CS307 Project 2 Report

Class: Lab 1

12011710 董宇航，12010225 张维宁

Contributions: 董宇航 50%， 张维宁 50%

Contributions belong to both :

1. The overall structure design, including ER graph, variable types, foreign keys, DDL statements used to generate database tables, etc.

2. Java program "methods.java" , implemented all APIs required by the project.

3. Design program test samples.

4. Design the function of outputting program running results as files.

Contributions belong to 董宇航 :

1. Java program "GoodLoader_proj2_database.java" , which is used to read CSV files and import data to four entity tables (supply_center, enterprise, product and staff) in the database.

2. Java program "GoodLoader_proj2_task.java" , which is used to read CSV and TSV files, and insert, delete and update data in the database according to the requirements of Task1, 2, 3 and 4.

3. Design user privileges.

Contributions belong to 张维宁 :

1. Java program "GUI.java" , displaying program running results in graphical interface.

2. Design database indexing.

Part 1: Briefly describe the table designs

Four tables represent four entities: supply center, enterprise, product, and employee.

Three tables represent relation between two entitiy table: rel_supply_center_enterprise,

rel_supply_center_product, rel_supply_center_staff

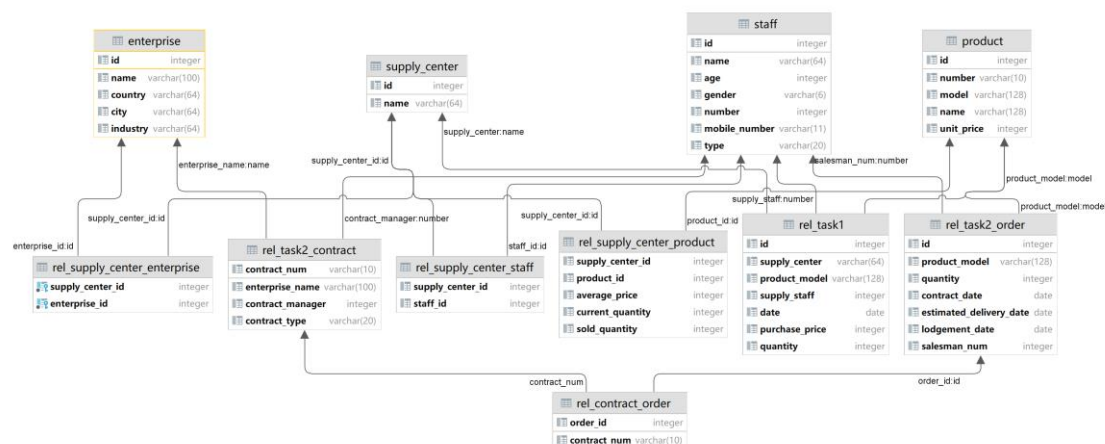Table "rel_task1" records incoming records involved in Task 1.

Table "rel_task2_contract" records contracts involved in Task 2.

Table "rel_task2_order" records orders involved in Task 2.

Table "rel_contract_order" represent relation between contracts and orders.
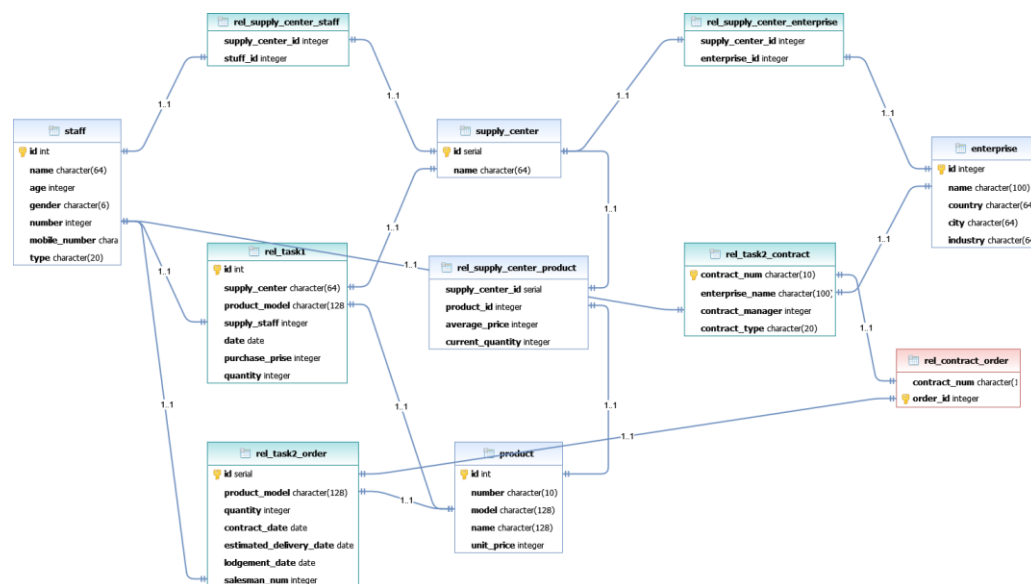
This ER graph is generated by online drawing website ： www.freedgo.com

This ER graph is generated by DataGrip.

Part 2:   Connction

Connecting Java program to database :

We modified it based on the demo program "GoodLoader.java" provided by teacher.

Several advanced features are used to improve performance, including batch, PreparedStatement, delayed commit, establishing only one connection to the database, and so on.

Connecting Java program to file :

We read CSV and TSV files by using BufferReader and FileReader.

We remove separators by using String.split(",") .

We output program running results as txt files, by using PrintStream.

Part 3: Basic API

For ease of understanding and memory, we numbered all APIs as per the project requirements document.

2) stockIn:

Start with a tabular query based on the type of person and the supply center where the person resides, using ResultSets rs = check. ExecuteQuery(); Gets the query structure, and if rs.next() = false, the input data is illegal.

Using nextval() and lastval() to get id, which is generated by increasing sequence (serial).

After executing the insert statement, immediately search the database for what just be inserted, and if it can be retrieved, the output inserts successfully.

3) placeOrder   4) updateOrder   5) deleteOrder :

Implementation is roughly the same as 2) stockIn.

When entering Data, call the String.replace method to change the character'\'to the character'-'. (For example, change 2021/06/07 to 2021-06-07.)

During the process, we tried to use views in the sql statement, encountered setbacks, and finally gave up using views.

6) ~ 13) select APIs:

6) gets the quantity of each type of staff, 7) gets the total contract quantity, and 8) gets the total order quantity.

We encountered setbacks when writing 9) , 10), and 11), so we split the part of task 2 into two tables as needed: contract table and order table, and wrote 9) to 13). In these methods we first used "\t" to separate data, and later changed to format String for output. After deciding to use GUI, considering that format String was not neat in GUI, we finally used the method of using "\t" to separate data.

Note that when we operate Task 2, to update the current_quantity and sold_quantity, we need to obtain the previous current_quantity and sold_quantity, then increasing or decreasing this order's quantity, otherwise the select will not get the correct quantity.

In an API function, we should first close the Statement, then return to main, in order to save memory.

In 12) and 13), instead of running Scanner inside the API and reading in the keywords, we should pass in the API the keywords we need to select as parameters.

Part 4 : Advanced Requirements:

4.1. API

Query the order list based on multiple parameters :

First check the incoming parameters to determine the number of non-empty parameters, generate a "where" restriction for each non-empty parameter, and count in real time. When the number of generated restrictions equals the number of non-null parameters, stop inserting "and" and end the method.

Bill Module:

Only the sales figures are counted, "group by" date can count by day.

Using the string to intercept the year and month, then "group by" can count by month.

4.2. Index

When creating the table, the index of the id in the four entity tables and the index of the model in the product table were added, and the index of the supply center id and the index of product id, enterprise id and staff id in three relational tables are added too.

After adding indexes, the speed of importing entity table data is reduced by about 5%, and the speed of importing relational table data is increased by about 85%.

4.3. User privileges

A super_user with all privileges;

Four staff users, with only their corresponding necessary table operation privileges.

4.4. GUI

We designed a simple and practical GUI, using different buttons to click different functions.

The method that needs to enter the content will pop up the input window, enter the content separated by english commas as required, and the return value will be displayed in the text area.
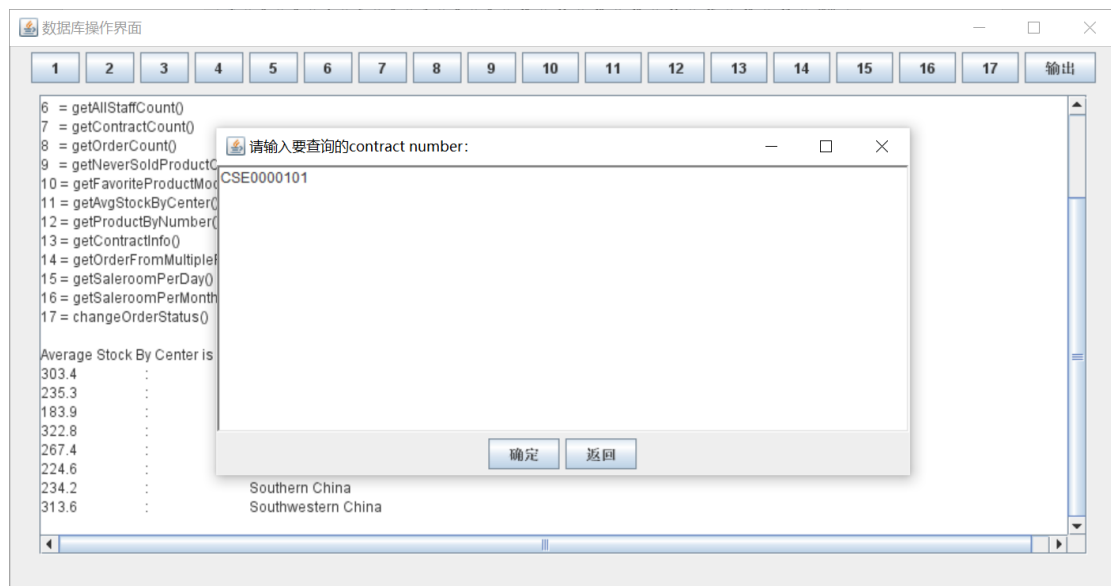
The user authority is determined by the entered login information. When the GUI interface is opened, the user login window will pop up first. If the login information is incorrect, an error will be reported. After the login is successful, the login window will be closed and the main window will pop up, and only the buttons that can be used by the user authority will be displayed.