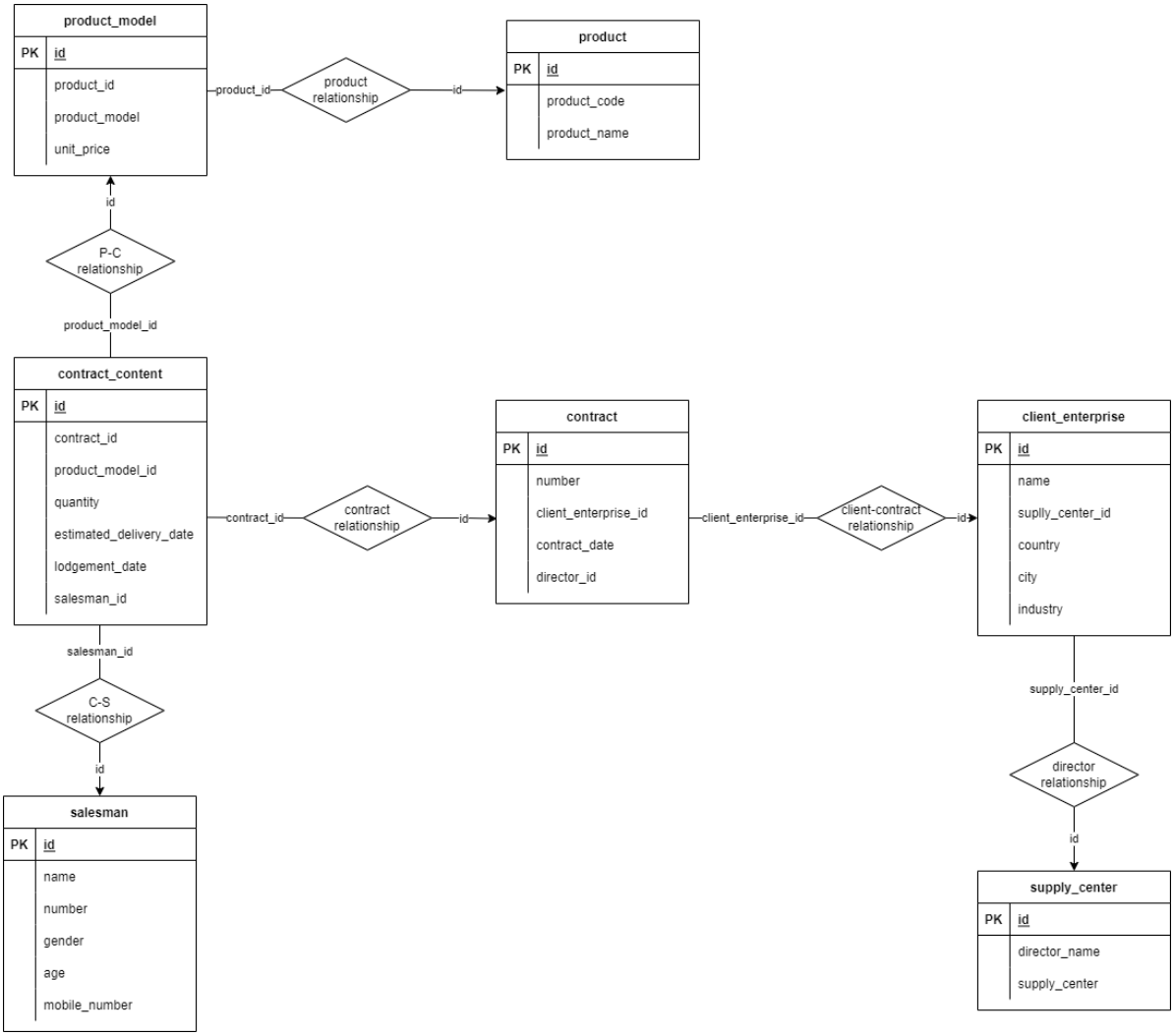
CS307 Project 1

Contributor:

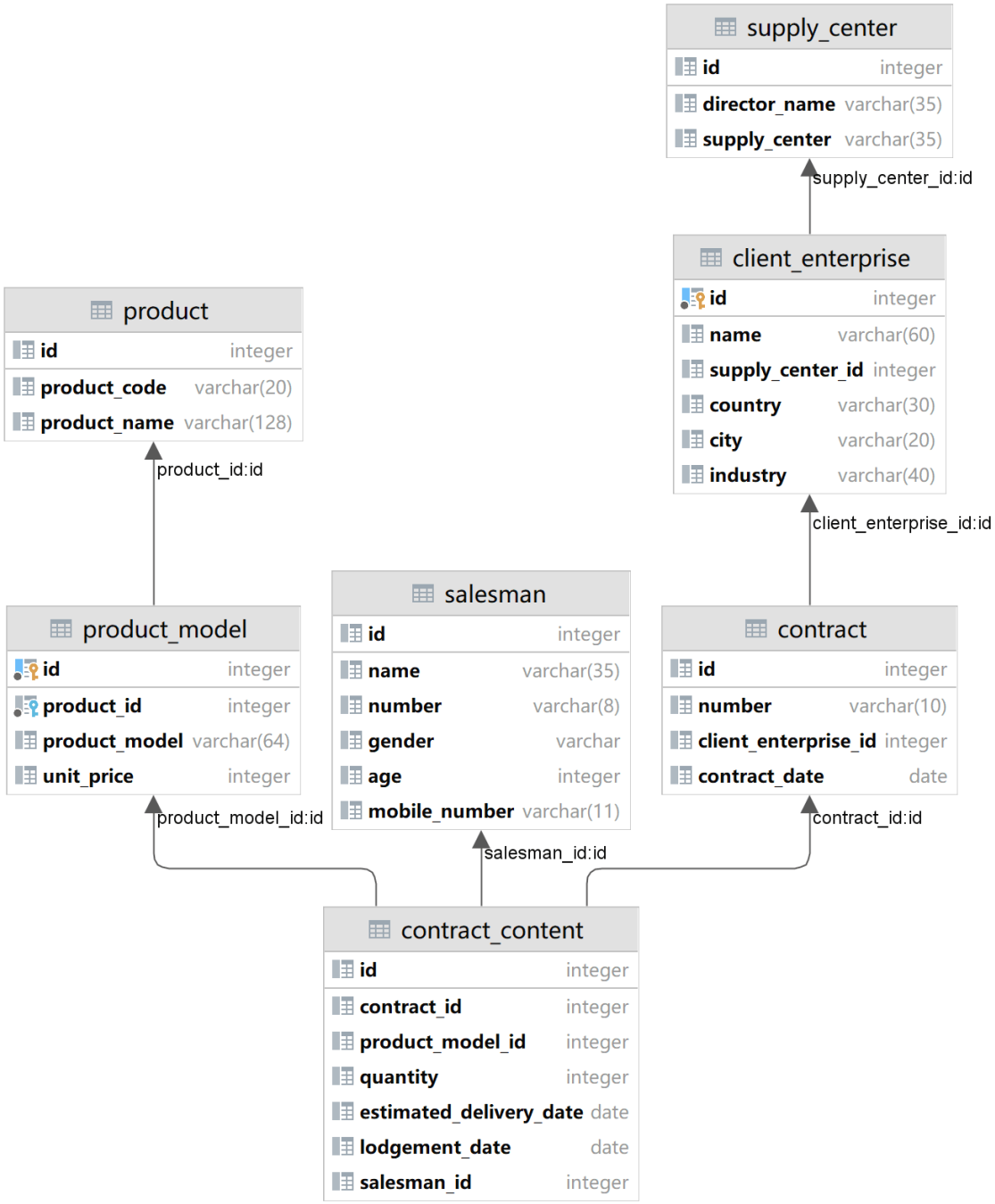
- 戴郭轶 12011211
- 周凡卜 12012519

Task 1: E-R Diagram

Software: `draw.io`



Task 2: Database Design



The id of each table is the serial primary key of each table.

Supply_center:

- director_name: As each director only in charge of one center, so we put it in the supply center
- supply_center: The name of supply center, unique

Client_enterprise:

- name: The name of enterprise
- supply center id: The foreign key which links to supply center
- country: Where the enterprise comes from
- city: Where the enterprise comes from
- industry: The enterprise's industry
- unique by (name, country, city, industry) as all elements can have the same name

Contract:

- number: The contract number, which is unique
- client enterprise id: The foreign key which links to client enterprise
- contract date: The date when the contract made
- unique by number, as each contract should be only made to one enterprise for ease of change.

Product:

- product code: The unique code of product
- product name: The name of it

Product model:

- product model: The unique name of the specified model
- unit price: Price of this model

Salesman:

- name: Name of salesman
- gender: Female or male
- age: Age of salesman
- mobile number: As name can be repeated, use it as unique

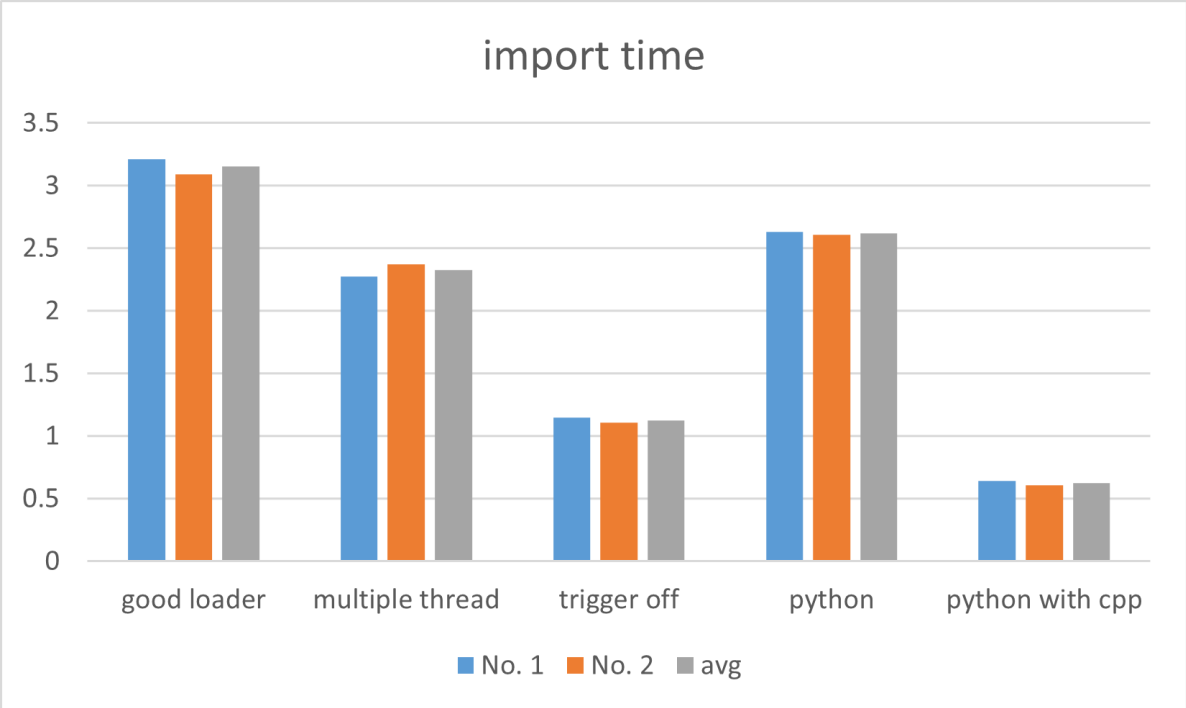
Contract content:

- product model id: Foreign key links to product model
- quantity: The quantity needed
- estimated delivery date: the estimated time
- lodgement date: the actual time
- salesman id: Who in charge of this item of contract
- As a contract may contain many items, divide them into a smaller table

Task 3: Data import

In this part, we tried two main ways to import data.

Time cost of each code.



```
-Java
  -goodloader
  -multiple thread
  -trigger off
-Python
  -pure python
  -cpp data process and python import
```

Java import

Java part, we first divide the csv into tables and store into memory temporarily.

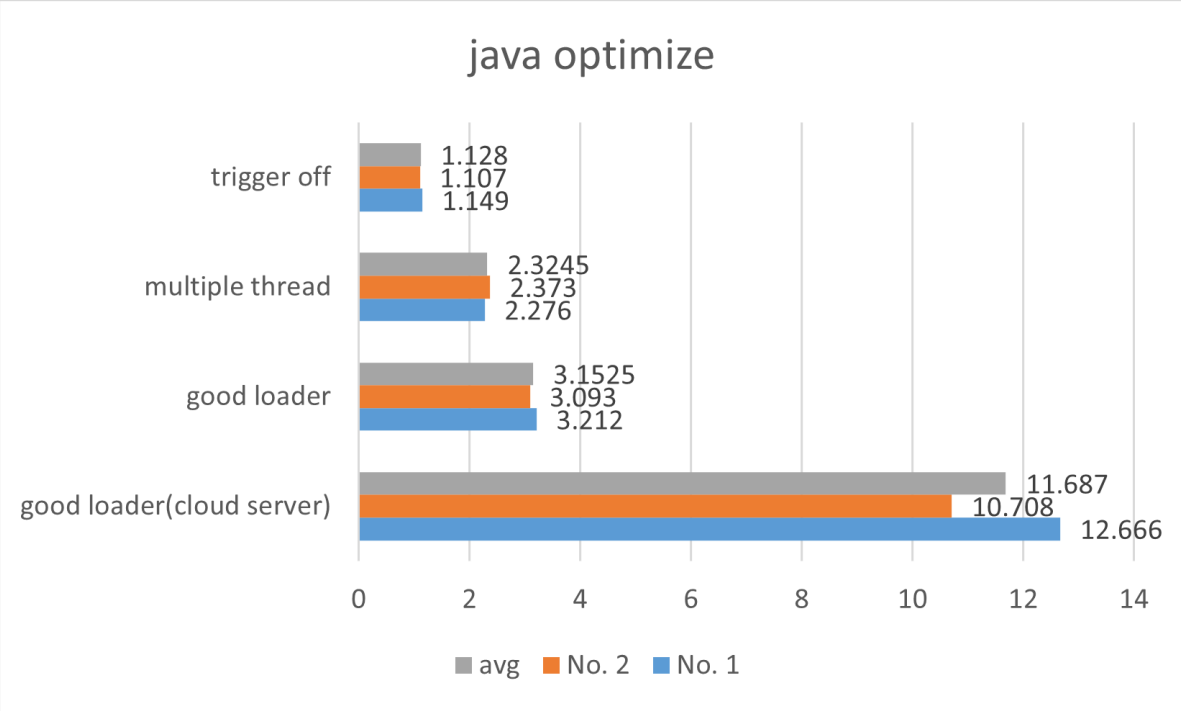
Secondly, according to the E-R diagram, we do top down insertion and get the order stored.

Then insert line by line.

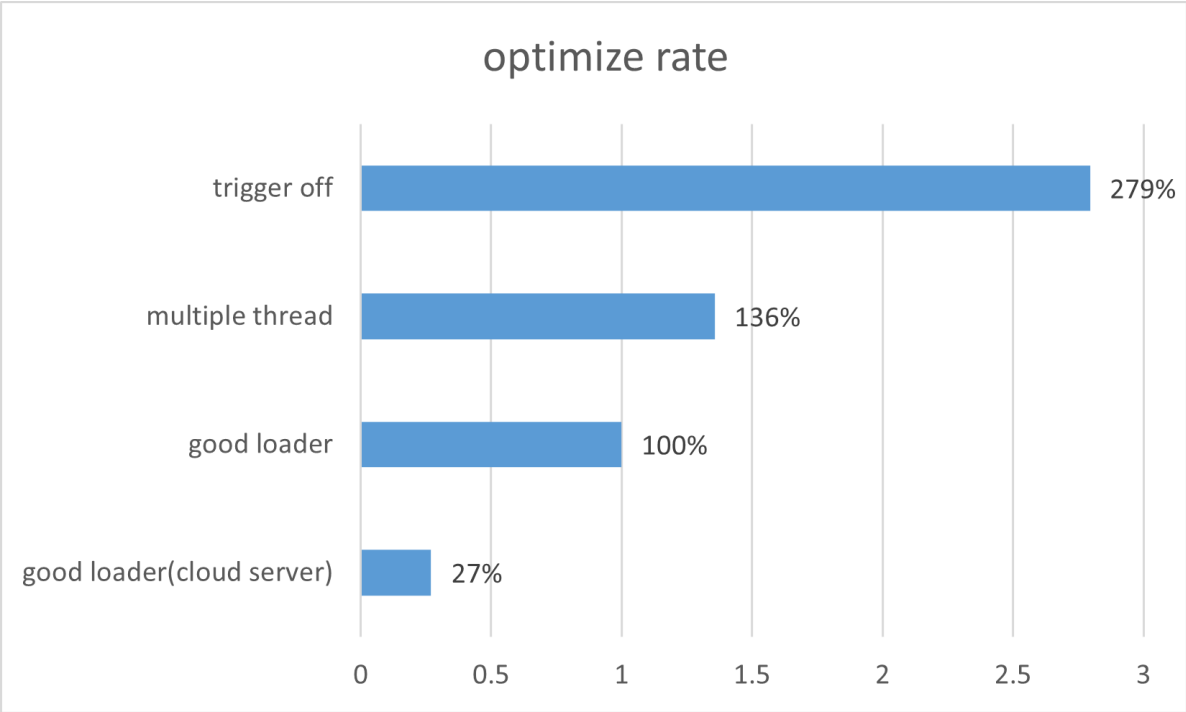
At first, we used our cloud server, but the time is too horrible. And we found the limit is mainly on network IO. So we changed to local postgresql server. Getting speed up of 3.71 times.

And according to the E-R diagram, we can find there are three row parallel, which can be used in multiple threading. Then we get a time record of 2.325s.

The rest part, I cancel the triggers as I can manually maintain the data availability. Finally, the time of java part reduced to average of 1.128s.



The optimize rate, with baseline of goodloader.



Python import

Firstly, i find that psycopg2 provides the function that allows us to copy data directly from a csv file to database with a efficiency same as function insert with batch. So i choose to pre_process the data and generate .csv files and import it with the functions. As we all know, if we write to much to the disk, the limitations is our disk io, so i choose the use python io.String that creates a csv like file and read it directly in memory.

I use cloudserver at the begining.

```
Python Console
pre_process time is 1.7836146354675293 seconds ---
client_enterprise time is 0.020674943923950195 seconds ---
director time is 0.018950939178466797 seconds ---
salesman time is 0.04094886779785156 seconds ---
product time is 0.020592451095581055 seconds ---
product_model time is 0.3956427574157715 seconds ---
contract time is 0.1230018138885498 seconds ---
contract_content time is 1.2080726623535156 seconds ---
total time:3.611499071121216
```

The contact_content time seems to long because it is not corresponding to the size proportion of contract, and at the same time, the network monitor find a peek at the rate of my vps network limitation, so i suppose that the limitation is network io.

The next test is on my own computer with the connection to localhost with the negligible latency and network limitation. The result is below:

```
pre_process time is 1.7985966205596924 seconds ---
client_enterprise time is 0.0030021667480468 75 seconds ---
director time is 0.0013298988342285156 seconds ---
salesman time is 0.01251530647277832 seconds ---
product time is 0.0020029544830322266 seconds ---
product_model time is 0.3069026470184326 seconds ---
contract time is 0.07787632942199707 seconds ---
contract_content time is 0.852492094039917 seconds ---
total time:3.055718421936035
```

The import time of contract_content is reduced to 0.85s. We can find that pre_process costs most time. So my next step is to optimize the pre_process period.

My first consideration is use muilty_threads(multi_thread.py). However, python has the fake multicore process, the pre_process costs much more than before:

```
Python Console
Time to read file: 0.963876485824585
pre_process time is 15.10898494720459 seconds ---
director time is 0.05306577682495117 seconds ---
client_enterprise time is 0.30851197242736816 seconds --
salesman time is 1.178116798400879 seconds ---
product time is 0.5765163898468018 seconds ---
product_model time is 2.84551739692688 seconds ---
contract time is 0.49003148078918457 seconds ---
contract_content time is 6.451136827468872 seconds ---
total import time is 11.902896642684937 seconds ---
total time:27.014883756637573
```

So we must change the ideas. The potentiality of python has been excavated. So i decided to use cpp as an substitute. The cpp code is read.cpp . The optimization of cpp is use stringstream to substitute fileout. It can be optimized for 0.1s. The result of cpp is:

```
"C:\Users\dgy\OneDrive - Trash Network\GitHub\cppstudy\Clion\cmake-build-debug\read.exe"
time:0.628

Process finished with exit code 0
```

Then import is with python(import_cpp.py):

```
Time to import supply_center.csv:]0;C:\Program Files\Python310\python.exeTime to import supply_center.csv:  0.04541468620300293
Time to import client_enterprise.csv:  0.0031986236572265625
salesman time is 0.009806394577026367 seconds
product time is 0.0020020008087158203 seconds
product_model time is 0.00860595703125 seconds
contract time is 0.0579679012298584 seconds
contract_content time is 0.8320021629333496 seconds
total import time is 0.9589977264404297 seconds
total time is 1.5869977264404298 seconds
```

We can find that the total time comsumption is reduced to 1.586s. Which is much more faster.(If i have a better ssd, it would be faster.)

If we turn off the trigger, the total time will be further reduced to 0.933s.

```
Time to import supply_center.csv:␀]0;C:\Program Files\Python310\python.exe␀Time to import supply_center.csv:  0.057051897048950195
Time to import client_enterprise.csv:  0.0020020008087158203
salesman time is 0.010014057159423828 seconds
product time is 0.002931356430053711 seconds
product_model time is 0.002997875213623047 seconds
contract time is 0.03300189971923828 seconds
contract_content time is 0.19499540328979492 seconds
total import time is 0.3059976100921631 seconds
total time is 0.9339976100921631 seconds
```

Part 4. Benchmark