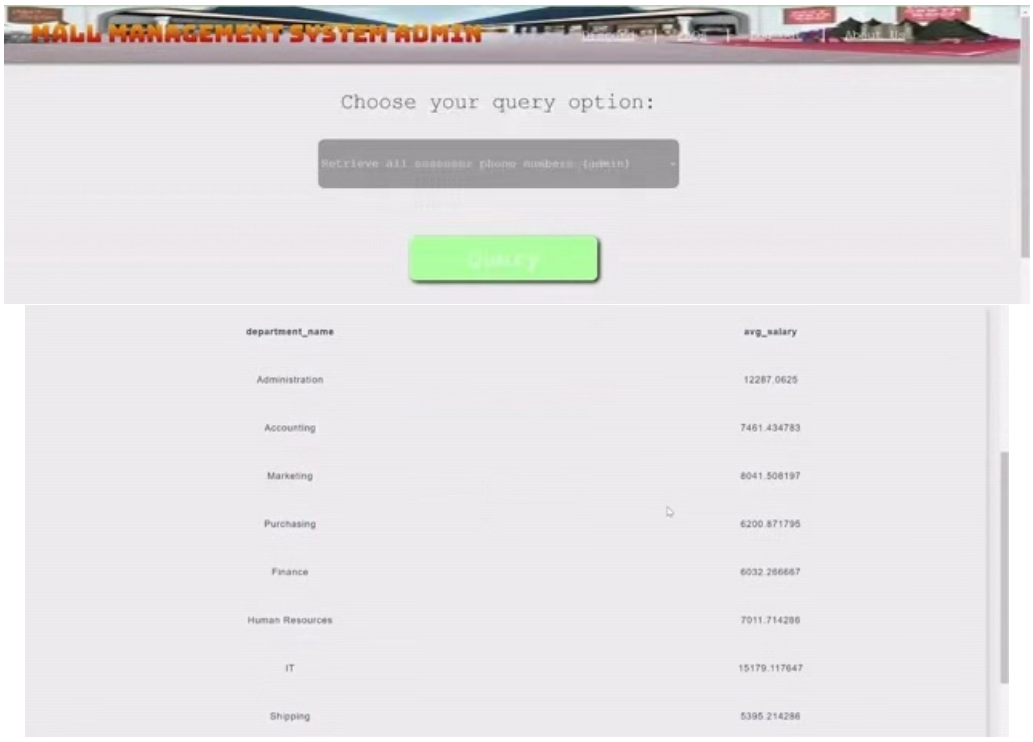


FIGURE 8: RETRIEVING AVG SALARY OF EMPLOYEES

```sql
SELECT d.department_name, AVG(e.salary) as avg_salary
FROM Employees e
JOIN Departments d ON e.department_ID = d.department_ID
GROUP BY d.department_name;
```

FIGURE 9: CORRESPONDING SQL CODE

The rest of the code and queries are not included due to the length of the report. Please refer to appendix.

4 CONCLUSION

Our *WeMall* project has successfully designed a comprehensive database management system for shopping mall operations. The database encompasses a wide range of entities such as customers, employees, inventory, and suppliers, and has been designed based on a thorough analysis of real-life shopping mall scenarios. A well-defined entity-relationship diagram has been formulated, and the schema has been normalized up to BCNF, despite being non-dependency preserving. The user interface has been developed via Vue.js and MySQL connection has been integrated, enabling the execution of various queries, including queries for products from a specific price range, category, brand, etc, and advanced queries for retrieving specific customer/employee information.

Overall, the database design and implementation for the *WeMall* project could be a decent contribution to the field of shopping mall management, providing potential robust and efficient solutions for handling various operations and entities in a shopping mall. Further research and improvements could be made to enhance the database's functionality and performance, thereby contributing to the field.

5 SELF-EVALUATION

WeMall can provide an abundance of useful features and functionalities. The highlights of our project are:

- The database structure is well-organized, with all schemas normalized to BCNF, ensuring data integrity and minimizing data redundancy.
- The website provides an intuitive and efficient visual interface that enables users to implement queries easily, promoting ease-of-use and a positive user experience. It can act as both a perspicuous platform for data visualization and a convenient system for data operation.

- The database contains a diverse range of tables and functionalities to cater to a wide variety of use cases and scenarios, ensuring versatility and scalability.

Of course, there exist deficiencies in our database system. Potential areas of improvement include:

- Incorporate a security system to protect user data and prevent unauthorized access. It would be crucial to ensure that the database is secure and prophylactic measures should be taken into account.
- Performance optimization techniques should be considered. As the database grows, it may become slower or more challenging to manage. It would be useful to consider implementing optimization techniques to improve the website's performance.
- Design a feedback/rating system. Conducting user testing and gathering feedback could help identify areas for improvement and enhance the website's overall user experience.
- Collect more realistic data. Most of the data used to demonstrate is coined. Incorporating actual data can help increase the value and relevance of the database, enabling it to provide more accurate and practical insights for users.

6 REFERENCE AND APPENDICES

The following references are for the implementation of the website part:
https://cn.vuejs.org/
https://expressjs.com/

Here you can check the source code and all related materials from the Github reqository with the link:
https://github.com/AlannnXu/CSC3170_final
Also,you can click the following button to go to the website.
Click Here

————————————————End of Report————————————————