# COMP 8045
# Practicum Report 9 Credits

BCIT Bachelor of Technology Program

**Student Name: Rui Ni**
**Student Number: A00679085**
**Date: March 22, 2015**

## Table of Contents

1. Introduction ..... 5
1.1 Project Problems ..... 6
1.1.1 Business and Task Flow ..... 6
1.1.2 Access Control ..... 6
1.1.3 Flexibility and Scalability ..... 6
1.2 Goals and Objectives ..... 6
1.2.1 Improving Database Performance by Optimizing SQL Statement ..... 6
1.2.2 Applying Data Mining Techniques ..... 6
2. Background ..... 7
2.1 Current Organizational Level ..... 7
2.2 Current Business Flow ..... 8
2.3 Current Business Problems ..... 9
2.3.1 Repeating and Confusing Records ..... 9
2.3.2 Hardly Manageable Financial Data and Slow Reports ..... 9
2.3.3 Inefficient Internal Communication and Inventory Management ..... 9
3. Project Statement ..... 10
4. Requirement Analysis ..... 13
4.1 System Functional Requirement ..... 13
4.1.1 Functional Modules ..... 13
4.1.2 System Actors ..... 17
4.1.3 Detailed Use Case ..... 18
4.2 System Non-Functional Requirement ..... 28
5. Solutions ..... 29
5.1 Access Control ..... 29
5.1.1 Alternative Solutions ..... 29
5.1.2 Chosen Solution ..... 29
5.2 Perform Improvement for Transaction Process ..... 30
5.2.1 Alternative Solutions ..... 30
5.2.2 Chosen Solution ..... 30
5.3 Perform Improvement for Analysis Process ..... 30
5.3.1 Alternative Solutions ..... 30

5.3.2 Chosen Solutions ..... 31
6. System Architecture Design ..... 32
6.1 Design Objectives ..... 32
6.2 MVC Framework ..... 32
6.3 System Environment ..... 33
6.4 Network Structure ..... 34
6.5 System Structure ..... 35
7. System Detail Design ..... 36
7.1 System Sequence Diagram ..... 36
7.2 Class Diagram ..... 38
7.3 Database Design ..... 50
7.3.1 RBAC System ..... 50
7.3.2 Human Resource Management ..... 55
7.3.3 Customer Relationship Management ..... 59
7.3.4 Supply Chain Management ..... 63
7.3.5 Financial Accounting ..... 69
8. Development and Implementation ..... 72
8.1 General ..... 72
8.2 Login and Logout ..... 72
8.3 Authorization Validation ..... 73
8.4 Add, Modify, Query, Delete ..... 73
8.5 Multiple Row Insert ..... 77
8.6 Reports and Charts ..... 79
9. Test Cases ..... 80
10. Research ..... 120
10.1 Optimizing SQL Statement ..... 120
10.1.1 Use of Indexes ..... 120
10.1.2 Other Optimizations ..... 127
10.2 Data Mining ..... 130
10.2.1 Introduction ..... 130
10.2.2 Research Problem ..... 130
10.2.3 Methodology ..... 130

10.2.4 Analysis ..... 133
10.2.5 Conclusion ..... 133
11. Conclusion ..... 134
Reference ..... 135
Change Log ..... 136

# 1. Introduction

The subject of the project is to provide a web-based database application with several functions related to following aspects: human resource management, customer relationship management, supply chain management, financial accounting and management accounting. The system is used to manage information of employees, customers, products, sales and stock.

Green Gong Mao Ltd. as the sponsor of the project is a fast growing company, which was founded in 1998 in China. The company is a distributor of different brands of beverages, mostly alcoholic beverages. Their business covers over five provinces in China, and there are 10 branches and over 500 hundred employees in this company. The customers of Green Gong Mao Ltd. are mostly dealers including convenience stores, restaurants, hotels and supermarkets.

This project is a MySQL database with PHP Web-based application for the company to improve the efficiency and convenience of the company's management. The system can be considered as a small version of Enterprise Resource Planning (ERP) system because the required functions are similar but less than the functions in a complete ERP system. This system is expected to include following functional areas and functions,

Human resources management

- Recruiting, payrolls, benefits, employee information management

Financial accounting

- General ledger, payables, receivables

Management accounting

- Costing

Customer relationship management

- Sales, customer information

Supply chain management

- Purchasing, inventory, transshipment

## 1.1 Project Problems

### 1.1.1 Business and Task Flow

In order to design and develop a suitable data model, the priority is to understand how the company operates in general and how each task is completed in detail. Therefore, the first problem need to be solved is how to interpret business flow and each task flow in order to design appropriate data model for the system.

### 1.1.2 Access Control

Without proper access control of database, some sensitive information such as customer and sales information can be accessed by unauthorized users. In order to prevent that, a suitable access control method need to be chosen and modified for the system according to the company's requirement.

### 1.1.3 Flexibility and Scalability

As mentioned above, Green Gong Mao Ltd. is a fast growing company, and they have already planned to change business method in the near future such as adding online ordering system and opening a new branch. The fact is a small change in business may cause a large change in data model and interface as well. Therefore, flexibility and scalability should be considered in data modeling instead of redeveloping a whole system.

## 1.2 Goals and Objectives

The objectives of the project are not only to deliver a system that meets the company's requirement, but also to provide what I have learned or anything may help me to improve the system.

### 1.2.1 Improving Database Performance by Optimizing SQL Statement

Database is the fundamental component of the whole system. Improving the performance of database will positively effect on the efficiency of users' tasks. In "Effective MySQL: Optimizing SQL Statement", Ronald Bradford (2011)1 suggests some methods such as use of index to optimize SQL statement to improve database performance. This will be an experiment included in this report, and useful findings will be applied in my own project to help me with database and query design.

### 1.2.2 Applying Data Mining Techniques

Data Mining can be a key process that discover and consistently use profitable information from data. It can be useful for decision making and predicting the effects of decisions. For instance, data mining can help companies to target their customers by clustering customers into groups based on customers' characteristics and demands, so that companies can make better marketing strategies based on information discovered.

---

[1] Ronald Bradford. (2011) *Effective MySQL: Optimizing SQL Statements*. The United States. MacGraw-Hill Companies, Inc.

Classification as data mining technique will be researched and applied in the data mining application.

## 2. Background

The practicum is to fundamentally help the company to solve the problems that occurred in various management processes including human resource management, customer relationship management, supply chain management, financial accounting, and management accounting. For now, the company uses Excel to keep all the records of their sales, inventory, employees' information and customers' information. The excel files are mostly sent by emails between branches which is insecure. For transshipment, branch managers can only check other branches' product storage by calling each branch manager which is inefficiency and inconvenient to response to customers' orders. The managers have realized the inefficiency, inconvenience and insecurity of their business process, so they have decided to change the current condition by using a customized web based application.

Currently, Green Gong Mao Ltd. has a commercial website under a rental apache server. The company has planned to have its own server in this year. This practicum as a part of their plan will be deployed under its own server. The application will be able for only authorized users to use through a browser-based interface. In the future, the company may extend the features of this application such as online ordering for customers to use.

### 2.1 Current Organizational Level

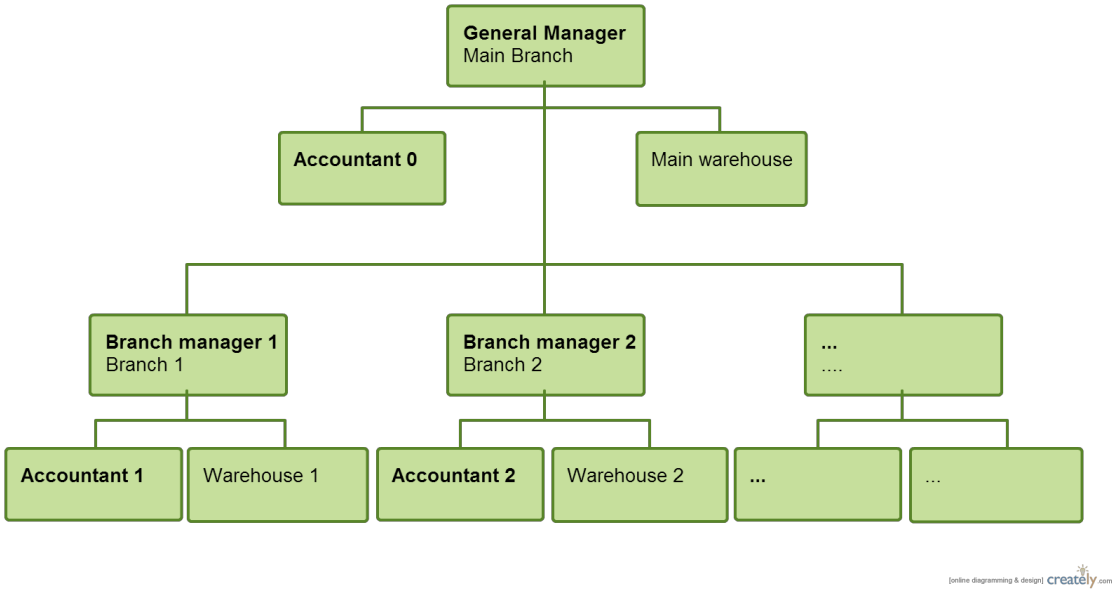



Figure 1 Current Organizational Level

It is necessary to know the inner structure and hierarchy of the company in order to understand how the company operates. The figure above illustrates the organizational level of the company and relationships among the roles: general manager, branch manager and accountant.

## 2.2 Current Business Flow

The following diagram represents the company's current business flow.

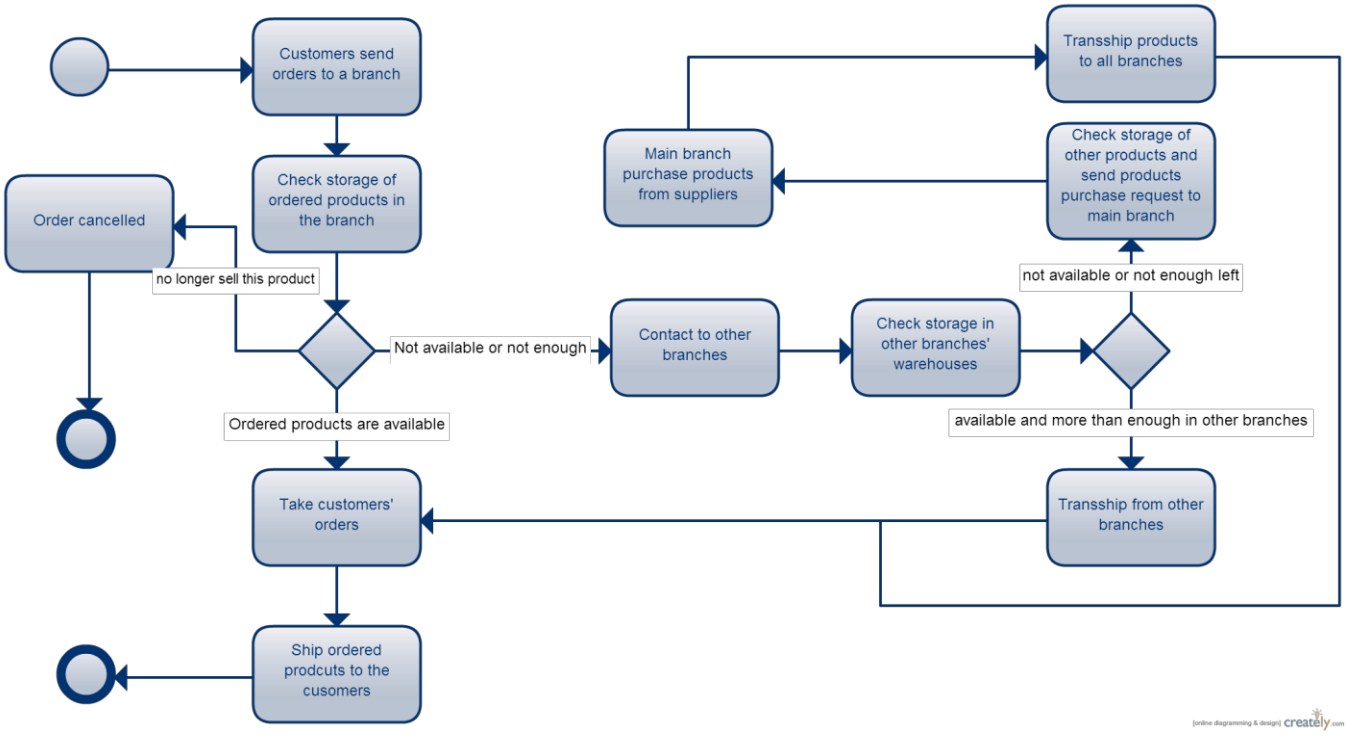



Figure 2 Current Business Flow

As the diagram above shows, the company's current business flow involves much communication among branches, between branches and customers, and between main branch and suppliers. The communication among branches is restricted at the moment. Sometimes, customers expect to get what they order in a short time. This makes transshipment so important especially under some situations such as temporary shortages of products. The reasons that to transship products from other branches instead of purchasing from suppliers are

1. Some suppliers may be far away from where branches located.
2. Small amount of products purchase increase the cost of products.
3. It increases uncertainty that ordered products may not be delivered to customers in time, which leads to negative effect on company's reputation, reparation for delay and even loss of customers.

## 2.3 Current Business Problems

### 2.3.1 Repeating and Confusing Records

According to information provided by the company, child branches are equal in all aspects. Each branch has its own warehouse. Main branch's warehouse is like a distribution centre to other branches' warehouses, because this is the first station of all products delivered from suppliers. All branches including main branch take orders from customers and deliver products. Only main branch has all customers' information, and child branches' lists of customers are separated from each other.

### 2.3.2 Hardly Manageable Financial Data and Slow Reports

For accounting, each branch has at least one accountant, and they send all records and reports about sales, salaries and other expenses with Excel files to main branch monthly by email. Even though they have certain formats for different records to keep the records from different branches consistent, it is still hard for accountants in main branch to bring different records together and analyze for different purposes. In addition, sending records including sensitive information such as customers' information by emails are highly insecure. Most important, management groups cannot make changes of product purchase, marketing strategy and human resource management in time based on monthly data and reports.

### 2.3.3 Inefficient Internal Communication and Inventory Management

However, the restriction of communication among branches causes the speed of process slower than it is expected to be. Managers cannot make quick decisions because they cannot check the product storage in other branches whenever they want directly instead of talking to other managers one by one on the phone. This makes storage management and transshipment extreme difficult.

## 3. Project Statement

Project Name: Information Management System Development
Project Sponsor: Green Gong Mao Ltd.
Project Start Date: 2015-07-20
Project Description: The purpose of the project is to develop a MySQL – PHP based web application system for human resource management, customer relationship management, supply chain management, financial accounting, and management accounting.

Project Scope: This project will include design and development for both MySQL database and web user interface.

Project Objectives:

This project should include following functions under different system users,

| **Branch Manager** | | |
|---|---|---|
| Required Function | Details | Involved Functions in Business |
| Manage employee profiles | Branch managers should be able to add new employee into their own branch, to view all employee profiles in the company, to edit and delete information of employees under their own branches | Recruiting, employee information management |
| Manage order records | Add order after taking customers' orders<br>Change information such as payment status and order status during sales process<br>Delete information after orders are cancelled<br>Branch managers can access only records belongs to their own branches | Sales, Customer Contact |
| Generate sales records | After orders are completed, branch managers can transfer completed order records to sales records by changing order status | Sales |
| View sales records | Branch managers can access only records belongs to their own branches | Sales |
| Manage delivery records | This is a function to monitor the process of delivering and delivered product information | Sales, Inventory |
| Manage product storage | Product storage information in all warehouses is shared. | Inventory |

| | | |
|---|---|---|
| Manage transshipment | Transshipment is like delivery. Its purpose is to monitor the process and save records. However, it involves two branches, and one branch manager controls the beginning and one branch manager controls the end of the process, “Deliver” -> “on the way” ->”receive”. | Inventory, Transshipment |
| Manage customer information | Customer information includes attributes such as contact information, bank information, category, and region. Branch managers can access only records belongs to their own branches. | Customer contact |
| Edit branch and warehouse information | Branch managers should be able to edit contact information of their own branches and warehouses. | |
| *Manage as function name under the role actually consists of four functions: add, view, edit and delete information.<br>*Administrator (General manager) as a role is also considered as a branch manager in main branch; however, general manager has full access to all information in the company which is different from other branch managers. | | |

| **Accountant** | | |
|---|---|---|
| Required Function | Details | Involved Functions in Business |
| Manage salaries and bonus | There are other attributes besides salaries and bonus in this function such as deduction, insurance and income tax. There are many type of bonus, especially sales bonus. Sales bonus is incentive payment which tied to defined targets of sales. | Payrolls, Benefit |
| Maintain records | Accountants can keep these records to be complete and up to date. | Payrolls, Benefit, Costing, Receivables, General ledger |
| Manage other expense | Accountants are able to keep records about basic expenses such as electric bill. | Costing |
| View employee profiles | Basically, this function is to allow accountants to check employees’ basic salary and other necessary data to manage payrolls. | Payrolls |
| View sales records | Salesperson’s information | Receivables, General |

| | | |
|---|---|---|
| | as an attribute is included for sale bonus records. | ledger, bonus |
| View purchase records | This function is to allow accountants to calculate costing and profits. | Costing, General ledger |
| Summarize and analyze data | Views and algorithms will be considered and designed to help accountants with data analysis | Payrolls, Benefit, Costing, Receivables, General ledger |
| Generate reports | Reports are generated based on the results of data summary and analysis. Reporting tools will be considered and applied to create direct reports with visual charts. | |
| *Accountants in main branch are allowed to access data other accountants under all branches. The other accountants can only access data under their own branches. | | |

| **Administrator( General Manager)** | | |
|---|---|---|
| Required Function | Details | Involved Functions in Business |
| Manage purchase records | General Manager is able to create purchase records after purchasing products from suppliers. | Purchasing, Costing |
| Manage product information | Product information includes many attributes such as product name, description, size type, category and price. | Purchasing, Inventory, Sales |
| Manage supplier information | Differ in customer information, supplier information contains supplied product information | Purchasing |
| Manage branch and warehouse information | Admin is able to add and delete branches and warehouses, and edit information of branches and warehouses. Admin is also able to manage the relationship between a branch and a warehouse. | |
| *This table only contains functions only Administrator (General manager) has. | | |

# 4. Requirement Analysis

## 4.1 System Functional Requirement

### 4.1.1 Functional Modules

There are five functional areas are required in the system including human resource management, customer relationship management, supply chain management, financial accounting, and management accounting. Each functional area is considered as a functional system while designing the system. Each functional system consists of several related functional modules.

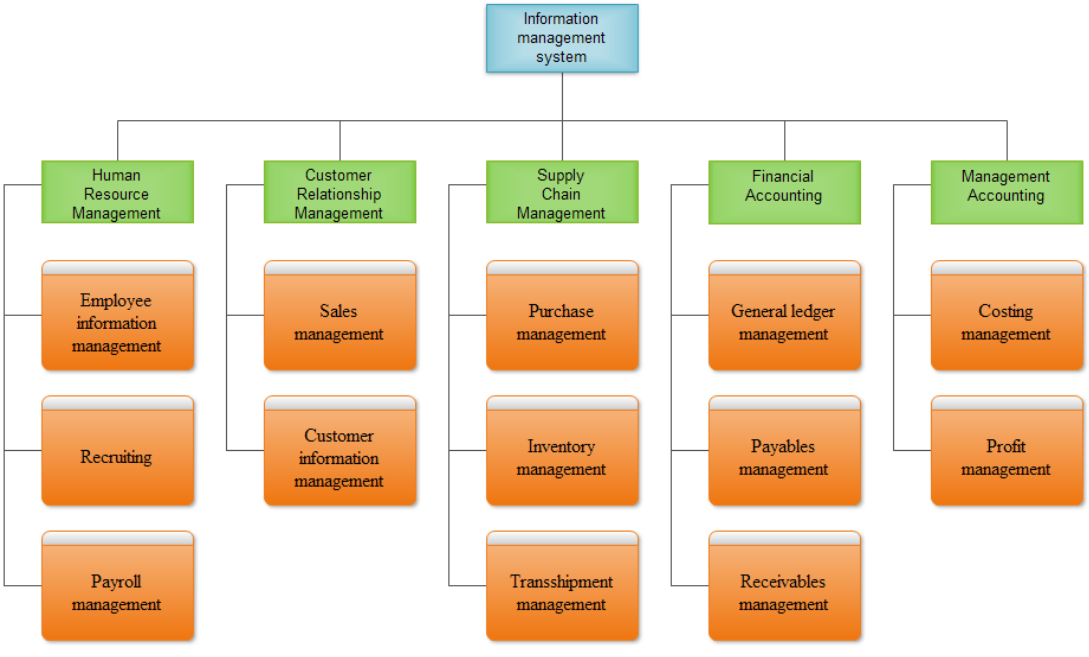



Figure 3 Functional modules structure

Package diagrams are used to categorize and introduce different functional modules.

1) Human Resource Management

From functional point of view, the human resource management system consists of three functional modules which are employee information management, recruiting, and payroll management, as the package diagram shows below.

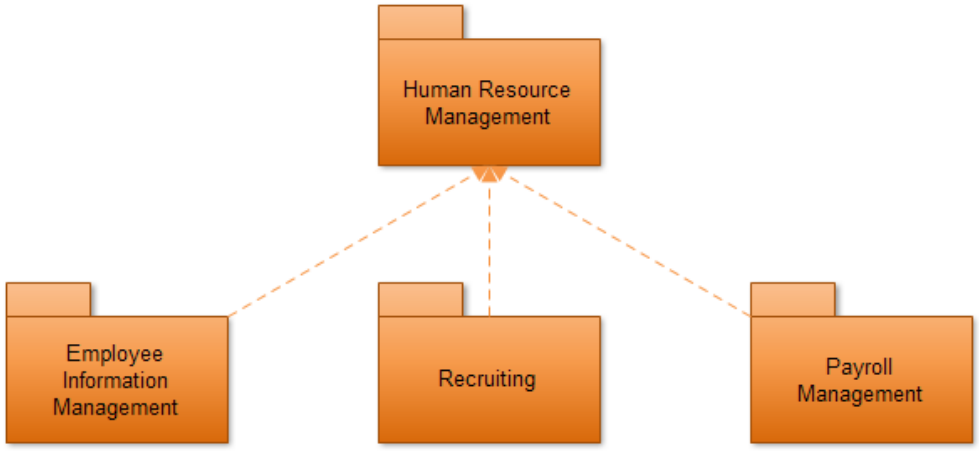



Figure 4 Package Diagram - Human Resource Management

2) Customer Relationship Management

The customer relationship management system consists of two functional modules which are sales management and customer information management, as the package diagram shows below.

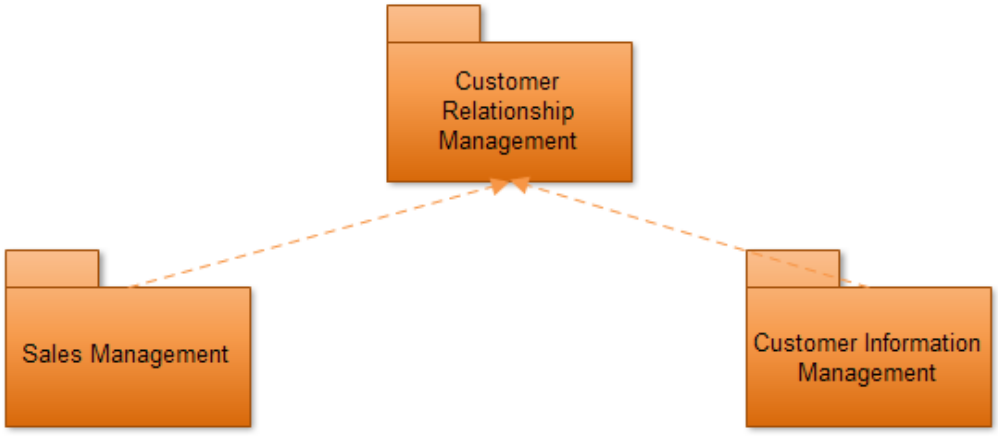



Figure 5 Package Diagram - Customer Relationship Management

3) Supply Chain Management

The supply chain management system consists of three functional modules which are purchase management, inventory management and transshipment management, as the package diagram shows below.

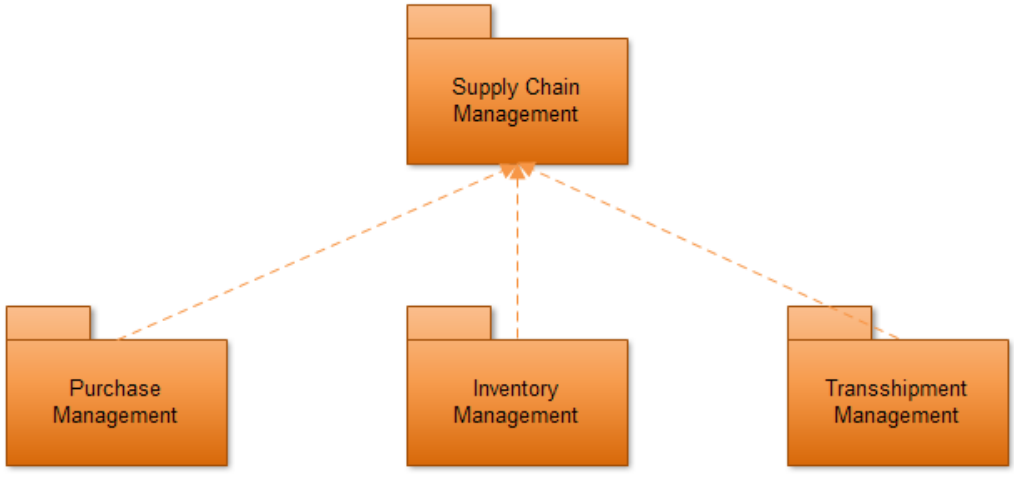



Figure 6 Package Diagram - Supply Chain Management

4) Financial Accounting

The financial accounting system consists of three functional modules which are general ledger management, payables management and receivables management, as the package diagram shows below.

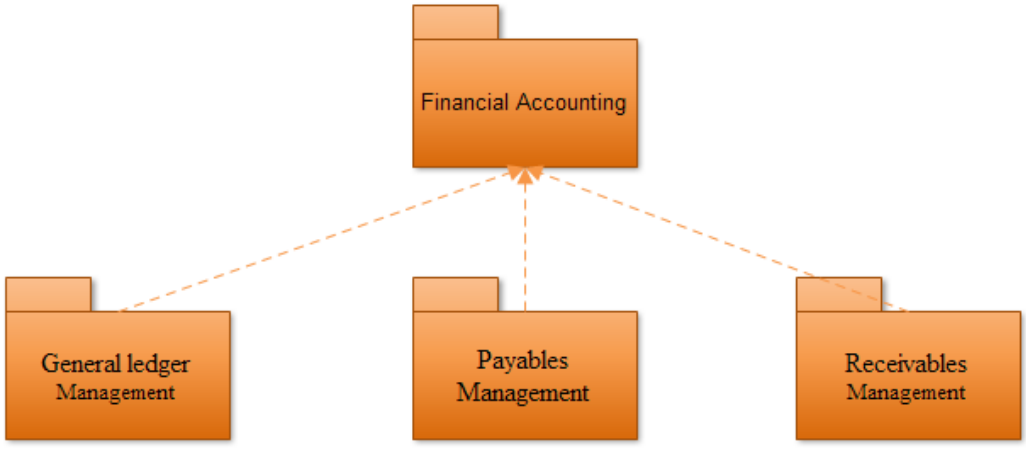



Figure 7 Package Diagram - Financial Accounting

5) Management Accounting

The management accounting system consists of two functional modules which are costing management and profit management, as the package diagram shows below.

Figure 8 Package Diagram - Management Accounting

### 4.1.2 System Actors

System actors as figure 8 shows

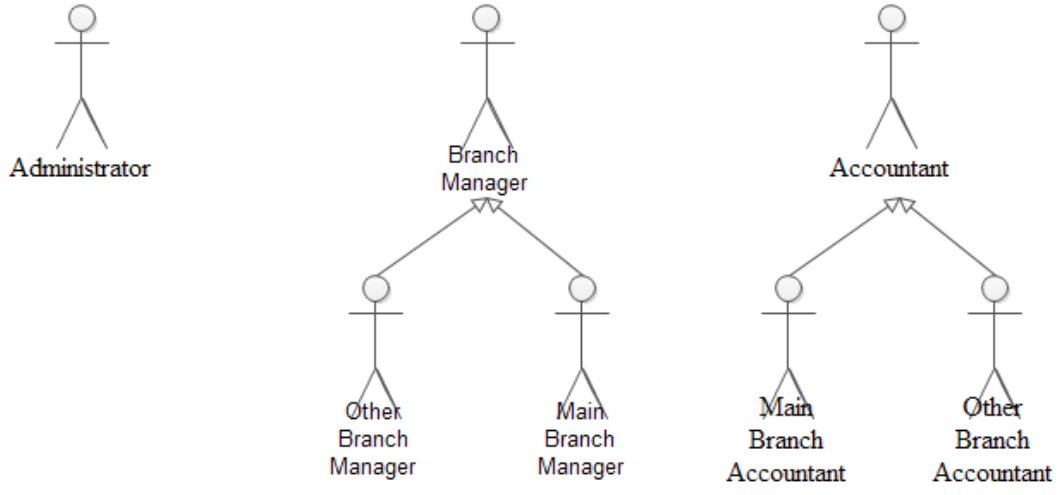



Figure 9 System Actors

As the company required, the system users are generally divided into administrators, branch managers and accountants. According to different permissions, branch managers are divided into main branch manager and other branch managers, and accountants are divided into main branch accountant and other branch accountants.

The differences between main branch manager and other branch managers are,

- Other branch managers cannot access purchase management in supply chain management
- Other branch managers cannot only access the functional modules to manage information belong to their own branches.
- Main branch manager has full access to all the functional modules above and information belong to all branches.

The differences between main branch accountant and other branch accountants are,

- Other branch accountants can only access the functional modules to manage information belong to their own branches.

Administrator is responsible for defining users, assigning privileges, and managing basic information of settings for the system. Branch managers are in charge of human resource management, customer relationship management, supply chain management, financial accounting, and management accounting. Accountants are only responsible for payroll management from human resource management and financial management in general.

### 4.1.3 Detailed Use Case

The use case of each functional module is explained separately in this section.

1) Human resource management
    a. Employee information management

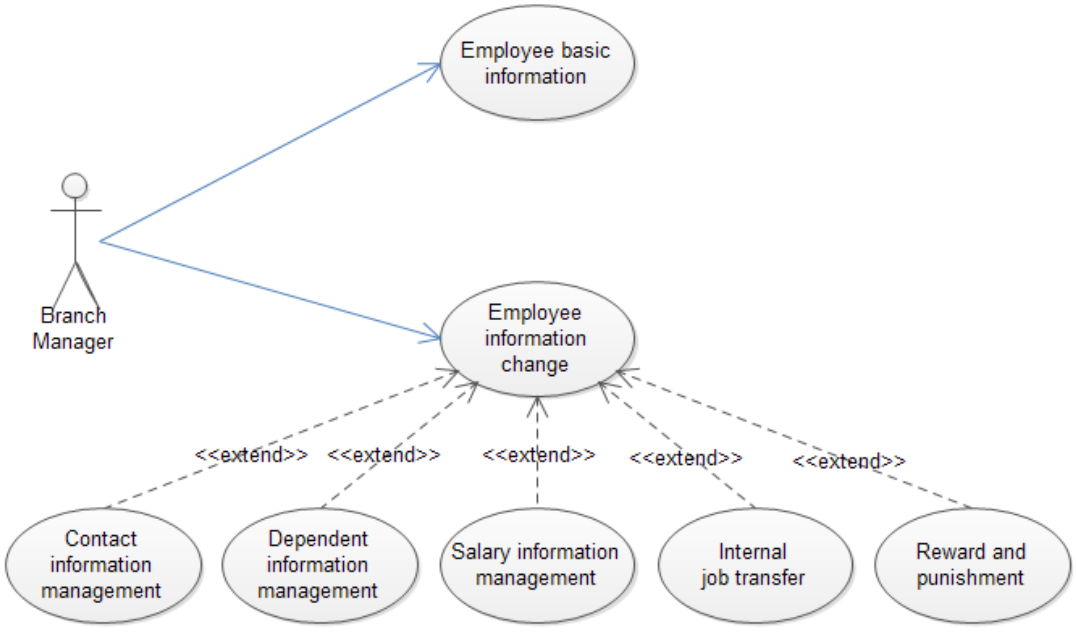



Figure 10 Use Case - Employee information management

Employee information management is one important module in human resource management. This module processes two main parts of information which are employee basic information and employee information changes. The changes include contact information, dependent information, salary information, internal job transfer, and reward and punishment information.

The only system actor here is branch manager. She/he is responsible for adding, deleting and modifying. In addition, there are connections between this module and confirm recruiting in recruiting module, and branch managers are only able to access employee information belong to their own branches except main branch manager.

b. Recruiting

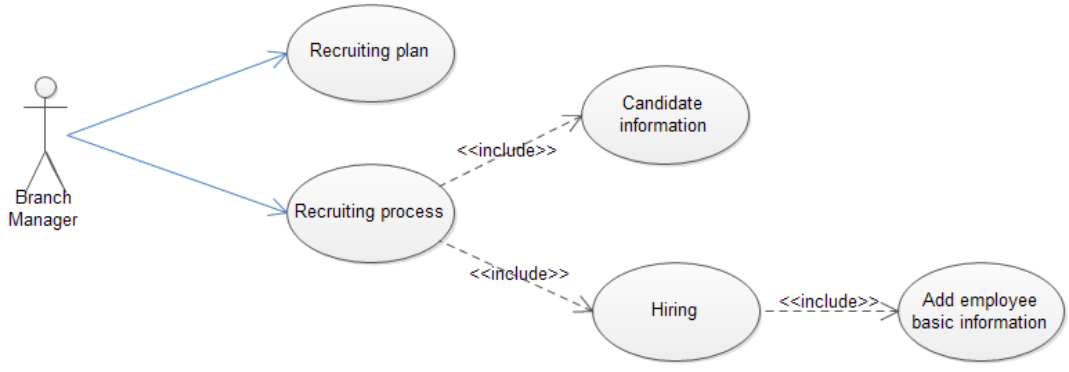



Figure 11 Use Case - Recruiting

In this module, there are two main process which are recruiting plan and recruiting process. Branch managers are able to add, delete and modify recruiting plan. During recruiting process, first, branch managers are able to add candidate information. Second, based on requirement, branch managers decide which candidates should be hired. Then candidate information will be modified and added into employee basic information.

c. Payroll management

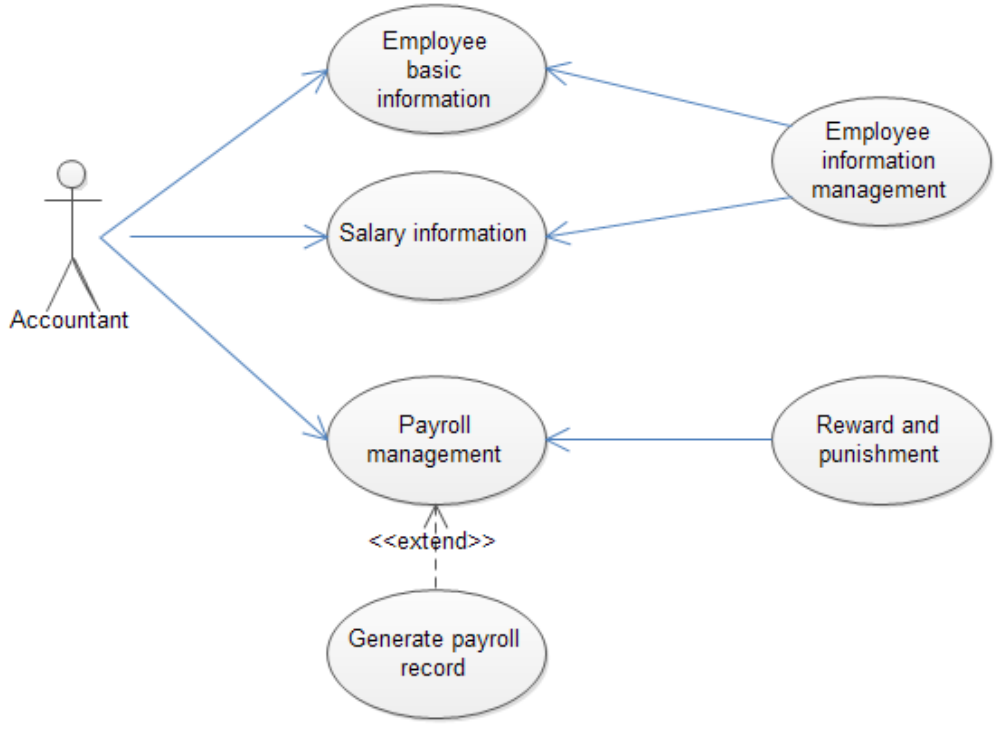



Figure 12 Use Case - Payroll management

Payroll management is also an important part in human resource management system. The main system actor in this module is accountant. In payroll management, employee list is from employee information management. Accountants generate monthly payroll records based on salary information and monthly reward and punishment records which are also from employee information management module.

2) Customer relationship management
    a. Customer information management

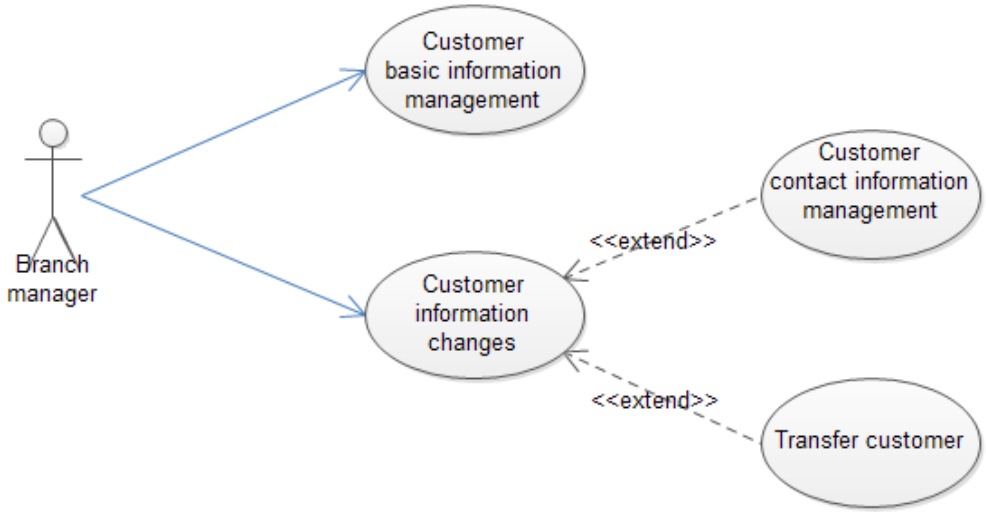



Figure 13 Use Case - Customer information management

Customer information management is the base of customer relationship management. In this module, branch managers are able to add, delete and basic information, bank information and contacts of customers. Each branch manager is also able to transfer their customers to other branches. In addition, branch managers are only able to access customer information belong to their own branches except main branch manager.

b. Sales management

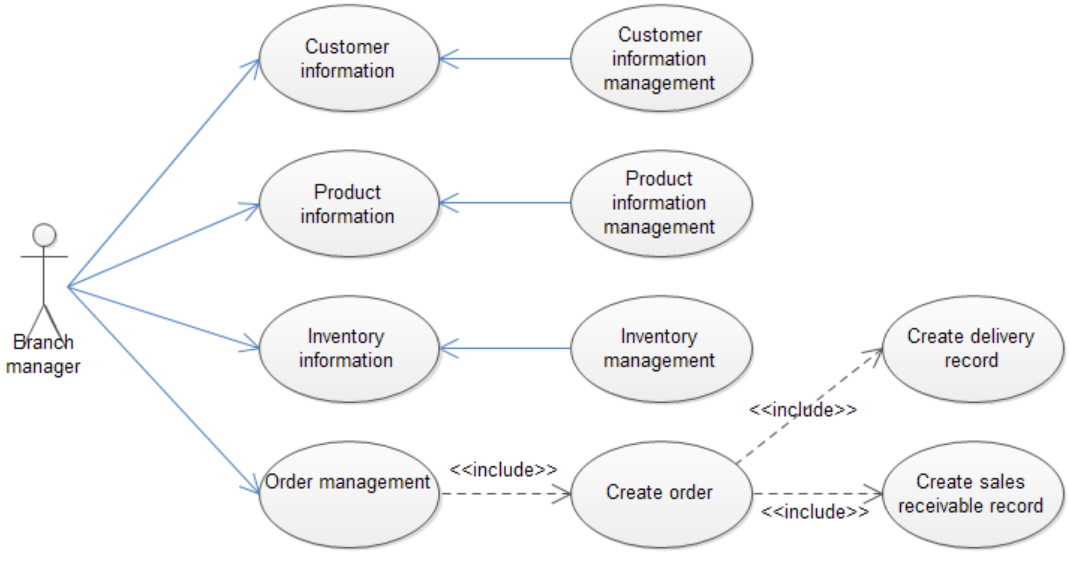



Figure 14 Use Case - Sales management

Sales management is the most important part of customer relationship management. There are several connections between sales management module and other modules.

- Customer information from customer information management.
- Products information from product information management.
- Inventory information from inventory management.
- Delivery records are generated for inventory management.
- Sales receivable records are generated for receivables management.

Branch manager as the main actor in this module, she/he is able to add order records created by customers. After checking inventory, branch managers can generate delivery records for inventory management and sales receivables records for receivable management.

3) Supply chain management
    a. Purchase management

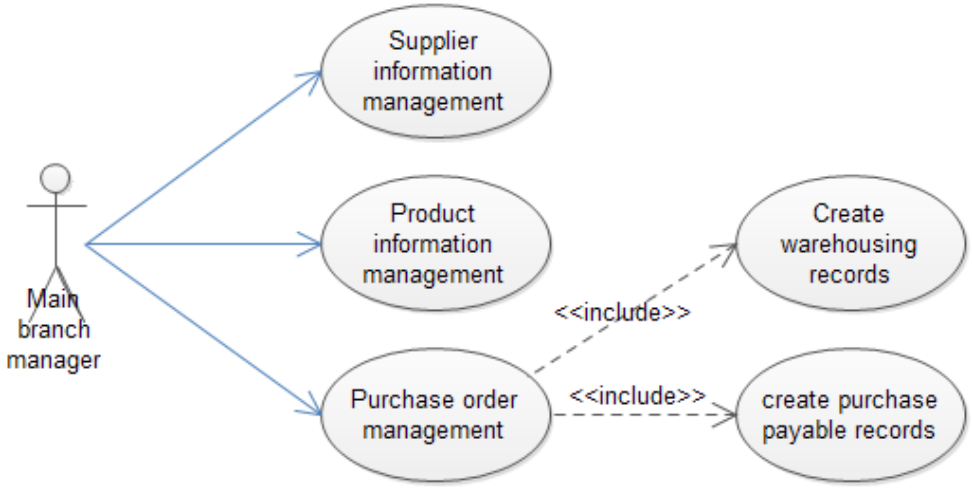



Figure 15 Use Case - Purchase management

Only main branch manager is responsible for purchase management. In this module, main branch manager is able to add, delete and modify supplier information and product information. She/he is also able to add purchase orders. In the meantime, new warehousing records are created and waiting to be confirmed, and new purchase payable records are also created0 for payable management.

b. Inventory management

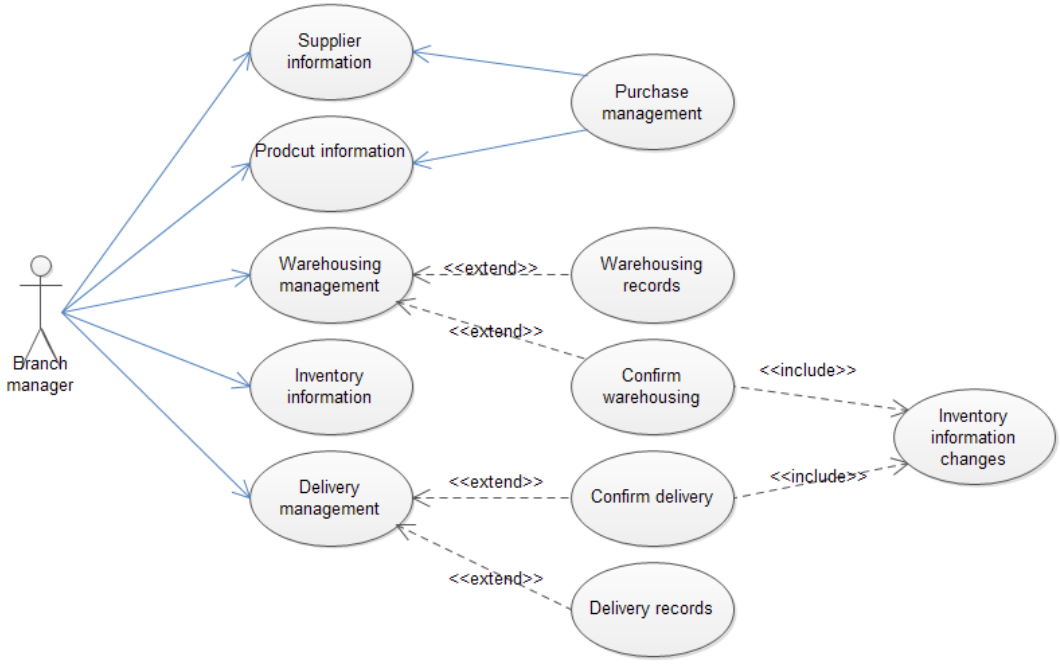



Figure 16 Use Case - Inventory management

Inventory management is the core of supply chain management system. Branch manager as the only system actor here is able to access supplier information and product information from purchase management. The two main functions in inventory management system are warehousing management and delivery management.

For warehousing management, branch managers are able to access warehousing records that generated from purchase management. After branch managers confirm that purchased products from supplier have been delivered into warehouse, inventory information of these products will be added or updated automatically.

For delivery management, branch managers are able to access delivery records that generated from sales management. After branch managers confirm that products ordered by customers have been shipped out from warehouse, inventory information of these products will be updated automatically.

Unlike customer information and employee information, branch managers are able to access inventory information from all branches in order to manage transshipment.

c. Transshipment management

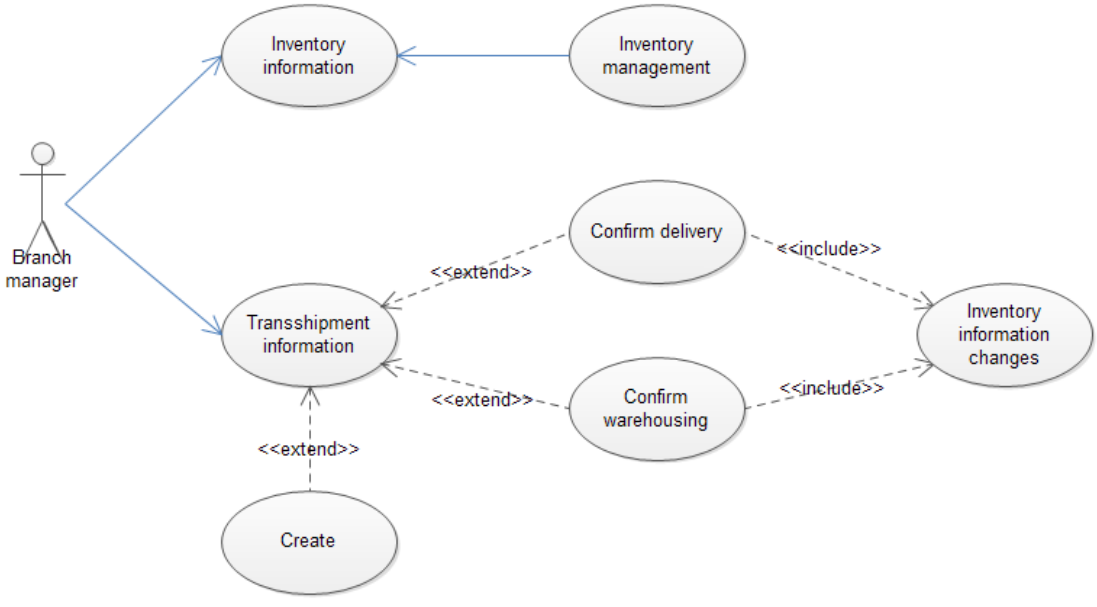



Figure 17 Use Case - Transshipment management

Transshipment management is used as an internal process for transshipping products from a warehouse under one branch to another. Branch managers are able to access inventory information not only from their own warehouse but also from warehouses under other branches.

In order to complete this process, first, a branch manager needs to create a transshipment record with which branch products transshipped from and which branch to. The branch manager who creates the record can be either sender or receiver. Second, after the sender, a branch manager, confirms that products have been shipped out from the warehouse under her/his branch, inventory information of these products under sender's branch and the status of this transshipment record will be updated automatically. Last, after the receiver, a branch manager, confirms that products have been delivered, inventory information of these products under receiver's branch and the status of this transshipment record will be added or updated automatically.

In addition, branch managers are able to access transshipment records when they are either sender or receiver except main branch manager.

4) Financial accounting
    a. General ledger management

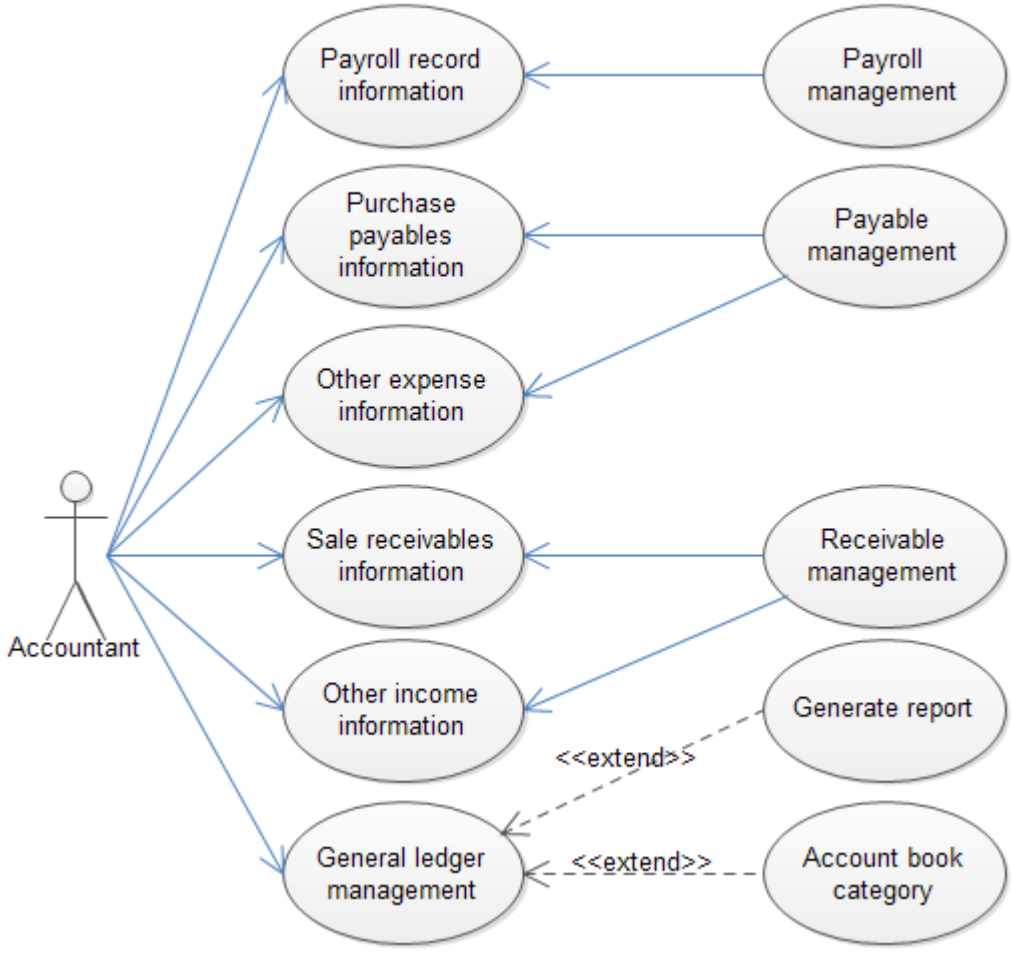



Figure 18 Use Case - General ledger management

The main system actor in this module is accountant. Accountants are able to access expense information include payroll record from payroll management in human resource management system and purchase payables and other expense information from payable management in financial accounting system. Based on the chosen account book categories, accountants are also able to gather different information mentioned above and generate reports.

In addition, accountants are only able to access information belong to their own branches except main branch accountants.

b. Payables management

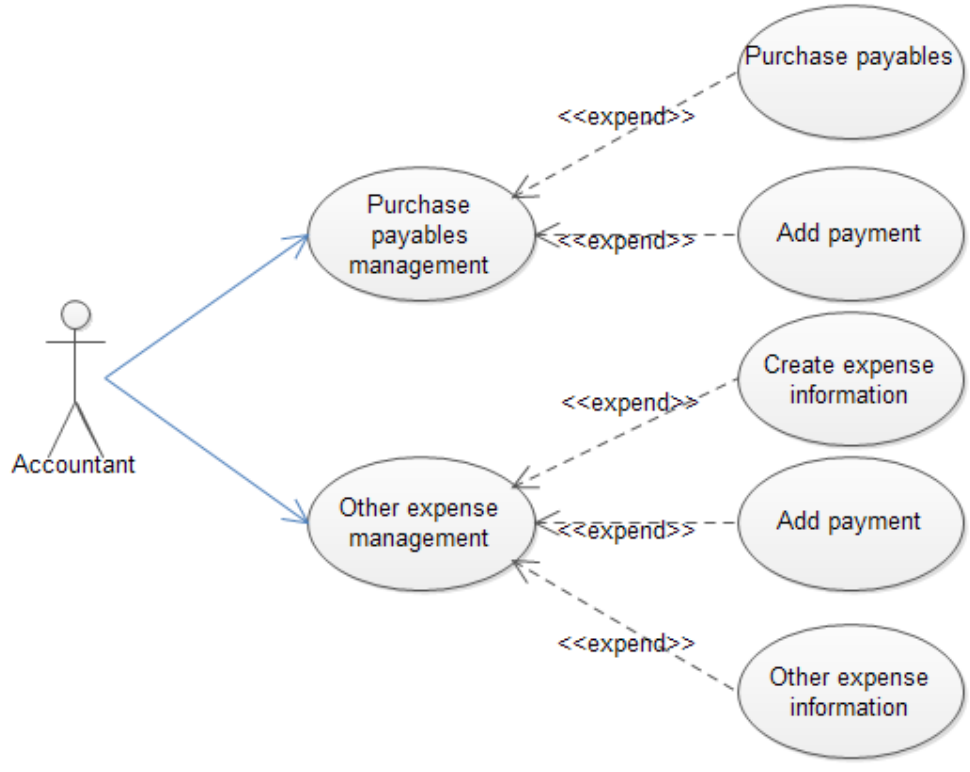



Figure 19 Use Case - Payables management

Purchase payables management and other expense management are the two main parts in this module. For purchase payable management, accountants are able to add several payments till all the debts are paid in order to complete a purchase payable record. The difference between purchase payables management and other expense management is that other expense records are created by accountants, and purchase payable records are generated from purchase management in supply chain management system.

c. Receivables management

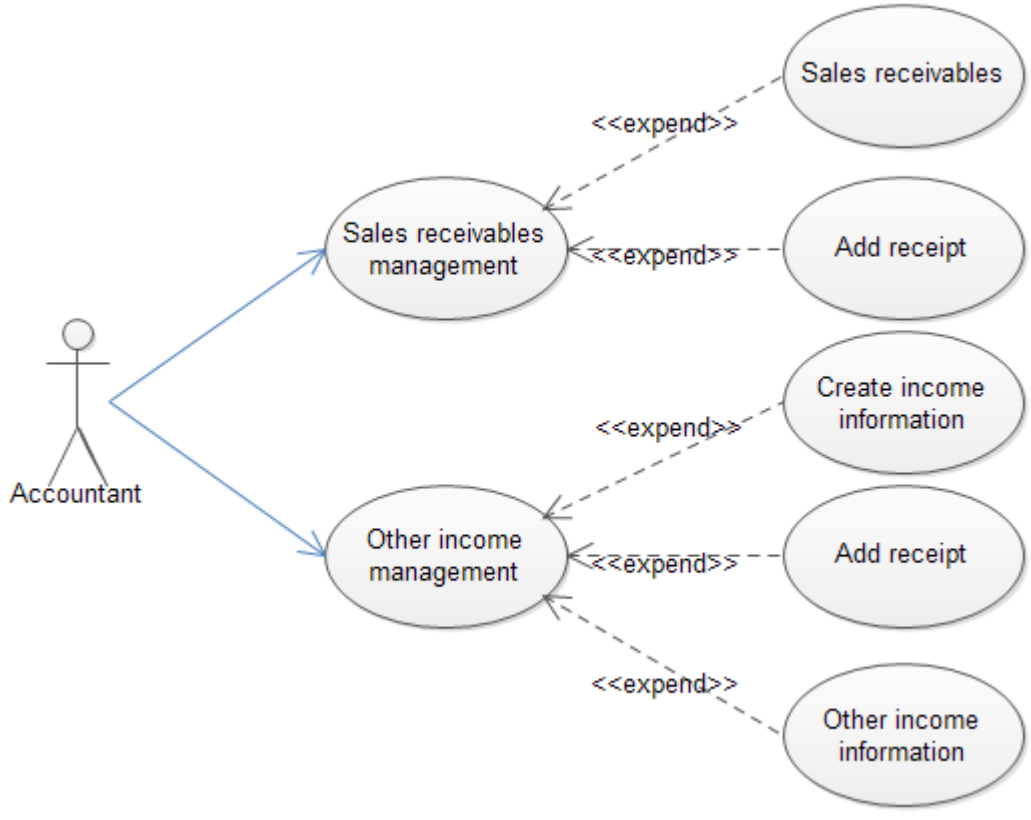



Figure 20 Use Case - Receivables management

Sales receivables management and other income management are the two main parts in this module. For sales receivables management, accountants are able to add several receipts till all the debts are paid in order to complete a sales receivable record. The difference between sales receivables management and other income management is that other income records are created by accountants, and sales receivables records are generated from sales management in customer relationship management system.

In addition, accountants are only able to access information belong to their own branches except main branch accountants.

5) Management accounting

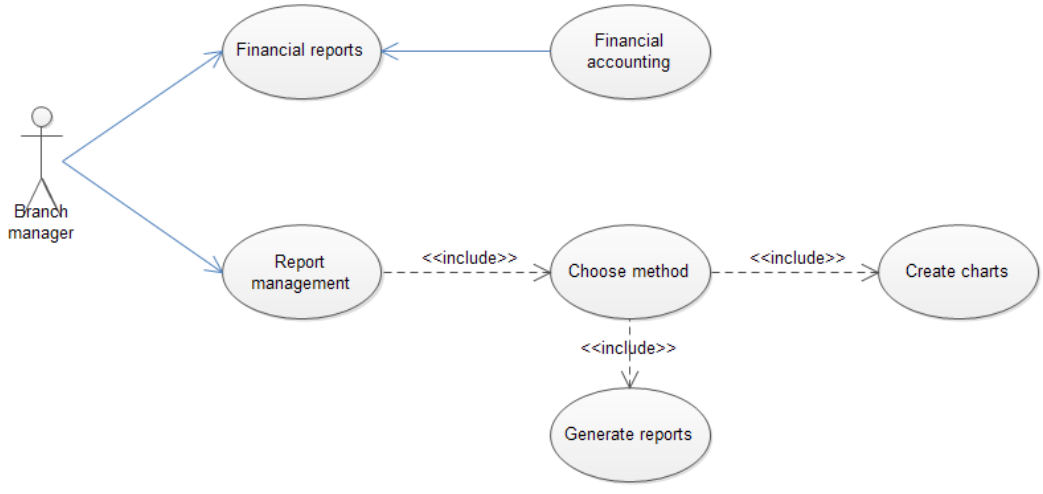



Figure 21 Use Case - Management accounting

The two modules in management accounting system are similar in use cases. The only difference is that only main branch manager is able to access profit management. Therefore, the use case is explained as whole.

In the management accounting system, branch managers are able to access any financial reports belong to their own branches except main branch manager. Different management accounting reports depend on the chosen method. In order to support for specific decision making, a method will be chosen first to collect statistical data. Based on statistical data, branch managers are able to generate data report and create charts to help them in decision making process.

## 4.2 System Non-Functional Requirement

The content of non–functional requirements is normally represented by qualitative indicators. Scope descriptions of qualitative indicators are helpful for measuring certain characteristics of the system.

The non-functional requirement of the system as table 1 shows below

| Qualitative indicators | Description of non-functional requirement |
|---|---|
| Availability | The system should be easy to use. System users should be able to use the system with simple training or no training at all. |
| Reliability | The system should be able to back up and restore data remotely in case of system or server failures. |
| Scalability | The expansion of users' tasks or functional modules should not affect the use of system. |

Table 1 System non-functional requirement

# 5. Solutions

## 5.1 Access Control

### 5.1.1 Alternative Solutions

Discretionary Access Control (DAC)

"Discretionary Access Control is a type of access control defined by the Trusted Computer System Evaluation Criteria as a means of restricting access to objects based on the identity of subjects and/or groups to which they belong."[2]

In database, the objects are schemas, tables and views. The types of discretionary privileges are subjects including

- Create schema, create table
- Create view
- Alter schema and attributes in tables
- Modify: insert, delete, and update tuples
- Select: retrieve (read) data by using select query

DAC requires permissions to be assigned to users who need access, and it is determined by the user with permission.

Mandatory Access Control (MAC)

"Mandatory access control is a type of access control by which the operating system constrains the ability of a subject or initiator to access or generally perform some sort of operation on an object or target."[3]

In database, access control is managed by the administrator. The usage and access policy are defined by the administrator to indicate which user has access to which schema, table and view. The access policy cannot be modified by users.

In contrast with DAC, MAC is determined by the administrator.

### 5.1.2 Chosen Solution

Role-based Access Control (RBAC)

Role-based access control system is applied to restrict system access to authorized users.

The advantages of RBAC system are

- RBAC can implement both DAC and MAC. It makes RBAC more flexible and scalable than the other two types of access control.

[2] Wikipedia http://en.wikipedia.org/wiki/Discretionary_access_control
[3] Wikipedia http://en.wikipedia.org/wiki/Mandatory_access_control

- One-time set up of a role and assign new users to a role instead of repeating assignment can decrease maintenance cost and increase efficiency
- Less skill required for users to modify.
- RBAC is one of the most flexible access control system due to that users can be assigned to different roles or role groups to gain access to the system.

The detailed application will be explained in 7.3.1 RBAC System on page 50.

## 5.2 Perform Improvement for Transaction Process

### 5.2.1 Alternative Solutions

Normalization in Database Design

Normalization is a process of organizing data in database. The process includes creating tables and establishing the relationships among the tables according to the designed rules. It makes database more flexible by eliminating redundancy and inconsistent dependency.

This process has been applied in database design of this project. However, the process could be more specific if it is chosen.

Reviewing Parameter Settings

Reviewing parameter settings of database can also be a solution to improve database performance. The parameter settings include buffer caches, memory locations, logs, undo, etc.

### 5.2.2 Chosen Solution

Writing SQL Statement

Optimizing writing SQL statement is part of the research about improving database performance by optimizing SQL statement. Optimizing writing SQL statement can effectively improve the speed of data input.

The detail research about improving database performance by optimizing SQL statement is explained in 10.1 Optimizing SQL Statement on page 120.

## 5.3 Perform Improvement for Analysis Process

### 5.3.1 Alternative Solutions

Creating Views

In database, a view is the result set of a stored query on the data. Creating views based on the frequent queries submitted by users can improve the speed of data output in order to improve database performance. This solution is implemented while designing queries. However, this can be mainly focused on as one of the solutions to improve database performance.

Data Warehouse

A data warehouse is a system used for data analysis and report. Data warehouse is designed for queries and data analysis which is the analytical processing rather than transaction processing. Data warehouse is the central repositories of integrated data from one or more disparate data sources. Appling data warehouse can be one of the solutions to improve analysis process of database performance.

However, data warehouse is not necessary in this project because at the moment the system is mainly used for keeping records rather than analyzing data. In the future, implementing data warehouse can be useful in data analysis if the company wants to apply more data analysis features.

### 5.3.2 Chosen Solutions

Query SQL statement and Use of Index

As a part of optimizing SQL statement researched, query SQL statement and use of index are researched and experimented. Some of the methods are applied in developing the system in order to improve analysis process.

The detail research about improving database performance by optimizing SQL statement is explained in 10.1 Optimizing SQL Statement on page 120.

Data Mining Techniques

Based on the data provided by the company, decision tree as one of the models in classification data mining technique is researched and experimented.

The detailed research report is in 10.2 Data Mining on page 130

# 6. System Architecture Design

## 6.1 Design Objectives

The goal of the system is to fundamentally help the company to solve the problems that occurred in various management processes including human resource management, customer relationship management, supply chain management, financial accounting, and management accounting.

The system mainly focuses on the company's daily management and operations. The information of each task during the processes of management and operation will be converted to collectable data. In the meantime, statistics and reports will be generated in order to support decision making and development processes of the company.

## 6.2 MVC Framework

Model-View-Controller (MVC) framework is used in this project to isolate application logic from the system. MVC framework is very useful for developers to provide information that users can easily understand.

There are three components of MVC framework,

1) Model is to manage the data, rules and logic of the application. A model can be activated either passively or by notification from a controller.
2) View is to represent information including user interfaces and the data user can interact with which is requested from a model or a controller directly.
3) Controller is to receive requests from users, then send commands to Model to prepare data needed by View. It can also send commands and data directly to View.

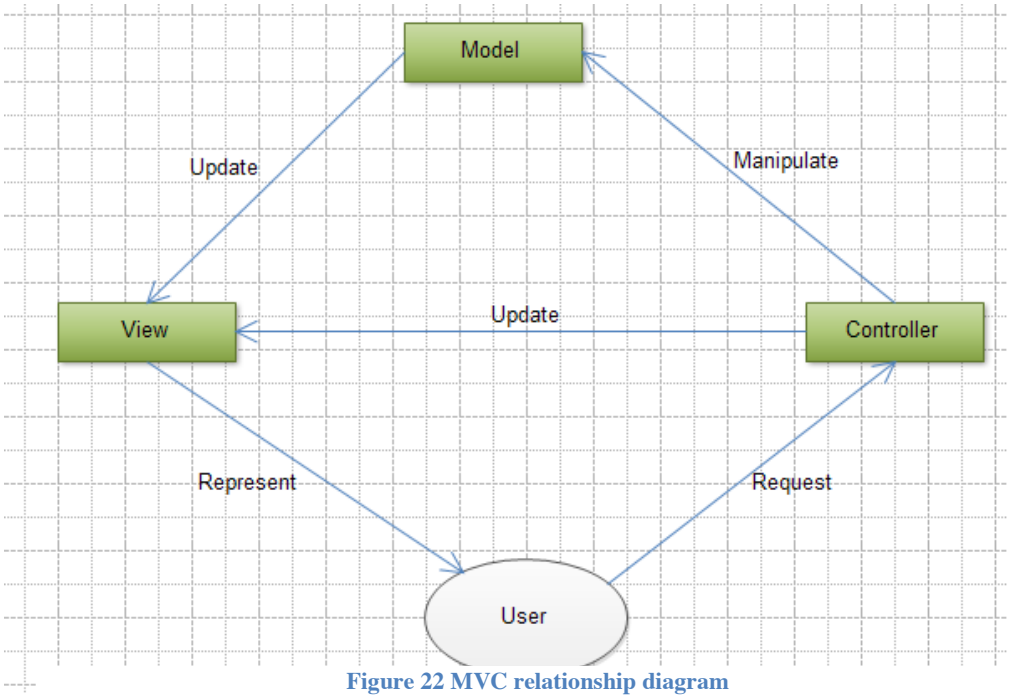



Figure 22 MVC relationship diagram

ThinkPHP is a free, open-source, object-oriented PHP development framework which developed by a Chinese company in Shanghai, follow the Apache 2 open source license issued. ThinkPHP was first published in 2006, and the company is still improving the framework. The version of ThinkPHP framework used in the project is 3.2.3, which is the latest version.

The project was developed by using Zend framework as PHP framework at first cause the sponsor did not require specifically which PHP framework to use. Due to that this system is a part of the company's undergoing project, it would be easier for the company to update framework and further development of the system by using ThinkPHP as PHP development framework for the whole project.

The reasons that ThinkPHP is specifically required are:

1) It is the most popular PHP development framework in China according to the sponsor of the project.
2) There are many third-party plugins meet Chinese market demands can be used for future development.
3) As this project is a part of the company's project, it is easier to implement, update and develop more features in the future by using same framework.

## 6.3 System Environment

**Server:**

Operating System: Windows XP/7/8

It can also be transplanted to UNIX or Linux operating system.

Database: MySQL 5.6.17 or higher

Web Server: Apache 2.4.9 or higher

Server-Side Scripting Language: PHP 5.5.12 or higher

**Client:**

Web browsers with Html5 and CSS3

## 6.4 Network Structure

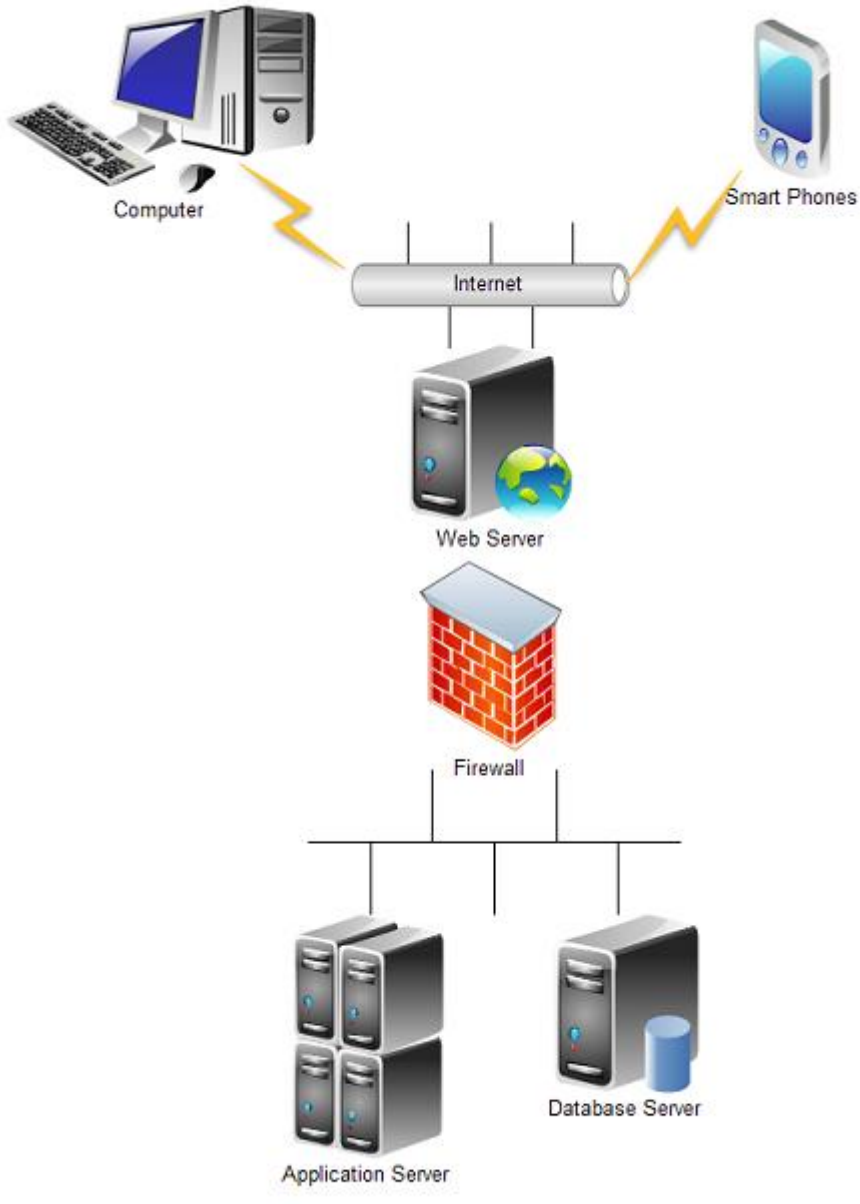



Figure 23 Network Structure

System users are able to access the user interface of the system by using web browsers on PCs and wireless devices such as smart phones through the Internet. By using the user interface, system users can send SQL command from application server to database server in order to complete tasks.

## 6.5 System Structure

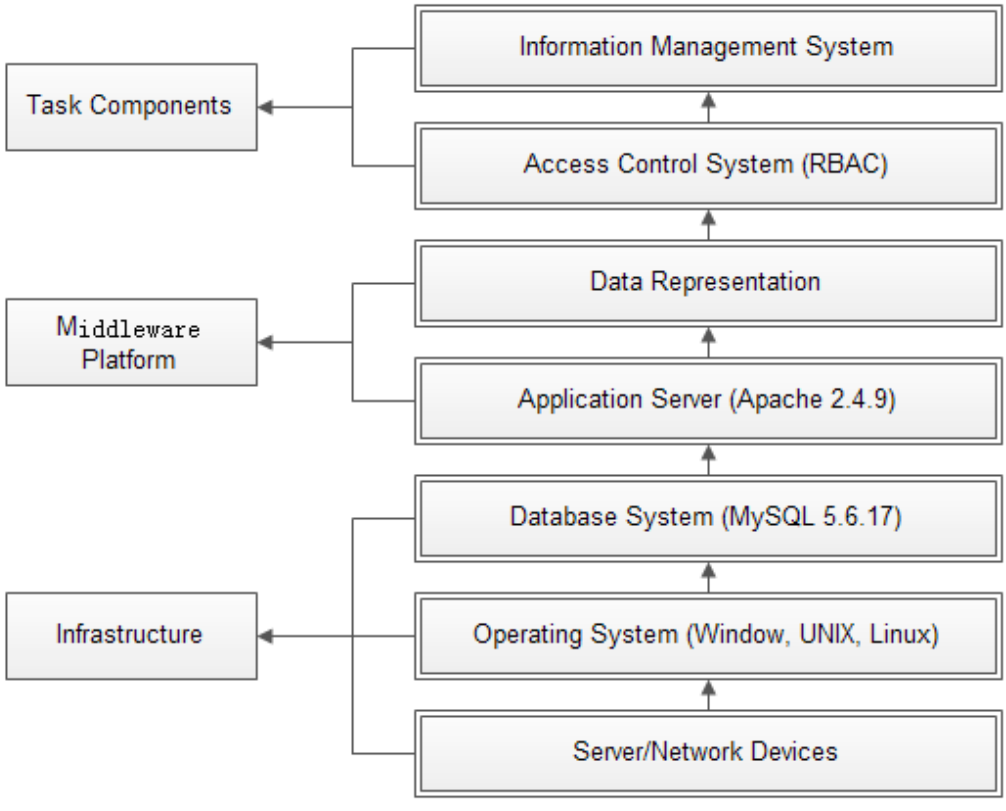



Figure 24 System Structure

The system can be generally divided into three main parts including infrastructure, middleware platform and task components. j

1) Infrastructure

Infrastructure consists of project support environment including server devices, network devices and operating systems. Database system is used to store all kinds of task data generated by the system, and it also provides data service in order to support data integration for system users.

2) Middleware Platform

Apache as application server provides stable and efficient operating environment for the application system.

3) Task Components

Information management system provides required functions in order to help system users to complete tasks. RBAC technology is used in order to provide a secure, flexible, scalable, and efficient access control system.

# 7. System Detail Design

System detail design is to clarify system design in detail based on system architecture design. It is mainly about the classes and objects used in system use cases in order to specify attributes and operations of each class.

"The Unified Modeling Language (UML) is a general-purpose modeling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system."[4]

UML is used to design this system based on the detailed analysis of business and task flow of the company.

## 7.1 System Sequence Diagram

All of the use cases are similar in system sequence; therefore "user tasks" in the system sequence diagram are used to generally represent all the use cases in the system sequence diagram.

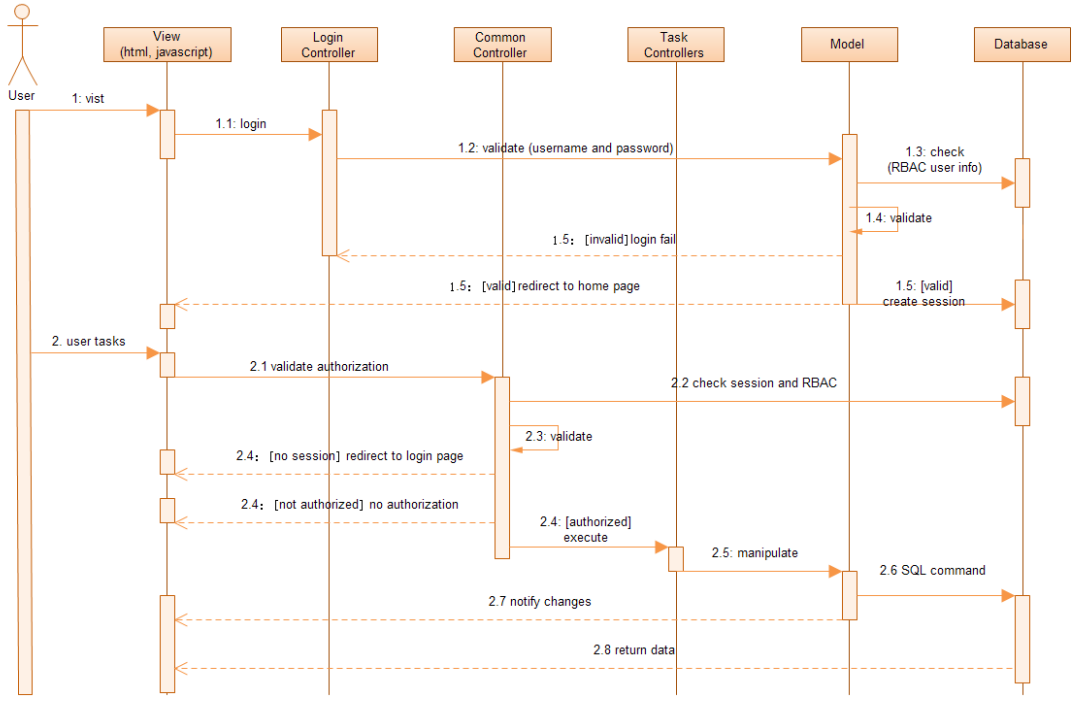



Figure 25 System sequence diagram

1) View
   View is the part of the system that system users directly deal with. In this part, the interfaces are also included which are designed by using mostly html and JavaScript, and some PHP code as well. In actual task processes, this part is replaced by html pages.

[4] Unified Modeling Language User Guide, The (2 ed.). Addison-Wesley. 2005. p. 496. ISBN 0321267974.

2) Login controller
   Login controller is the initial controller associated with RBAC system in order to validate username and password when system users or unauthorized users try to login the system. After system users successfully login in, session information of users will be added into database for system users' authorization of each function.
3) Common controller
   Common controller is used to validate if system users have authorization to invoke a certain function under a task controller. In order to do that, every task controller extends common controller. First, common controller checks session information. If there is no session information or the user's session is expired, common controller will redirect the user to login page. If session information is valid, RBAC system will be associated with in order to validate if the user has authorization to invoke a certain function. Error message will be returned if the user is unauthorized.
4) Task controller
   There are several functions under each task controller in order to complete system users' tasks. After a task controller is executed, related models will be invoked in order to manage data, rules and logic. A task controller can also return data and notify changes directly to view if there is no model related.
5) Model
   Models are used to manage data, rules and logic of application. A model can be activated manually or passively by executing a related controller. Then, return prepared data to view.

## 7.2 Class Diagram

1) Employee Information Management

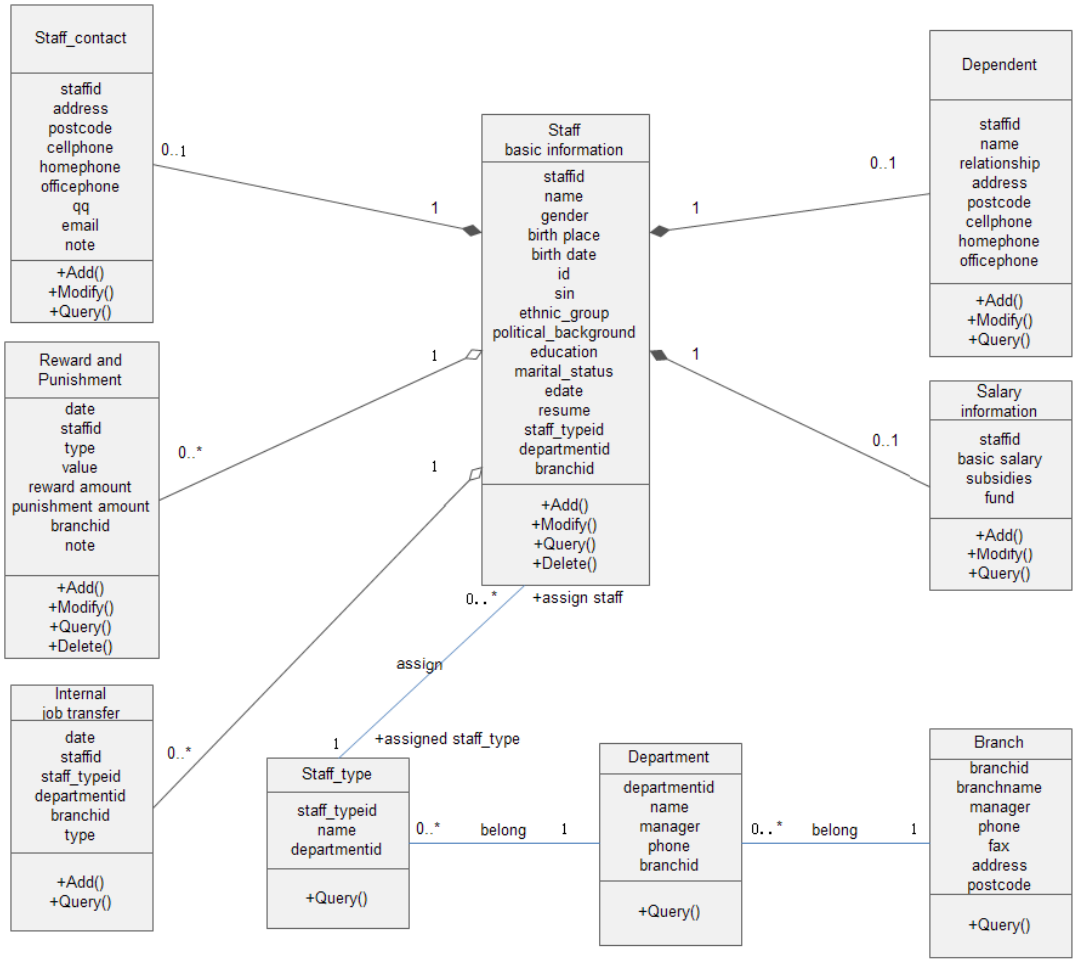



Figure 26 Class Diagram - Employee information management

The core of employee information management is staff basic information. Reward and punishment and internal job transfer information connect to staff basic information through staff id. The relationships between them and staff basic information are one-to-many. Internal job transfer information is to keep information of staff type, staff department and staff branch changes.

Through employee information management, staff information including basics, salary, and contact can be added, modified, searched and deleted as whole. Reward and punishment records are able to be added, modified, searched and deleted. Internal job transfer records are able to be added and searched.

2) Recruiting

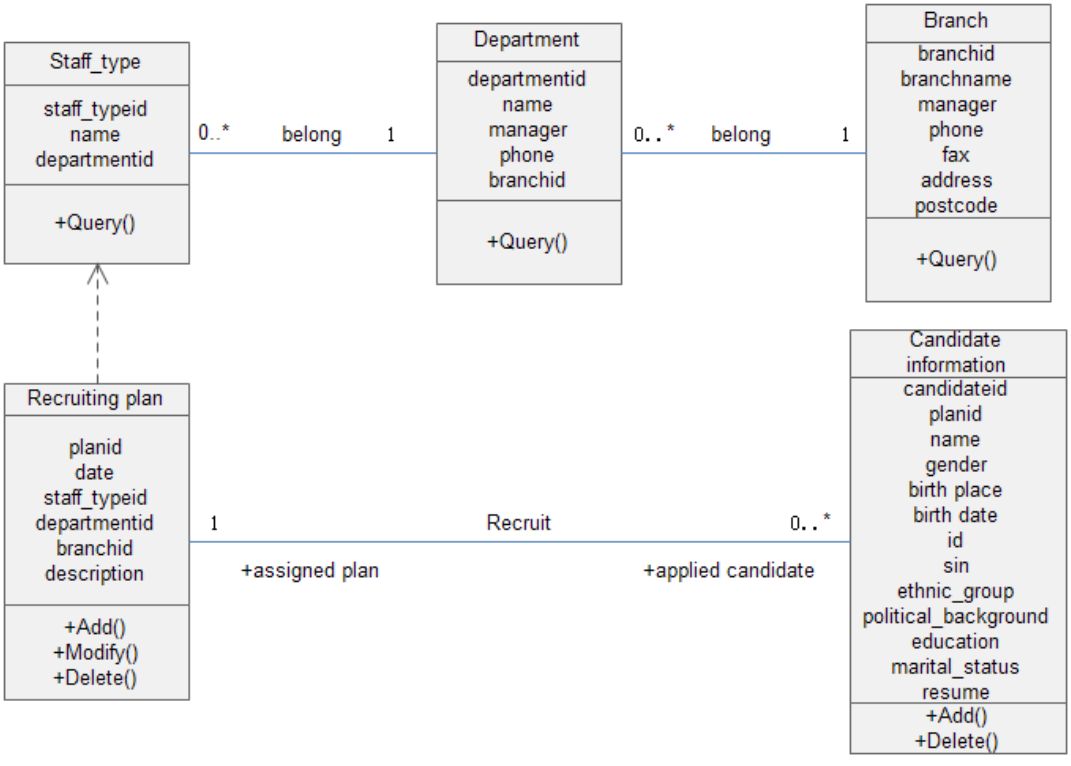



Figure 27 Class Diagram - Recruiting

The main parts of recruiting are recruiting plan and candidate information. They are connected through recruiting plan id. Recruiting plan depends on staff type, department, and branch information.

Through recruiting management, recruiting plan is able to be added, modified and deleted. Candidate information is able to be added and deleted.

3) Payroll management

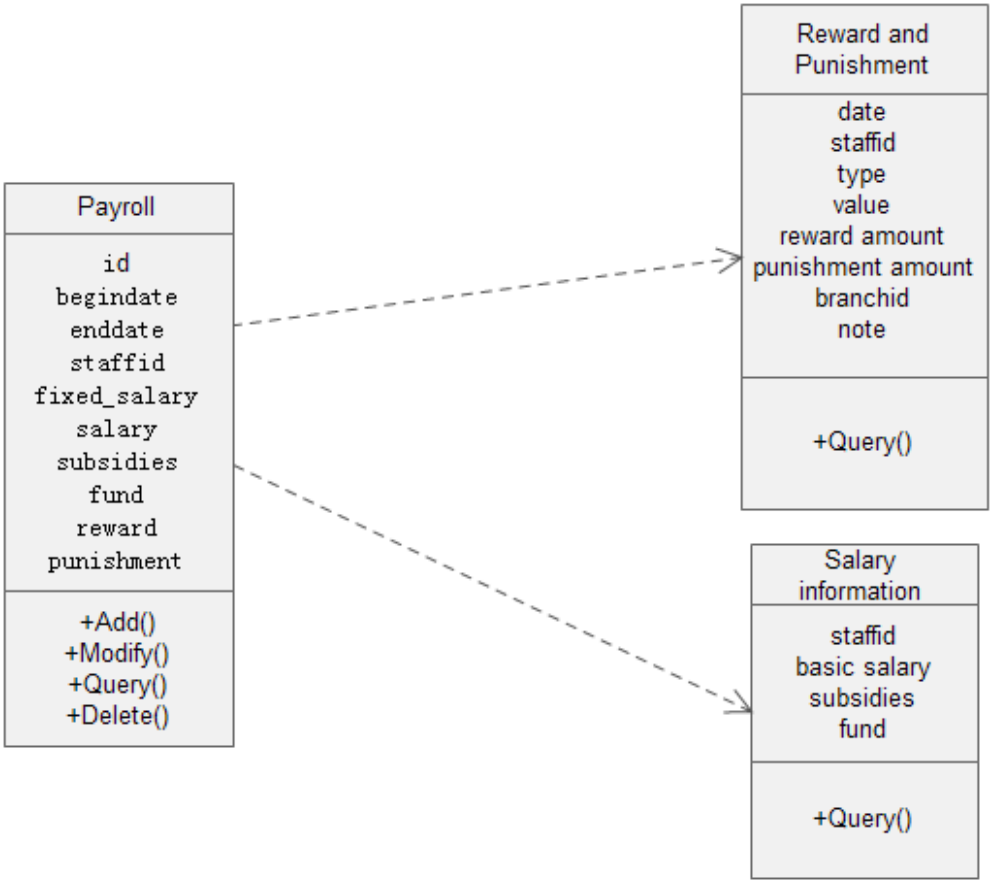



Figure 28 Class Diagram - Payroll management

Payroll is the core in this class. Payroll depends on reward and punishment and salary information from employee information management.

Through payroll management, payroll records can be added by summarizing total reward and punishment value and salary during selected time period. Payroll records are also able to be searched and deleted.

Object diagram is introduced to explain payroll records.

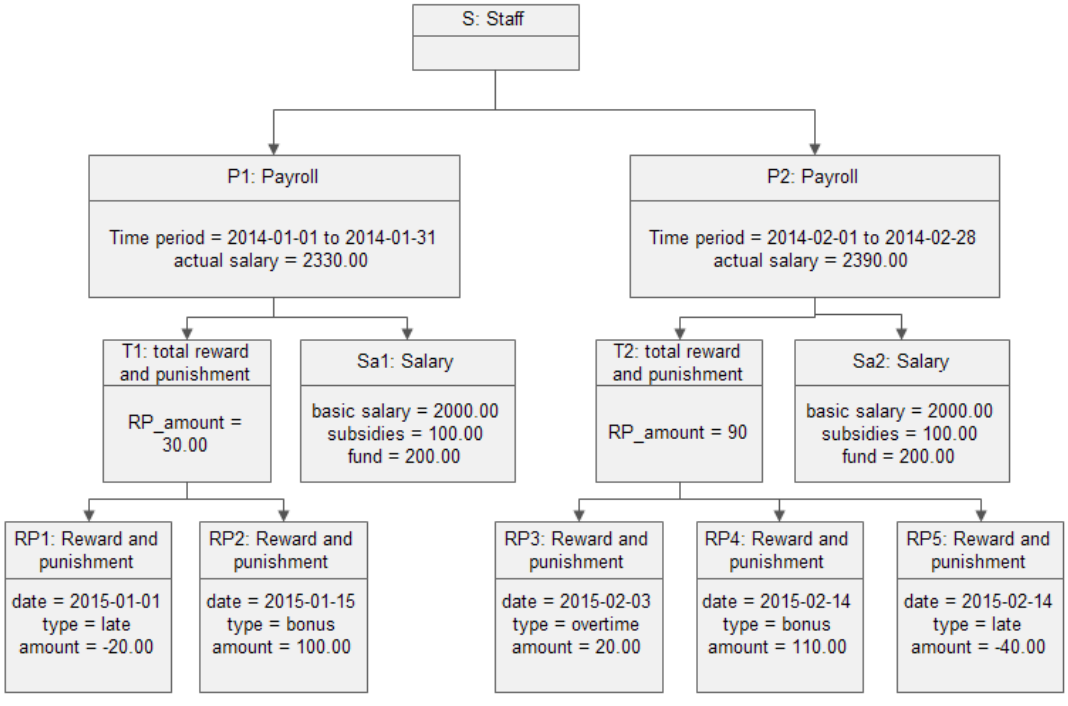



**Figure 29 Object Diagram – Payroll records**

Each payroll record of an employee not only contains total reward and punishment amount and current salary information, it also summarizes them as actual salary. The time period in payroll is for summarizing all reward and punishment records during this period of time as total reward and punishment in a payroll record.

4) Customer Information Management

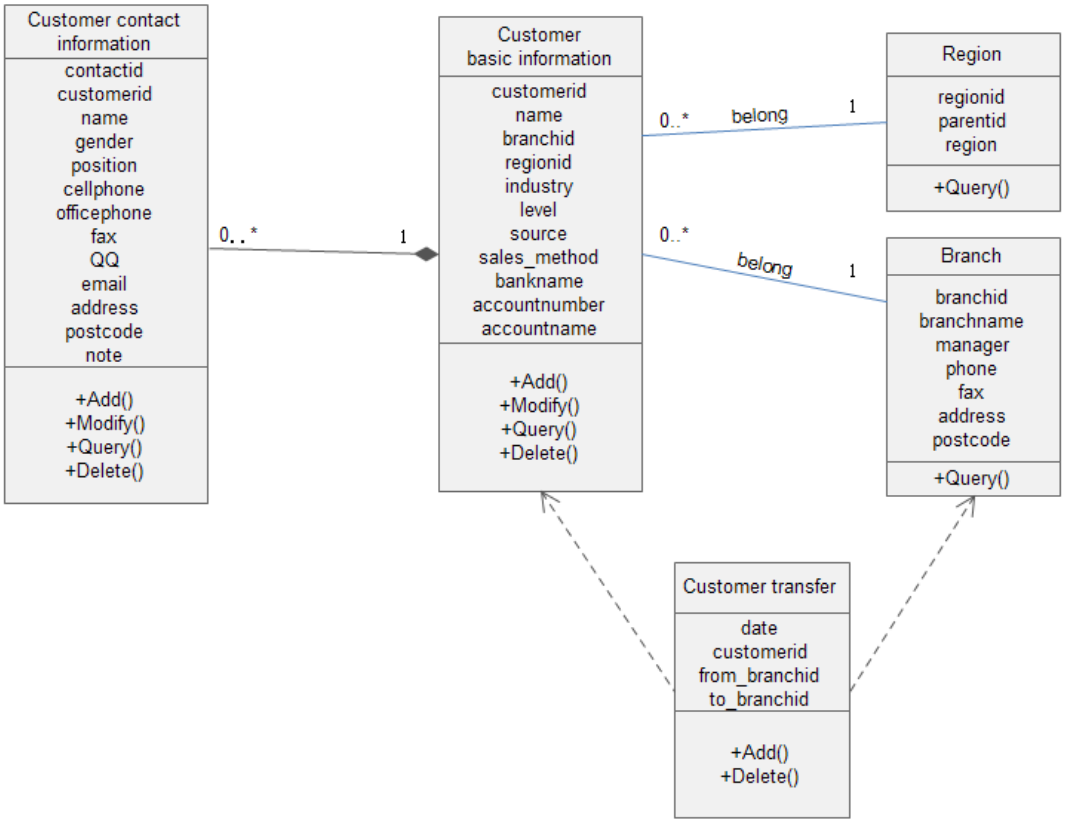



Figure 30 Class Diagram - Customer information management

Customer basic information is the core here. Customer contact information is connected to customer basic information through customer id. The relationship between contact information and customer basic information is one-to-many. Customer information is also connected to region and branch information through region id and branch id. The relationships between customer information and them are one-to-many. Customer transfer is used to keep records after customers are transferred from a branch to another.

Through customer information management, both customer basic information and contact information are able to be added, modified, searched and deleted. Customer transfer records can only be added or deleted.

5) Sales Management

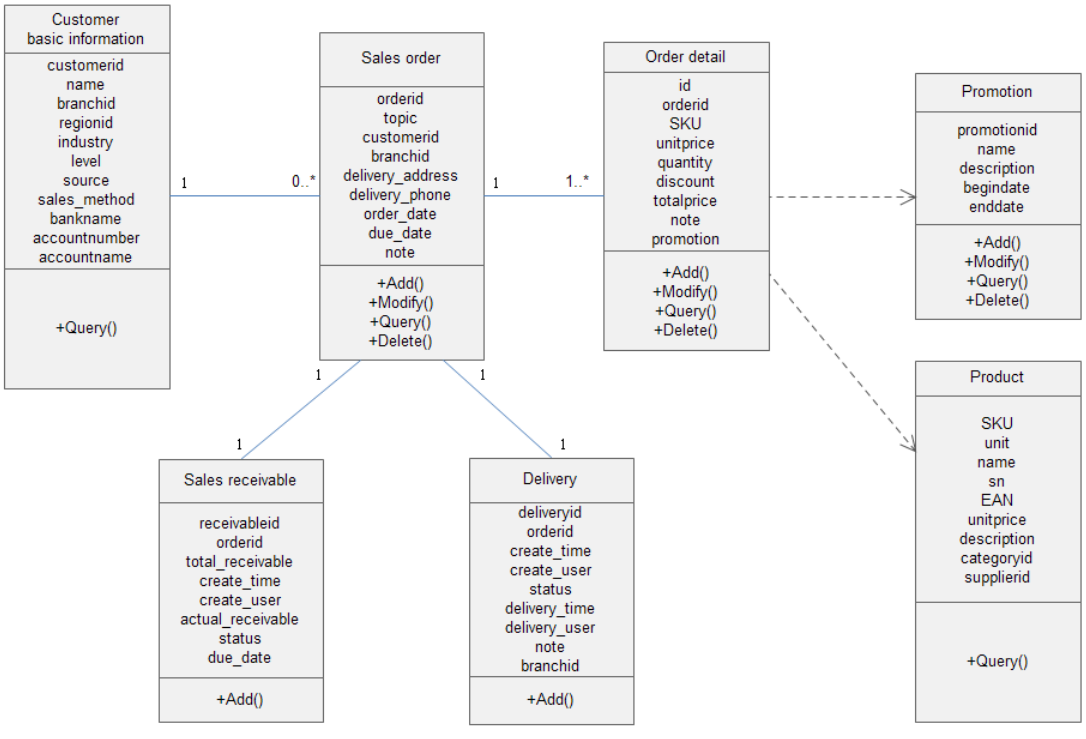



**Figure 31 Class Diagram - Sales management**

Sales order and order detail is the core of sales management. They are connected through order id. The relationship between sales order and order detail is one-to-many. Each order detail record represents an ordered product and related information in a sales order. Order detail depends on promotion and product information. Promotion is connected to order detail through promotion id, and product is connected to order detail through SKU. Sales receivable is connected to sales order through ordered, and the relationship is one-to-one. Delivery is also connected to sales order as same as sales receivable.

Through sales management, promotion can be added, modified, searched and deleted. Sales order can be added, modified, searched and deleted. Order details can be added, modified, searched and deleted through sales order. Sales receivable and delivery record can be added through confirming sales order.

Object diagram is introduced to explain the process of sales order.

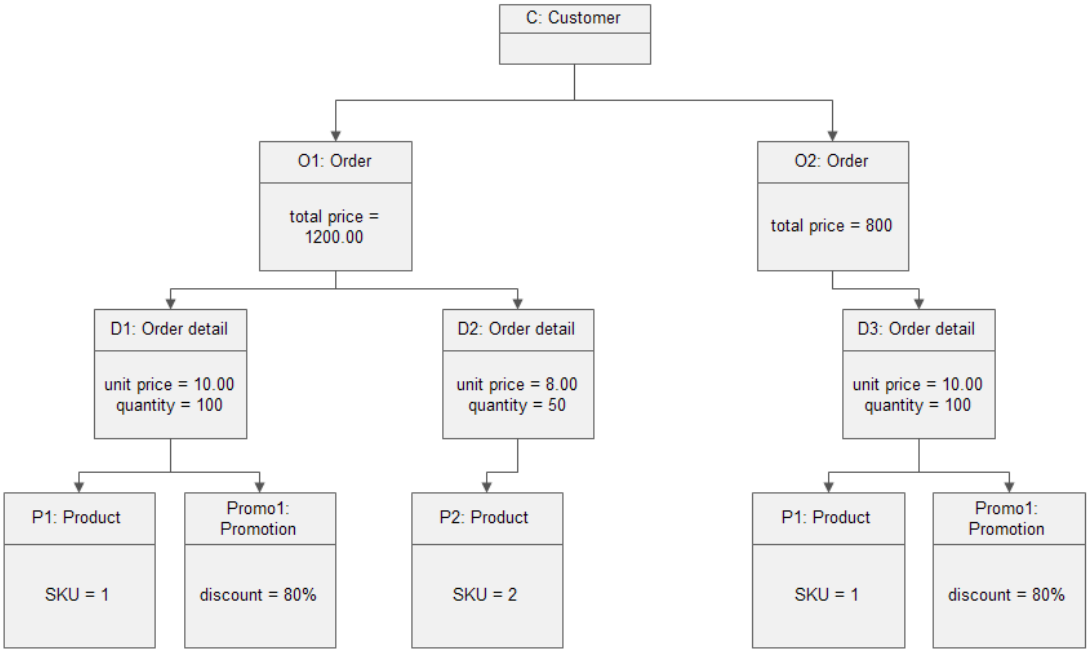



**Figure 32 Object Diagram - Sales order**

Each order from a customer contains at least one detail record. Each detail record is about only one ordered product record. It contains not only unit price and quantity but also SKU and discount of related promotion if a promotion plan is applied for the specific sale of product. Total price of each other detail record is calculated and summarized as total price in order record.

6) Purchase Management

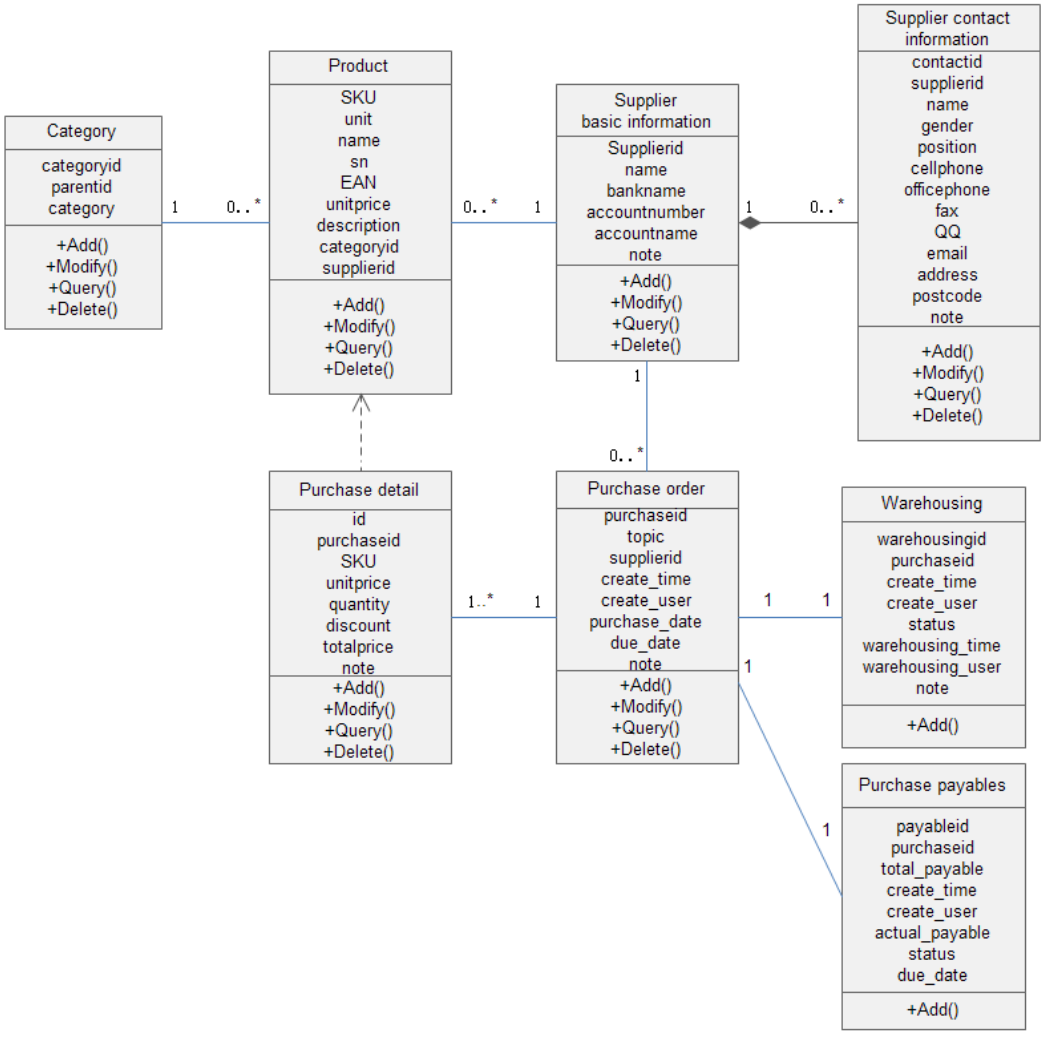



Figure 33 Class Diagram - Purchase management

There are three main parts in purchase management: supplier information, product and purchase order. Supplier basic information and supplier contact information are connected through supplier id, and the relationship is one-to-many. Product is connected to category and supplier basic information through category id and supplier id respectively, and both of relationships are one-to-many. Purchase order is connected to supplier basic information through supplier id, and the relationship is one-to-many. Purchase detail, warehousing, purchase payables are all connected to purchase order through purchase id. The relationship between purchase detail and purchase order is as same as the relationship between sales detail and sales order, which is one-to-many.

Through purchase management, supplier basic information and product category information can be added, modified, searched and deleted. Supplier contact information

can be added, modified, searched and deleted through supplier basic information. Product records can be added after selecting category and supplier information. The records can also be modified, searched and deleted. Purchase order can be added after selecting supplier information. Purchase detail records can be added, modified, searched, and deleted through purchase order, and purchase detail records depend on product information. After confirming purchase order, warehousing records and purchase payable records can be added.

7) Inventory Management

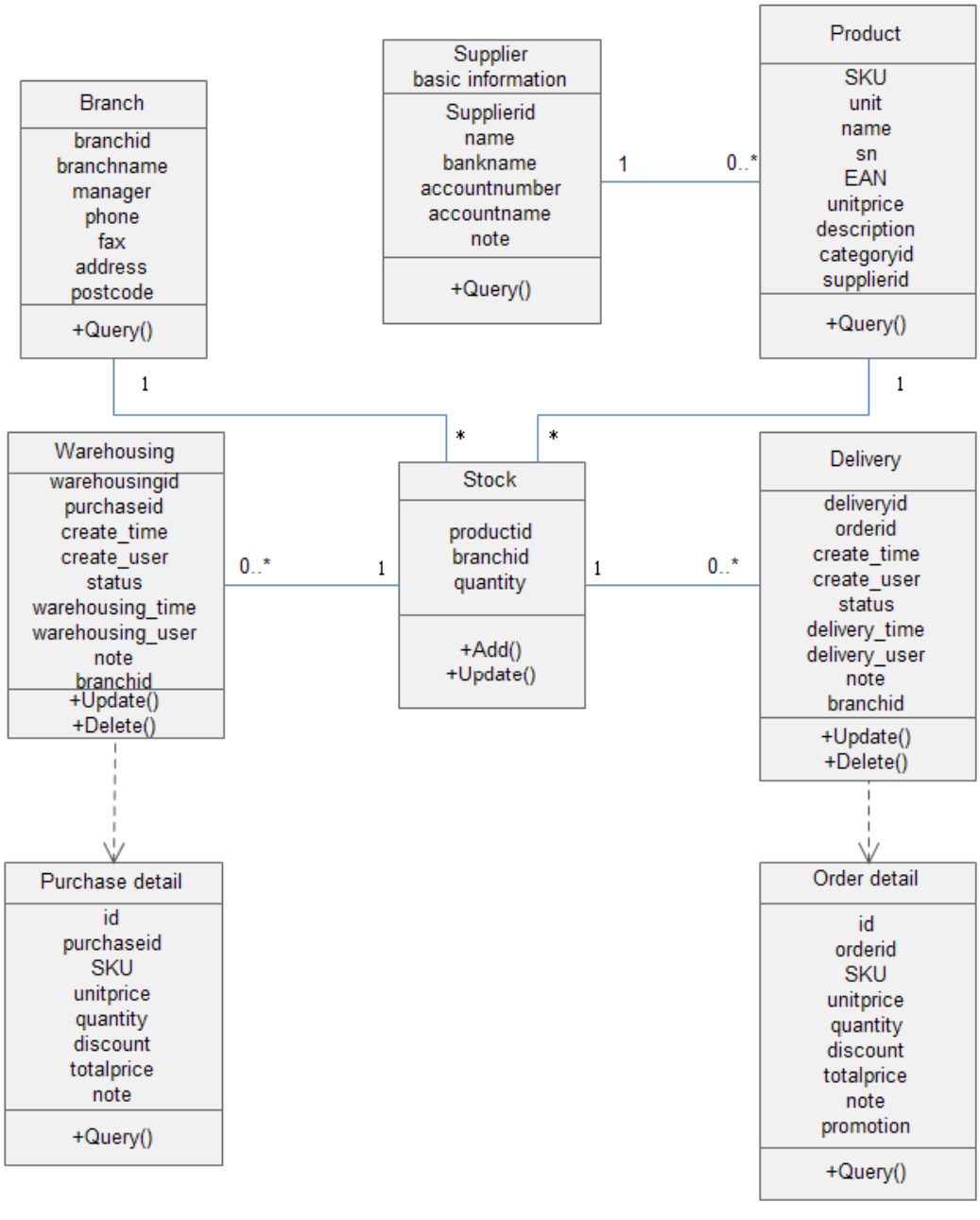



Figure 34 Class Diagram - Inventory management

The core of inventory management is stock class. It is the link of the many-to-many relationship between branch and product. Stock is connected to them through branch id and SKU respectively. Warehousing and delivery are connected to stock through branch id, and both of the relationships are many-to-one.

Through Inventory management, stock records can be added and updated by updating status in warehousing and delivery records. The data need to be added or updated is stored in purchase detail and order detail. For instance, after confirming products are delivered into warehouse (update warehousing status), stock is able to get SKU and quantity from purchase detail through purchase id in warehousing record. If certain product exists under the branch id got from warehousing, quantity of the certain product will be added up. If not, a new record will be added into stock. Warehousing records are added from purchase management, and delivery records are added from sales management. Records in both of them can be deleted here in case purchase orders and sales orders are cancelled.

8) Transshipment Management

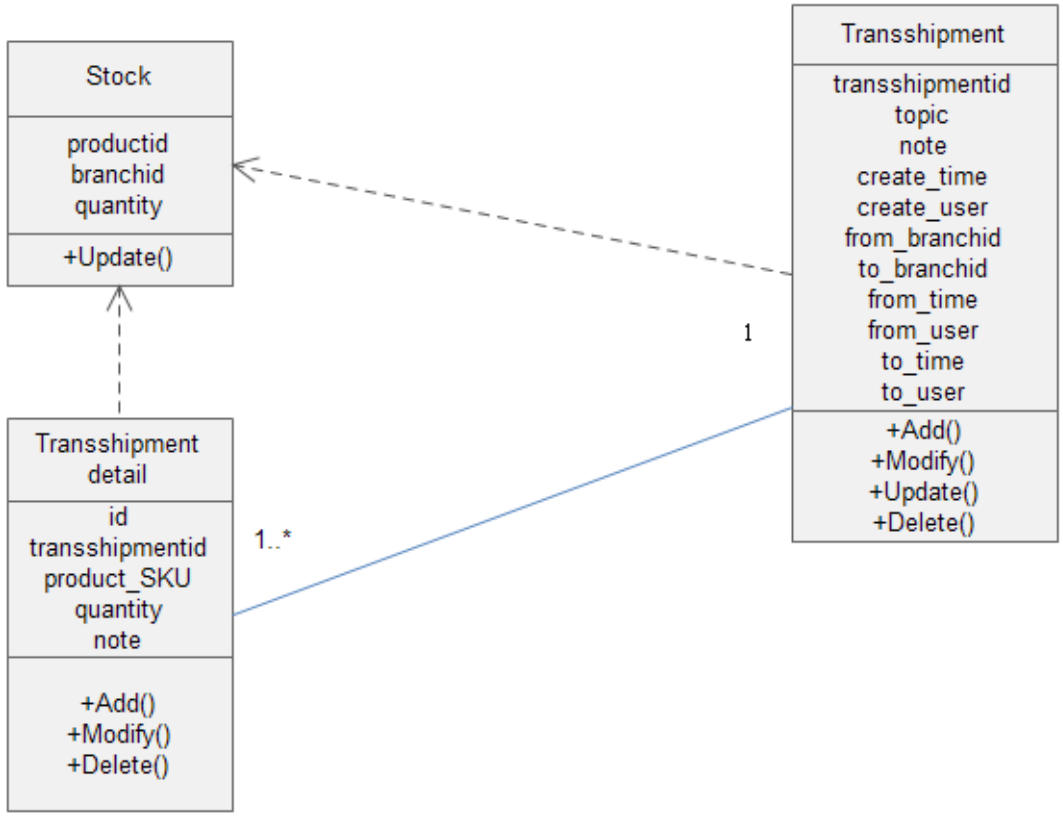



Figure 35 Class Diagram - Transshipment management

Transshipment is the core here. Transshipment detail is connected to transshipment, and the relationship is many-to-one. Both transshipment and transshipment detail depend on stock.

Through transshipment management, transshipment is able to be added, modified, searched and deleted. Transshipment detail records are able to be added, modified, searched and deleted through transshipment. After updating transshipment status, records in stock can be added and updated from transshipment detail records.

9) Payables Management

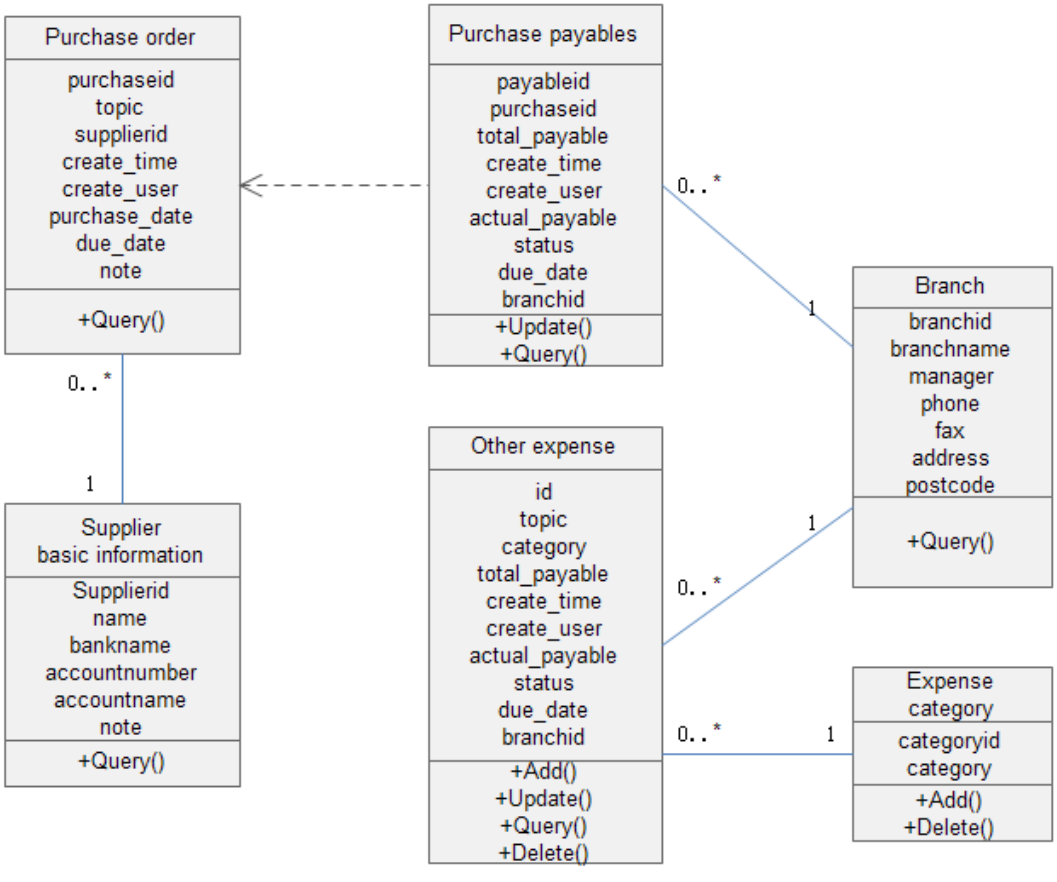



Figure 36 Class Diagram - Payables management

The core in payables management is purchase payables and other expense. Both of them are connected to branch through branched. The relationships are many-to-one. Expense category is used to categorize different other expense records. It is connected with other expense through category id.

Through payables management, purchase payables can be updated and searched. Records in purchase payables are added from purchase management. Unlike, purchase payables, in order to add other expense record, expense category need to be selected. Expense category can be added or deleted. Records in other expense can also be deleted unlike purchase payables.

10) Receivables Management

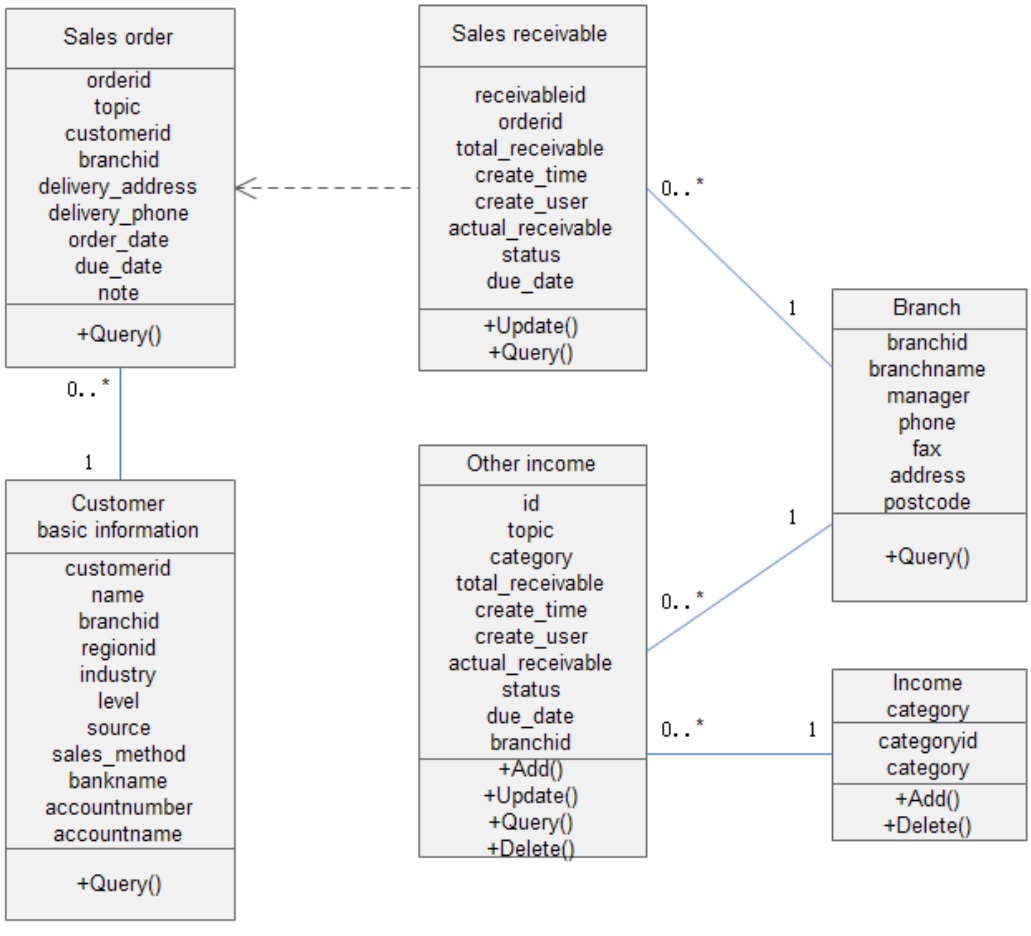



Figure 37 Class Diagram - Receivables management

Receivables management is similar to payables management in class design. The only difference between payables management and receivables management is the use. Receivables management is to manage income information which includes sales receivables and other income.

11) General Ledger Management

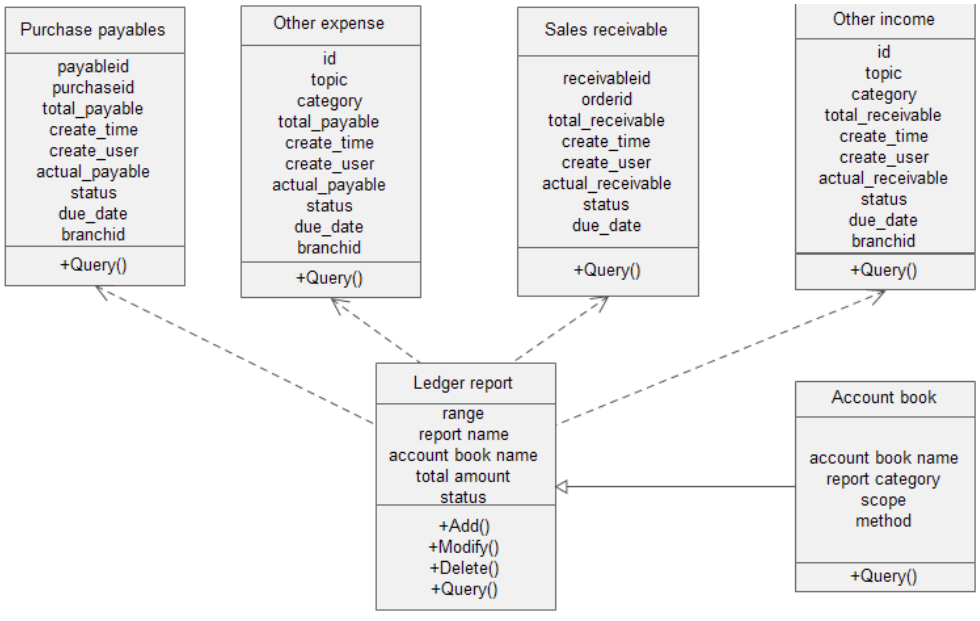



Figure 38 Class Diagram - General ledger management

General ledger management is to mainly generate financial report. The core here is ledger report. In order to generate a financial report, first, a designed account book will be chosen. Then specific range will be selected for the report. For instance, if it is a report about monthly income, then sales receivable and other income will be involved. After the range such as time period is selected, the system will be able to calculate the summary of total income during the period of time.

## 7.3 Database Design

In this section, database details will be explained. Details are divided based on different subsystems. Designs of main tables in each subsystem are included.

### 7.3.1 RBAC System

In RBAC system, access control is based on roles, and an individual user belong to one or more groups. Users have assigned roles and assigned branches. The process of defining role is based on the analysis of organization operations in the company.

Access is grouped by role name, and the use of resource is limited to associated role of authorized users. For instance, in human resource management, branch managers are able to add, modify, update and delete reward and punishment in payroll management; however, accountants are only able to use information in reward and punishment.

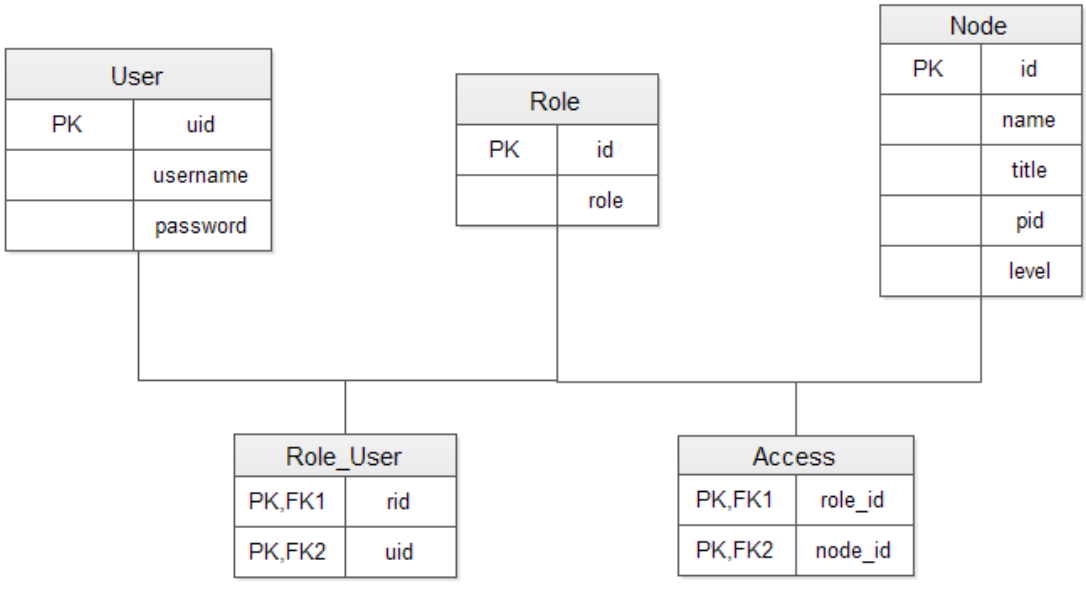



Figure 39 RBAC system based on web application

**User and Role**

Users are the subject of the system. Role is introduced to separate the relationship between user and privilege. Depending on user properties, a user is assigned to a designed role. Privileges are assigned to roles instead of users.

**Assignment**

There are two parts in assignment. Role_user is user assignment is to assign users to defined roles. Access is permission assignment is to assign permission of access and operations to roles.

**Role Hierarchies (Role parent id)**

Role hierarchies are considered as role groups. Role groups can be used to define levels of different roles in a role group. A role can be assigned to a role group in order to assign privileges.

**Permission (Node)**

Permissions describe privileges of access and operations that roles have.

**Session**

A session represents the relationship between a user and a role. A user needs to activate roles by creating sessions in order to get assigned privileges.

**Database Schema**

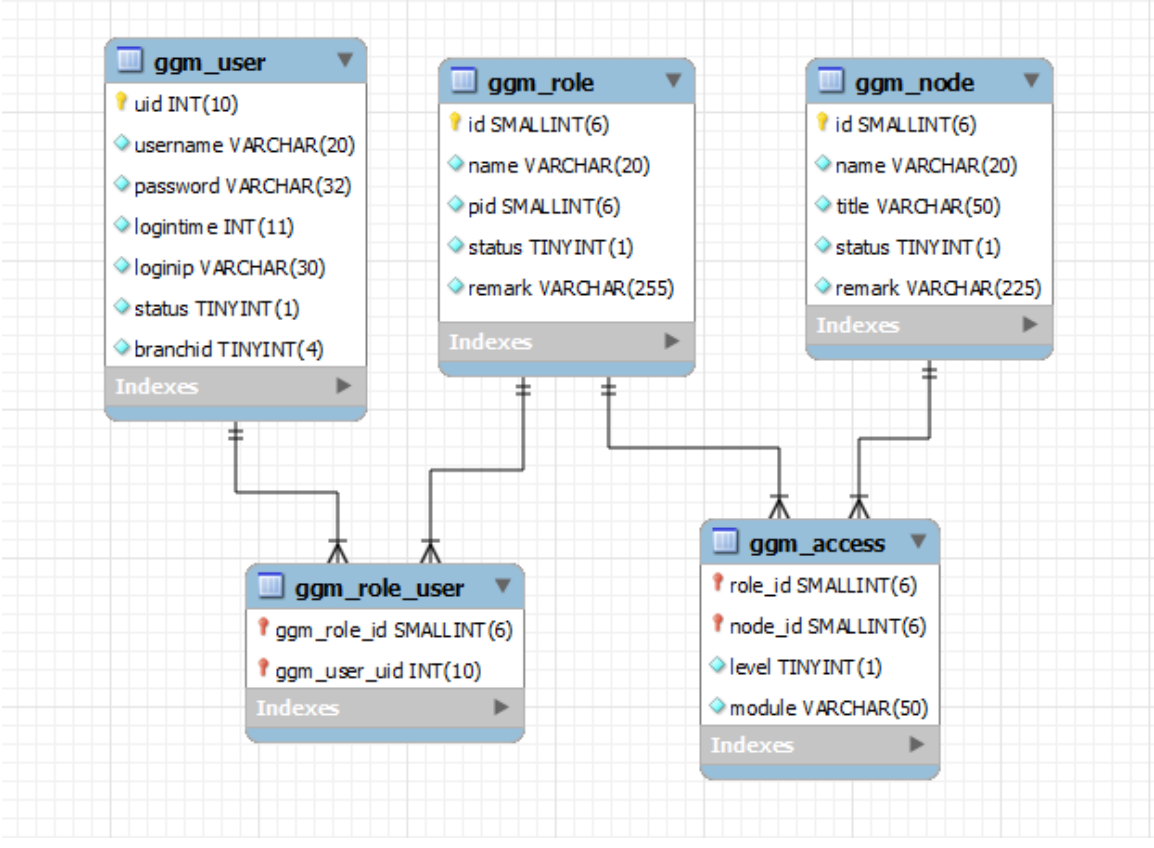



Figure 40 Database Schemas - RBAC

**Table Design**

1) ggm_user table

Field name: uid, username, password, logintime, loginip, status, branched

Primary key: uid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| uid | INT(10) | ✓ | ✓ | ✓ | | | | ✓ | |
| username | VARCHAR(20) | | ✓ | ✓ | | | | | |
| password | VARCHAR(32) | | ✓ | | | | | | |
| logintime | INT(11) | | ✓ | | | | | | |
| loginip | VARCHAR(30) | | ✓ | | | | | | |
| status | TINYINT(1) | | ✓ | | | ✓ | | | '0' |
| branchid | TINYINT(4) | | ✓ | | | | | | |

Figure 40 ggm_user table

2) ggm_role table

Field name: id, name, pid, status, remark

Primary key: id

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| id | SMALLINT(6) | ✓ | ✓ | | | ✓ | | ✓ | |
| name | VARCHAR(20) | | ✓ | | | | | | |
| pid | SMALLINT(6) | | ✓ | | | | | | 0 |
| status | TINYINT(1) | | ✓ | | | ✓ | | | 0 |
| remark | VARCHAR(20) | | ✓ | | | | | | '-' |

Figure 41 ggm_role table

3) ggm_role_user table
   Field name: id, uid
   Primary key: id, uid
   Foreign key: id, uid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| role_id | TINYINT | ✓ | ✓ | | | | | | |
| user_id | INT(10) | ✓ | ✓ | | | | | | |

Figure 42 ggm_role_user table

4) ggm_node
   Field name: id, name, title, status, remark, sort, pid, level
   Primary key: id

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| id | SMALLINT(6) | ✓ | ✓ | | | ✓ | | ✓ | |
| name | VARCHAR(20) | | ✓ | | | | | | |
| title | VARCHAR(50) | | ✓ | | | | | | '-' |
| status | TINYINT(1) | | ✓ | | | | | | '0' |
| remark | VARCHAR(255) | | ✓ | | | | | | '-' |
| sort | SMALLINT(6) | | ✓ | | | ✓ | | | 0 |
| pid | SMALLINT(6) | | ✓ | | | ✓ | | | |
| level | TINYINT(1) | | ✓ | | | ✓ | | | |

Figure 43 ggm_node table

5) ggm_access

Field name: role_id, node_id, level, module

Primary key: role_id, node_id

Foreign key: role_id, node_id

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| role_id | SMALLINT(6) | ☐ | ☑ | ☐ | ☐ | ☑ | ☐ | ☐ | |
| node_id | SMALLINT(6) | ☐ | ☑ | ☐ | ☐ | ☑ | ☐ | ☐ | |
| level | TINYINT(1) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| module | VARCHAR(50) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | '-' |
| | | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |

Figure 44 ggm_access table

6) session

Field name: session_id, session_expire, session_data

Primary key: role_id, node_id

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| session_id | VARCHAR(255) | ☐ | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | |
| session_expire | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| session_data | BLOB | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| | | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |

Figure 45 session table

### 7.3.2 Human Resource Management

**Database Schema**

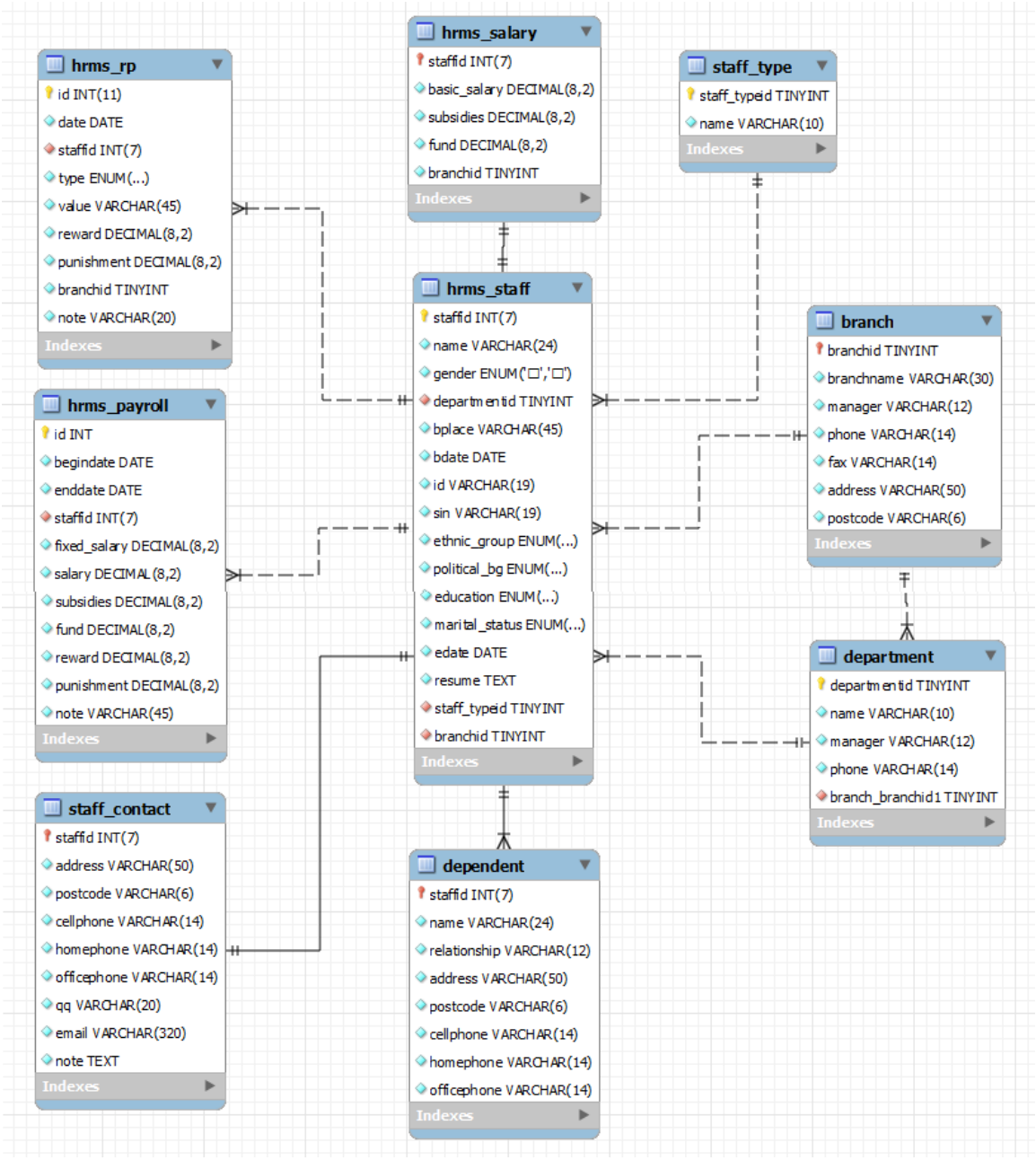



Figure 46 Database Schema - Human resource management

## Table Design

1) Employee basic information table

Field name: staffid, name, gender, departmentid, bplace, bdate, id, sin, ethnic_group, political_bg, education, marital_status, edate, resume, branchid, staff_typeid

Primary key: staffid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| staffid | INT(7) | ✓ | ✓ | | | ✓ | ✓ | ✓ | |
| name | VARCHAR(24) | | ✓ | | | | | | |
| gender | ENUM('男','女') | | ✓ | | | | | | |
| departmentid | TINYINT(4) | | ✓ | | | | | | |
| bplace | VARCHAR(45) | | ✓ | | | | | | |
| bdate | DATE | | ✓ | | | | | | |
| id | VARCHAR(19) | | ✓ | ✓ | | | | | |
| sin | VARCHAR(19) | | ✓ | | | | | | |
| ethnic_group | ENUM('-','汉族','壮族... | | ✓ | | | | | | |
| political_bg | ENUM('-','党员','团员... | | ✓ | | | | | | |
| education | ENUM('-','小学','初中... | | ✓ | | | | | | |
| marital_status | ENUM('-','单身','已婚... | | ✓ | | | | | | |
| edate | DATE | | ✓ | | | | | | |
| resume | TEXT | | ✓ | | | | | | |
| branchid | TINYINT(4) | | ✓ | | | | | | |
| staff_typeid | TINYINT(4) | | ✓ | | | | | | |

Figure 47 Employee basic information table

2) Employee contact table

Field name: staffid, address, postcode, cellphone, homephone, officephone, qq, email, note

Primary key: staffid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| staffid | INT(7) | ✓ | ✓ | | | ✓ | ✓ | | |
| address | VARCHAR(50) | | ✓ | | | | | | '-' |
| postcode | VARCHAR(6) | | ✓ | | | | | | '-' |
| cellphone | VARCHAR(14) | | ✓ | | | | | | '-' |
| homephone | VARCHAR(14) | | ✓ | | | | | | '-' |
| officephone | VARCHAR(14) | | ✓ | | | | | | '-' |
| qq | VARCHAR(20) | | ✓ | | | | | | '-' |
| email | VARCHAR(320) | | ✓ | | | | | | '-' |
| note | TEXT | | ✓ | | | | | | |

Figure 48 Employee contact table

3) Employee salary table

Field name: staffid, basic_salary, subsidies, fund, branchid

Primary key: staffid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| staffid | INT(7) | ✓ | ✓ | | | ✓ | ✓ | | |
| basic_salary | DECIMAL(8,2) | | ✓ | | | | | | '0.00' |
| subsidies | DECIMAL(8,2) | | ✓ | | | | | | '0.00' |
| fund | DECIMAL(8,2) | | ✓ | | | | | | '0.00' |
| branchid | TINYINT(4) | | ✓ | | | | | | |

Figure 49 Employee salary table

4) Dependent table

Field name: staffid, name, relationship, address, postcode, cellphone, homephone, officephone,

Primary key: staffid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| staffid | INT(7) | ✓ | ✓ | | | ✓ | ✓ | | |
| name | VARCHAR(24) | | ✓ | | | | | | '-' |
| relationship | VARCHAR(12) | | ✓ | | | | | | '-' |
| address | VARCHAR(50) | | ✓ | | | | | | '-' |
| postcode | VARCHAR(6) | | ✓ | | | | | | '-' |
| cellphone | VARCHAR(14) | | ✓ | | | | | | '-' |
| homephone | VARCHAR(14) | | ✓ | | | | | | '-' |
| officephone | VARCHAR(14) | | ✓ | | | | | | '-' |
| | | | | | | | | | |

Figure 50 Dependent table

5) Employee reward and punishment table

Field name: id, date, staffid, type, value, reward, punishment, branchid, note

Primary key: id

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ✓ | ✓ | | | | | ✓ | |
| date | DATE | | ✓ | | | | | | |
| staffid | INT(7) | | ✓ | | | ✓ | ✓ | | |
| type | ENUM('奖金','加班','... | | ✓ | | | | | | |
| value | TINYINT(4) | | ✓ | | | | | | '0' |
| reward | DECIMAL(8,2) | | ✓ | | | | | | '0.00' |
| punishment | DECIMAL(8,2) | | ✓ | | | | | | '0.00' |
| branchid | TINYINT(4) | | ✓ | | | | | | '0' |
| note | VARCHAR(20) | | ✓ | | | | | | '-' |
| | | | | | | | | | |

Figure 51 Employee reward and punishment table

6) Payroll table

Field name: id, begindate, enddate, staffid, fixed_salary, salary, subsidies, fund, reward, punishment, note

Primary key: id

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ✓ | ✓ | | | | | ✓ | |
| begindate | DATE | | ✓ | | | | | | |
| enddate | DATE | | ✓ | | | | | | |
| staffid | INT(7) | | ✓ | | | ✓ | ✓ | | |
| fixed_salary | DECIMAL(8,2) | | ✓ | | | | | | '0.00' |
| salary | DECIMAL(8,2) | | ✓ | | | | | | '0.00' |
| subsidies | DECIMAL(8,2) | | ✓ | | | | | | '0.00' |
| fund | DECIMAL(8,2) | | ✓ | | | | | | '0.00' |
| reward | DECIMAL(8,2) | | ✓ | | | | | | '0.00' |
| punishment | DECIMAL(8,2) | | ✓ | | | | | | '0.00' |
| note | VARCHAR(45) | | ✓ | | | | | | '-' |

Figure 52 Payroll table

### 7.3.3 Customer Relationship Management

**Database Schema**

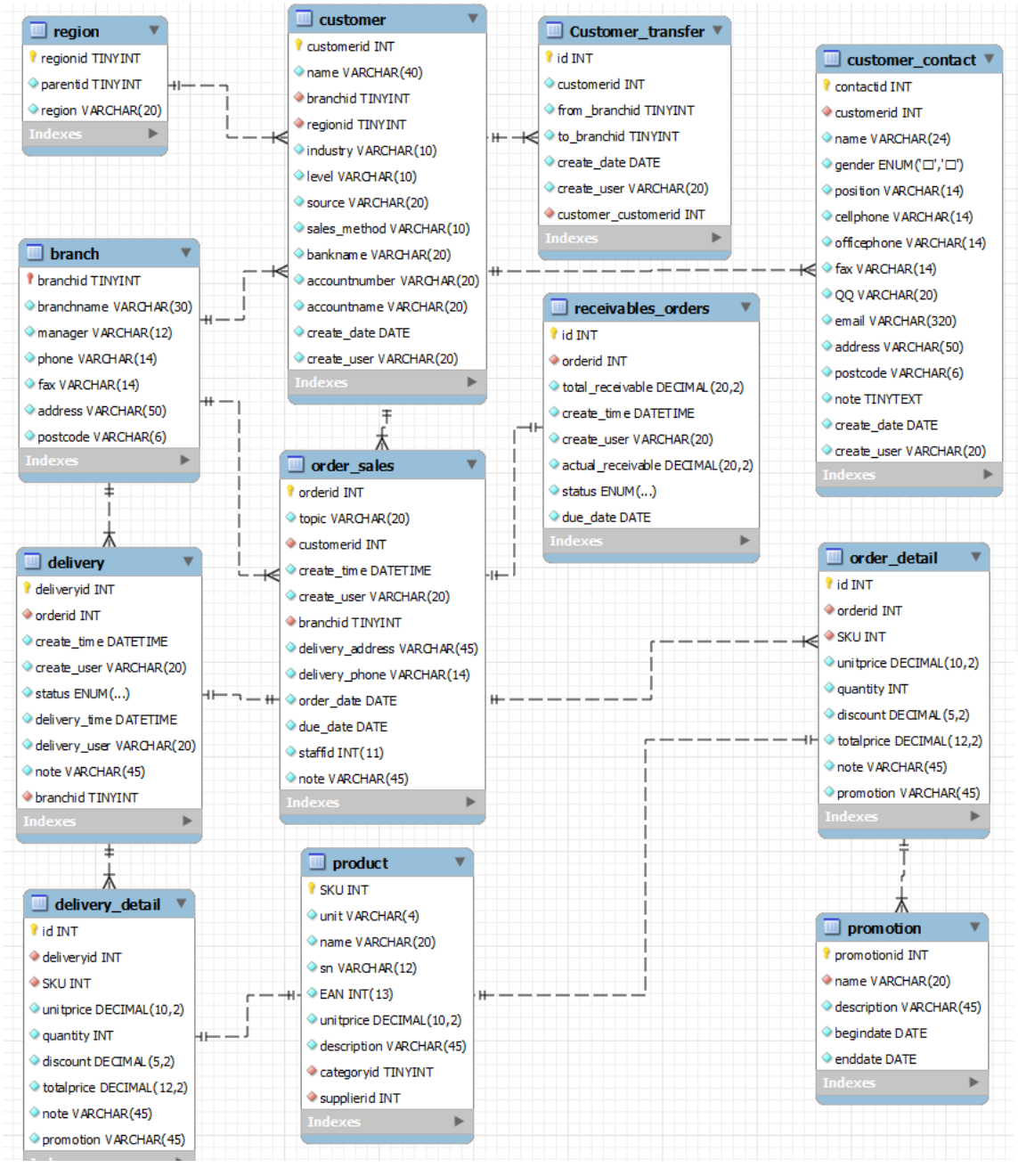



Figure 53 Database Schema - Customer relationship management

## Table Design

1) Customer basic information table

Field name: customerid, name, branchid, regionid, industry, level, source, sales_method, bankname, accountnumber, accountname, create_date, create_user

Primary key: customerid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| customerid | INT(11) | ✓ | ✓ | | | | | ✓ | |
| name | VARCHAR(40) | | ✓ | | | | | | '-' |
| branchid | TINYINT(4) | | ✓ | | | | | | |
| regionid | TINYINT(4) | | ✓ | | | | | | |
| industry | VARCHAR(10) | | ✓ | | | | | | '-' |
| level | VARCHAR(10) | | ✓ | | | | | | '-' |
| source | VARCHAR(20) | | ✓ | | | | | | '-' |
| sales_method | VARCHAR(10) | | ✓ | | | | | | '-' |
| bankname | VARCHAR(20) | | ✓ | | | | | | '-' |
| accountnumber | VARCHAR(20) | | ✓ | | | | | | '-' |
| accountname | VARCHAR(20) | | ✓ | | | | | | '-' |
| create_date | DATE | | ✓ | | | | | | |
| create_user | VARCHAR(20) | | ✓ | | | | | | |

Figure 54 Customer basic information table

2) Customer contact table

Field name: contactid, customerid, name, gender, position, cellphone, officephone, fax, QQ, email, address, postcode, note, create_date, create_user

Primary key: contactid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| contactid | INT(11) | ✓ | ✓ | | | | | ✓ | |
| customerid | INT(11) | | ✓ | | | | | | |
| name | VARCHAR(24) | | ✓ | | | | | | '-' |
| gender | ENUM('男','女') | | ✓ | | | | | | |
| position | VARCHAR(14) | | ✓ | | | | | | '-' |
| cellphone | VARCHAR(14) | | ✓ | | | | | | '-' |
| officephone | VARCHAR(14) | | ✓ | | | | | | '-' |
| fax | VARCHAR(14) | | ✓ | | | | | | '-' |
| QQ | VARCHAR(20) | | ✓ | | | | | | |
| email | VARCHAR(320) | | ✓ | | | | | | '-' |
| address | VARCHAR(50) | | ✓ | | | | | | '-' |
| postcode | VARCHAR(6) | | ✓ | | | | | | '-' |
| note | TINYTEXT | | ✓ | | | | | | |
| create_date | DATE | | ✓ | | | | | | |
| create_user | VARCHAR(20) | | ✓ | | | | | | |

Figure 55 Customer contact table

3) Customer transfer table

Field name: id, customerid, from_branchid, to_branchid, create_date, create_user

Primary key: id

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ✓ | ✓ | | | | | ✓ | |
| customerid | INT(11) | | ✓ | | | | | | |
| from_branchid | TINYINT(4) | | ✓ | | | | | | |
| to_branchid | TINYINT(4) | | ✓ | | | | | | |
| create_date | DATE | | ✓ | | | | | | |
| create_user | VARCHAR(20) | | ✓ | | | | | | |
| | | | | | | | | | |

Figure 56 Customer transfer table

4) Sales order table

Field name: orderid, topic, customerid, create_time, create_user, branchid, delivery_address, delivery_phone, order_date, due_date, staffid, note

Primary key: orderid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| orderid | INT(11) | ✓ | ✓ | | | | | ✓ | |
| topic | VARCHAR(20) | | ✓ | | | | | | '-' |
| customerid | INT(11) | | ✓ | | | | | | |
| create_time | DATETIME | | ✓ | | | | | | |
| create_user | VARCHAR(20) | | ✓ | | | | | | |
| branchid | TINYINT(4) | | ✓ | | | | | | |
| delivery_address | VARCHAR(45) | | ✓ | | | | | | |
| delivery_phone | VARCHAR(14) | | ✓ | | | | | | |
| order_date | DATE | | ✓ | | | | | | |
| due_date | DATE | | ✓ | | | | | | |
| staffid | INT(11) | | ✓ | | | | | | |
| note | VARCHAR(45) | | ✓ | | | | | | '-' |

Figure 57 Sales order table

5) Order detail table

Field name: id, orderid, SKU, unitprice, quantity, discount, totalprice, note, promotion

Primary key: id

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☑ | |
| orderid | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| SKU | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| unitprice | DECIMAL(10,2) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | '0.00' |
| quantity | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| discount | DECIMAL(5,2) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| totalprice | DECIMAL(12,2) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| note | VARCHAR(45) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | '-' |
| promotion | VARCHAR(45) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | '-' |

Figure 58 Order detail table

6) Promotion table

Field name: promotionid, name, description, begindate, enddate

Primary key: promotionid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| promotionid | INT(11) | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☑ | |
| name | VARCHAR(20) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| description | VARCHAR(45) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | '-' |
| begindate | DATE | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| enddate | DATE | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | |

Figure 59 Sales order table

### 7.3.4 Supply Chain Management

**Database Schema**

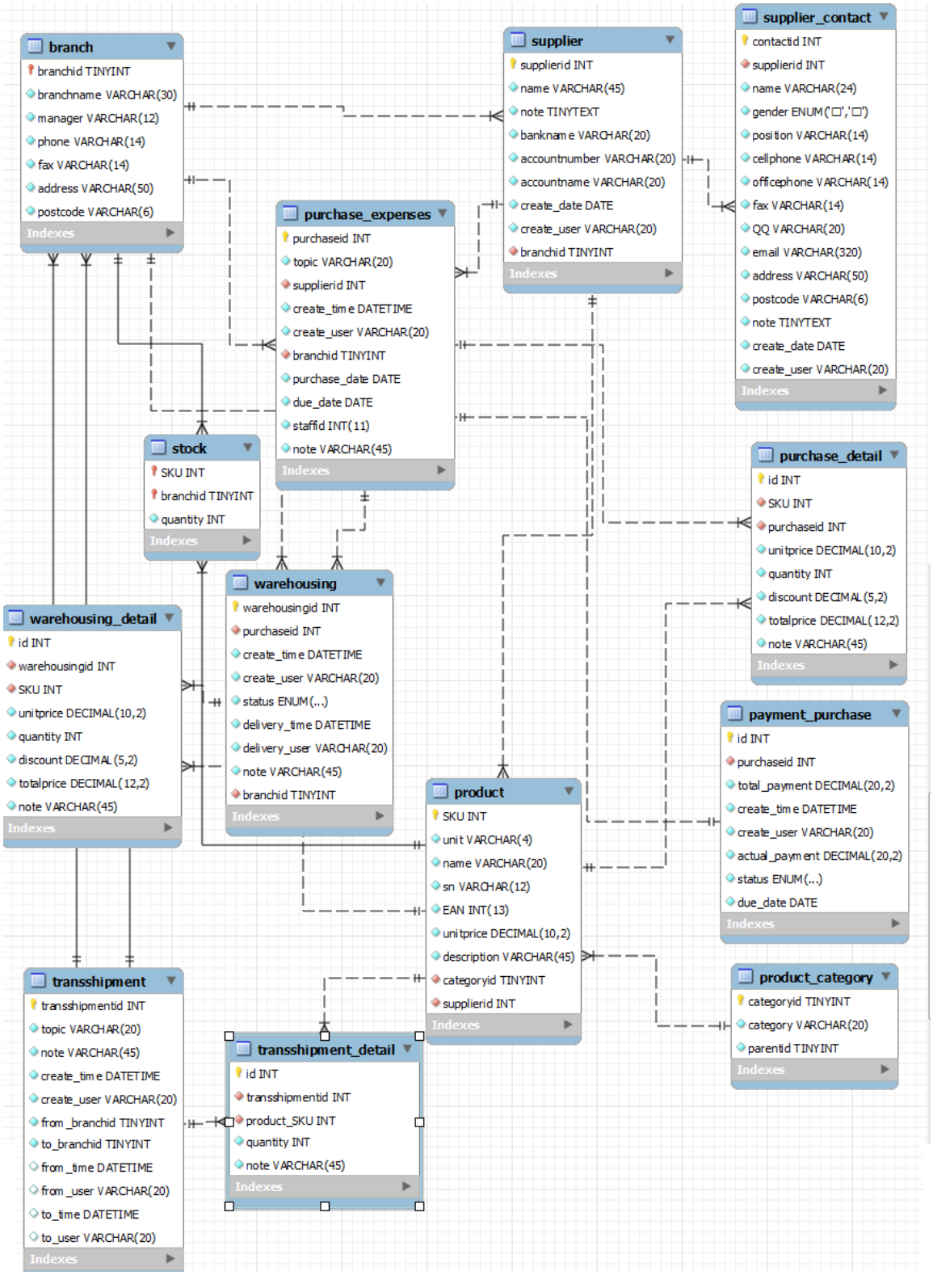



Figure 60 Database Schema - Supply chain management

## Table Design

1) Supplier basic information table

Field name: supplierid, name, note, bankname, accountnumber, accountname, create_date, create_user

Primary key: supplierid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| supplierid | INT(11) | ✓ | ✓ | | | | | ✓ | |
| name | VARCHAR(45) | | ✓ | | | | | | |
| note | TINYTEXT | | ✓ | | | | | | |
| bankname | VARCHAR(20) | | ✓ | | | | | | '-' |
| accountnumber | VARCHAR(20) | | ✓ | | | | | | '-' |
| accountname | VARCHAR(20) | | ✓ | | | | | | '-' |
| create_date | DATE | | ✓ | | | | | | |
| create_user | VARCHAR(20) | | ✓ | | | | | | |

Figure 61 Supplier basic information table

2) Supplier contact table

Field name: contactid, supplierid, name, gender, position, cellphone, officephone, fax, QQ, email, address, postcode, note, create_date, create_user

Primary key: contactid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| contactid | INT(11) | ✓ | ✓ | | | | | ✓ | |
| supplierid | INT(11) | | ✓ | | | | | | |
| name | VARCHAR(24) | | ✓ | | | | | | '-' |
| gender | ENUM('男','女') | | ✓ | | | | | | |
| position | VARCHAR(14) | | ✓ | | | | | | '-' |
| cellphone | VARCHAR(14) | | ✓ | | | | | | '-' |
| officephone | VARCHAR(14) | | ✓ | | | | | | '-' |
| fax | VARCHAR(14) | | ✓ | | | | | | '-' |
| QQ | VARCHAR(20) | | ✓ | | | | | | '-' |
| email | VARCHAR(320) | | ✓ | | | | | | '-' |
| address | VARCHAR(50) | | ✓ | | | | | | '-' |
| postcode | VARCHAR(6) | | ✓ | | | | | | '-' |
| note | TINYTEXT | | ✓ | | | | | | |
| create_date | DATE | | ✓ | | | | | | |
| create_user | VARCHAR(20) | | ✓ | | | | | | |

Figure 62 Supplier contact table

3) Product table

Field name: SKU, unit, name, sn, EAN, unitprice, description, categoryid, supplierid

Primary key: SKU

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| SKU | INT(11) | ✓ | ✓ | ✓ | | | | | |
| unit | VARCHAR(4) | | ✓ | | | | | | '-' |
| name | VARCHAR(20) | | ✓ | | | | | | |
| sn | VARCHAR(12) | | ✓ | | | | | | '-' |
| EAN | INT(13) | | ✓ | ✓ | | | | | |
| unitprice | DECIMAL(10,2) | | ✓ | | | | | | '0.00' |
| description | VARCHAR(45) | | ✓ | | | | | | '-' |
| categoryid | TINYINT(4) | | ✓ | | | | | | |
| supplierid | INT(11) | | ✓ | | | | | | |

Figure 63 Product table

4) Purchase order table

Field name: purchaseid, topic, supplierid, create_time, create_user, branchid, purchase_date, due_date, staffid, note

Primary key: purchaseid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| purchaseid | INT(11) | ✓ | ✓ | | | | | ✓ | |
| topic | VARCHAR(20) | | ✓ | | | | | | '-' |
| supplierid | INT(11) | | ✓ | | | | | | |
| create_time | DATETIME | | ✓ | | | | | | |
| create_user | VARCHAR(20) | | ✓ | | | | | | |
| branchid | TINYINT(4) | | ✓ | | | | | | |
| purchase_date | DATE | | ✓ | | | | | | |
| due_date | DATE | | ✓ | | | | | | |
| staffid | INT(11) | | ✓ | | | | | | |
| note | VARCHAR(45) | | ✓ | | | | | | '-' |

Figure 64 Purchase order table

5) Purchase detail table

Field name: id, SKU, purchaseid, unitprice, quantity, discount, totalprice, note

Primary key: id

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ✓ | ✓ | | | | | ✓ | |
| SKU | INT(11) | | ✓ | | | | | | |
| purchaseid | INT(11) | | ✓ | | | | | | |
| unitprice | DECIMAL(10,2) | | ✓ | | | | | | '0.00' |
| quantity | INT(11) | | ✓ | | | | | | |
| discount | DECIMAL(5,2) | | ✓ | | | | | | |
| totalprice | DECIMAL(12,2) | | ✓ | | | | | | |
| note | VARCHAR(45) | | ✓ | | | | | | '-' |

Figure 65 Purchase detail table

6) Inventory stable

Field name: SKU, branchid, quantity

Primary key: SKU, branchid,

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| SKU | INT(11) | ✓ | ✓ | | | | | | |
| branchid | TINYINT(4) | ✓ | ✓ | | | | | | |
| quantity | INT(11) | | ✓ | | | | | | '0' |

Figure 66 Inventory table

7) Warehousing table

Field name: warehousingid, purchaseid, create_time, create_user, status, delivery_time, delivery_user, note, branchid

Primary key: warehousingid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| warehousingid | INT(11) | ✓ | ✓ | | | | | ✓ | |
| purchaseid | INT(11) | | ✓ | | | | | | |
| create_time | DATETIME | | ✓ | | | | | | |
| create_user | VARCHAR(20) | | ✓ | | | | | | |
| status | ENUM('空单','待入货... | | ✓ | | | | | | |
| delivery_time | DATETIME | | ✓ | | | | | | |
| delivery_user | VARCHAR(20) | | ✓ | | | | | | |
| note | VARCHAR(45) | | ✓ | | | | | | '-' |
| branchid | TINYINT(4) | | ✓ | | | | | | |

Figure 67 Warehousing table

8) Delivery table

Field name: deliveryid, orderid, create_time, create_user, status, delivery_time, delivery_user, note, branchid

Primary key: deliveryid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| deliveryid | INT(11) | ✓ | ✓ | | | | | ✓ | |
| orderid | INT(11) | | ✓ | | | | | | |
| create_time | DATETIME | | ✓ | | | | | | |
| create_user | VARCHAR(20) | | ✓ | | | | | | |
| status | ENUM('空单','待发货... | | ✓ | | | | | | |
| delivery_time | DATETIME | | ✓ | | | | | | |
| delivery_user | VARCHAR(20) | | ✓ | | | | | | |
| note | VARCHAR(45) | | ✓ | | | | | | '-' |
| branchid | TINYINT(4) | | ✓ | | | | | | |

Figure 68 Delivery table

9) Transshipment table

Field name: transshipmentid, topic, note, create_time, create_user, from_branchid, to_branchid, status, from_time, from_user, to_time, to_user

Primary key: transshipmentid

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| transshipmentid | INT(11) | ✓ | ✓ | | | | | ✓ | |
| topic | VARCHAR(20) | | ✓ | | | | | | '-' |
| note | VARCHAR(45) | | ✓ | | | | | | '-' |
| create_time | DATETIME | | ✓ | | | | | | |
| create_user | VARCHAR(20) | | ✓ | | | | | | |
| from_branchid | TINYINT(4) | | ✓ | | | | | | |
| to_branchid | TINYINT(4) | | ✓ | | | | | | |
| status | ENUM('空单','待出库... | | ✓ | | | | | | |
| from_time | DATETIME | | | | | | | | NULL |
| from_user | VARCHAR(20) | | | | | | | | NULL |
| to_time | DATETIME | | | | | | | | NULL |
| to_user | VARCHAR(20) | | | | | | | | NULL |

Figure 69 Transshipment table

10) Transshipment detail table

Field name: id, transshipmentid, SKU, quantity, note

Primary key: id

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☑ | |
| transshipmentid | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| SKU | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| quantity | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | '0' |
| note | VARCHAR(45) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | '-' |

**Figure 70 Transshipment detail table**

### 7.3.5 Financial Accounting

**Database Schema**

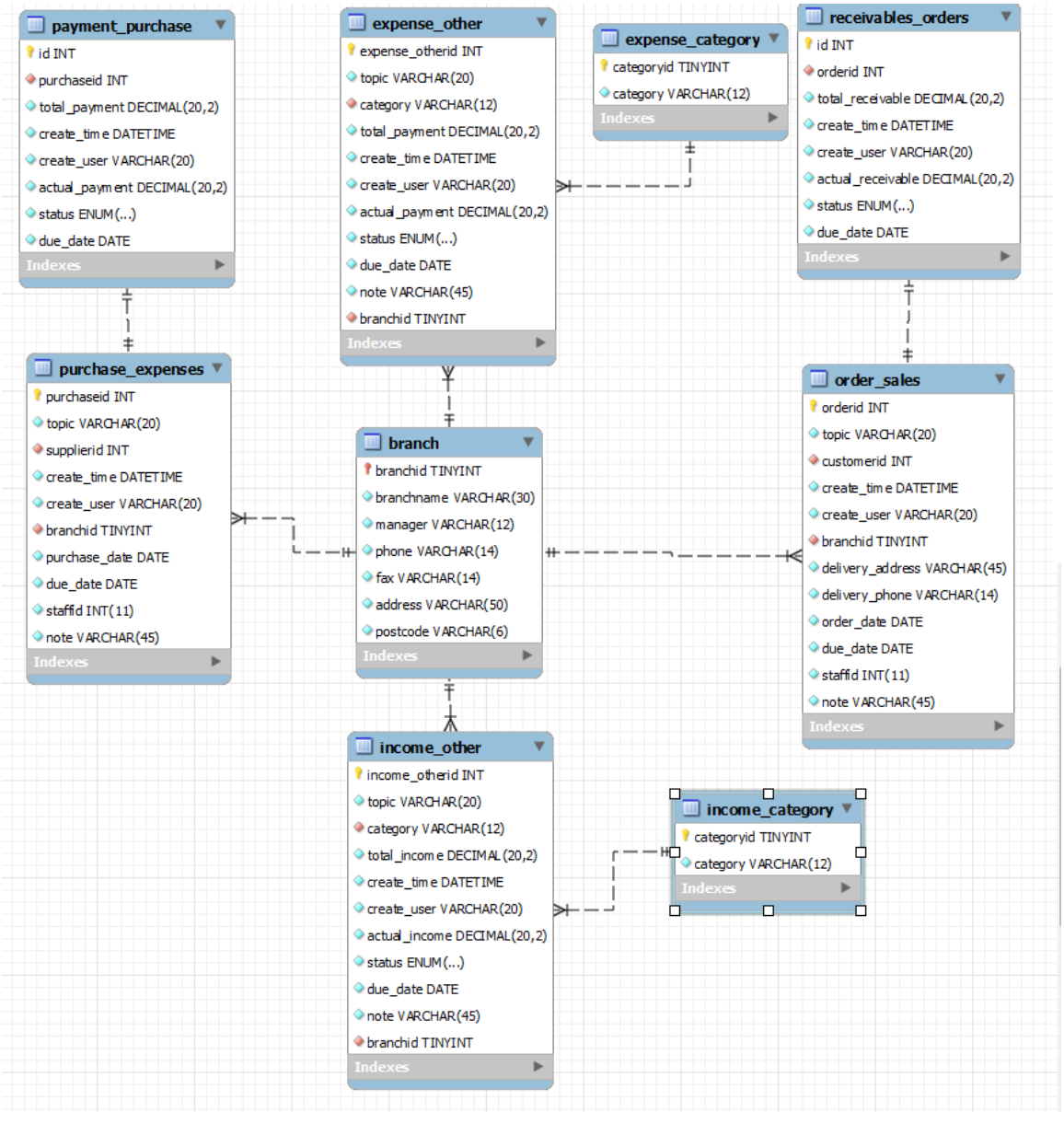



Figure 71 Database Schema - Financial accounting

**Table Design**

1) Sales receivable table

Field name: id, orderid, total_receivable, create_time, create_user, actual_receivable, status, due_date

Primary key: id

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ✓ | ✓ | | | | | ✓ | |
| orderid | INT(11) | | ✓ | | | | | | |
| total_receivable | DECIMAL(20,2) | | ✓ | | | | | | |
| create_time | DATETIME | | ✓ | | | | | | |
| create_user | VARCHAR(20) | | ✓ | | | | | | |
| actual_receivable | DECIMAL(20,2) | | ✓ | | | | | | |
| status | ENUM('未付款','部分... | | ✓ | | | | | | |
| due_date | DATE | | ✓ | | | | | | |

Figure 72 Sales receivable table

2) Other income table

Field name: income_otherid, topic, category, total_income, create_time, create_user, actual_income, status, due_date, note, branchid

Primary key:

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| income_otherid | INT(11) | ✓ | ✓ | | | | | ✓ | |
| topic | VARCHAR(20) | | ✓ | | | | | | |
| category | VARCHAR(12) | | ✓ | | | | | | |
| total_income | DECIMAL(20,2) | | ✓ | | | | | | |
| create_time | DATETIME | | ✓ | | | | | | |
| create_user | VARCHAR(20) | | ✓ | | | | | | |
| actual_income | DECIMAL(20,2) | | ✓ | | | | | | |
| status | ENUM('未付款','部分... | | ✓ | | | | | | |
| due_date | DATE | | ✓ | | | | | | |
| note | VARCHAR(45) | | ✓ | | | | | | '-' |
| branchid | TINYINT(4) | | ✓ | | | | | | |

Figure 73 Other income table

3) Purchase payable table

Field name: id, purchaseid, total_payment, create_time, create_user, actual_payment, status, due_date

Primary key: id

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ✓ | ✓ | | | | | ✓ | |
| purchaseid | INT(11) | | ✓ | | | | | | |
| total_payment | DECIMAL(20,2) | | ✓ | | | | | | |
| create_time | DATETIME | | ✓ | | | | | | |
| create_user | VARCHAR(20) | | ✓ | | | | | | |
| actual_payment | DECIMAL(20,2) | | ✓ | | | | | | |
| status | ENUM('未付款','部分... | | ✓ | | | | | | |
| due_date | DATE | | ✓ | | | | | | |

Figure 74 Purchase payable table

4) Other expense table

Field name: expense_otherid, topic, category, total_payment, create_time, create_user, actual_payment, status, due_date, note, branchid

Primary key:

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | Default |
|---|---|---|---|---|---|---|---|---|---|
| expense_otherid | INT(11) | ✓ | ✓ | | | | | ✓ | |
| topic | VARCHAR(20) | | ✓ | | | | | | |
| category | VARCHAR(12) | | ✓ | | | | | | |
| total_payment | DECIMAL(20,2) | | ✓ | | | | | | |
| create_time | DATETIME | | ✓ | | | | | | |
| create_user | VARCHAR(20) | | ✓ | | | | | | |
| actual_payment | DECIMAL(20,2) | | ✓ | | | | | | |
| status | ENUM('未付款','部分... | | ✓ | | | | | | |
| due_date | DATE | | ✓ | | | | | | |
| note | VARCHAR(45) | | ✓ | | | | | | '-' |
| branchid | TINYINT(4) | | ✓ | | | | | | |

Figure 75 Other expense table

# 8. Development and Implementation

## 8.1 General

a. The following programming language are used for developing the system

- PHP
- JavaScript
- HTML5
- CSS3

b. MySQL is used as the database system

c. ThinkPHP is used as PHP development framework.

d. Zurb Foundation framework is used for developing user interface pages.

e. CanvasJS is used as reporting tool to generate charts.

## 8.2 Login and Logout

a. Validate login information

```
Public function login(){
    if(!IS_POST) E('页面不存在');
    $username = I('username');
    $pwd = I('password', '', 'md5');
    $user = M('ggm_user')->where(array('username' => $username))->find();
    if (!user || $user['password'] !=$pwd) {
        $this->error('账号或密码错误');
    }
    if($user['status']) $this->error('用户被锁定');
    $data = array(
        'uid' =>$user['uid'],
        'logintime' => time(),
        'loginip' => get_client_ip()
    );
    M('ggm_user')->save($data);
```

System users submit username and password from login page to login controller by post. Password is encrypted by using MD5. If function cannot find username in ggm_user table or password doesn't much the submitted password, error message will be returned. If the user's status is not 1 which means locked, the message 'user is locked' will be returned.

After successfully login, user's login time and login IP will be updated.

b. Create session

```
M('ggm_user')->save($data);
session(C('USER_AUTH_KEY'), $user['uid']);
session('branchid', $user['branchid']);
session('username', $user['username']);
session('logintime', date('Y-m-d H:i:s',$user['logintime']));
session('loginip', $user['loginip']);
```

Session data will be added into session table by invoking session function. Session data contains user id, branch id, username, login time, and login IP.

c. Load interface

```
    RBAC::saveAccessList();
    $this->redirect('Home/Index/index');
}
```

d. Logout

```
public function logout(){
    session_unset();
    session_destroy();
    $this->redirect('Home/Login/Index');
}
```

## 8.3 Authorization Validation

```
Class CommonController extends Controller{
    public function _initialize(){
        if(!isset($_SESSION[C('USER_AUTH_KEY')])){
            $this->redirect('home/login/index');
        }

        $notAuth = in_array(MODULE_NAME, explode(',', C('NOT_AUTH_MODULE'))) ||
            in_array(ACTION_NAME, explode(',', C('NOT_AUTH_ACTION')));

        if (C('USER_AUTH_ON')) {
            Rbac::AccessDecision() || $this->error('没有权限');
        }
    }
}
```

By using AccessDecision () function in RBAC class package in ThinkPHP, _initialize function will check access table to validate if the user is authorized to invoke certain functions.

```
Class ProductController extends CommonController{
    public function search(){
        $where1['parentid'] = "0";
        $this->pcategory = M('product_category')->wher
```

_initialize function is under common controller. Other task controllers need to extend this common controller in order to use this function before users execute specific functions under task controllers.

## 8.4 Add, Modify, Query, Delete

Functions about add, modify, query and delete will be explained in general by using employee information management as an example.

**Add**

```
public function add(){
    $this->department = M('department')->select();
    $this->staff_type = M('staff_type')->select();
    $this->branchid = I('session.branchid');
    $this->display();
}
```

Add() function is used to prepare data and load add page for users to fill in information. In this case, department information and staff type information are loaded for users to select in order to complete the process of adding new employee information.

```
<div class="row">
  <div class="medium-3 columns">
    <label>所属部门</label>
    <select name="departmentid">
      <option value="-">Select Department</option>
      <foreach name='department' item='d'>
            <option value="{$d['departmentid']}">{$d.name}</option>
      </foreach>
    </select>
  </div>
  <div class="medium-3 columns">
    <label>员工类型</label>
    <select name="staff_typeid">
      <option value="-">Select Staff Type</option>
      <foreach name='staff_type' item='st'>
            <option value="{$st['staff_typeid']}">{$st.name}</option>
      </foreach>
    </select>
  </div>
  <div class="medium-3 columns end">
    <label>雇佣日期</label>
    <input name="edate" type="text" placeholder="YYYY-MM-DD" class="datepicker"></input>
  </div>
</div>
```

<foreach> is ThinkPHP's internal label for listing each row in data array. The function is as same as foreach() function in PHP. While loading pages, <foreach> will be translated to PHP. In this case, the purpose is to set all department names and all staff types as each option for users to select. When users submit this form to controller, departmentid and staff_typeid are submitted instead of department name and staff type.

```
Public function addhandle(){
    IF (!IS_POST) E('页面不存在');
    if (M('hrms_staff')->add($_POST)) {
        $this->success('添加成功',U('Home/staff/index'));
    }else{
        $this->error('添加失败');
    }
}
```

Addhandle() function is to handle posts submitted by users from add page. Success() and error() are used to send message or redirect users to certain pages.

```
<body>
  <form data-abide method="post" action="{:U('Home/Staff/addhandle')}">
  <ul class="tabs" data-tab>
```

Action inside <form> label determines where data is submitted to.

**Modify**

Unlike add () function, modify () function need a parameter or a group of parameters to determine which specific data need to be modified.

```
<a href="{:U('Home/Staff/edit', array('staffid' => $v['staffid']))}"><i
class="fa fa-edit"></i></a>
```

In this case, one staffid is submitted to edit () function under StaffController.

```
public function edit(){
    $this->department = M('department')->select();
    $this->staff_type = M('staff_type')->select();
    $where['staffid'] = I('staffid');
    $data = M('staff_view')->where($where)->find();
    $this->assign('staff',$data);
    $this->display();
}
```

Compare to add () function, edit () not only prepares department information and staff type information but also get staff information in staff table by staffid.

```
public function edit_handle(){
    IF (!IS_POST) E('页面不存在');
        $hrms_staff = M('staff');
        $staffid = I('staffid');
        $where['staffid'] = $staffid;
        if ($hrms_staff->where($where)->setField($_POST)){
            $this->success('修改成功',U('Home/staff/edit',array('staffid' => $staffid)));
        }
        else{
            $this->error('修改失败');
        }
```

Edit_handle () function is used to handle data submitted from modify page. Function setField() is to save only part of the submitted data which is changed instead of saving all data into database.

**Query**

Search conditions and search values can be submitted by users to search () function.

```
public function search(){

    $branchid = I('session.branchid');
    if($branchid!=1){
        $where['branchid'] = $branchid;
    }

    $condition1 = I('condition1');
    $value1 = I('value1');

    if($condition1!=null && $value1!=null){
        $where[$condition1] = $value1;
    }
    $condition2 = I('condition2');
    $value2 = I('value2');
    if($condition2!=null AND $value2!=null){
        $where[$condition2] = $value2;
    }
    $condition3 = 'gender';
    $gender = I('gender');
    if($gender!=null){
        $where[$condition3] = $gender;
    }
```

Condition columns such as gender can be predefined depending on different tasks. Conditions columns such as staffid, ID and staff name can be selected depending on which condition users want to use to search.

```
$staff_view = M('staff_view');
$data = $staff_view->where($where);

$row = I('row');
$count = $data->count();
$page = new \Think\Page($count,$row);
$limit = $page->firstRow . ',' . $page->listRows;
$page->setConfig('theme','
    <span class="TOTAL_ROW">共 %TOTAL_ROW% 条记录</span>
    %UP_PAGE%<span class="TOTAL_PAGE">%NOW_PAGE% / %TOTAL_PAGE%</span>
    %DOWN_PAGE% %FIRST% %LINK_PAGE% %END%');
$data = $staff_view->where($where)->limit($limit)->select();
$this->assign('page',$page->show());

$this->assign('staff_view',$data);
$this->display('index');
```

Data matches search conditions will be chosen from a table or a model. Page () function is used to control number of data rows can be displayed on each page. $row can be defined by users based on the number of data rows they are comfortable to deal with on each page. Then chosen data will be assigned to certain page.

**Delete**

```
public function del(){
    if (I('staffid')==null){
        $getid = $_REQUEST['id'];
        $getids = implode(',', $getid);
        if(!$getid){
            $this->error('未选择任何记录');
        }else{
            $staffid = is_array($getid) ? $getids : $getid;
        }
    }else{
        $staffid = I('staffid');
    }
    $hrms_staff = M('hrms_staff');
    if (M('hrms_staff')->delete($staffid)){
        $this->success('删除成功',U('Home/staff/index'));
    }
    else {
        $this->error('删除失败');
    }
}
```

Del () function can be used to delete either one or multiple rows of data based on the number of data records are selected by users.

## 8.5 Multiple Row Insert

```
<div class="medium-6 columns">
    <select name="SKU[]">
        <optgroup label="选择[产品]">
            <option value="">选择[产品]</option>
        </optgroup>
        <foreach name='category' item='c'>
        <optgroup label="{$c.category}">
            <foreach name='stock' item='s'>
            <if condition="$s['category'] eq $c['category']">
            <option value="{$s['SKU']}">{$s.name} |&nbsp型号:{$s
            .sn} |&nbsp库存:{$s.quantity} |&nbsp单价:{$s.
            unitprice}</option>
            </if>
            </foreach>
        </optgroup>
        </foreach>
    </select>
</div>
<div class="medium-2 columns">
    <label for="promotion" class="inline">促销计划:</label>
</div>
<div class="medium-3 columns">
    <select name="promotion[]">
        <option value="">选择[促销计划]</option>
        <foreach name="promotion" item="p">
        <option value="{$p.name}">{$p.name}</option>
        </foreach>
    </select>
```

In order to support multiple data row insert, data need to be submitted as arrays. In this case, SKU, promotion and other variables are ended with '[]'.

```
<script type="text/javascript">
  $(function(){
      $('#add-input').click(function(){
          var obj = $(this).parents('div#select-input').clone();
          obj.find('#add-input').remove();
          $('#last').before(obj);
      });
  });
</script>
```

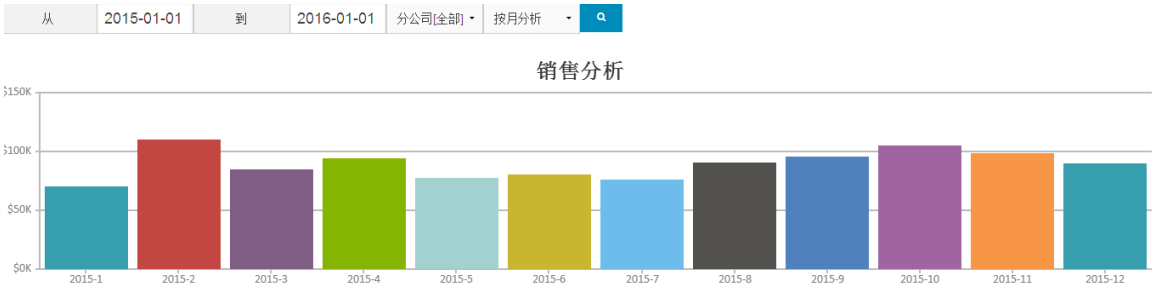
This function is used to copy and paste part of html code in order to add extra space for users to fill in.

```
public function add_order_detail_handle(){
    $data = array();
    $length = count($_POST['SKU']);
    for($i=0;$i<$length;$i++){
        $data[$i]['SKU'] = $_POST['SKU'][$i];
        $data[$i]['promotion'] = $_POST['promotion'][$i];
        $data[$i]['unitprice'] = $_POST['unitprice'][$i];
        $data[$i]['quantity'] = $_POST['quantity'][$i];
        $data[$i]['note'] = $_POST['note'][$i];
        $data[$i]['discount'] = $_POST['discount'][$i];
        $data[$i]['totalprice'] = $_POST['totalprice'][$i];
        $data[$i]['orderid'] = $_POST['orderid'][$i];
    }

    if(M('order_detail')->addall($data)){
        $this->success('添加成功',U('Home/Sales/order'));
    }
    else{
        $this->error('添加失败');
    }
}
```

After data is submitted by users, a loop is used to arrange data in order to insert data one row at a time.

## 8.6 Reports and Charts




This is an example of a branch's monthly sales report chart. In this case, account book is already selected by user which is sales report. In order to generate this report, users need to select time period, branch and methods to calculate data. Depending on selected methods, sales report can be daily, weekly, monthly, quarterly and yearly.

```
<script type="text/javascript">
  window.onload = function () {
    var chart = new CanvasJS.Chart("chartContainer", {
      title:{
        text: "销售分析"
      },
      data: [//array of dataSeries
        { //dataSeries object

         /*** Change type "column" to "bar", "area", "line" or "pie"***/
        type: "column",
        name: "First Quarter",
        dataPoints: [
        <?php
          foreach ($branch_sales as $k => $v) {
          echo '{ label: "';
          echo $v[$condition1];
          echo '", y:';
          echo $v['total_income']/1000;
          echo '},';
          }
        ?>
        ]
       }
       ],
        axisY:{
          prefix: "$",
          suffix: "K"
        }
     });

    chart.render();
  }
```

After calculated data is loaded, data will be assigned to inside CanvasJS's chart function through PHP code to generate charts.

## 9. Test Cases

In this section, some of the test cases are demonstrated as examples of testing process. One or two test cases per use case are selected based on the importance and diversity of tests.

The test cases are set up mainly based on the applied functions in each user cases. Boundary tests are implemented in order to analyze the boundary value of input. Extreme values are used for boundary tests.

In the testing process, I invited four actual system users to help me to test the system based on their daily tasks. Boundary tests are implemented and analyzed by myself.

The test list is as the following table, * indicates the test cases which are selected as examples.

| Test No. | Testing | Expected Outcome |
|---|---|---|
| | **Login** | |
| ***1** | login with valid username and password | Redirected to home page |
| ***2** | Login with incorrect username or password | An error message |
| ***3** | Access other pages without login | Redirected to login page |
| | **Recruiting** | |
| ***4** | Add candidate information | Data added |
| ***5** | Boundary test for | Boundary value of input boxes analyzed |
| **6** | Modify recruiting plan and candidate information | Data updated |
| **7** | Delete recruiting plan and candidate information | Data deleted |
| **8** | Hire candidate | Data transferred to staff basic information |
| | **Employee information management** | |
| **9** | Add staff information including basic information, contact, dependent, and salary information | Data added |
| **10** | Boundary test for adding staff information | Boundary value of input boxes analyzed |
| ***11** | Modify staff information | Data updated |
| ***12** | Boundary test for modifying staff information | Boundary value of input boxes analyzed |
| **13** | Job transfer | Staff information updated, job transfer record added |
| ***14** | Add reward and punishment | Data added |
| **15** | Boundary test for adding reward and punishment form | Boundary value of input boxes analyzed |
| **16** | Modify reward and punishment | Data updated |
| **17** | Delete reward and punishment | Data deleted |

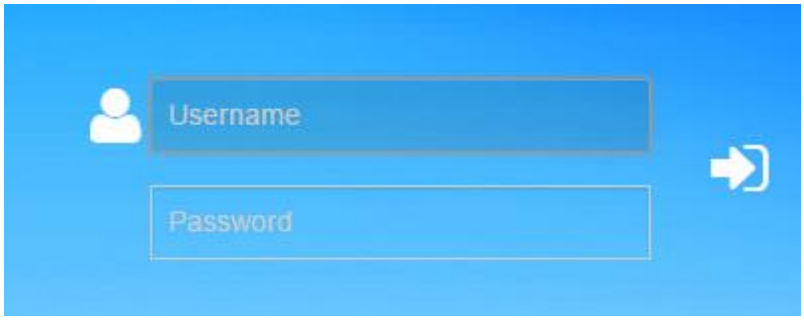
| | | |
|---|---|---|
| **18** | Search for staff information, and reward and punishment | Conditions working, results match searching values |
| | **Payroll** | |
| ***19** | Add payroll | Data added, calculated correctly |
| ***20** | Boundary test for adding payroll | Boundary value of input boxes analyzed |
| **21** | Delete payroll | Data deleted |
| **22** | Search for payroll information | Conditions working, results match searching values |
| | **Customer information management** | |
| ***23** | Add customer basic information and contact information | Data added |
| ***24** | Boundary test for adding customer information form | Boundary value of input boxes analyzed |
| **25** | Modify customer information including basic and contact information | Data updated |
| ***26** | Search for customer information | Conditions working, results match searching values |
| ***27** | Boundary test for searching for customer information | Boundary value of input boxes analyzed |
| | **Sales management** | |
| ***28** | Add sales order and order details | Data added |
| ***29** | Boundary test for adding sales order and order details | Boundary value of input boxes analyzed |
| **30** | Modify sales order and order details | Data updated |
| **31** | Delete sales order including order details | Data deleted |
| | **Purchase management** | |
| **32** | Add supplier basic information and contact information | Data added |
| **33** | Modify supplier information including basic and contact information | Data updated |
| **34** | Delete supplier information including basic and contact information | Data deleted |
| **35** | Add product category | Data added |
| ***36** | Add product | Data added |
| ***37** | Boundary test for adding product | Boundary value of input boxes analyzed |
| **38** | Modify product information | Data updated |
| **39** | Delete product information | Data deleted |
| **40** | Add purchase order and order details | Data added |
| **41** | Boundary test for adding purchase order and order details | Boundary value of input boxes analyzed |
| **42** | Modify purchase order and order details | Data updated |
| **43** | Delete purchase order including order details | Data deleted |
| | **Inventory management** | |
| **44** | Confirm warehousing records | Stock data updated, warehousing |

| | | |
|---|---|---|
| | | data updated |
| ***45** | Confirm delivery records | Stock data updated, delivery data updated |
| **46** | Add transshipment | Stock data updated, transshipment data added |
| **47** | Boundary test for adding transshipment | Boundary value of input boxes analyzed |
| ***48** | Search for transshipment details | Conditions working, results match searching values |
| **Payables management** | | |
| ***49** | Add other expense | Data added |
| ***50** | Boundary test for adding other expense | Boundary value of input boxes analyzed |
| **51** | Add payment for both purchase expense and other expense | Data updated |
| **52** | Modify other expense | Data updated |
| **53** | Search for purchase expense and other expense | Conditions working, results match searching values |
| **54** | Delete both purchase expense and other expense | Data deleted |
| **Receivables management** | | |
| **55** | Add other income | Data added |
| **56** | Boundary test for adding other income | Boundary value of input boxes analyzed |
| ***57** | Add payment for both purchase income and other income | Data updated |
| ***58** | Boundary test for adding payment | Boundary value of input boxes analyzed |
| **59** | Search for purchase income and other income | Conditions working, results match searching values |
| **60** | Delete both purchase income and other income | Data deleted |
| **Management Accounting** | | |
| ***61** | Generate charts | Methods working, data calculated correctly, charts represented correctly |

Most expected outcome of the tests are required page redirecting and success message; therefore, they will not be specifically included in the table above.

**Test 1 Login with valid username and password**

System users should be able to login in the system with username and password.

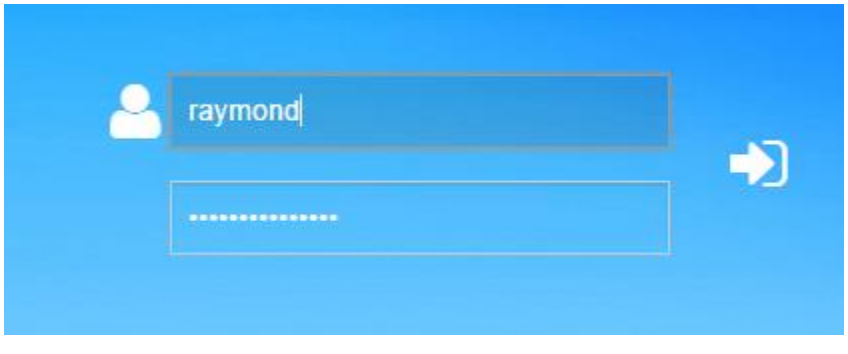



Input correct username and password

If the username and password are correct, the system will redirect user to home page of the system.

The words in red on the screenshot above are the translation for the side navigation.

**Test 2 Login with wrong password or wrong username**




If the username or wrong word does not match the record in the user table of RBAC system, wrong message will be sent to user.

The error message is shown as the screenshot above.

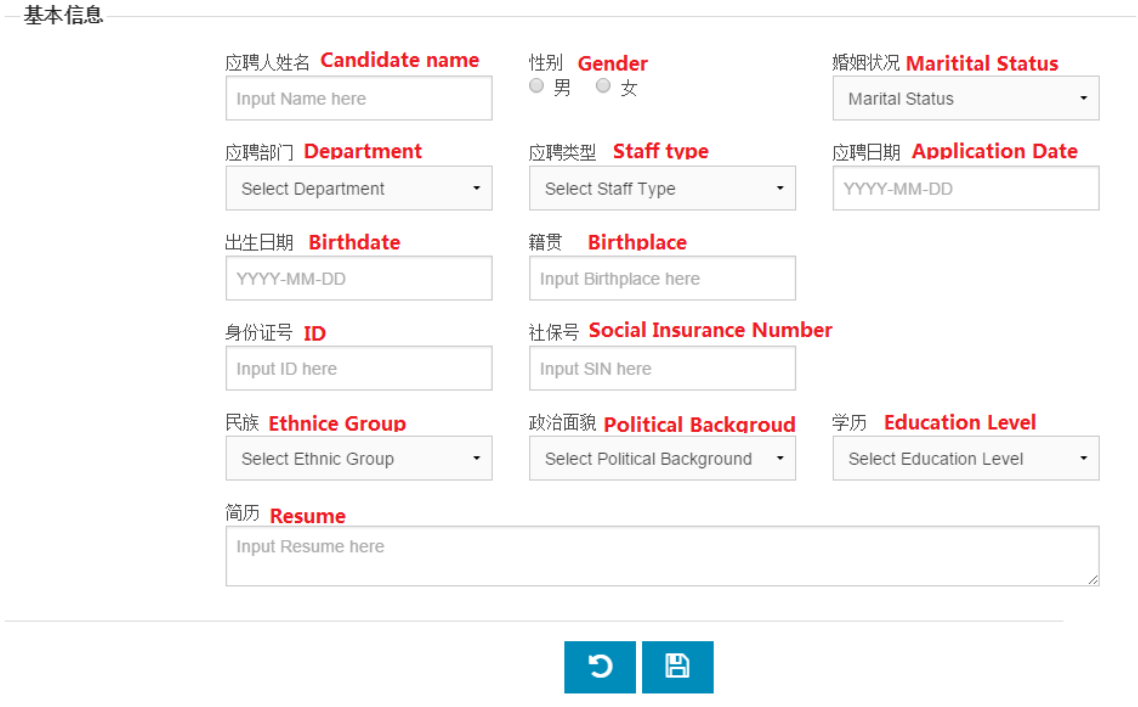
The system will deny access if the input username or password is incorrect.

**Test 3 Access without login**

If unauthorized user want to access home page or any other pages without login, they will be redirected to login page. Unauthorized users are not able to login the system without login.

### Test 4 Adding Candidate Information

Users should be able to input candidate information by filling in the form below.




Once the candidate information is filled correctly, user will be able to click button to submit this form.

The success message is shown below after submitting the form.

**Test 5 Boundary Test**

Several boundary tests are implemented for the input boxes. One of the boundary tests is shown below as an example.

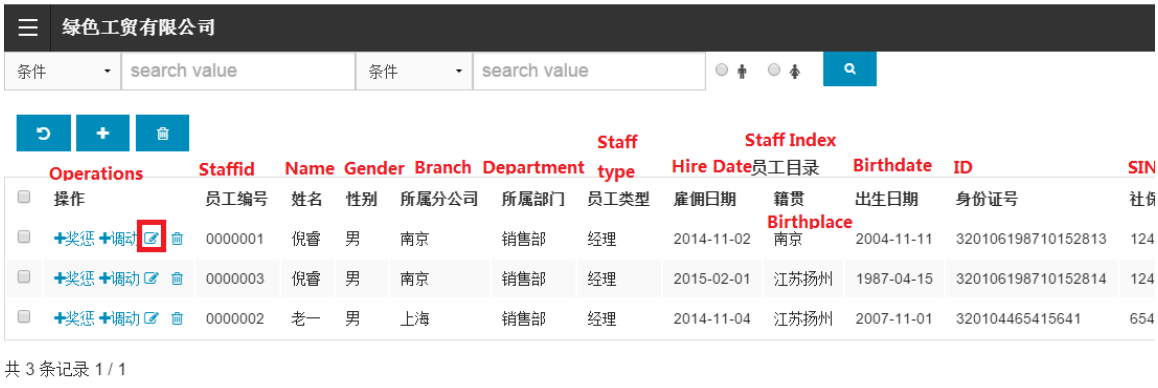
The ID number in China is 18 digits. If the input ID number in the form is not valid, an error message will be shown below the input box as the screenshot below.

Boundary test analysis is as following,

- Candidate name is required and less than 24 characters, which is 12 Chinese letters.
- Gender is required.
- Marital status is required.
- Department is required.
- Staff type is required.
- Birthdate and Application date are required, and the date format can only be "YYYY-MM-DD" as 2015-02-01.
- ID number is required and exact 18 digits.
- SIN is not required, but the SIN is no more than 18 digits.
- Resume can be less than 1024 characters.

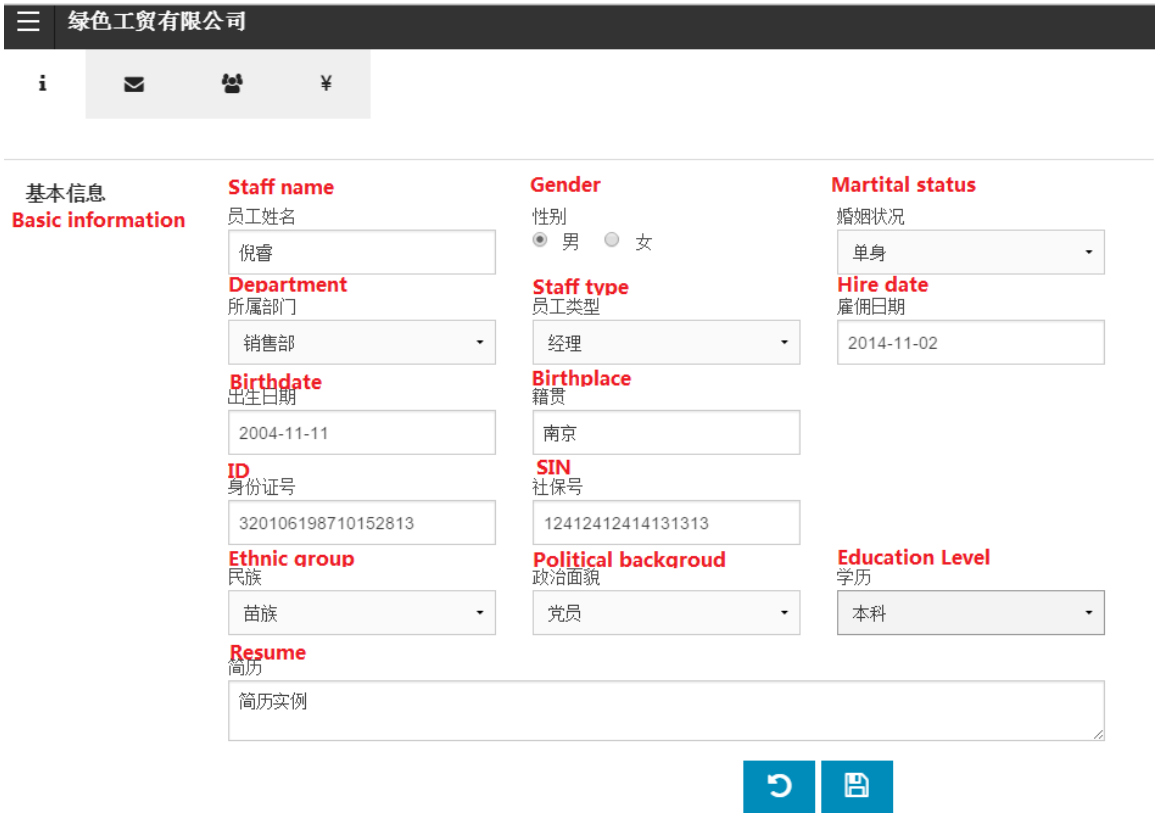
Two test cases of employee information management are selected as example.

**Test 11 Modify Employee Information**



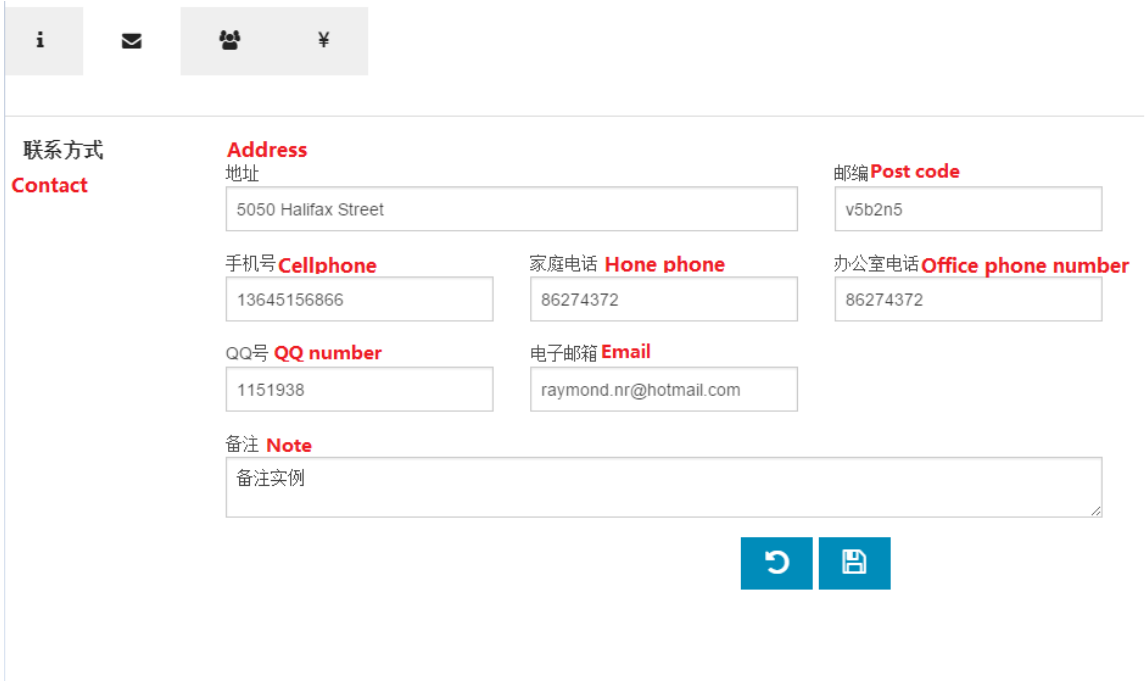

Users should be able to access staff information modify page by clicking the button in front of each row.

The modify page will show the current information of the employee users want to modify.

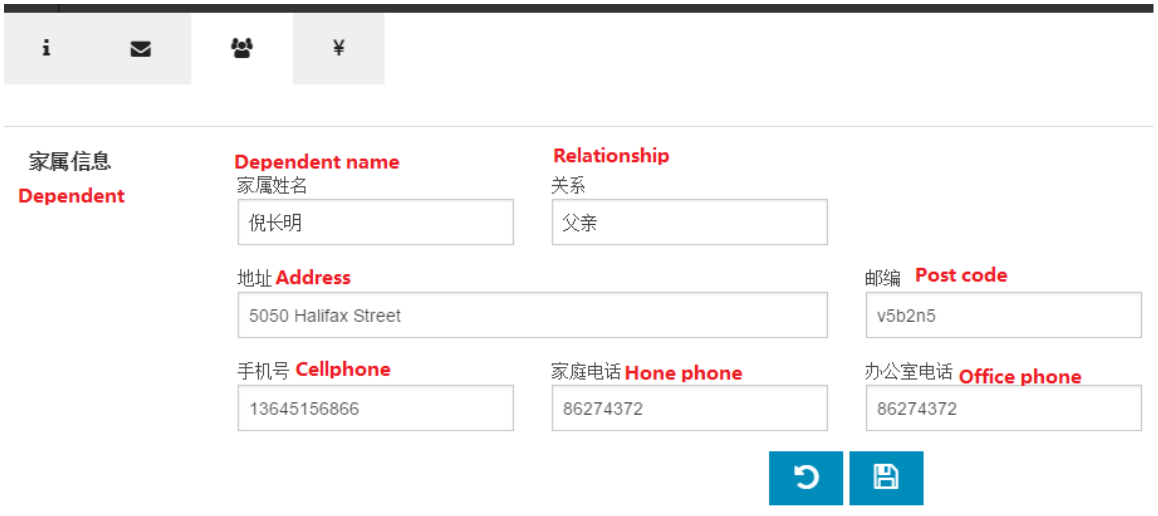



Users should also be able to access the employee's contact information, and dependent information and salary information by click the tabs above the form.

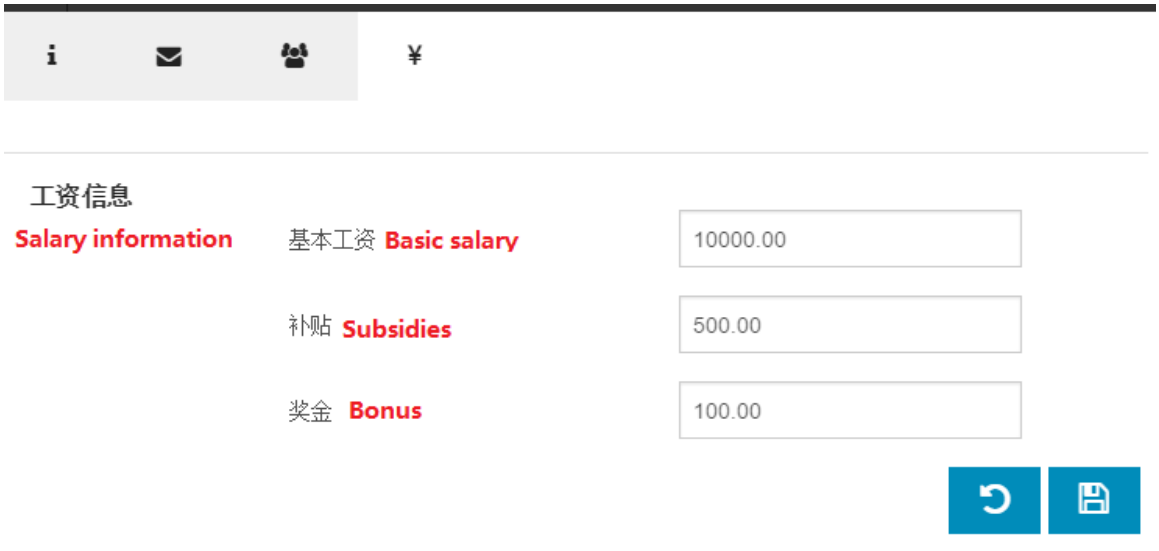
The screenshot of contact information is shown below,




The screenshot of dependent information is shown below,




The screenshot of salary information is shown below,




After modifying employee information, users are able to submit the changes by clicking the button , and success message will be sent to users as the screenshot below.

**Test 12 Boundary Test for Modify Employee Information**

Due to that the basic information of employee is as same as candidate information in recruiting, boundary tests are for the input boxes in contact information, dependent information, and salary information.

Email input box test is shown as an example of the boundary tests.

If the email format is not correct, an error message will be shown below the input box as below.

Boundary test analysis is as following,

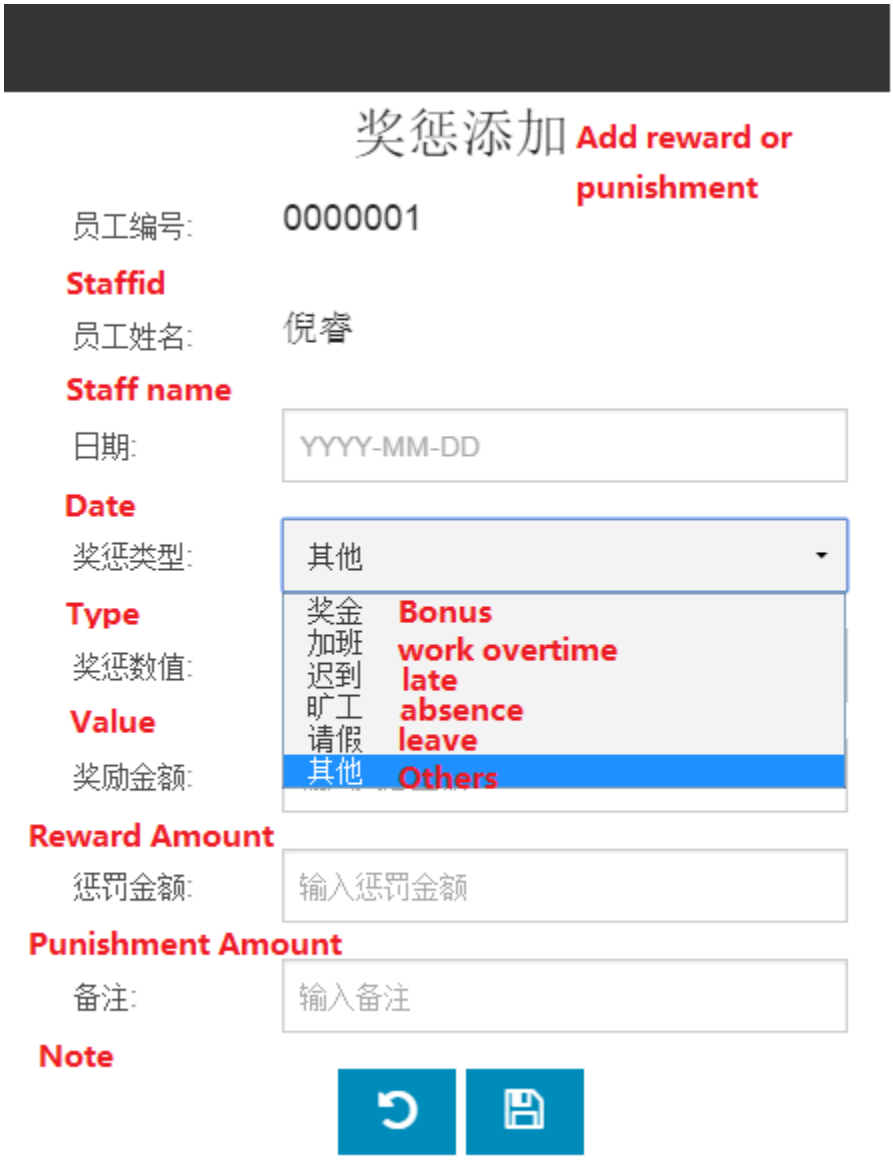
- Address is allowed less than 50 characters.
- Post code can only be 6 digits.

- Cellphone number, home phone number and officer phone number can be less than 14 characters, and only dash and integers are allowed.
- QQ number can be less than 20 characters.
- Email can less than 320 characters, and only email format as *@*.* is allowed.
- Note can be less than 1024 characters.
- Dependent name can be less than 24 characters.
- Relationship can be less than 10 characters.
- Basic salary, subsidies and bonus can be 0.00 to 999999.99.

**Test 14 Add Reward and Punishment Record**

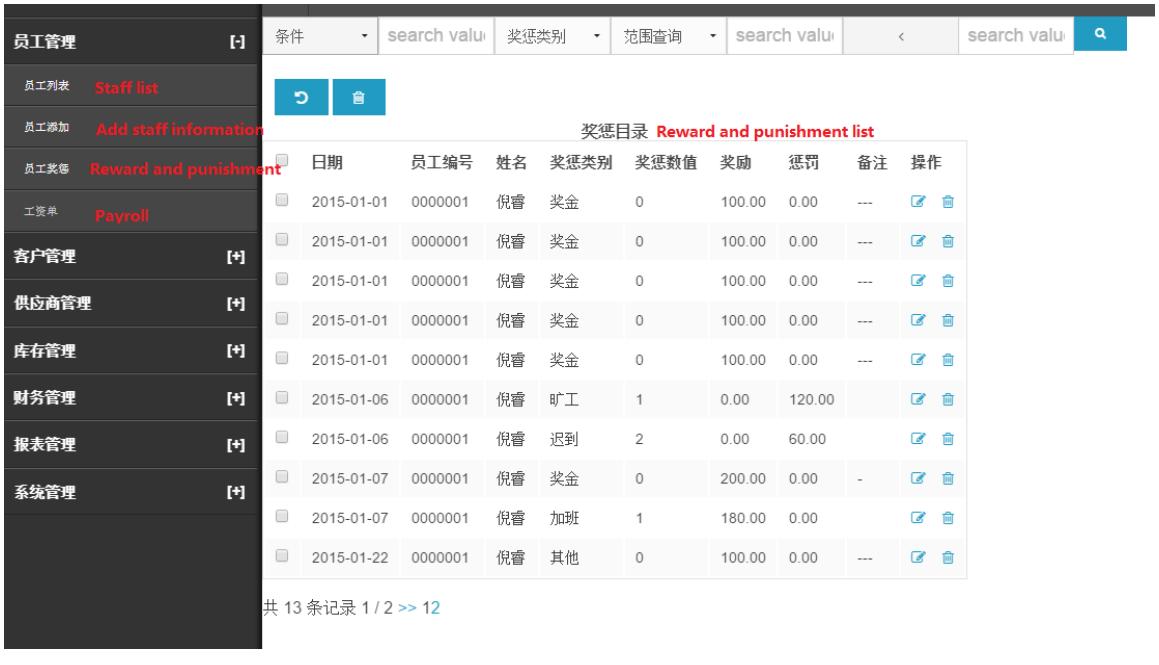
Users should be able to add reward and punishment record for an employee by clicking the button +奖惩, which means "+ reward or punishment" in front of each row to access add reward and punishment page.

The add reward and punishment page should be able to show staffid and staff name of the selected employee as the screenshot below.




Users should be able to input reward and punishment information by filling in the form above, and then click the button to submit a reward and punishment record for the selected employee.

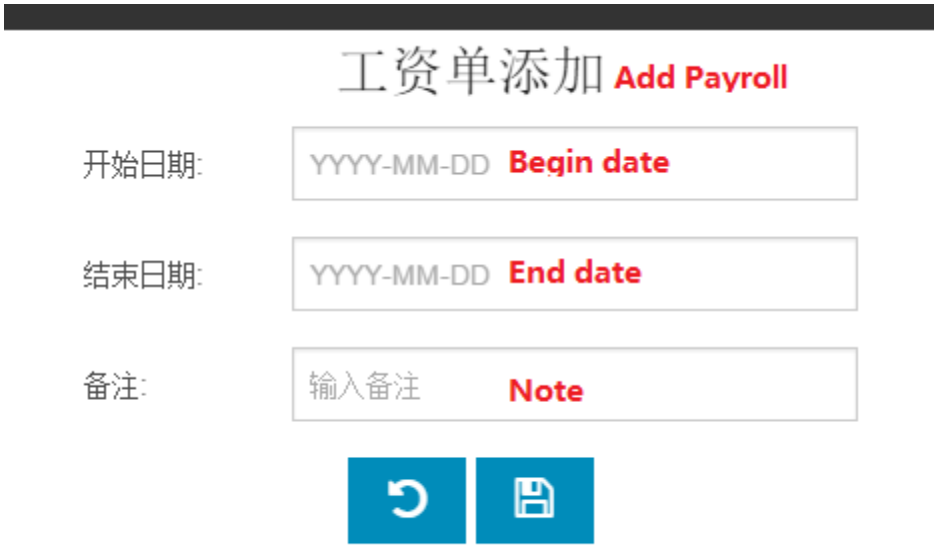
A success message will be sent to users if the reward and punishment record is added successfully. Then the page will be automatically redirected to reward and punishment list.



**Test 19 Add payroll**

Users should be able to access payroll list in order to invoke the add payroll function. The payroll list is as the screenshot below.

By clicking the button above the payroll list, users should be able to access the add payroll page.




Users need to set begin date and end date first in order to summarize total reward and punishment.

In order to complete the process, at least one employee needs to be selected by check the check box in front of each employee's information.

After the process is completed, payroll records will be added into payroll list as the screenshot below.

### Test 20 Boundary Test for adding payroll

As same as the other boundary tests above, the boundary tests for the input boxes during the process are implemented.

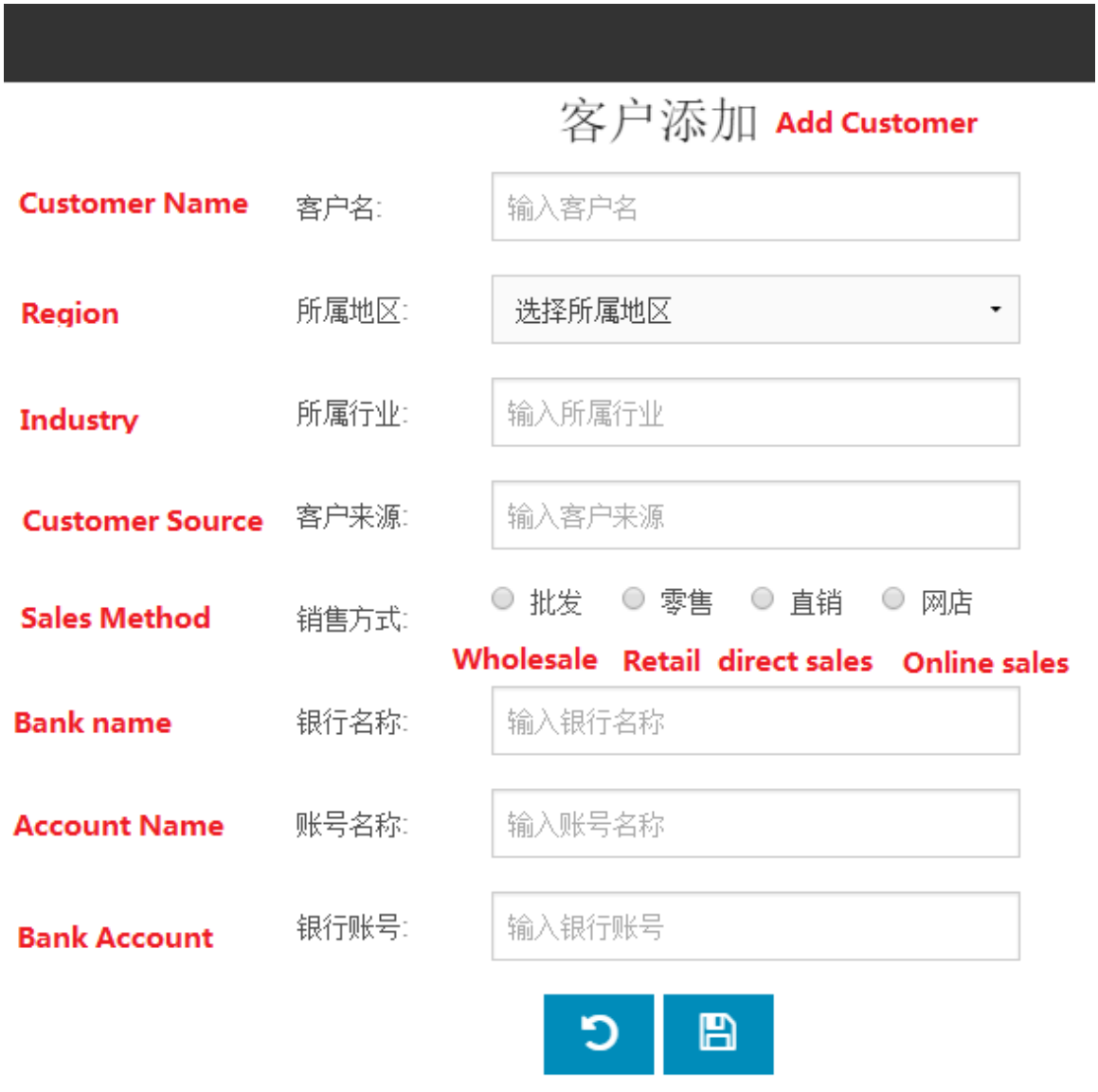
Boundary test analysis is as following,

- The date format of begin date and end date can be only “YYYY-MM-DD” as “2015-01-01”.
- Note can be less than 20 characters.
- At least one employee record needs to be selected.

### Test 23 Add Customer Information

In order to add a customer information record, users should be able to access customer list first. The customer list is as the screenshot below

By clicking the button above the customer list, users should be able to access add customer page as the screenshot below.



After the form is filled, users should be able to click the button to submit in order to complete the process. If the customer information is added successfully, a success message will be shown to users and page will be redirected to customer list as the following screenshot.

**Test 24 Boundary Test for Adding Customer Information**

Boundary tests are implemented for the input boxes during the process. The boundary test for bank account is demonstrated as an example.

If the bank account is empty, an error message will be shown to users. In this case, the error message is “Bank account is required”.

Boundary Test Analysis is as following,

- All information in input boxes during the process is required.
- Customer name can be less than 40 characters.
- Industry can be less than 10 characters.

- Customer source, bank name, account name, and bank account can be less than 20 characters.
- One of the sales methods must be selected.

### Test 26 Search for Customer Information

Users should be able to use the search bar above the customer list to search for specific customer or a group of customer by filling in search conditions.

### No condition

If there is no condition selected or search value filled, all of the customer will be shown by clicking the searching button .

The search results are as the following screenshot.

### One condidtion

If there is one condition selected, the system should be able to show only the customers match this condition. The search result is as the following screenshots.

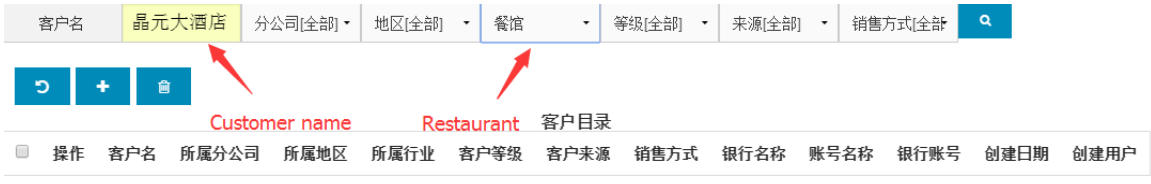
The result above is shown by using industy as an condition and hotel as search value.

The result above is shown by using industy as a condition and restaurant as search value.

**Multible conditions**

If there is more than one condition, the search result should match all selected conditions. The search results are as following screenshots.

The result above is shown by using two conditions: customer name is “晶元大酒店” which is the hotel’s name and industry is “hotel”.




The result above is shown by using two conditions: customer name is “晶元大酒店” and industry is “restaurant”. There is no customer record shown because there is no record matches both condition at same time.

**Test 27 Boundary Test for Searching Customer Information**

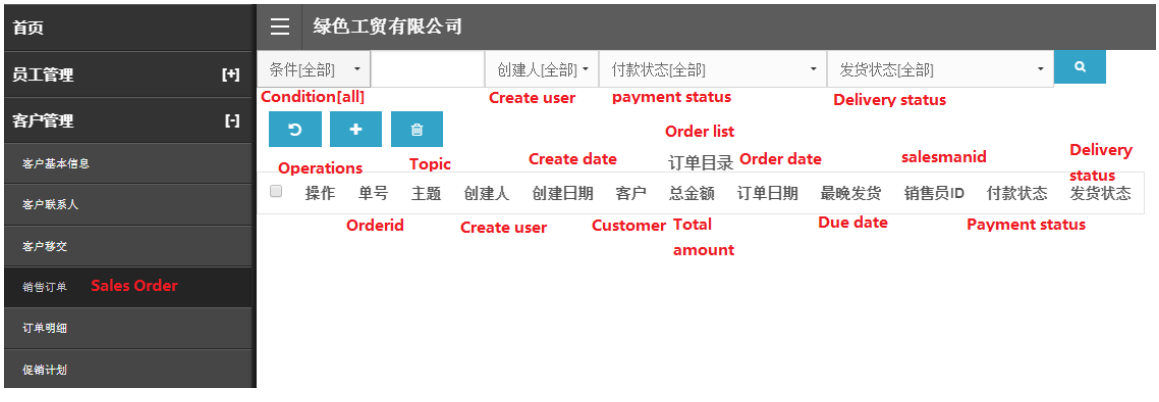
The boundary tests for the selectable conditions in the search bar are implemented. The search values under these selectable conditions can only be shown if there is existing value in at least one customer record. The industry condition is shown as an example below.

Boundary test analysis is as following,

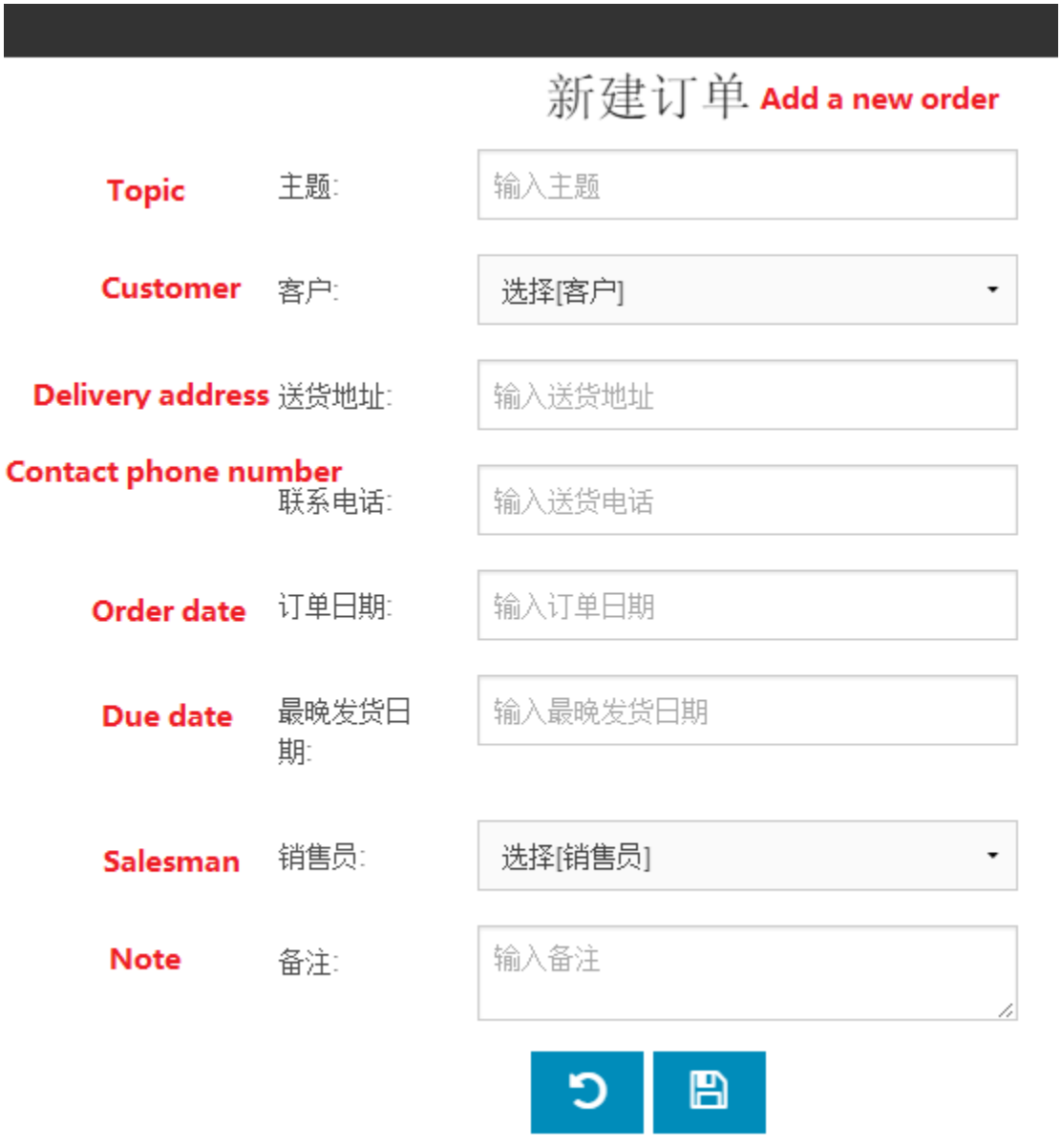
- All search values under the selectable conditions can be shown only if they exist in at least one customer record.
- At least one condition needs to be selected to search for certain customer or customer group; otherwise, all the customers will be shown.
- All conditions can be used at same time to search for customer information.

**Test 28 Adding Order Record**

In order to add a sales order record, users should be able to access order list page first. The order list page is as the following screenshot.




By clicking the button above the order list, users should be able to access add order page as the following screenshot.




After filling the order information, users should be able to submit this form by clicking the button . An success message will be shown to users if the order is added successfully, and the page will be automatically redirected to order list as following screenshot.

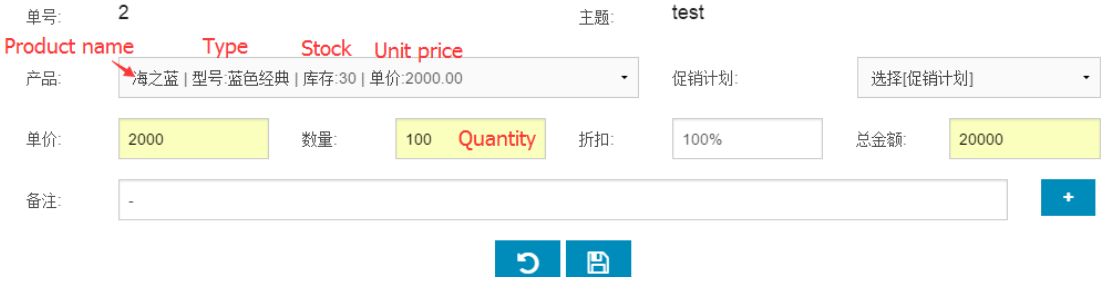
In order to complete the order, users need to add order details by clicking the link 录入明细, which means add order details, in front of each order record. The add order details page is as following screenshot.

Users should also be able to add more fields for adding more than one order detail record at same time by clicking the button . The page after adding two more fields is as the following screenshot.

After the order details are filled, user should be able to click the button to submit the form. If the order details are added successfully, a success message will be shown, and the page will be redirected to order list.

**Test 29 Boundary Test**

Boundary tests for the input boxes during the process are implemented. The input boxes of product and quantity in order details page are demonstrated as an example.



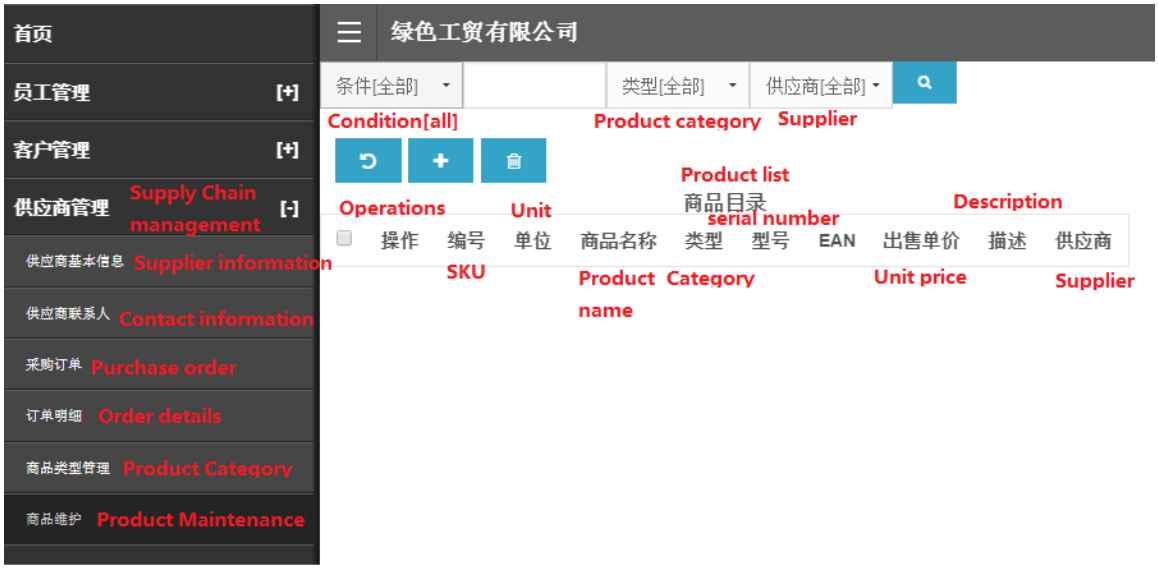

If the stock quantity of the selected product is less than the order quantity, an error message will be shown if users try to submit the form. In this case, the stock quantity of selected product is only 30, but the ordered quantity is 100. The error message is shown as the following screenshot.

The page will be redirected to back to add order detail page.

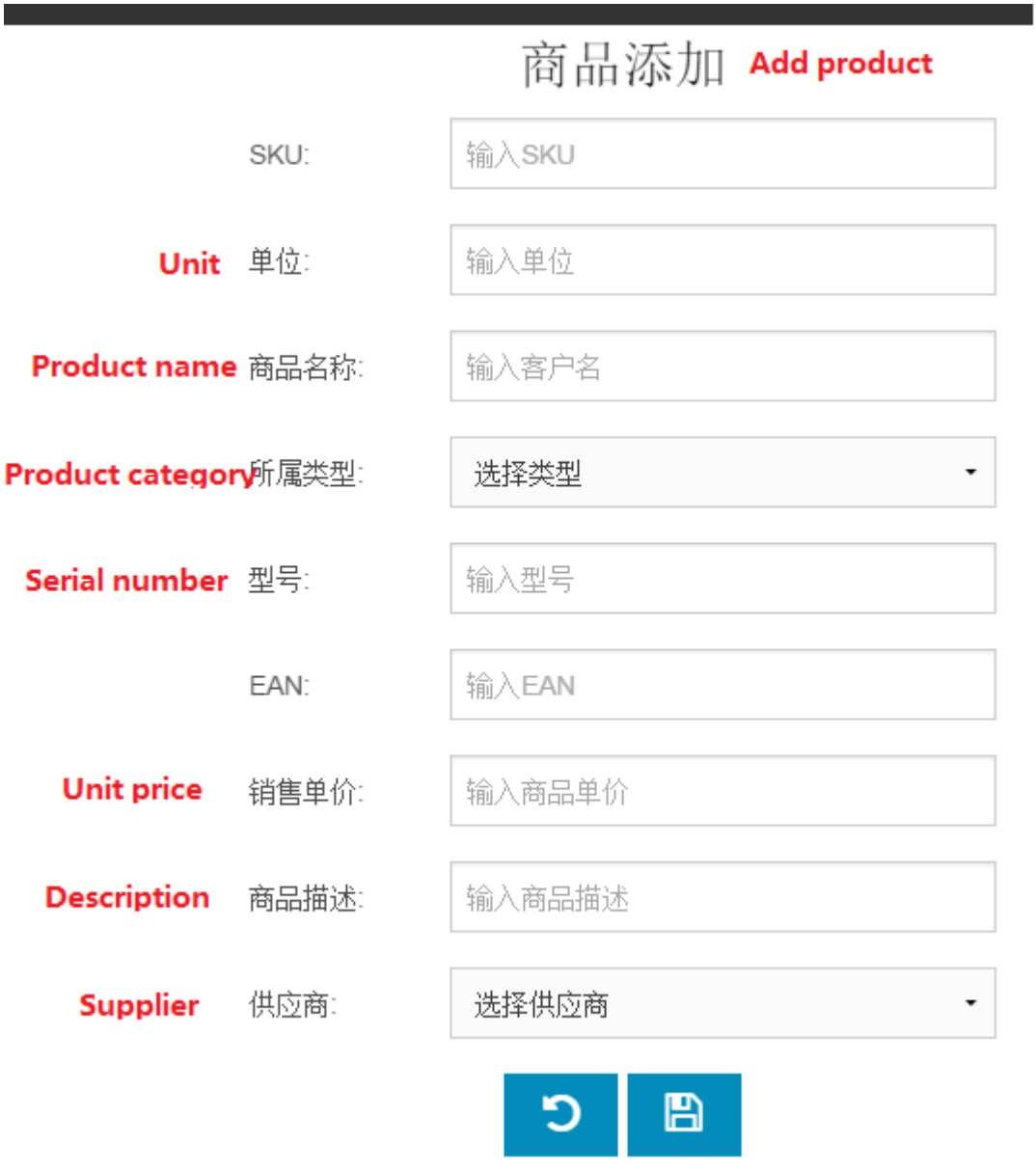
- All information in input boxes are required except notes.
- Boundary test analysis is as following,
- Topic in sales order can be less than 20 characters.
- Delivery address in sales order can be less than 45 characters.
- Contact phone in sales order can be less than 14 characters.
- The format of order date and due date in sales order can be only “YYYY-MM-DD”.
- Note in sales order can be less than 45 characters.
- Unit price in order detail can be from 0.00 to 99999999.99.
- Quantity in order detail can be less than stock quantity of selected product.
- Discount in order detail can be from 0.00 to 999.99.
- Note in order detail can be less than 45 characters.

**Test 36 Add Product Information**

In order to add a new product, users should be able to access product maintenance page first. The product maintenance page is as the following screenshot.




By clicking the button above the product list, users should be able to access add product page as the following screenshot.




After the product information is filled, users should be able to click the button to submit the form. If the product information is added successfully, a success message will be shown to users, and the page will be redirected to product list.

**Boundary Test for adding product**

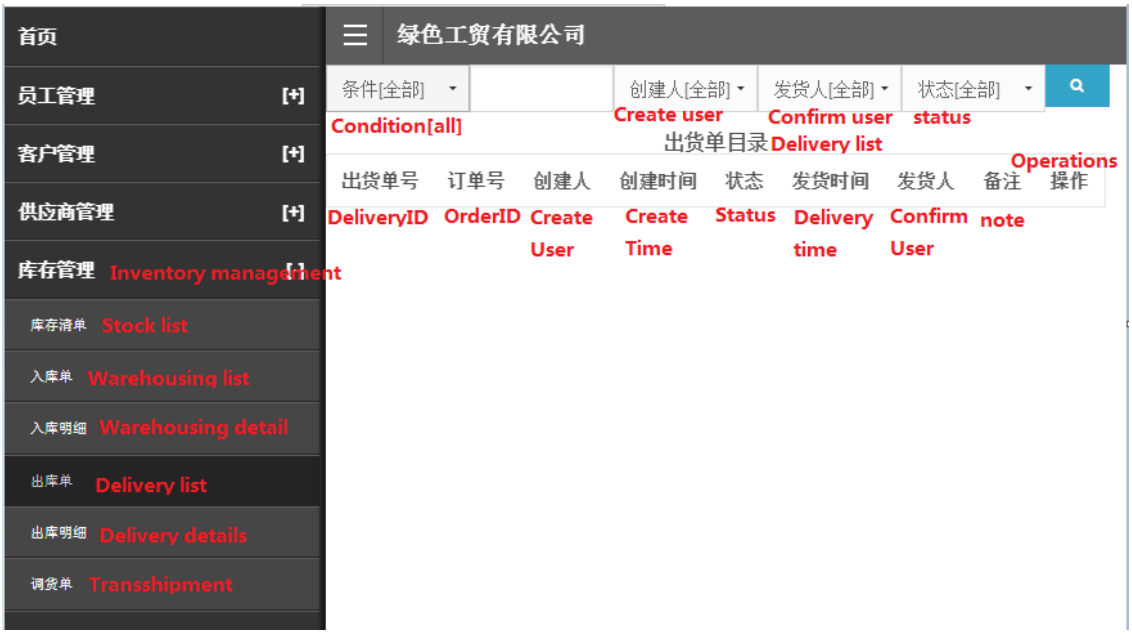
The boundary tests for all the input boxes during the process are implemented.

The boundary test analysis is as following,

- Only serial number and description are not required.
- The SKU number can be between 1 and 99999999.
- Unit can be less than 4 characters.
- Product name can be less than 20 characters.
- Serial number can be less than 12 characters.
- EAN can only 13-digit integer.
- Unit price can be between 0.00 and 99999999.99.
- Description can be less than 45 characters.

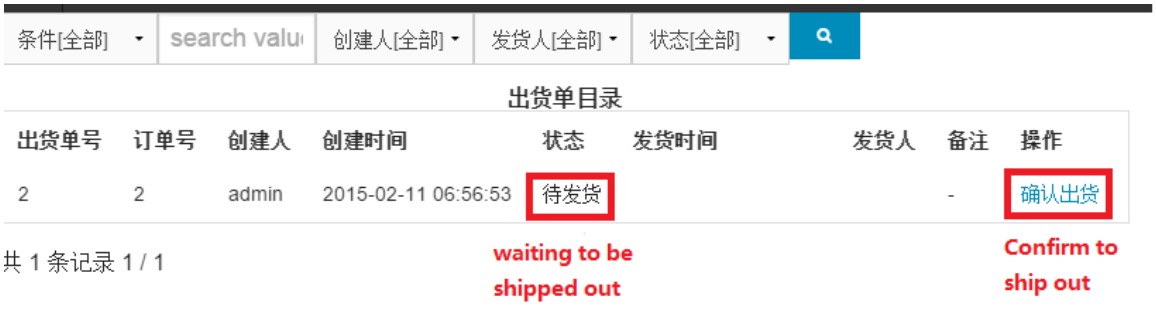
**Test 45 Confirm Delivery**

In order to confirm a delivery record, users should be able to access delivery list page as the following screenshot.




There are two steps in this process: confirm to ship out and confirm delivered.

Step1: Confirm to ship out




When the delivery status shows “waiting to be shipped out”, it means that the ordered products are ready. Users should be able to see and click the link 确认出货 to confirm that the products are already shipped out from warehouse. Once the the link is clicked, an success message will be shown, and the the stock quantity of the ordered products will be updated by minus the ordered quantity of the products in order details.

Screenshot of Stock quantity before shipping out is shown below

Screenshot of stock quantity after shipping out is shown below

Step2: Confirm delivered

After the link is clicked, the delivery status is changed to “shipped out”, which means that the products are on the way to customer. The time of shipping out and confirm users are updated too as the following screenshot.

After the products are delivered, users are able to click the link 完成送货 , which means delivered confirm, in order to complete the delivery process. An success message will be shown to users as the following screenshot.

Then the page will be automatically redirected to delivery list page as the following screenshot

### Test 48 Search for Transshipment details

In order to search for transshipment details, users should be able to access transshipment list page first. The page is as the following screenshot.

User should be able to use the conditions in search bar to search for specific transshipment records. The conditions are as the following screenshot.

As same as the tests in Test 27 Search for Customer Information, three tests are implemented including no selected condition, one selected condition and multiple selected conditions.

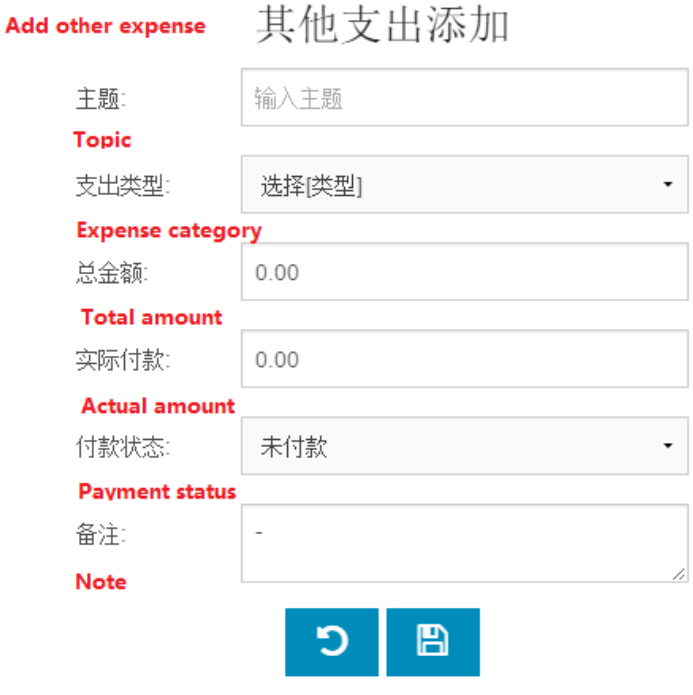
In order to check the details of each transshipment record, users need to click the button in front of each record.

The transshipment details will be shown as the following screenshot.

**Test 49 Add Other Expense Record**

In order to add other expense records, users should be able to access other expense list first. The page of other expense is as the following screenshot.

By clicking the button above other expense list, users should be able to access add other expense page. The page is as the following screenshot.



After filling in the expense record, users should be able to click the button below the form to submit the new record. A success message will be shown to users if the record is added successfully, and the page will be redirected to other expense list.

**Test 50 Boundary Test for adding other expense**

The boundary tests for the input boxes during the process are implemented.

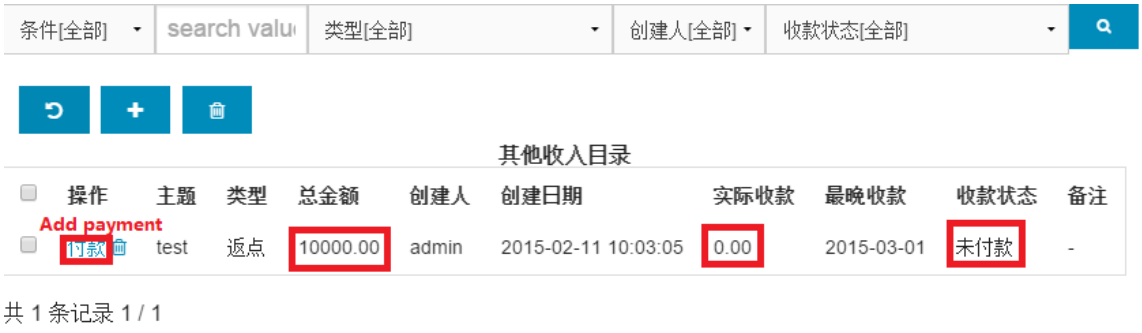
The boundary test analysis is as following,

- Only actual payment and note are not required.
- Topic can be less than 20 characters.
- Total amount and actual payment can be between 0.00 and 99999999.99.
- Note can be less than 45 characters.

**Test 57 Add Payment**

Sometimes customers make payment by instalments. In order to keep tracking the payment, users need to add several payments to complete an order receivable record.

In order to add a payment, users should be able to access receivables list first. Here other income list is used as an example. The page of other income is shown as the following screenshot.




By clicking the link 付款, which means add payment, in front of each record, users should be access add payment page as the following screenshot.

After filling in the amount of this payment, users should be able to click to submit. A success message will be shown and the page will be redirected to other income page as the following screenshot.

The actual payment is updated by adding the payment. The payment status is also changed to “部分欠款” which means arrears.

Once all debts are paid, the status will be automatically changed to “paid” as the following screenshot.

**Test 58 Boundary Test for adding payment**

The only boundary test during the process is for the payment amount input box.

The boundary test is as following

- The payment amount can be between 0.01 and 99999999.99.
- However, if the summary of actual receivable and payment amount is greater than total amount, an error message will be sent to users.

**Test 61 Generates Charts**

Sales report is demonstrated as an example in management accounting.

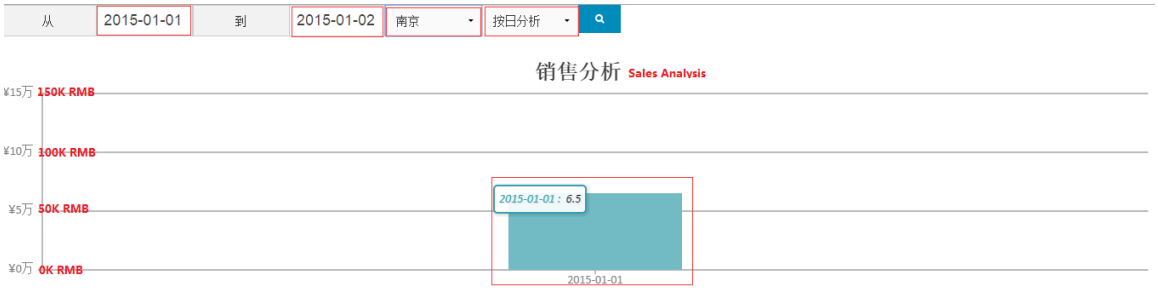
In order to generate sales report, users should be able to access the sales report page first as the following screenshot.

Users need to fill in the conditions in order to generate the sales report. The conditions are beginning date, end date, branch, and methods. There are four methods including daily, weekly, monthly and quarterly analysis.

Some test data is added into database for testing purpose.

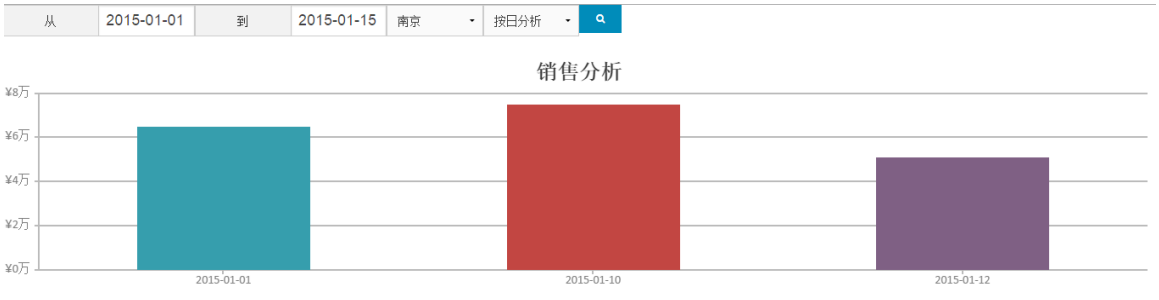
Step 1: the system should be able to summarize sales amount between selected beginning date and selected end date. Therefore daily analysis is selected as an example.

The daily analysis result between 2015-01-01 and 2015-01-02 is shown as the screenshot below.




There is only one summary of sales amount which is 2015-01-01.

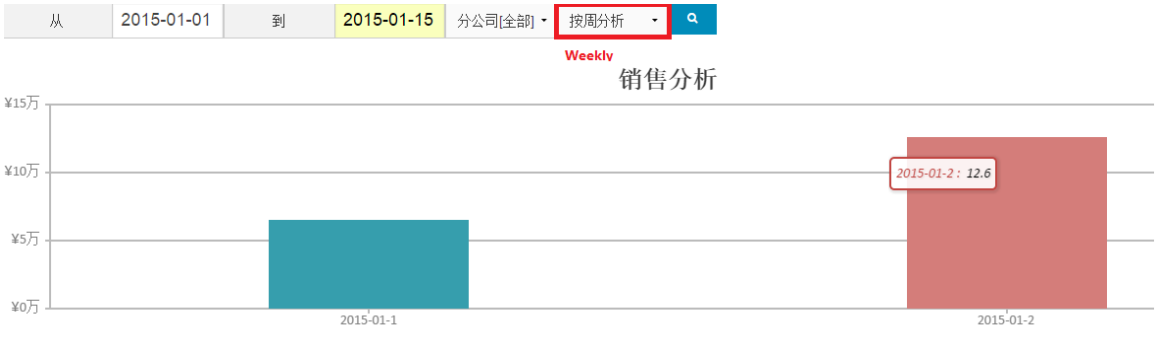
Then end date is changed to 2015-01-15, and the result is shown below.




There are three summaries in this chart because these three dates are in selected time of period and only these are the only dates with sales records in selected time of period.

Step 2: the system should be able to summarize sale amount by day, week, month, and quarter.

The test result of weekly analysis is shown below



There are two summaries in this chart: 2015-01-1 means the first week in Jan, 2015. Sales amount of 2015-01-10 is 7.5 and 2015-01-12 is 5.1. There is no record between 2015-01-13 and 2015-01-14; therefore, the summary of week 2 is correct, which is 12.6.

The similar tests are implemented to test monthly and quarterly analysis method, and the results are shown correctly based on data inserted for testing purpose.

# 10. Research

## 10.1 Optimizing SQL Statement

In "Effective MySQL: Optimizing SQL Statement", Ronald Bradford (2011) states that "optimizing SQL statements is a key component of improving performance and scalability."[5]

### 10.1.1 Use of Indexes

The most important part in "Effective MySQL: Optimizing SQL Statement" is the use of indexes. In Chapter 3 "Understanding MySQL index", the author states that creating the right indexes is one of the most significant techniques for SQL performance tuning.[6]

There are several approaches introduced in Chapter 1 "The Five Minute DBA" including

- Confirm slow query

  Confirm slow query is the step to verify the slow performance of a certain query is not a unique instance that might occurs due to other factors such as a system bottleneck or locking rather than the query itself.

  The certain query need to be executed and timed repeatedly in order to determine the response time of a query. If the query is not a SELECT statement, the query needs to be rewritten as a SELECT statement for verification purpose.

- Identify a missing index

  Identifying a missing index is an important process of optimizing SQL statement based on the existing structure and the size of the table that the slow query is related with.

  The purpose of this approach is to find alternative solutions to apply in order to optimize this SQL statement.

- Apply a new index

  There are many factors to be considered to add a new index and to deploy it accordingly. This is a very important part of choosing the correct approach to optimizing the slow query.

- Verify new index

  After a new index is applied, the optimized SQL statement needs to be executed to confirm the improvement of the new statement and the effectiveness of the new index.

---

[5] Ronald Bradford. (2011) *Effective MySQL: Optimizing SQL Statements*. Introduction, page 1. The United States. MacGraw-Hill Companies, Inc.
[6] Ronald Bradford. (2011) *Effective MySQL: Optimizing SQL Statements*. Chapter 3, page 29. The United States. MacGraw-Hill Companies, Inc.

**Research Problems**

The purposes of the research is to answer the following questions

- If applying a new index is effective for improving a certain slow query?
- How much improvement can be provided by applying a new index?
- How exactly to improve database performance by applying new index in this project or future enhancement?

In order to answer these questions, several experiments are conducted by using the approaches above. Test group and control group are used to determine the differences before and after applying a new index.

**Experiments**

Staff table in this project is chosen as an example for this experiment. There are 1,000 rows of example data in staff table for experiment purpose. The staff table is created as below

```
CREATE TABLE `staff` (
  `staffid` int(7) unsigned zerofill NOT NULL AUTO_INCREMENT,
  `name` varchar(24) COLLATE utf8_unicode_ci NOT NULL,
  `gender` enum('男','女') COLLATE utf8_unicode_ci NOT NULL,
  `bplace` varchar(45) COLLATE utf8_unicode_ci NOT NULL,
  `bdate` date NOT NULL,
  `sin` varchar(19) COLLATE utf8_unicode_ci NOT NULL,
  `ethnic_group` enum('-','汉族','壮族','回族','满族','维吾尔族','苗族','彝族','土家族','藏族',
                      '蒙古族','朝鲜族','其他') COLLATE utf8_unicode_ci NOT NULL,
  `political_bg` enum('-','党员','团员','群众') COLLATE utf8_unicode_ci NOT NULL,
  `education` enum('-','小学','初中','中专','高中','大专','本科','硕士','博士','博士后')
              COLLATE utf8_unicode_ci NOT NULL,
  `marital_status` enum('-','单身','已婚','离异') COLLATE utf8_unicode_ci NOT NULL,
  `edate` date NOT NULL,
  `resume` text COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`staffid`)
) ENGINE=InnoDB AUTO_INCREMENT=0 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

The query need to be optimized as an example is

```
Select * from staff where education ='';
```

This is a query to find staff information based on their education level.

1) Confirm slow query

By using SHOW PROFILES command, execution time of the query can be calculated.

```
SET @@profiling = 0;
SET @@profiling_history_size = 0;
SET @@profiling_history_size = 100;
SET @@profiling = 1;
select * from staff where education = '大专';
show profiles;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| Query_ID | Duration | Query |
|---|---|---|
| 52 | 0.00005700 | SHOW WARNINGS |
| 53 | 0.00171875 | select * from staff where education = '大专' |

Education = “大专” means that the education level is diaploma.

In this case, the execution time is 0.00171875 seconds which equals to 1.71875 milliseconds (ms).

The query is executed for 12 times. The shortest and longest duration are excluded, and the other 10 duration records are selected for calculating average execution time of the query.

| | | |
|---|---|---|
| 53 | 0.00171875 | select * from staff where education = '大专' |
| 54 | 0.00193875 | select * from staff where education = '大专' |
| 55 | 0.00161300 | select * from staff where education = '大专' |
| 56 | 0.00178950 | select * from staff where education = '大专' |
| 57 | 0.00183925 | select * from staff where education = '大专' |
| 58 | 0.00234225 | select * from staff where education = '大专' |
| 59 | 0.00185575 | select * from staff where education = '大专' |
| 60 | 0.00164350 | select * from staff where education = '大专' |
| 61 | 0.00171025 | select * from staff where education = '大专' |
| 62 | 0.00182875 | select * from staff where education = '大专' |
| 63 | 0.00165425 | select * from staff where education = '大专' |
| 64 | 0.00177600 | select * from staff where education = '大专' |

Therefore, query id no.55 and no.58 are excluded. The average execution time of this query is 1.77547ms.

2) Identify missing indexes

```
1 • explain select * from staff where education = '本科';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | staff | ALL | NULL | NULL | NULL | NULL | 1000 | Using where |

Education = "本科" means that the education level is bachelor degree.

According to Ronald Bradford (2011), "the EXPLAIN command is the essential tool for determining the Query Execution Plan (QEP) of the SQL statement. This command can offer insight to the MySQL cost-based optimizer and provide details about access strategies the optimizer might have considered and what strategy the optimizer is expected to choose when running the SQL statement."[7]

In this result, there is no possible key which means there is no index in this table at all. Key refers to the indexes used when executing this query, which is also none in this result. There are 1,000 rows processed for this query.

3) Applying a new index

Based on the results above, in order to improve the performance of this query is to add an index on education column in staff table. Staff_test table is created as a control group for adding an index in instead of staff table as below

```
CREATE TABLE staff_test LIKE staff;

INSERT INTO staff_test SELECT * FROM staff;

ALTER TABLE staff_test ADD INDEX education_index(education);
```

4) Verify new index

```
1 • explain select * from staff_test where education = '大专';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | staff_test | ref | education_index | education_index | 1 | const | 3 | Using index condition |

[7] Ronald Bradford. (2011) *Effective MySQL: Optimizing SQL Statements*. Chapter 2, page 12. The United States. MacGraw-Hill Companies, Inc.

The QEP result of the query after adding an index on education indicates that education_index is the only index in this table, and it is used when executing the query. Key_len refers to key length. It is data length of the index, which is 1 byte in this case. There are only 3 rows of data are processed which is much less compare to the number of rows processed before adding an index on education column.

This is an improvement from the number of rows processed. However, in order to confirm the effectiveness, the average execution time of the query after adding an index needs to be calculated.

| | | |
|---|---|---|
| 95 | 0.00047000 | select * from staff_test where education = '大专' |
| 96 | 0.00031425 | select * from staff_test where education = '大专' |
| 97 | 0.00036850 | select * from staff_test where education = '大专' |
| 98 | 0.00032600 | select * from staff_test where education = '大专' |
| 99 | 0.00039150 | select * from staff_test where education = '大专' |
| 100 | 0.00031375 | select * from staff_test where education = '大专' |
| 101 | 0.00030825 | select * from staff_test where education = '大专' |
| 102 | 0.00037075 | select * from staff_test where education = '大专' |
| 103 | 0.00031200 | select * from staff_test where education = '大专' |
| 104 | 0.00032250 | select * from staff_test where education = '大专' |
| 105 | 0.00031350 | select * from staff_test where education = '大专' |
| 106 | 0.00033775 | select * from staff_test where education = '大专' |

This is the test results of the query's execution time after adding an index. The shortest and longest duration are excluded; therefore, the average execution time of the query is 0.337ms. Compared to the average execution time before adding an index on education column which is 1.77547ms, the current average execution time is only about 19.0% of the original. Therefore, the query is improved about 81%, and the average execution time is reduced about 1.44ms per 1000 records.

5) Other implication

However, there are other implications for adding an index. In Chapter "Creating MySQL Indexes", the author Ronald Bradford (2011) explains three aspects of impacts for adding indexes including DML impact, DDL impact and disk space impact. In this research, two impacts related are experimented. DDL impact is due to that the increased number of data records will cost more time of adding an index.

a) DML Impact

According to Ronald Bradford (2011), adding indexes to a table affects the performance of writes.

| | | |
|---|---|---|
| 109 | 0.00373050 | INSERT INTO `test`.`staff` (... |
| 110 | 0.00207400 | INSERT INTO `test`.`staff` (... |
| 111 | 0.00150825 | INSERT INTO `test`.`staff` (... |
| 112 | 0.00160500 | INSERT INTO `test`.`staff` (... |
| 113 | 0.00197525 | INSERT INTO `test`.`staff` (... |
| 114 | 0.00228975 | INSERT INTO `test`.`staff` (... |
| 115 | 0.00199475 | INSERT INTO `test`.`staff` (... |
| 116 | 0.00094800 | INSERT INTO `test`.`staff` (... |
| 117 | 0.00214700 | INSERT INTO `test`.`staff` (... |
| 118 | 0.00216075 | INSERT INTO `test`.`staff` (... |
| 119 | 0.00194000 | INSERT INTO `test`.`staff` (... |
| 120 | 0.00173475 | INSERT INTO `test`.`staff` (... |

The timed result of inserting without indexes as above, and the average execution time of insert SQL statement is 1.943ms

| Query_ID | Duration | Query |
|---|---|---|
| 136 | 0.00272300 | INSERT INTO `test`.`staff_test... |
| 137 | 0.00213475 | INSERT INTO `test`.`staff_test... |
| 138 | 0.00286625 | INSERT INTO `test`.`staff_test... |
| 139 | 0.00255600 | INSERT INTO `test`.`staff_test... |
| 140 | 0.00235200 | INSERT INTO `test`.`staff_test... |
| 141 | 0.00265825 | INSERT INTO `test`.`staff_test... |
| 142 | 0.00153850 | INSERT INTO `test`.`staff_test... |
| 143 | 0.00310975 | INSERT INTO `test`.`staff_test... |
| 144 | 0.00155075 | INSERT INTO `test`.`staff_test... |
| 145 | 0.00263250 | INSERT INTO `test`.`staff_test... |
| 146 | 0.00216475 | INSERT INTO `test`.`staff_test... |
| 147 | 0.01635275 | INSERT INTO `test`.`staff_test... |

The timed result of inserting with one index as above, and the average execution time of insert SQL statement is 2.475ms.

Therefore, the average execution time of insert SQL statement increased 0.532ms after adding an index.

b) Disk Space Impact

```
SHOW TABLE STATUS FROM test WHERE name = 'staff' OR name=
'staff_test';
```

Test
**test.staff**

**Table Details**

| | |
|---|---|
| Engine: | **InnoDB** |
| Row format: | **Compact** |
| Column count: | **12** |
| Table rows: | **1000** |
| AVG row length: | **114** |
| Data length: | **112.0 KiB** |
| Index length: | **0.0 bytes** |
| Max data length: | **0.0 bytes** |
| Data free: | **0.0 bytes** |
| Table size (estimate): | **112.0 KiB** |

Test
**test.staff_test**

**Table Details**

| | |
|---|---|
| Engine: | **InnoDB** |
| Row format: | **Compact** |
| Column count: | **12** |
| Table rows: | **1000** |
| AVG row length: | **114** |
| Data length: | **112.0 KiB** |
| Index length: | **16.0 KiB** |
| Max data length: | **0.0 bytes** |
| Data free: | **0.0 bytes** |
| Table size (estimate): | **128.0 KiB** |

In comparison of the table status without indexes on the left and the table status with an index on the right, the index length increased 16.0KiB per 1000 table rows, and the estimated table size increased 16.0KiB per table rows as well.

6) Determine which index to use
For complex queries with more than one condition, first, possible indexes need to be added first for testing.

The tested query is

```
select * from staff_test where education = 'diplomat' and marital_status = 'single';
```

The possible indexes are education_index and marital_status_index, therefore both of them are added in staff table.

```
1 • explain select * from staff_test where education = '大专' and marital_status = '单身';
2
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| id | select_ty | table | type | possible_keys | key | key_len | ref | rows | Extra |
|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | staff_test | ref | education_index,idx_staff_test_marital_status | education_index | 1 | const | 3 | Using index condition; Using where |

The QEP shows that the query chose education_index over martital_status_index. The reason that the QEP use education_index over marital_status_index is that education is generally more unique than marital_status.

```
select count (*) from staff_test where education = 'single';
```

Result is 355.

```
select count(*) from staff_test where marital_status = 'single';
```

Result is 3.

In conclusion, the use of indexes has significant impact on improving the performance of queries in order to improve database performance. However, the use of indexes also has its disadvantages such as performance decrease of writes and increase of data storage size.

As the results in experiments above shows, by adding one index on a table with 1,000 records,

- The performance of the query example is improved about 81%, and the average execution time is reduced about 1.44ms per 1000 records.
- The average execution time of insert statement increased 0.532ms.
- The table size increased 16.0KiB

Therefore, in order to apply a new index as a way to optimize SQL statement, there are several factors need to be considered.

- User feedbacks
  It would be better to consider use of indexes based on users' daily tasks before users complain about them. However, as a database administrator, optimizing SQL statement should be based on user feedbacks towards certain task functions in order to improve system performance and satisfy users' demands.
- Proportion of query and write towards the related table
  Due to that use of indexes also has negative impact on the performance of writes, avoiding unnecessary use of indexes is important to balance the performance of queries and writes.
- Existing indexes on the related table
  Based on identified missing indexes and existing indexes on the related tables, some indexes may need to be deleted by determining which index to use.
  Some indexes can be combined as a multi column index in order to maximize system memory.

### 10.1.2 Other Optimizations

In Chapter "Hidden Performance Tips", Ronald Bradford (2011) suggested several tips to optimizing SQL statement such as index data types, MySQL caching and combining SQL statements.[8]

**Data Type and Column Types**

[8] Ronald Bradford. (2011) *Effective MySQL: Optimizing SQL Statements*. Chapter 8, page 161. The United States. MacGraw-Hill Companies, Inc.

According to Ronald Bradford, “these optimizations include using specific data types and/or column types. The benefit is a smaller disk footprint producing fewer disks I/O and results in more index data being packed in available system memory.”

The general idea is to minimize data length especially those included in indexes. For instance, ENUM () is an ideal data type. The reasons are

- An ENUM column provides additional data integrity with an implied check constraint.
- An ENUM column provides additional data integrity with an implied check constraint.
- The values of an ENUM column are better for readability. For example, if you have a status field, the use of an ENUM is compact in data size and provides a meaningful description of the column value.

**MySQL Caching**

The MySQL query cache is a useful tool for improving query performance. According to MySQL Reference Manual[9], by using following settings, execution time of repeating queries can be reduced.

This query_cache_type variable can be set to the following values:

- A value of 0 or OFF prevents caching or retrieval of cached results.
- A value of 1 or ON enables caching except of those statements that begin with SELECT SQL_NO_CACHE.
- A value of 2 or DEMAND causes caching of only those statements that begin with SELECT SQL_CACHE.

Setting the GLOBAL query_cache_type value determines query cache behavior for all clients that connect after the change is made. Individual clients can control cache behavior for their own connection by setting the SESSION query_cache_type value.

**Combining SQL statement**

According to Ronald Bradford, combining repeating SQL commands as one instead of executing them one by one can reduce total execution time.

[9] MySQL 5.7 Manual, 8.9.3.3 Query Cache Configuration. From http://dev.mysql.com/doc/refman/5.7/en/query-cache-configuration.html

```
SET @@profiling = 0;
SET @@profiling_history_size = 0;
SET @@profiling_history_size = 100;
SET @@profiling = 1;
select * from staff where staffid = '1';
select * from staff where staffid = '2';
select * from staff where staffid = '3';
select * from staff where staffid = '4';
select * from staff where staffid = '5';
select * from staff where staffid in (1,2,3,4,5);
SHOW PROFILES;
```

| | | |
|---|---|---|
| 347 | 0.00026350 | select * from staff where staffid = '1' |
| 348 | 0.00029700 | select * from staff where staffid = '2' |
| 349 | 0.00027700 | select * from staff where staffid = '3' |
| 350 | 0.00024000 | select * from staff where staffid = '4' |
| 351 | 0.00025275 | select * from staff where staffid = '5' |
| 352 | 0.00031050 | select * from staff where staffid in (1,2,3,4,5) |

The tested time results indicate that the total execution time of these five select statements is 1.33025ms.The combined select statement uses 0.31050ms, which is only about 23.3% of total execution time.

## 10.2 Data Mining

### 10.2.1 Introduction

Nowadays, people are dealing with larger and larger data in different fields. Traditional data analysis can no longer satisfy the demands of modern society. The process of knowledge discovery needs to be faster, and the results need to be more accurate. "Data Mining is the process of automatically discovering useful information in large data repositories." (Tan, Steinbach, Kumar)[10]

Data mining techniques have been applied in many field including sales, marketing, customer retention, and buyer behavior. Data Mining Techniques are generally categorized into description and prediction methods. Different data mining methods are chosen depending on different purposes.

In this research, decision tree as a data mining technique is chosen for research and experiment in order to determine whether it can be applied to predict sales chance of certain products or not.

### 10.2.2 Research Problem

The research problem is that whether decision tree model is able to identify a strategy based on customers' characteristics and purchase behaviors in order to predict sales chance of certain new product.

### 10.2.3 Methodology

1) Translate into data mining problem

The new product the company want to sell may have no connection with customers. In order to build connections between the new product and customers, a similar product need to be chosen based product category, model and product price. Variables are customers' characteristics and purchase behaviors.

The result will be delivered as percentages and rules based on the variables used in the model.

The result will be used to predict which group of customers may be interested in the new product the company want to sell and the possibilities.

2) Select appropriate data

Data is selected from customer information and sales information.

The variables selected from customer information are customer name, customer category, sales model, company size and region.

[10] Tan, P., Steinbach, M., Kumar, V. (2006) Introduction to Data Mining. Boston: Pearson Education.

The variables selected from sales information are selected product and related customer ids.

The number of selected data which is customer information is only 53 due to the limited information that my sponsor is able to provide.

3) Get to know the data

Among these variables in selected data, most of them are categorical variables, and one numeric variable.

Customer category is categorical variable that to describe category the customers belong to. There are generally five customer categories including individual, hotel, restaurant, super market and convenience store.

Sales model is also categorical variable to describe how the products are sold to the customer. The sales models include wholesale and retail.

Company size is numeric variable. It uses number of branches that customers have to describe the company scale.

Region is categorical variable to describe where the customer located.

There are some missing values in company size due to sometimes the customers are individuals that do not have this attribute. Some data in region and customer category is inconsistent.

4) Create a Model set

The model set is created as following

| Name | Customer id | Customer Category | Region | Company Size | If Purchased product before |
|---|---|---|---|---|---|
| Function | ID | Attribute | Attribute | Attribute | Label |
| Data Type | Integer | Text | Text | Integer | Text |

5) Fix problems with data

Miss values in company size are imputed with ‘0’ in order to include them as targets instead of throwing them out. Data in customer category and region are changed in order to keep them consistent.

6) Build Models

Decision tree model is applied to identify the rules based on the model set.

7) Access Models

In this model, the decision tree is splitting based classification error. In "Introduction to Data Mining", the authors explained that classification error is one of the measures to determine best way to split a node.[11]

The formula of classification error at a node t is:

$$Error(t) = 1 - \max_i P[i \mid t]$$

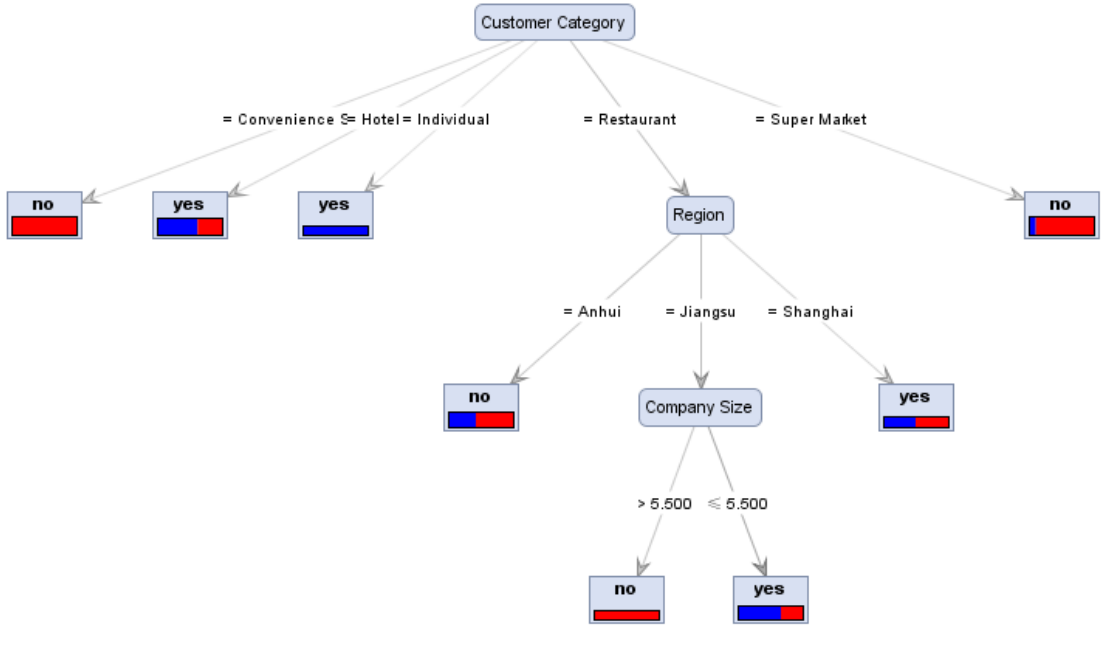
As the authors state,

"Maximum (1 -1/n): when records are equally distributed among all classes, implying least interesting information

Minimum (0.0): when all records belong to one class, implying most interesting information"

This means that the lower error of a node is the better splitting.

By using Rapidminer5, the decision tree generated as following




[11] *Introduction To Data Mining*. Pang-Ning Tan, Michigan State University, Michael Steinbach, University of Minnesota, Vipin Kumar, University of Minnesota. PPT slides: Chapter 4, page 53.

### 10.2.4 Analysis

The possibilities can be shown by calculating the percentage of "yes" at a leaf. The detailed calculation is as following table.

| Decision Tree Result Calculation | | | | | |
|---|---|---|---|---|---|
| **Customer Category** | Region | Company Size | Yes | No | Percentage |
| **Convenience Stores** | - | - | 0 | 9 | 0% |
| **Individuals** | - | - | 5 | 0 | 100% |
| **Hotels** | - | - | 5 | 3 | 62.5 |
| **Supermarket** | - | - | 1 | 8 | 11.1% |
| **Restaurant** | Anhui | - | 3 | 4 | 42.9% |
| **Restaurant** | Shanghai | - | 2 | 2 | 48.3% |
| **Restaurant** | Jiangsu | >5.5 | 0 | 5 | 0% |
| **Restaurant** | Jiangsu | ⩽5.5 | 4 | 2 | 66.7% |

The analysis of the decision tree indicates that

- The possibility of selling this product to convenience stores as a group of customers is 0%.
- The possibility of selling this product to hotels as a group of customers is 62.5%.
- The possibility of selling this product to individuals is 100%.
- The possibility of selling this product to supermarkets as a group of customers is 11.1%.
- The possibility of selling this product to restaurants in Anhui province is 42.9%.
- The possibility of selling this product to restaurants in Shanghai is 48.3%.
- If the restaurants in Jiangsu province have more than 5 branches, the possibility of selling this product is 0%
- If the restaurants in Jiangsu province have less than 6 branches, the possibility of selling this product is 66.7%

However, this model cannot be deployed due to insufficient selected data and lack of practical variables. This is why the result shows that 100% possibility of selling the selected product to a whole customer category, which is highly unlikely in real world. There should be much more available data to be selected in order to improve the confidence level of the test results. In addition, the provided variables about customers' characteristics are insufficient and not appropriate enough to be selected as the factors to represent a customer.

### 10.2.5 Conclusion

In conclusion, data mining techniques such as decision tree are often used to discover useful information from large set of data. If the data selection is too small, the results discovered by applying data mining techniques may not be reliable to use. In addition, the diversity of selected data can be very useful for different purposes. Therefore, in this case, data collection is a very important during daily tasks.

## 11. Conclusion

This system is an important part of the company's undergoing project which is enterprise management platform. This system is designed and developed for improving human resource management, customer relationship management, supply chain management and financial management of the company. The goal of system design is to achieve unified company information management in order to improve efficiency of daily tasks and reduce costs.

The Unified Modeling Language is applied for documenting the system. Detailed analysis and design of the system consists five functional modules including human resource management, customer relationship management, supply chain management, financial accounting and management accounting. ThinkPHP as MVC framework development tool is used for designing and developing user interface of the system.

The provided functions of the system meet the company's requirement. However, there were many problems occurred during the process of system development. For instance, due to lack of work experience and communication with the company, ThinkPHP was not used while developing user interface. This caused a lot of codes need to be rewritten. In addition, during discussion of system requirement, there were a lot of changes due to the different understandings of system function.

The project is a great experience to me because I have never done such project before. I have learned so many things during the process. For instance, from the very beginning, I have learned how to communicate with client and understand what they need from the project. Although I made some mistakes as I mentioned before, learning from these mistakes can prevent me from making same mistakes again in the future. The project is also a learning process for me. During this process, I have enhanced some of my skills such as MySQL database design, PHP and HTML coding, and the project challenges me to learn more technologies such MVC framework, Optimizing SQL statement and data mining techniques.

There are still many parts of the system can be enhanced in the future such as functions and system performance. For instance, use the findings of the research about optimizing SQL statement can be used for improving the performance of database in order to improve the system performance. As the company is fast growing, other functions related to electronic business such as online ordering may be needed in the future. In addition, other data mining techniques such association may be able to apply in the future for the company to make decisions such as targeting potential customers.

## Reference

Ronald Bradford. (2011) *Effective MySQL: Optimizing SQL Statements*. The United States. MacGraw-Hill Companies, Inc.

MySQL 5.7 Manual, 8.9.3.3 Query Cache Configuration. From http://dev.mysql.com/doc/refman/5.7/en/query-cache-configuration.html

Tan, P., Steinbach, M., Kumar, V. (2006) Introduction to Data Mining. Boston: Pearson Education.

Unified Modeling Language User Guide, The (2 ed.). Addison-Wesley. 2005. p. 496. ISBN 0321267974.

## Change Log

| Chapter | Page | Change |
|---|---|---|
| 1 Introduction | 4 | Grammar fixed |
| 1.1.1 Business and Task Flow | 5 | Language/Grammar fixed |
| 1.2.1 Improving Database Performance by Optimizing SQL statement | 5 | Language/Grammar fixed |
| 2.2 Current Business Flow | 7 | Language/Grammar fixed |
| 2.3 Current Business Problem | 8 | Language/Grammar fixed |
| 3 Project Statement | 11 | Repeating information deleted |
| 4.1.3 Detailed Use Case | 20,21 | Grammar fixed |
| 4.1.3 Detailed Use Case | 25,26 | Repeating information deleted |
| 5 Solutions | 29,30,31 | Rewrite<br>Alternative solutions added |
| 6.5 System Structure | 32 | Language fixed |
| 7.2 Class Diagram | 44 | Language fixed |
| 9 Test Cases | 77 | Language/Grammar fixed |
| 9 Test Cases | 79 | Mistake corrected |
| 9 Test Cases | 110 | Test number corrected |
| 10.1 Optimizing SQL Statement | 117-121 | Language/Grammar fixed |
| 10.1 Optimizing SQL Statement | 123 | Mistake corrected |
| 11 Conclusion | 131 | Language/Grammar fixed |