CS307 Project2 Report

Basic Information

Group members:

(1) Name：王思懿； SID： 12010339； Lab session : Lab Class 1

(2) Name：黄慧惠； SID： 12010336； Lab session : Lab Class 1

Contribution:

1. **王思懿 50%：**

In basic requirements:

APIs(1,4,5,6,7,8,9)

In advanced requirements:

Query the order list based on multiple parameters, and the parameters

can be null or not.

Encapsulate the features and implement a real back-end server instead of several

independent scripts or programs.

A usable and beautiful GUI design for data presentation.

1. **黄慧惠 50%：**

In basic requirements:

APIs(2,3,10,11,12,13)

In advanced requirements:

Design the Bill Module.

Apply database connection pools.

Proper use of user privileges, procedures, indexing, and views in a reasonable manner.

Support high-concurrent or distributed access with proper pressure tests.

Original Data

The table creation statement is as follows:

```
create table center
(

    id   int primary key,
    name varchar(200) unique not null

);
create table enterprise
```

```
(
    id            int primary key,
    name          varchar(200) unique not null,
    country       varchar(200)        not null,
    city          varchar(200),
    supply_center varchar(200)        not null,
    industry      varchar(200)        not null,
    foreign key (supply_center) references center (name)
);
create table model
(
    id         int primary key,
    number     char(7)            not null,
    model      varchar(200) unique not null,
    name       varchar(200)       not null,
    unit_price int                not null
);
create table staff
(
    id            int primary key,
    name          varchar(200)   not null,
    age           int            not null,
    gender        varchar(10)    not null,
    number        char(8) unique not null,
    supply_center varchar(200)   not null,
    mobile_number char(11)       not null,
    type          varchar(200)   not null,
    foreign key (supply_center) references center (name)
);
```

Basic Requirements

1.APIs for manipulating the original data:

First, find the table with original columns as the foreign keys, and find the corresponding columns.Then when doing update and delete operations, first do operations on the table with the original table column as foreign key, and then operate on the original table. Insert and Select can be handled normally

2.stockIn:

First create an inventory table, the table creation statement is as follows：

```
create table inventory
(
    center               varchar(200) not null,
    product_model        varchar(200) not null,
    quantity             int          not null,
    total_purchase_money int          not null,
    sales_quantity       int,
    profits              int,
    primary key (product_model, center),
    foreign key (product_model) references model (model),
    foreign key (center) references center (name)
);
```

Then, when importing data, first check whether the supply center exists in the center table, then check whether the product exists in the model table, and finally check whether the personnel exist, whether the supply center is correct and whether the identity is correct in the staff table, and then check whether the product exists in the staff table.

If it can be added to the warehouse, the information of the warehouse will be changed, and the corresponding cost of the purchased product will be changed.

3.placeOrder:

First create the contract and orders table, the tables creation statement is as follows：

```
create table contract
(
    contract_num     char(10)     not null primary key,
    enterprise       varchar(200) not null,
    contract_manager varchar(30)  not null,
    contract_date    date         not null,
    contract_type    varchar(30),
    supply_center    varchar(200),
    foreign key (enterprise) references enterprise (name),
    foreign key (contract_manager) references staff (number),
    foreign key (supply_center) references center (name)
);
create table orders
(
    contract_num       char(10)     not null,
    product_model      varchar(200) not null,
    quantity           int          not null,
    salesman_num       char(8)      not null,
    estimated_delivery date         not null,
    lodgement_date     date         not null,
    primary key (product_model, contract_num, salesman_num),
    foreign key (product_model) references model (model),
    foreign key (salesman_num) references staff (number),
    foreign key (contract_num) references contract (contract_num)
);
```

Then, check whether the person is a salesman, and whether the inventory quantity in the inventory table is satisfied, which is in line with the insertion, and the corresponding inventory quantity is reduced and sales are increased.

4.updateOrder:

First, reading the table data from the tsv file. Note Handling illegal data: If we can select orders with the same salesman number, product model and contract number from the original order form, the updated data is legal. Calculate inventory using the updated data and data selected before and then update the inventory. Remove orders with a quantity of 0.

5.deleteOrder:

Sort the original order table, then delete the corresponding order, and finally update the inventory

6.getAllStaffCount:

Using the sql statement of select count.

10.getFavoriteProductModel:

The sql statement is as following:

```
select product_model,sum(quantity)s from orders group by product_model order by
s desc limit 1;
```

In the model table, divide by product_model, calculate the sales of each product_model, and select the first row after sorting from high to low.

Advanced Requirements

1.Query the order list based on multiple parameters, and the parameters can be null or not:

If there is no argument , using * in the sql statement in the stmt=con.prepareStatement().if not, let the corresponding column name replaces *. Then use a hashmap to record columns that is Interger or Date because there is some difference between the handling of the text/char/varcher and interger/date.

2.Design the Bill Module:

The sql statement is as following:

```
create or replace view bill as
select center                                      center,
       sum(total_purchase_money)                   total_purchase_money,
       sum(profits)                                profits,
       sum(profits) - sum(total_purchase_money) total_profits
from inventory
group by center;
```

Because the view has the characteristics of changing from time to time according to the data, the view is used here to make the bill table

3.Encapsulate the features and implement a real back-end server instead of several independent scripts or programs. Also with a usable and beautiful GUI design for data presentation:

The front end uses vue, the back end uses springboot. and embed our own API to operate on 7 tables. At the back end, we connected our own database and used our own API to operate on update and delete.Use the library and API in Springboot for Insert and select, and page processing such as paging.

4.Apply database connection pools.Also with the bonus that support high-concurrent or distributed access with proper pressure tests:

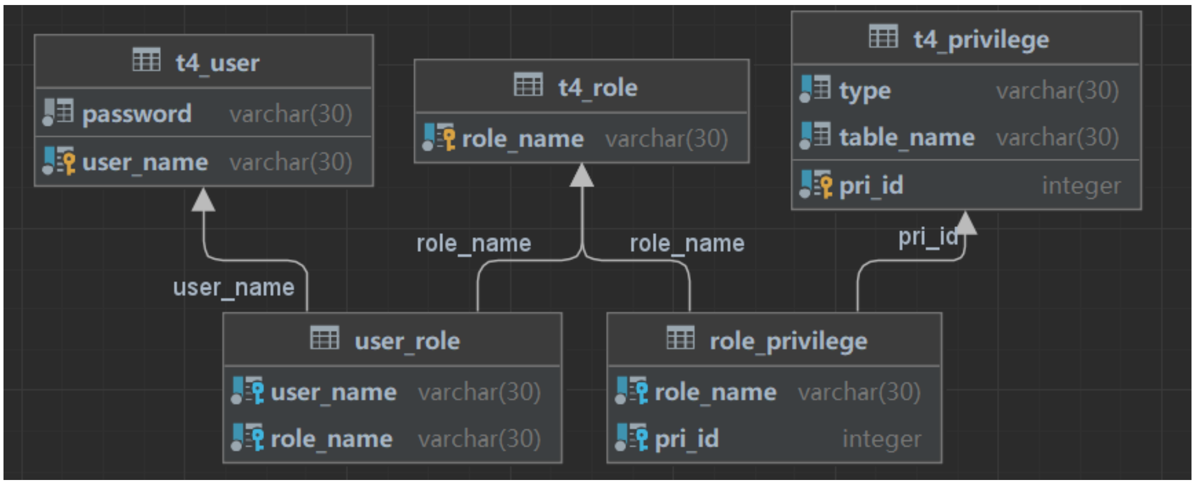
We use the hikari database connection pool that comes with springboot, and limit the amount of concurrency by setting the number of database connection pools, and finally perform pressure testing through the jmeter pressure testing software.

5.Proper use of user privileges, procedures, indexing, and views in a reasonable manner:

user privileges:

This privileges management is based at RBAC model. Role-based access control (RBAC) is a policy-neutral access-control mechanism defined around roles and privileges. The components of RBAC such as role-permissions, user-role and role-role relationships make it simple to perform user assignments.(refer to wikipedia)

The table built:



procedures:

The sql statement is as following:

```
create or replace function staff_check()
    returns trigger
as
$$
declare
trigger_supply_center varchar;
begin
    select name into trigger_supply_center from center where
name=new.supply_center;
if trigger_supply_center is  null
    then return null;
else return new;
    end if;
end;
$$
    language plpgsql;
```

When adding data to the staff table, the trigger will check whether the center of the staff is in the center table. If it exists, it will be added normally. If it does not exist, it cannot be added.

indexing:

The sql statement is as following:

```
create index contract_supply_center
on contract (supply_center);
```

Since there are few types of centers in the table, it is a very efficient behavior to use index for center.

views：

This is implemented in the bill table