Report for Project One of CS307

## Basic Information

Team of 12010903林雨航, 12012338曾宪清 in Lab 2.

|  |  |  |
| --- | --- | --- |
|  | 林雨航 | 曾宪清 |
| Task 1 | 15% | 0 |
| Task 2 | 25% | 0 |
| Task 3 | 0 | 25% |
| Task 4 | 10% | 25% |

## Task 1: E-R Diagram

I followed instinct and made a very different design from the E-R diagram above at beginning. But it is hard to draw the E-R diagram for it. Then I find that textbook shows the translation from E-R diagram to relationship in section 6.7, so I think I should make E-R diagram first then translate E-R into postgresql code, rather than make database design first then draw a E-R for it. That is, task2 should be done at the same time with task 1. Hence, I start all over again.

I find a very useful theory in section 6.9.1: if we can use the primary key of an entity

set as an attribute of another entity set, then there must be a relationship set between two entity sets. Following this theory, I design the E-R diagram below.

The diagram is generated from website <https://www.freedgo.com/>, on 2022/4/4.

图示

描述已自动生成

In section 6.5.2 of textbook, it writes that “we permit only one arrow out of a nonbinary relationship set”. Hence the drawing of “Contract” and “Order” above is incorrect in theory.

What needs to be emphasized is that I do have tried to solve this problem.

In section 6.5.2, textbook says “A simpler approach is to use functional dependencies, which we study in Chapter 7 (Section 7.4)”, but after some attempt to reading materials at section 7.4, I found it too difficult to learn functional dependency only by textbook.

Also in section 6.5.2, textbook suggest to “treat each instance of the non-binary relationship set as an entity”, but the book does not explain the drawing method of connecting two tables directly by lines in E-R, nor explain whether this drawing is legal.

In section 6.9.4, textbook offers an example to transmit a many-to-many ternary relationship into three binary many-to-one relationship. But after thinking, I think this method can’t be used in my problem. Textbook also writes “There may not be a way to translate constraints on the ternary relationship into constraints on the binary relationships” and gives an many-to-one example later, which is like my problem.

图示

描述已自动生成 So I asked this problem in QQ group. Yuxin Ma says just need to put mapping cardinality on the line, like examples in slides. Later he offers an example of drawing on one website:

Since the cardinality of 1 can be replaced by a directed line, then I maintain the drawing method at the beginning (the E-R diagram above). But to prevent ambiguity, I still decide to explain each relationship set in my E-R diagram:

**model\_class**: An entity in *product\_class* is associated with any number (zero or more) of entities in *product\_model*. An entity in *product\_model*, however, can be associated with at most one entity in *product\_class*.

**order**: A particular combination of entities from *header, product\_model* can be associated with at most one combination of entities from *sales* and *attached information*. A particular combination of entities from *sales* and *attached information* can be associated with any number of combinations of entities from *header* and *product\_model*. Thus, the primary key for the relationship *Order* is the union of primary keys in *header* and *product\_model*.

**contract**: A particular entity from *header* can be associated with at most one combination of entities from *client* and *supply.* A particular combination of entities from *client* and *supply* can be associated with any number of entity from *header*. Thus, the primary key for the relationship *contract* is the primary key of *header*.

## Task 2: Database Design

Because of our special thinking about task1 and task2, the work of task2 has already been done in task1, especially primary keys for each relationship. The rest of this phrase is just to translate E-R diagram into postgresql code. All entity sets with their primary keys are shown in E-R diagram clearly, and all relationship set with their primary keys have been explained in task 1, so no more repetition here. Instead, we introduce our thought about E-R diagram.

Looking into **contract\_info.csv**, it can be clearly seen that this table consists of four main parts:

|  |  |
| --- | --- |
| Meaning of table | Content |
| unique contract information | contract number, contract date |
| supplier in SUSTC | supply center, director  (this relationship is pointed out in the project requests) |
| client who purchases goods in SUSTC | client enterprise, city, industry, country |
| salesman for client | salesman number, salesman name, gender, age, mobile phone |
| purchased product | product code, product name, unit price |

Combining the contract in real life, we created a relationship set to connect these parts. (The connected part does not include salesman because we do not find explicit relation between salesman and contract number in **contract\_info.csv**)

|  |  |
| --- | --- |
| Meaning of table | Content |
| relationship set to connect contract, supplier and client | contract number, supply center, client enterprise |

Then we look **contract\_info.csv** more carefully and find that one contract number can be associated with more than one same product code, but only associated with one different product model. Also, unit price may be different for different model even these models belong to the same product code. So, we break “purchased product” into two more smaller part, and add one relationship set to connect them:

|  |  |
| --- | --- |
| Meaning of table | Content |
| the specific model of product | product model, unit price |
| the abstract classification of product | product code, product name |
| relationship set to connect them | product model, product code |

Now look back at what have not been classified: the information of salesman, quantity of products, estimated delivery date and lodgement date of product. Among these entities, quantity of products is associated with product model in each contract number. So we create another relationship set to connect all remains together:

|  |  |
| --- | --- |
| Meaning of table | Content |
| the order in one contract | product model, contract number, salesman number, quantity, estimated delivery date, lodgement date |

Now, all of design of database is finished, corresponding resultant E-R diagram is shown in task 1. After checking, all of designs meets the requirements in project file.

Easy to see, most of type is *varchar*. The exceptions are below:

*date* : contract date, estimated delivery date, lodgement date

*int* : unit price, quantity, age (of the salesman)

Since only city and lodgement may be NULL, so we set all other attributes to NOT NULL.

To be more rigorous, we set maximal length for every varchar. For some fixed-length variable like phone number, we set its type into char. Most variables, like country, do not have an explicit maximal length, we sort the content of this variable in **contract\_info.csv** in the order of decreasing length to find the maximal length in the table. Then we look back at the code examples in each lab class. Combine **contract\_info.csv** and lab together, we set the maximal length for each variable as below.

|  |  |
| --- | --- |
| Variable | Type |
| contract number | char(10) |
| client enterprise | varchar(100) |
| supply center | varchar(30) |
| country | varchar(50) |
| city | varchar(20) |
| industry | varchar(50) |
| product code | varchar(7) |
| product name | varchar(100) |
| product model | varchar(100) |
| unit price | integer |
| quantity | integer |
| contract date | date |
| estimated delivery date | date |
| lodgement date | date |
| director | varchar(30) |
| salesman name | varchar(50) |
| salesman number | char(11) |
| gender | varchar(6) |
| age | integer |
| mobile phone | char(11) |

**Task 3: Data Import**

To complete this task, I firstly reference the “data-import-material” offered in sakai. In the video, 4 optimisation methods are summarised to improve the efficiency of importing data: 1. only connect and close databse once 2. use the *preparestatement* instead of *statement* 3. Batch processing mechanism 4. commit only once after all statements have been executed.

However, we cannot get each column in and insert required information into 9 tables designed by ourselves, since it will lead to the Error:” Duplicate key violates unique/primary constraint”. To avoid such error, I decided to split contract\_info.csv into 9 corresponding .csv files as required in 3 steps.

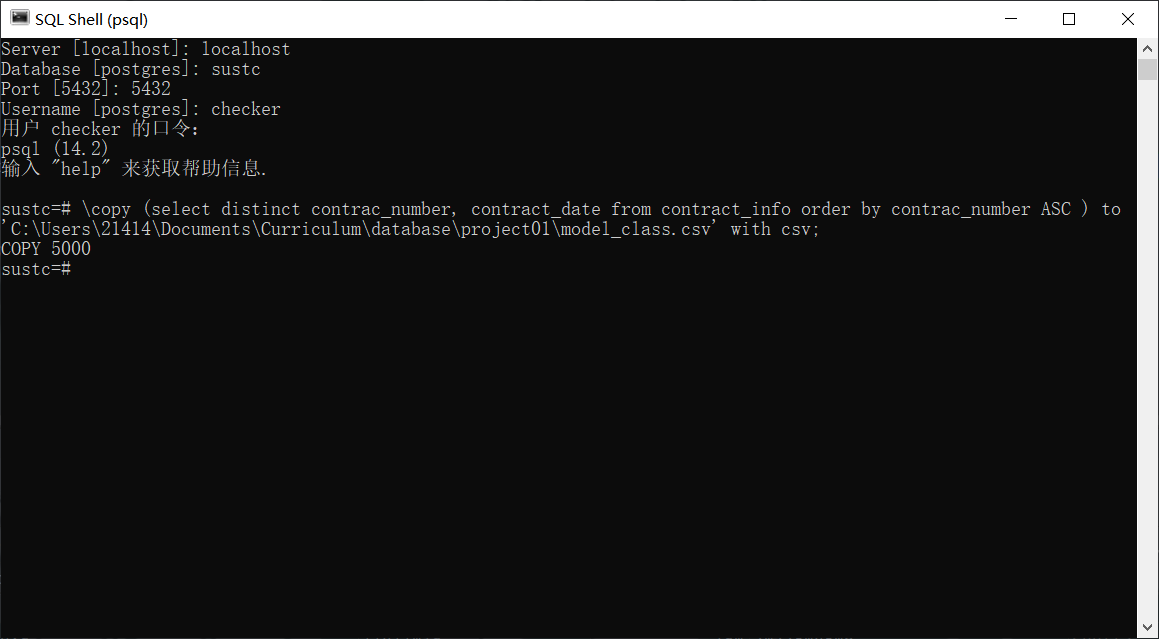
1. To get accessed to all given data, I created a corresponding table *contract\_info* in database, which has the same 20 columns. Based on the above improvement, I wrote *CSVLoader.java* to import the whole data into the table *contract\_info* reference to the given *goodloader.java.* Within the *loadData* method, there are two particular points to note:

① if the city is “NULL” or the lodgement date is blank in files, we should set them to *null*

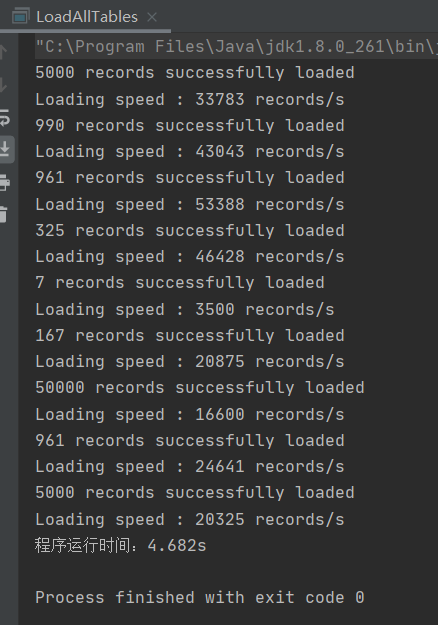
1. **if** (parts[2].equals("NULL"))
2. stmts[5].setNull(3, Types.VARCHAR);
3. **else**
4. stmts[5].setString(3, parts[2]);

② the date information should be transformed from *String* into *java.sql.Date.*

1. java.sql.Date cont\_date = **new** java.sql.Date(**new** java.util.Date(parts[1].replace("-", "/")).getTime());
2. Call the *\copy* command in psql to get 9 splited files.

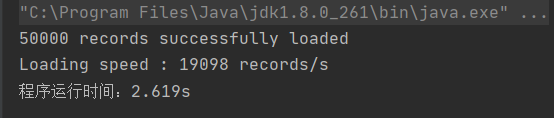


1. Write *LoadAllTables*.java to import 9 files obtained in step 2 into corresponding tables. The actual time cost is as followed:

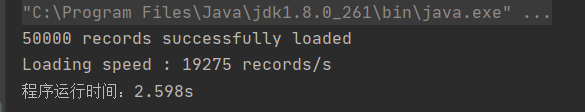


After successfully importing data, I tried to achieve in other ways.

1. Firstly, I consider improvement in file reading. The original *goodloader.java* apply *BufferedReader* to get accessed to the file. Thus I rewrote *CSVLoader.java* to get *LoadWithCSVpackage.java* with importing com.csvreader.CsvReade and get file input via CsvReader. The import efficiency of two program are showed as followed, however, there is no much difference.



CSVLoader.java

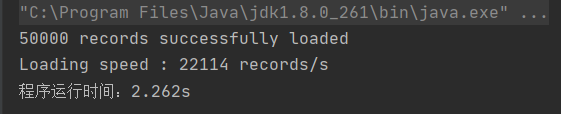


LoadWithCSVpackage.java

1. Since the execute t \copy command in psql is executed very rapidly, I considered implement the command in java program with CopyManager class.
2. file = **new** FileInputStream(filename);
3. String copy = "COPY " + "contract\_info" + " FROM STDIN DELIMITER AS ','";
4. copyManager.copyIn(copy, file);

The runtime cost is as followed:

We can see that the speed of single operation and the whole runtime are both improved.



(All the above programs are timed from the first statement of the main method until the last statement )