Database Management Systems(COP 5725) Project

Instructor: Dr. Daisy Zhe Wang

TA:
Kun Li, Yang Chen, Yibin Wang
kli, yang, yibwang@cise.ufl.edu

November 9, 2012

Department of Computer and Information Science and Engineering
University of Florida

Association Rules Mining

1 Project description

1.1 Project Overview

For this project, you need to use Java to access an Oracle database and accomplish the association rule data mining task using the Apiriori algorithm. **Please do your own implementation. We will be checking similarities between the solutions.** For this project, you will be asked to write code that mines frequent itemsets and association rules. The data set consists of two database tables:

1. Trans(TransID, ItemID)
2. Items(ItemID, ItemName)

The first table lists all of the database transactions that you will be mining, and the second table gives the actual names for the items that you are mining (such as ”Merlot Cheddar”, ”Red Potatoes”, etc.) Here are some of the vocabulary you'll need to understand this project:

- **Itemset** is a set of database items.
- **Transaction** is a set of database items that were purchased together, perhaps as part of a cash register transaction.
- **Frequent Itemset (FI)** is a set of database items that all appear in at least s% of the database transactions, where s is the user-defined support level.
- **Candidate Frequent Itemset** is an itemset that might be frequent, but we have not yet counted how many transactions it appears in.
- **Apriori Rule** is a rule that is used to construct candidate FIs of size i + 1 from all FIs of size i.
- **Association Rule (AR)** is a rule of the form {Merlot Cheddar, Red Potatoes} $\rightarrow$ {Romano}. All of the items in the rule must appear at least s% of the time (that is, all of the items in the association rule must together constitute an FI). Also, if the left-hand-size of the rule is present in a transaction, then the right-hand-side must also be present c% of the time, where c is is the user-specified confidence.

1.2 Tasks

There are 4 tasks you need to finish in this project:

- In Task 1, given a specific support level(such as s = 45%), your program should return all of the (single)items in the database that appear at least s% of the time. So your output might be:The following items appear in 45% of the database transactions: {Merlot Cheddar}, {Romano}, {Red Potatoes}.

- In Task 2, given a specific support level(such as s= 45%), your program should return all of the single items or pairs of items that appear in the database at least s% of the time. So your output might be: The following items appear in 45% of the database transactions: {Merlot Cheddar}, {Romano}, {Red Potatoes}, {Merlot Cheddar, Romano}, {Merlot Cheddar, Red Potatoes}
- In Task 3, given a specific support level and the maximum FI size, your program should find all of the frequent itemsets that are no larger (in terms of the number of items that they contain) than the user-specified size.
- In Task 4, this task is about mining association rules. Given a specific support level, a specific confidence and the maximum number of items in an association rule, your program would then first find all frequent itemsets at the user-specified size, and then use those FIs to generate all possible association rules.

1.3 A Quick Note and Hints

You are not allowed to maintain and manipulate data structures in Java, if you can avoid it. Use the database as much as you can!

For example, one of the things that you'll need to do to is to apply the "apriori rule" to obtain candidate FIs of size (i + 1) from all of the actual FIs of size i. This rule states:

"An itemset of size (i + 1) cannot appear in s% of the database transactions unless all of its subsets of size i appeared in s% of the database transactions."

Of course, you can do this by writing a bunch of code in Java. But you can also do it by sending SQL queries off to the database, which could be a heck of a lot easier. For example, imagine that you have created a database table FISet(ISetID, ItemID) that you use to hold all of the FIs of size i (that is, each FI of size i has a unique ISetID value, and the i different items contained in each itemset are spread across i different tuples in FISet; those i tuples all have the same ISetID). The first thing that you'll want to do is to find pairs of size-i FIs that can be "glued together" to find a candidate FI of size i + 1 (for example: the itemsets ABCD and BCDE can be glued together to form ABCDE). You can find all such pairs with the following SQL query:

```
SELECT DISTINCT IS1.ISetID AS ID1, IS2.ISetID AS ID2
FROM FISet IS1, FISet IS2
WHERE IS1.ISetID > IS2.ISetID AND i + 1 =
      (SELECT COUNT (DISTINCT ItemID)
       FROM FISet IS3
       WHERE IS1.ISetID = IS3.ISetID OR IS3.ISetID = IS2.ISetID);
```

Given a specifc ID1, ID2 pair, you can get the list of ItemIDs that results from combining them with the following query:

```
SELECT DISTINCT ItemID
FROM FISet IS
WHERE IS.ISetID = ID1 OR IS.ISetID = ID2;
```

Once you have such a candidate itemset, but before you actually do to the database to see how many times it appears, you'll want to verify that all of its subsets of size i are also FIs – if they are not, then according to the apriori rule, the candidate itemset cannot possibly be an FI. You can do this check using the database as well. If all of the candidate itemsets's item IDs are stored in the database table TEMP, then the following query will tell you how many of its size-i subsets are FIs:

```
SELECT COUNT(*)
FROM (SELECT FISet.ISetID
      FROM TEMP, FISet
      WHERE TEMP.ItemID = FISet.ItemID
      GROUP BY FISet.ISetID
      HAVING COUNT(*) = i);
```

If this query does not come back with the number i + 1, then the candidate itemset that is in TEMP cannot possibly be frequent (because it has broken the apriori rule) and you don't have to see how many times it appears in the database.

These are just a few examples of how, by using the database to manage your data, you may be able to reduce the amount of coding that you need to do in the external language.

2 Input and Output

You must exactly follow the input and output format described in the following subsections. If you failed to follow the format, your program may not be able to parse the input file and the TAs' grading script may not be able to interpret your output. Also the input files, output files and your code files should be in the same directory. Don't add any extra file path except the file name to the path argument when program tries to read/write some files.

2.1 Input

There are three input files, namely items.dat, trans.dat and system.in.
The system.in contains one and only one test case for each task.
The first line in the system.in contains the your credential to access the oracle database.
The second line, third line, fourth line, fifth line are the parameters for TASK1, TASK2, TASK3, TASK4 respectively.

```
username = your_user_name, password = your_password
TASK1: support = 6%
TASK2: support = 7%
TASK3: support = 7%, size = 3
TASK4: support = 2%, confidence = 95%, size = 3
```

2.2 Output

There is one output file for each task. Your program should generate four output files, namely, system.out.1, system.out.2, system.out.3 and system.out.4.

Sample output for TASK1:

```
{Merlot Cheddar}, s=45%
{Red Potatoes}, s=57%
```

Sample output for TASK2:

```
{Merlot Cheddar}, s=10%
{Merlot Cheddar, Romano}, s=15%
{Merlot Cheddar, Red Potatoes}, s=20%
```

Sample output for TASK3:

```
{Merlot Cheddar, Romano}, s=15%
{Merlot Cheddar, Red Potatoes}, s=20%
{Merlot Cheddar, Red Potatoes, Romano}, s=21%
```

Sample output for TASK4:

```
{{Merlot Cheddar, Red Potatoes} – > {Romano}}, s=10%, c=11%
{{Merlot Cheddar, Diaper} – > {Beer}}, s=11%, c=33%
```

2.3 Project Report

You are required to submit a 1-2 pages TXT report. Name it report.txt. Note the small case 'r' and NOT capital 'R'. Your report must contain a brief description telling us how you developed your code, what difficulties you faced and what you learned. Remember this has to be a simple text file and not a MSWORD or PDF file. It is recommended that you use vi or gedit or pico/nano applications in UNIX/Linux to develop your report.

2.4 Additional Instructions

1. You should tar all you java code files including the txt report in a single tar file using this command on Linux/SunOS:
   tar cvf arm.tar <file list to compress and add to archive>

2. Your project must be named 'arm.tar' and also your java file which contains the main function must be named 'arm.java'.

3 Automatic Testing Script

The TAs will run the following script to grade your program on the CISE Linux machines. We will use the same data files as the provided data files. But the output files will not be provided.

```
#!/bin/bash
source /usr/local/etc/ora11.csh
tar xvf arm.tar;
./plagiarism-detector
rm *.class system.out.* report.txt
cp /path/to/item.dat .
cp /path/to/trans.dat .
javac -cp ojdbc5.jar arm.java
```

```
cp /path/to/system.in .
cp /path/to/item.dat .
cp /path/to/trans.dat .
cp /path/to/correct.system.out-1 .
cp /path/to/correct.system.out-2 .
cp /path/to/correct.system.out-3 .
java -cp ojdbc5.jar:. arm
java autograder
cat report.txt
```

4 Grading Guidelines:

It is your responsibility to make sure you follow the the input and output format. Your project will be graded 0 if either your program cannot read the input files provided by the TAs or the TAs' autograder cannot parse your output files.

- TASK1 15%
- TASK2 15%
- TASK3 15%
- TASK4 30%
- Project report 25%

5 Late Submission Policy:

Late submissions are not encouraged but we rather have you submit a working version than submit a version that is broken or worse not submit at all. Every late submission will be penalized 10% for each day late for up to a maximum of 3 days from the due date.

6 Helpful Resources:

6.1 Access CISE Oracle database

You need to create a CISE Oracle account to access the CISE Oracle database. Please refer to the following CISE website:
http://www.cise.ufl.edu/help/database/oracle.shtml
However some information in the above website is not up to date. The location for the jdbc driver is no longer valid.
Note: To use Oracle's JDBC on Department Solaris machines, add the following to your CLASSPATH: /usr/local/libexec/oracle/app/oracle/product/11.2.0/client_1/ojdbc5.jar.
The solution for you is just downloading the attached 'ojdbc5.jar' and put it into your source code.

6.2 How to execute sql script in Java

You will need to write procedures to implement the core part of this project. The following two lines of code enable you to import sql script containing procedure code to the database.
Process p = Runtime.getRuntime().exec("sqlplus username@orcl/passwd @arm.sql");
p.waitFor();
Please append "exit;" to the end of your procedure code file 'arm.sql'. This line of code enable your to exit the sqlplus environment.

6.3 Tutorials

You can have a look at the following tutorials that might be helpful in doing this project:

1. http://en.m.wikipedia.org/wiki/Apriori_algorithm#section_language
2. http://infolab.stanford.edu/ ullman/fcdb/oracle/or-jdbc.html
3. http://docs.oracle.com/cd/B25329_01/doc/appdev.102/b25108/xedev_jdbc.htm
4. JDBC is covered a bit in your textbook, there are tons of books on JDBC at UF libraries, and Google will get you more info on JDBC than you ever wanted to know.