Project 1 Report

Basic Information of Group:

1. 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

2. Contribution:

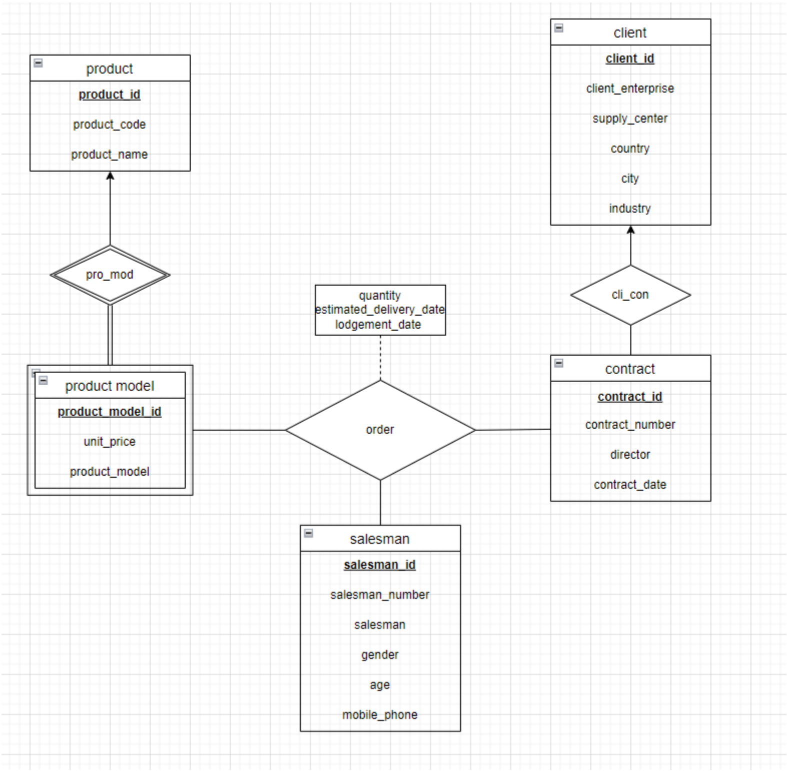
   (1) 王思懿 50%：Task 1 , Task 2 , Benchmarking with file APIs, Comparison, Database index and data visualization in Task 4

   (2) 黄慧惠 50%：Task 3 , Benchmarking with Database APIs, Make test data, User privileges management,transaction management in Task 4

Task 1: E-R Diagram

E-R DiagramTask 1: E-R Diagram

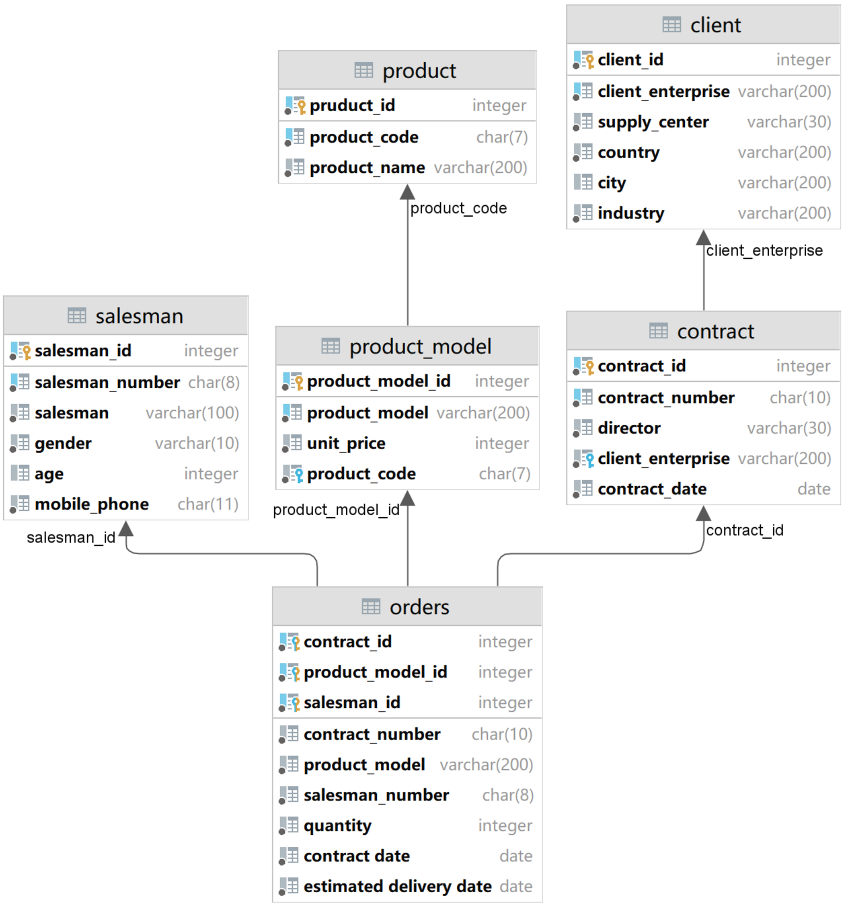
E-R Diagram



Drawn by draw.io Diagram

Task 2

1. **the E-R diagram generated by DataGrip.**



2. **Briefly describe the table designs and the meanings of each table and column.**

(1)client: the table is to store clients' information.

client_id: auto_increase sequence, which can be used to distinguish clients. The primary key of client.

client_enterprise: the name of enterprise where client from.

supply_center: the corresponding Supply Center to the client enterprise.

country: the country where the client enterprise is located.

city: the city where the client enterprise is located.

industry: the industry where the client enterprise resides.

(2)contract: the table to store contracts' information. Current size is 5000

contract_id: auto_increase sequence, which can be used to distinguish contracts. The primary key of contract.

contract_number: the unique identifier of each contract; each value likes CSEXXXXXXX, from CSE0000000 to CSE0004999.

contract_date: the date of creating the contract.

director: the person responsible for this contract.

client_id: the client_id of the client which we have contracts with; the foreign key of contract;references client

(3)product: the table to store the information of products

product_id: auto_increase sequence, which can be used to distinguish products. The primary key of product.

product_code: the unique identifier of the product; each value is a mix of letters.

product_name: the name of the product. Each product code has only one product name.

(4)product model: the table to store the information of product model; product model is the specific model of the product.

product_model_id: auto_increase sequence, which can be used to distinguish product models. The primary key of product model.

product model: the name of product model;Each project model has its own unit price.

unit_price: the unit price of the product model.(In our project, the unit price of the same product model is considered to be the same)

product_id: the product_id of the product of which the product model is specific model; the foreign key of product model; references product

(5) salesman: the table to store the information of salesmen.

salesman_id: auto_increase sequence, which can be used to distinguish salesmen. The primary key of salesman.

salesman: the name of the salesman.

salesman_number: the number of the salesman.

gender: the gender of the salesman.

age: the age of the salesman.

mobile_phone: the mobile phone number of the salesman.

(6)order: the table to store the information of orders made in all the contracts

contract_id: the contract_id of the contract containing the order; the foreign key of order; references contract.

product_model_id: the product_model_id of the product model the order contains; the foreign key of order; references product model.

salesman_id: the salesman_id of the salesman for the order; the foreign key of order; references salesman.

contract date: the date of creating the contract with the corresponding contract_id.

estimated_delivery_date: the estimated date of the delivery of the product model.

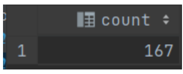
lodgement_date: the actual date of the delivery of the product model.

Task 3

Proof of correctness:

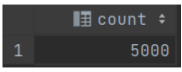
rows of each table:

client:



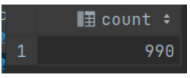
	count
1	167

contract:



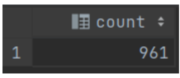
	count
1	5000

salesman:



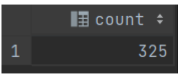
	count
1	990

product_model:



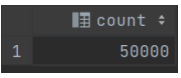
	count
1	961

product:



	count
1	325

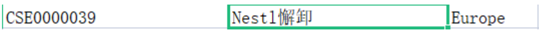
orders:



	count
1	50000

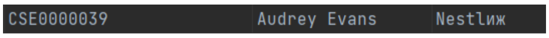
utf-8:

In csv file:



CSE0000039	Nest1懈卸	Europe

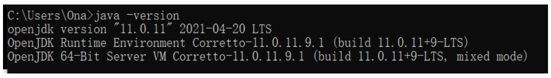
In db:



CSE0000039	Audrey Evans	Nestlиж

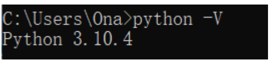
use java and python to import to postgres:

java version:



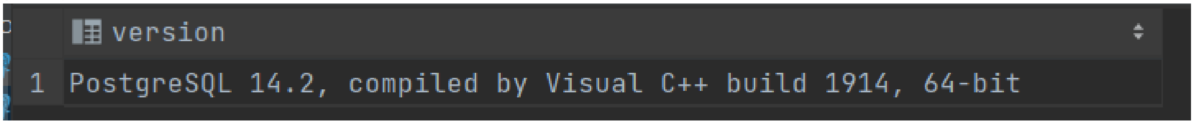
```
C:\Users\Ona>java -version
openjdk version "11.0.11" 2021-04-20 LTS
OpenJDK Runtime Environment Corretto-11.0.11.9.1 (build 11.0.11+9-LTS)
OpenJDK 64-Bit Server VM Corretto-11.0.11.9.1 (build 11.0.11+9-LTS, mixed mode)
```

python version:



```
C:\Users\Ona>python -V
Python 3.10.4
```

postgres version:



	version
1	PostgreSQL 14.2, compiled by Visual C++ build 1914, 64-bit

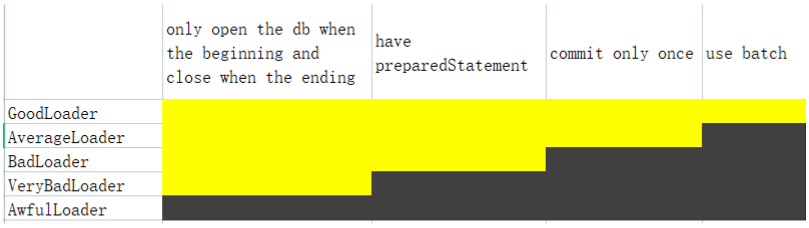
Notation for java :

Before running you should "-v contract_info.csv" in the editor configuration,and the file should in the same directory of the .java file. Also, you should import a jar package for connecting the postgres.

Notation for python :

Before running you should make sure the "contract_info.csv" file is in the same directory of the .py file.

Details of each .java and .py file:



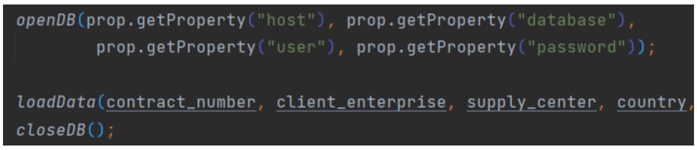
	only open the db when the beginning and close when the ending	have preparedStatement	commit only once	use batch
GoodLoader	🟨	🟨	🟨	🟨
AverageLoader	🟨	🟨	🟨	⬛
BadLoader	🟨	🟨	⬛	⬛
VeryBadLoader	🟨	⬛	⬛	⬛
AwfulLoader	⬛	⬛	⬛	⬛

The black color means it doesn't satisfy the headers and the yellow color means it satisfies the header.

For AwfulLoader:

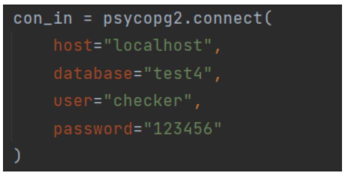
The database open and close each time to insert the statement.

In java:



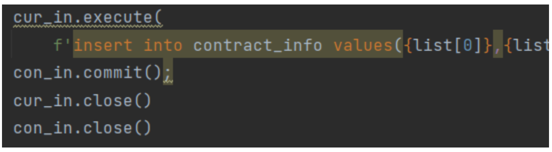
```java
openDB(prop.getProperty("host"), prop.getProperty("database"),
        prop.getProperty("user"), prop.getProperty("password"));

loadData(contract_number, client_enterprise, supply_center, country,
closeDB();
```

In python:



```
con_in = psycopg2.connect(
    host="localhost",
    database="test4",
    user="checker",
    password="123456"
)
```

then after loading the data,

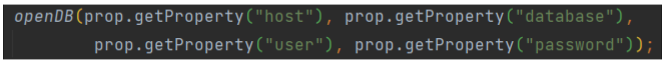


```
cur_in.execute(
    f'insert into contract_info values({list[0]},{list
con_in.commit();
cur_in.close()
con_in.close()
```

For VeryBadLoader:

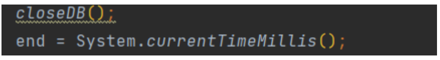
To compare with the AwfulLoader, it improves that open the database only at the beginning and close it only after all the data having been inserted into the table.

In java:



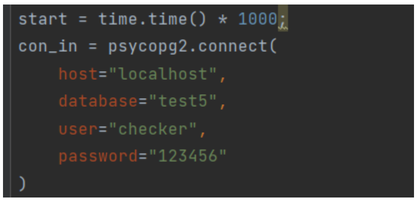
```
openDB(prop.getProperty("host"), prop.getProperty("database"),
        prop.getProperty("user"), prop.getProperty("password"));
```

then after loading all the data,



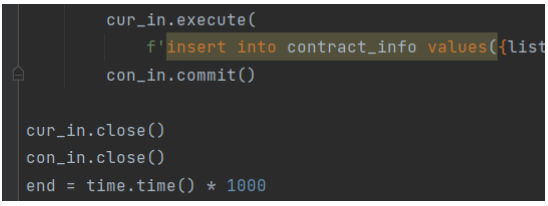
```
closeDB();
end = System.currentTimeMillis();
```

In python:



```
start = time.time() * 1000;
con_in = psycopg2.connect(
    host="localhost",
    database="test5",
    user="checker",
    password="123456"
)
```

then after loading all the data,

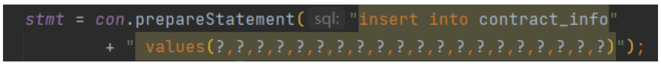


```
        cur_in.execute(
            f'insert into contract_info values({list
        con_in.commit()

cur_in.close()
con_in.close()
end = time.time() * 1000
```

For BadLoader:

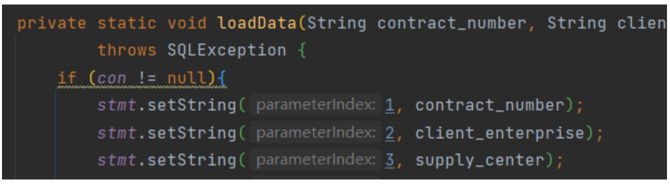
To compare with the VeryBadLoader, it improves that it has the preparedstatement.

In java:

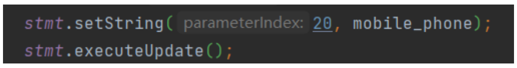


```
stmt = con.prepareStatement( sql: "insert into contract_info"
        + " values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");
```

then it can be used in

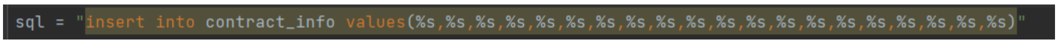


```
private static void loadData(String contract_number, String clien
        throws SQLException {
    if (con != null){
        stmt.setString( parameterIndex: 1, contract_number);
        stmt.setString( parameterIndex: 2, client_enterprise);
        stmt.setString( parameterIndex: 3, supply_center);
```



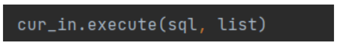
```
stmt.setString( parameterIndex: 20, mobile_phone);
stmt.executeUpdate();
```

In python:



```
sql = "insert into contract_info values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
```

then it can be used in

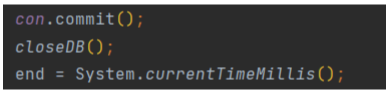


```
cur_in.execute(sql, list)
```

For AverageLoader:

To compare with theBadLoader, it improves that it commits the statements only once.

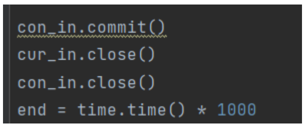
In java:



```
con.commit();
closeDB();
end = System.currentTimeMillis();
```

Commit and close the connection.

In python:



```
con_in.commit()
cur_in.close()
con_in.close()
end = time.time() * 1000
```

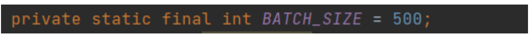
Commit and close the connection.

For GoodLoader:

To compare with the AverageLoader, it improves that it has the batch processing.

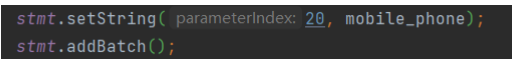
In java:

set the batch size:



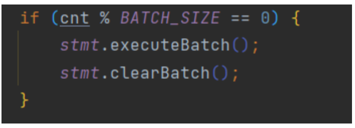
```
private static final int BATCH_SIZE = 500;
```

when loading data:



```
stmt.setString( parameterIndex: 20, mobile_phone);
stmt.addBatch();
```

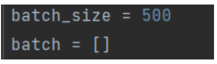
when executing data:



```
if (cnt % BATCH_SIZE == 0) {
    stmt.executeBatch();
    stmt.clearBatch();
}
```

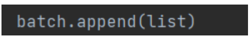
In python:

set the batch size:



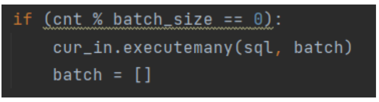
```
batch_size = 500
batch = []
```

when loading data:



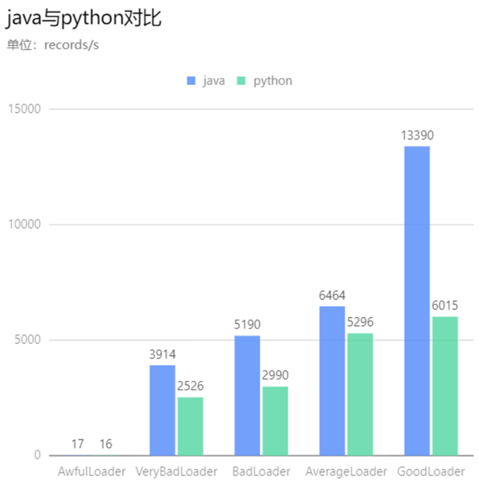
```
batch.append(list)
```

when executing data:



```
if (cnt % batch_size == 0):
    cur_in.executemany(sql, batch)
    batch = []
```

comparison between java and python:



use java threads to import:

Notation:

It is same as the notation for java in the above.

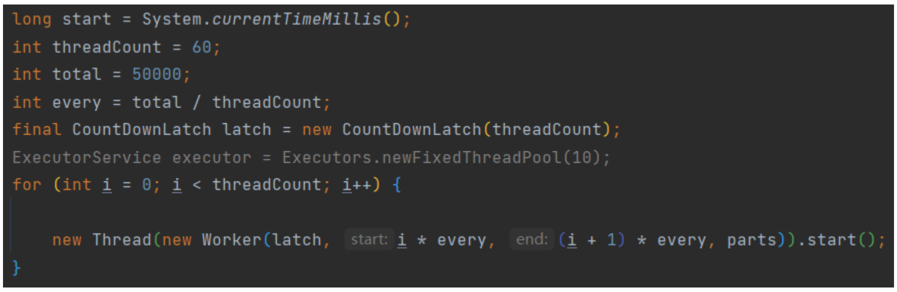
The commons:

This program is like VaryBadLoader, it commits every time the data has been loaded and doesn't have preparedstatement and batch processing. But it opens only at the beginning and closes only when all the data has been imported to the table.

The difference:

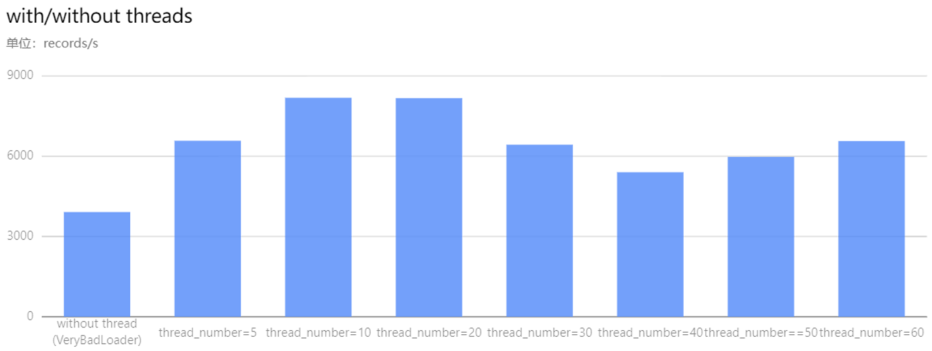
It use the thread pool to insert more then one statement each time.

This is adding the threads into the thread pool.



```
long start = System.currentTimeMillis();
int threadCount = 60;
int total = 50000;
int every = total / threadCount;
final CountDownLatch latch = new CountDownLatch(threadCount);
ExecutorService executor = Executors.newFixedThreadPool(10);
for (int i = 0; i < threadCount; i++) {

    new Thread(new Worker(latch,  start: i * every,  end: (i + 1) * every, parts)).start();
}
```

comparison between java without threads and with threads



Conclusion:

Task 4

Basic Requirements

1. A description of your test environment

CPU: 11th Gen Intel(R) Core(TM) i7-11800H

Memory: 16GB (3200MHZ)

Primary hard drive: Solid State Drive

Operating system: Windows 10,64-bit operating system, based on X64 processor

DBMS: postgreSQL-12.1-3

Programing language: Java

The development environment:

(1) the version of the language: java version "17.0.2" 2022-01-18 LTS
Java(TM) SE Runtime Environment (build 17.0.2+8-LTS-86)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.2+8-LTS-86, mixed mode, sharing)

(2) the specific version of the compilers: javac 17.0.2

2. A specification of how you organize the test data in the DBMS and the data file.

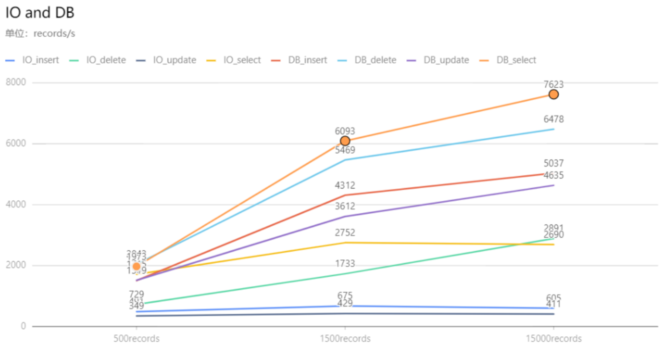
(1)the DDLs of the tables:

(2)the data format of the files: CSV

(3)How to organize the test data in the data file?

Firstly,I create a class named obIn to represent the object similar to entities in table,which contains same attributes with entities. Then, I use com.csvreader I have learned from the Internet which only contains standard file APIs (such as java.io). Each line means an obIn(object) so I new an obIn named ob to store the contents of the line. Next I add this obIn ob in an Arraylist . Finally I can operation on this Arraylist to simulate corresponding operations of database and complete instructions.

3.



Advanced Tasks

For High concurrency and transaction management

Basic imformation:(Copy form the Internet)

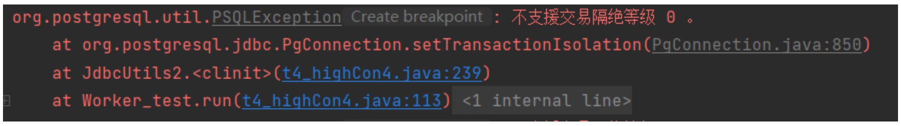
In order to solve the problem related to "multiple threads requesting the same data", transactions are usually isolated from each other with locks. And JDBCAPI supports different types of transactions, they are assigned or determined by the Connection object. The following 5 transaction isolation levels are defined in JDBC:

1. TRANSACTION_NONE_JDB does not support transactions.
2. TRANSACTION_READ_UNCOMMITED uncommitted read, indicating that a transaction can see another transaction change before committing. This level allows dirty reads, non-repeatable reads, and dummy reads.
3. TRANSACTION_READ_COMMITED has committed to read, indicating that reading uncommitted data is not allowed. This level allows non-repeatable reads and dummy reads.
4. TRANSACTION_REPEATABLE_READ Repeatable read, indicating that the transaction is guaranteed to be able to read the same data again without failure. Fixed non-repeatable reads, but false reads still occur.
5. TRANSACTION_SERIALIZABLE is serializable. The highest isolation level for transactions, which can prevent dirty reads, non-repeatable reads, and virtual reads.

Basic Settings:

Number of operations: 5000.
The sleep time between addition commit and subtraction commit is 100ms.

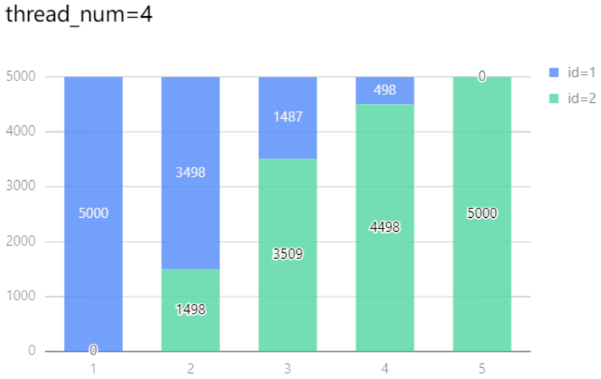
TRANSACTION_NONE_JDB



```
org.postgresql.util.PSQLException Create breakpoint : 不支援交易隔绝等级 0 。
    at org.postgresql.jdbc.PgConnection.setTransactionIsolation(PgConnection.java:850)
    at JdbcUtils2.<clinit>(t4_highCon4.java:239)
    at Worker_test.run(t4_highCon4.java:113) <1 internal line>
```

Connection.TRANSACTION_READ_UNCOMMITTED

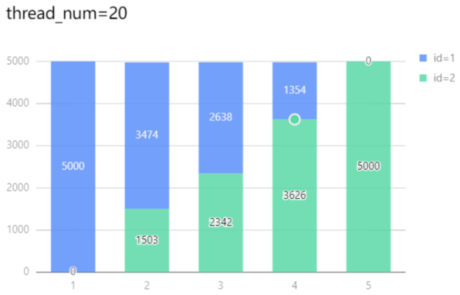
When thread_num=4



The sum of time2, time3, time4 is 4996 which is 4 less than 5000, exactly the number of threads.

And we can get that all the number in id1 are removed to id2.

When thread_num=20



The sum of time2, time3, time4 is 4996 which is 20 less than 5000, exactly the number of threads.

And we can get that all the number in id1 are removed to id2.

Conclusion:

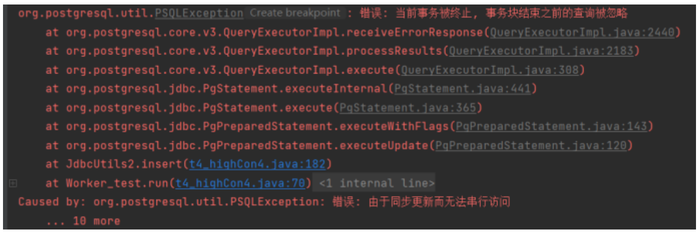
This transaction management allows multiple threads to access and may appear dirty read which is not safe.

TRANSACTION_READ_COMMITED

This have almost same result of TRANSACTION_READ_UNCOMMITTED.

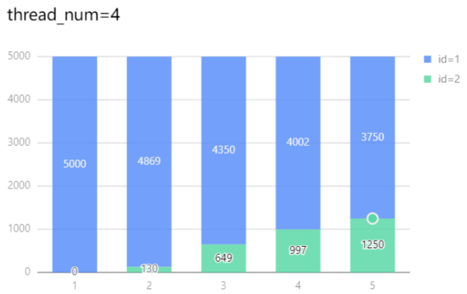
TRANSACTION_REPEATABLE_READ

when in this transaction management it will have error:



```
org.postgresql.util.PSQLException Create breakpoint : 错误: 当前事务被终止, 事务块结束之前的查询被忽略
    at org.postgresql.core.v3.QueryExecutorImpl.receiveErrorResponse(QueryExecutorImpl.java:2440)
    at org.postgresql.core.v3.QueryExecutorImpl.processResults(QueryExecutorImpl.java:2183)
    at org.postgresql.core.v3.QueryExecutorImpl.execute(QueryExecutorImpl.java:308)
    at org.postgresql.jdbc.PgStatement.executeInternal(PgStatement.java:441)
    at org.postgresql.jdbc.PgStatement.execute(PgStatement.java:365)
    at org.postgresql.jdbc.PgPreparedStatement.executeWithFlags(PgPreparedStatement.java:143)
    at org.postgresql.jdbc.PgPreparedStatement.executeUpdate(PgPreparedStatement.java:120)
    at JdbcUtils2.insert(t4_highCon4.java:182)
    at Worker_test.run(t4_highCon4.java:70) <1 internal line>
Caused by: org.postgresql.util.PSQLException: 错误: 由于同步更新而无法串行访问
    ... 10 more
```

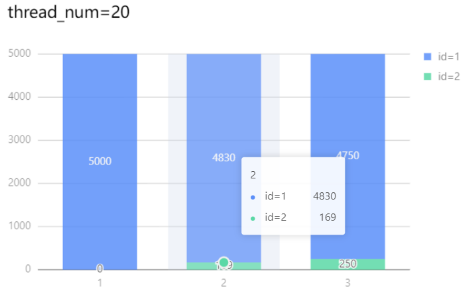
When thread_num=4



The sum of time2, time3, time4 is 4999 which is only 1 less than 5000, exactly no relation with the number of threads.

And we can't get that all the number in id1 are removed to id2.

When thread_num=20



The sum of time2 is 4999 which is only 1 less than 5000, exactly no relation with the number of threads.

And we can't get that all the number in id1 are removed to id2.

Conclusion:

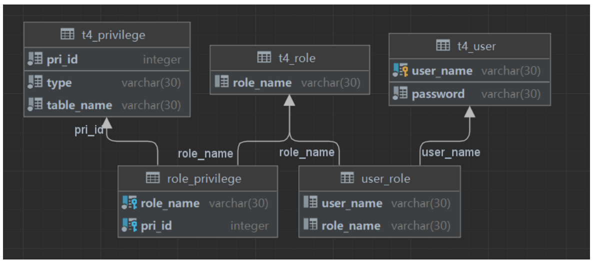
This transaction management allows only one thread to access at the same time, which is safe, but it will waste time and resources, and a transaction rollback mechanism is required after an error is reported to finally get the desired result.

User privileges management

Basic infomation:

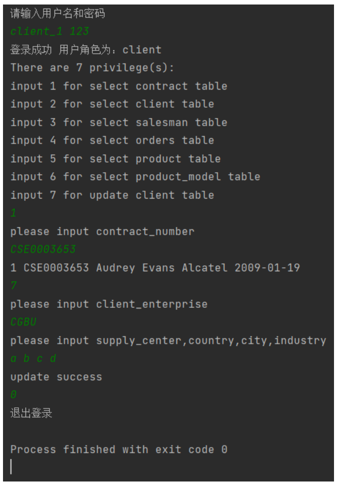
This privileges management is based at RBAC model. Role-based access control (RBAC) is a policy-neutral access-control mechanism defined around roles and privileges. The components of RBAC such as role-permissions, user-role and role-role relationships make it simple to perform user assignments.(refer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

The table i build:



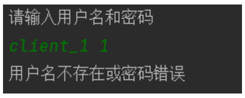
The performance of the privileges management:

For example:



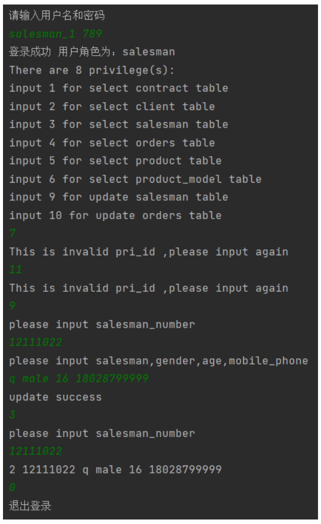
```
请输入用户名和密码
client_1 123
登录成功 用户角色为：client
There are 7 privilege(s):
input 1 for select contract table
input 2 for select client table
input 3 for select salesman table
input 4 for select orders table
input 5 for select product table
input 6 for select product_model table
input 7 for update client table
1
please input contract_number
CSE0003653
1 CSE0003653 Audrey Evans Alcatel 2009-01-19
7
please input client_enterprise
CGBU
please input supply_center,country,city,industry
a b c d
update success
0
退出登录

Process finished with exit code 0
```

However when the user are not in the user table or the passward is wrong:



```
请输入用户名和密码
client_1 1
用户名不存在或密码错误
```

For another example:

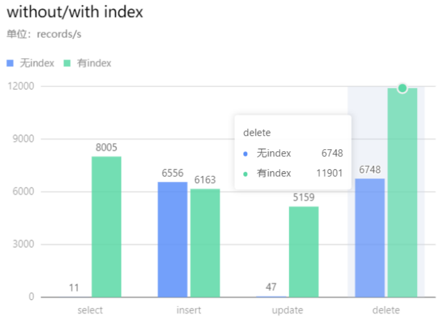


```
请输入用户名和密码
salesman_1 789
登录成功 用户角色为：salesman
There are 8 privilege(s):
input 1 for select contract table
input 2 for select client table
input 3 for select salesman table
input 4 for select orders table
input 5 for select product table
input 6 for select product_model table
input 9 for update salesman table
input 10 for update orders table
7
This is invalid pri_id ,please input again
11
This is invalid pri_id ,please input again
9
please input salesman_number
12111022
please input salesman,gender,age,mobile_phone
q male 16 18028799999
update success
3
please input salesman_number
12111022
2 12111022 q male 16 18028799999
0
退出登录
```

3 . Database index and file IO

The database index is data structure of sorting in the database management system (DBMS), used to assist in rapid query, update the data in the database table.

In order to test the usage of database index, we , magnitude increases the amount of data, and tests update, delete, insert, and select operations.



(1)We test select operation. Select 25593 pieces of data from 150000 pieces of data, which mainly complete by select and WHERE clauses. Select with index is almost 800 times than select without index. We can get a conclusion from the picture that index help speed up SELECT queries and WHERE clauses greatly.

(2)We test insert operation. Insert 450000 pieces of data into a table, which mainly complete by insert and WHERE clauses. Insert with index has similar speed with insert without index. we can guess there is a conclusion that when frequent operations operated on the columns with index, the index's function to speed up declines and insert operation is not closely relate to sorting, so index doesn't help o lot in insert.

(3)We test update operation. Update 50000 pieces of records in 150000 pieces of records, which mainly complete by update, set and WHERE clauses. Update with index is almost 100 times than update without index. We can think there is a conclusion that there is a conclusion that index speed up insert more than update.

(4) We test delete operation. Delete 50000 pieces of records in 150000 pieces of records, which mainly complete by delete and WHERE clauses. Delete with index is only 2 times than delete without index. Because when we delete data , frequent operations operated on the columns with index which is similar with insert operation, so the speed of deletion with index is slow, but as the amount of data it has changed is much smaller than insert , finally it is not slow like insert.

Indexes help speed up SELECT queries and WHERE clauses, but they slow down data entry when using UPDATE and INSERT statements. Indexes can be created or dropped without affecting the data.