Fall 2011

Data Mining and Data Warehouse ------ Maximal Simplex Method

Madhuri Gollu

Data Mining and Data Warehouse

------ Maximal Simplex Method

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Madhuri Gollu

Dec 2011

SAN JOSÉ STATE UNIVERSITY

The Undersigned Writing Project Committee Approves the Writing Project

Titled

Data Mining and Data Warehouse – Maximal Simplex Method

By

Madhuri Gollu

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Tsau Young Lin, Department of Computer Science 11/22/2011

Dr. Soon Tee Teoh, Department of Computer Science 11/22/2011

Mr. Sridhar Dhavala, Senior Software Engineer, Wells Fargo11/22/2011

ACKNOWLEDGEMENTS

I am very grateful to my advisor Dr. Tsau Young Lin for his continuous support, guidance, encouragement, and valuable advices to my project. I would like to thank my committee members Dr. Soon Tee Teoh and Sridhar Dhavala for their valuable suggestions to my project. I also want to thank my husband, Mr. Kishore Bankuru and my brother, Mr. Kiran Kumar Gollu for their valuable suggestions, encouragement and support thorough out the project and masters program.

Abstract

Association Rule Mining is a widely used method for finding interesting relationships from large data sets. The challenge here is how to swiftly and accurately discover association rules from large data sets. To achieve this, this paper will (1) build a data warehouse system that simulates the secondary storage and represents a database by bit patterns, and (2) implement a new geometric algorithm to find association rules, called Maximal Simplex Algorithm. The data warehouse consists of very long bit columns. Each column is an item or an attribute value pair and a row represents a transaction or a tuple in a database. A bit value 1 in a row represents the transaction contain this item or the tuple contains this value. In this Maximal Simplex Algorithm, we interpret the set of bit columns as a set of independent vertices in a high dimension Euclidean space. The main idea is for each vertex, we find its star neighborhood, namely to find all simplexes that contains this vertex. An n-dimensional simplex is called n-simplex. An n-simplex represents the association rule of length n+1. Based on the experimental results, Maximal Simplex method improves the performance of association rule mining. And also it is possible to achieve parallel computing by using the data warehouse system.

Table of Contents

List of Tables

List of Figures

1. Introduction

Data Mining can be defined as the retrieval of hidden patterns from the large databases. Data mining is useful to predict the future behavior and proactive the business. The relationship among two sets can be expressed as an association rule in a transaction database. For example, (A} -> (B} is a rule and can be explained as follows: If A is purchased in a database transaction, it is possible that B can also be purchased in the same database transaction.

The enigma of association rule mining is to extract all the association rules whose support and confidence are greater than the user-specific minimum value. The problem can be bifurcated into sub problems. (1) Discovering all items combinations whose transaction support is greater than the user-specific minimum support (2) utilize the large item sets to produce the desired association rules. Most of the data mining algorithms were fall into these two categories.(1) candidate-generation-test approach (2) pattern growth approach . Apriori follows the (1) approach, which requires multiple scans of the database. FP-Growth tree the (2) approach, which requires scanning the whole database twice. Data Warehouse is a source of data for mining algorithms. In Data Warehouse, we store the bit pattern table transactions as 32-bit integer values in the files.

In this paper, I have done research on two areas (1) building a data warehouse system which simulates the secondary storage (2) implement a maximal simplex method algorithm. In data warehouse system database data is represented in bit

pattern. Bit computation is fast and effective to find the association rules. By accessing data from data warehouse system avoids the multiple scans of the database to find large item set association rules. By using this system, we can achieve better performance of the mining algorithms.

In section 2, mathematical preliminary of the implemented algorithms is discussed. In section 3, building a data warehouse system and database bit pattern representation is discussed. In section 4 discusses about the FP-Growth Tree algorithm. In section 5, proposes Maximal simplex method algorithm and its explanation with example. The computation results of Maximal simplex method are presented in section 6. Section 7 provides a conclusion and future work followed by references.

2. Mathematical Preliminary

> A finite abstract simplicial complex is a set of objects, called vertices, a^1… a^{α}, and a set K of subsets s^i_p of the vertices, where (p+1) is the number of vertices in the simplex s^i_p and i is an indexing superscript; the simplexes of K satisfy the condition that any subset of a simplex of K is also a simplex of K. The dimension if s^i_p is *p* and the dimension of K is the largest of the dimensions of its simplexes. (Hilton, P.J., 1962, p. 41)

In Maximal Simplex Method, we first the high frequency items which are called as vertices for the Simplex. Start with vertex A and build the 2- dimension simplex, 3-dimension simplex and so on till n-dimensional simplex. Once we finish building the top dimension simplex for vertex A, any subset of vertex A simplex is also considered as a simplex of vertex A.

3. Building a Data Warehouse System

Association rule mining is a time-consuming task while accessing data directly from the database. To avoid costly database scans, build a data warehouse system using bit pattern tables. The bitmap technique has been popularly used by a variety of products and it was proposed in 1960's.Bit pattern representation is useful to perform fast bit operations like logical (AND, OR, or NOT).Data

warehouse system reduce the program running time while accessing data and this can be used to achieve parallel computing.

3.1 Algorithm:

a. Convert the raw database table into new bit pattern table. Each column in old table can be converted as multiple columns based on the distinct values of the column.

b. Change all the transactions values to either 1 or 0. The column values are set to 1 whose transactions have the corresponding attribute value.

c. Read 32 bits of column at a time and convert that as an integer value and store in the disk file until it reaches the end of transactions.

3.2 Conversion of database into bit pattern table

Below table is showing information about Car name and Car Types.

Item No	**Car name**	**Car Type**
C1	Ford Focus	Compact
C2	Honda Accord	Full-size
C3	Acura TL	Mid-size
C4	Nissan Altima	Mid-size
C5	BMW Z-series	Sports car
C6	Toyota Yaris	Subcompact

Table 1: Car Database table

Convert the original table into bit pattern table by using above mentioned algorithm. The above table contains six different car names and car types. In the below table, each car name and car type become a separate column for constructing the bit pattern table.

Car Name							**Car Type**			
Ford Focus	**Honda Accord**	**Acura TL**	**Nissan Altima**	**BMW Z-series**	**Toyota yaris**	**Subcompact**	**Compact**	**Mid-size**	**Full-size**	**Sports car**
1	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	0	1	1	0	0	0	0

Table 2: Converted Car bit pattern table

3.3 Disk File Structure

In this paper, the database bit pattern data is organized in disk files. Each disk file contains several control areas, which are called as virtual cylinders. Each column or attribute of the database is called a cylinder. Every control area has one sequence set and a number of control intervals, which are called as virtual tracks. The number of tracks in a control area is determined by the hardware characteristic of the machine. Each control interval contains the database transaction data.

While building data warehouse system, we have considered each control area contains seven tracks of data and each track can hold 4K data. Sequence set will have seven compartments and each compartment contains the disk pointer of each track starting address. Each disk pointer is 8 bytes long. So each sequence set requires 7 tracks * 8 bytes i.e. 56 bytes. Every control area holds 4K * 7 i.e. 28K data.

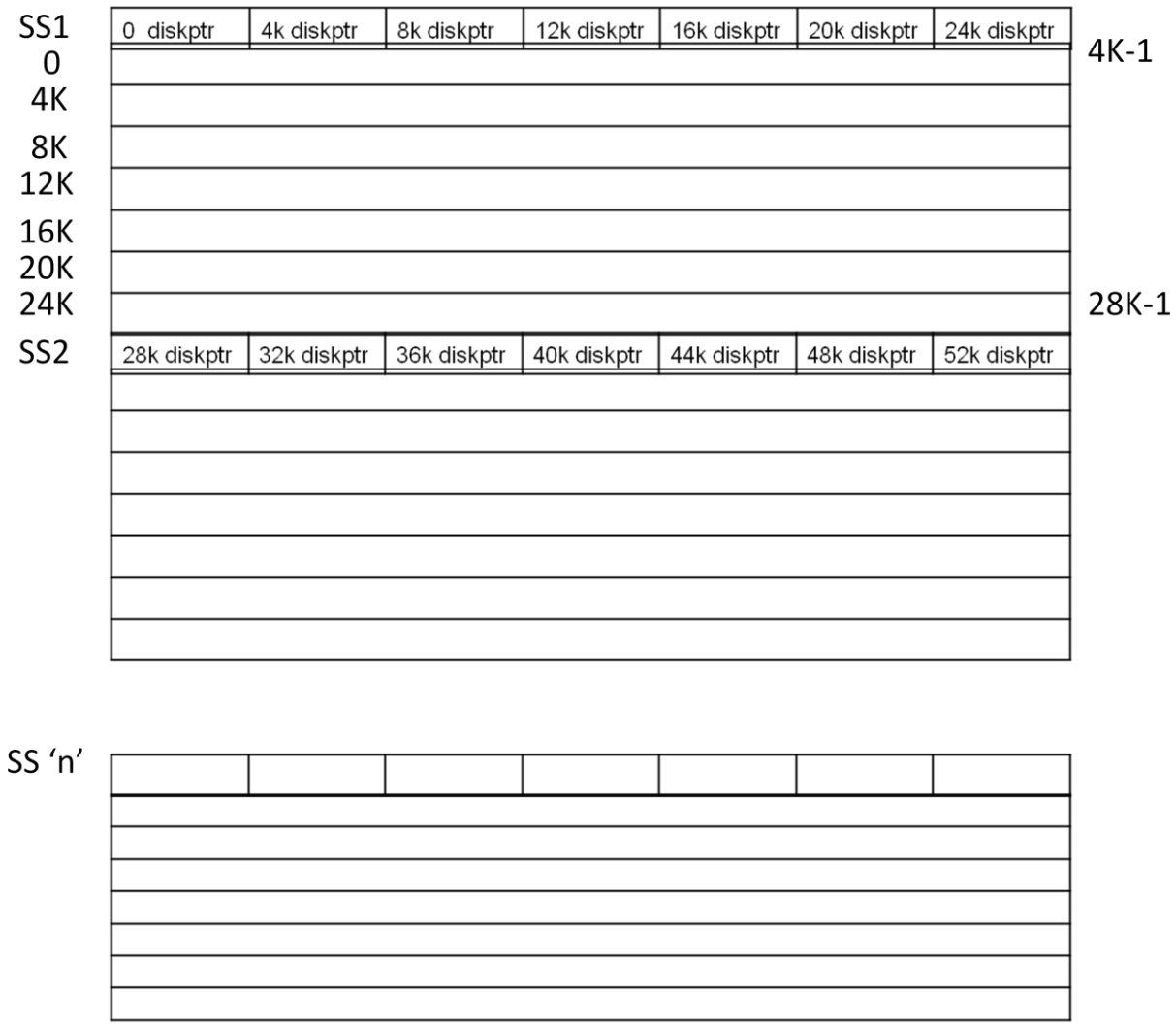

Figure 1: Disk File Structure

From the above figure, we notice that the disk file contains data of 'n' cylinders. First cylinder needs 56 bytes for sequence set1 (SS1) plus 28 K data where K = 1024. We assume first cylinder data is starting from 0 offset and ending with (28 K -1) offset.

3.4 Data Warehouse System Implementation Details

The following Java file is used to convert database table into simulated hard disk files.

File Name: Build_DataWarehouse.java

For accessing data from Oracle database used JDBC connection driver. Read each attribute data by querying the database and store the transaction data in the disk files in the form of integers. For testing the sample data build three different disk files and store the part database transaction data on each file and calculated the correct location of file seek pointers and store the data. Properly handled the file input output operations.

4. FP Growth Tree

4.1 Overview

FP-Growth Tree Algorithm follows different approach for finding frequent item sets. This Algorithm falls into divide and conquer approach. Instead, this algorithm builds the data structure called FP-Tree to store the data and directly extracts the item sets from the tree. Initially, FP tree holds the root node which is set to NULL. Each node contains two values. One is Item label and other value is frequency count of the item.

Let us discuss about the FP-Growth Tree Algorithm and FP-Tree Construction.

4.2 FP-Growth Tree Algorithm

1. In preprocessing step FP-Growth Algorithm scans the entire data set once and finds all the frequent items that is all the items that appear more than the support value. And remaining infrequent items which appear in fewer transactions will be discarded. Since, based on the frequency count infrequent items cannot be part of the frequent item set. Each transaction items need to sort in descending order based on their frequency in the database. In below mentioned example A is the most frequent item followed by B, C, D, E, F, X, Y, and Z.

2. The data set will be scanned one more time to construct the FP-Tree path for each transaction. First transaction gives the frequent item set as (A, B, C, D, E}, all the nodes will be created with their item name. A new branch will be created from null -> A -> B -> C -> D -> E with frequency count of 1.

3. The second transaction, (A, B, C, D, E}, shares a common prefix items. So the frequency count of the each node will be added by 1.

4. If the transaction doesn't contain the common prefix new nodes will created and linked with root of the tree.

5. This process will continue until every transaction in the data set has been mapped onto one of the branches in the FP-Tree.

4.3 FP-Tree Construction

A dataset contains nine transactions and eleven items named as A, B, C, D, E, F, G, H, X, Y, and Z. The items G and H will be discarded from the frequent item list since those two items are infrequent.

Item Name	**Frequency Count**
A	5
B	5
C	4
D	4
E	4
F	4
X	3
Y	3
Z	3
G	**0**
H	**0**

Table 3: Frequency count of the data set

Transaction ID	Items in the transaction	Sorted frequent items
1	**A, B, C, D, E**	A, B, C, D, E
2	**A, B, C, D, E**	A, B, C, D, E
3	**A, B, C, D**	A, B, C, D
4	**A, B, C, D**	A, B, C, D
5	**A, B, E, F**	A, B, E, F
6	**E, F, X**	E, F, X
7	**F, X, Y, Z**	F, X, Y, Z
8	**F, Y, Z**	F, Y, Z
9	**X, Y, Z**	X, Y, Z

Table 4: Data set transactions

The following is the process to build the FP-Growth tree. For first transaction, (A, B, C, D, E) branch will be drawn with count value as 1.

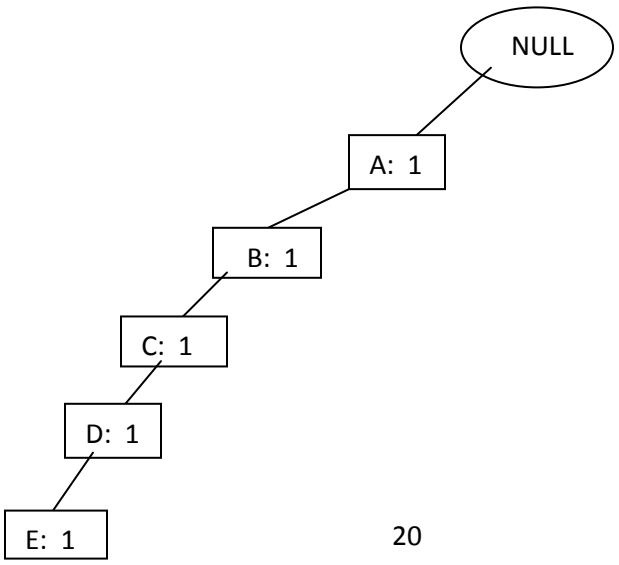

Figure 2: FP- Tree after first transaction

After second transaction, (A, B, C, D, E) share the common prefix (A, B, C, D, E) with first transaction. So each node counter will be updated.

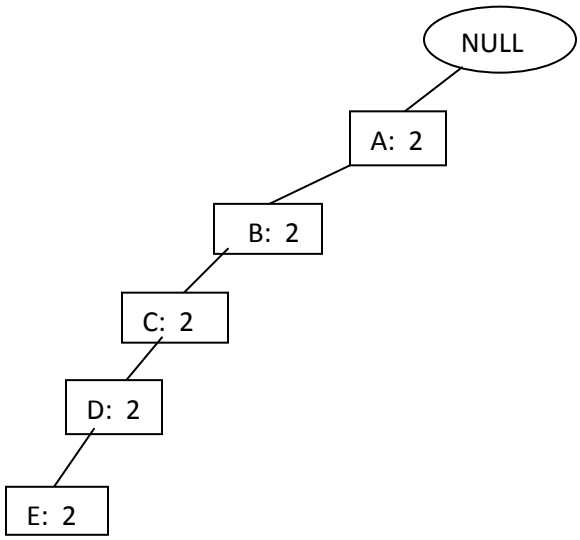

Figure 3: FP- Tree after second transaction

After fifth transaction (A, B, E, F), nodes (A, B) share the common path with previous transactions and then create (E, F) nodes with counter 1.

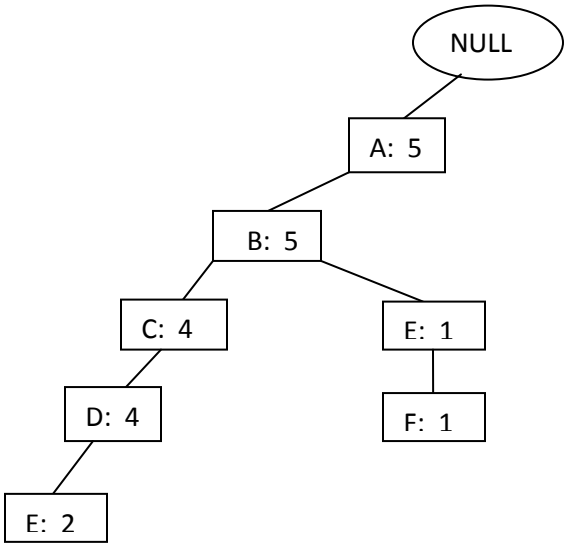

Figure 4: FP-Tree Construction after fifth transaction

The following we continue to construct the tree for remaining all transactions. The below drawn tree is the complete FP-tree for above mention dataset.

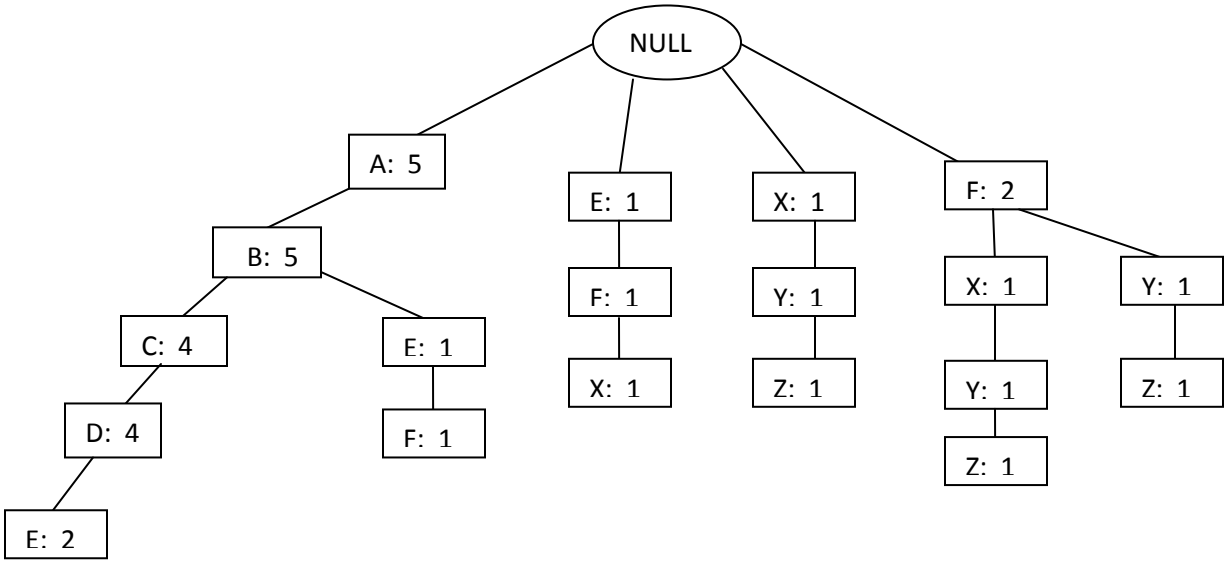

Figure 5: Complete FP-Tree after all transactions

4.4 Find Frequent Patterns from FP-Tree

4.4.1 FP-Tree with Node Links

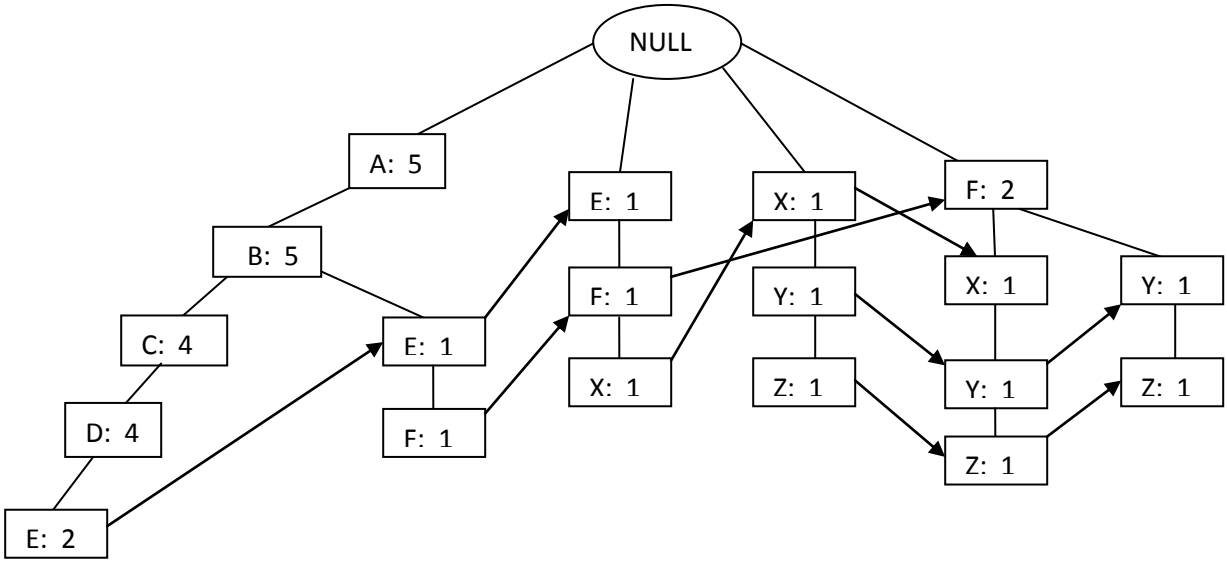

Figure 6: Complete FP-Tree with Node-Links

In FP tree each node contains Item name, count value and Node link pointer. The lines with arrow represents that those two nodes share the same item name. If the same item name is not shared among other branches of the FP-Tree, the node-link pointer contains null value. For example, (E:2) and (E:1) have a line connection between them since those nodes have the same item name.

4.4.2 Conditional FP Base and FP-tree construction

By using above figure we can determine Conditional FP Tree and Frequent patterns for each item name. We will start finding the frequent patterns with least frequency count item since it contains less number of node links. For example if we consider the node Z, FP-tree has three branches. Those are (X, Y, Z), (F, X, Y, Z), (F, Y, Z). From Z count value, (X, Y :1), (F, X, Y :1) and (F, Y :1). Those 3 three values become the conditional

FP-base for Z. From the above the relations we construct the Conditional FP-Tree. Here we are assuming minimum support value is 2.

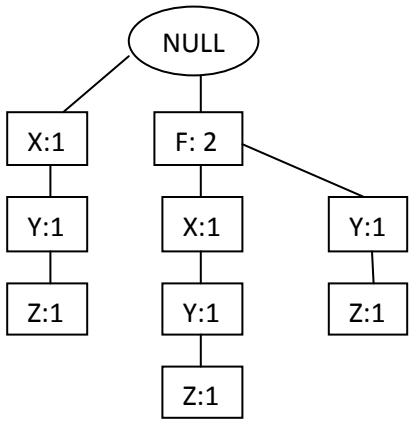

Figure 7: Conditional FP-Base for Item Z

From the above figure if we exclude (X:1) from (F, X, Y, Z) branch we will the frequent pattern (F, Y , Z : 2). If we separate (F, X, Y, Z) and (F, Y, Z) without sharing F node, We can combine (X, Y, Z) and (F, X, Y , Z) relations by excluding F. The frequent pattern will be (X, Y , Z :2). If we do the same the process for all the item names we find frequent patterns for complete FP-tree.

4.4.3 Frequent Pattern Table generated from FP-tree:

Item Name	Conditional FP-base	Conditional FP-Tree	Frequent Patterns
Z	{X:1, Y:1},{F:1, X:1, Y:1}, { F:1, Y:1}	{X :2, Y :2},{F:2, Y:2} | Z	X Z :2, Y Z:2, F Z:2, X Y Z:2, F Y Z:2
Y	{X:1},{F:1, X:1},{F:1}	{F:2},{X:2} | Y	F Y :2, X Y:2

X	{E:1, F:1},{F :1}	{F:2} | X	F X:2
F	{A:1,B:1,E:1},{E:1}	{E:2} | F	E F:2
E	{A:2,B:2,C:2,D:2}	{A:2,B:2,C:2,D:2} | E	A E:2, B E:2, C E:2,D E:2,A B E:2, A C E:2, A D E:2, B C E:2, B D E: 2,C D E:2,A B C E:2, A B D E:2, B C D E:2, A B C D E:2,A C D E:2
D	{A:4, B:4, C:4}	{A:4, B:4, C:4} | D	A D:4, B D:4, C D:4, A B D:4, A C D:4, B C D:4, A B C D:4
C	{A:4, B:4}	{A:4, B:4} | C	A C:4, B C:4, A B C:4
B	{A:5}	{A:5} | B	A B:5
A	0	0	-----

Table 5: Frequent Pattern generation table

5 Maximal Simplex Method

5.1 Overview

Association Rule Mining can visualize in geometric way. From this method we can determine all the frequent items with respect to each vertex (Attribute). In this method, scan the entire column data and perform bit operations for determining the association rules. Maximal Simplex method is related to the FP-Growth algorithm. The dimension of set of vertices is the largest dimension of its simplexes. A 1 vertex (0-simplex) is point, the connection between 2 vertices is open segment (1-simplex), 3 vertices is a triangle (2-simplex), 4 vertices is tetrahedron (3-simplex) etc. “The simplexes of K satisfy the condition that any subset of a simplex of K is also a simplex of K” (Hilton, P.J., 1962, 41) is a called a closed condition which is equivalent to Apriori mining algorithm.

5.2 Maximal Simplex Method Algorithm

Each vertex in the graph will be part of high frequency item whose count value is greater than the minimum support value. Every line between two vertices is the part of high frequency items. Start with first vertex and find all the direct connections to other vertices which will be called as ‘Star Neighborhood’ of that vertex. First we start with 2- dimensional simplex and then 3- dimensional simplex and so on till n-dimensional simplex. While build n-dimensional simplex, we recursively find the lower dimensional simplexes. That means we build the top level simplex for vertex

A. We will follow the same process for all vertices (high frequency items) to build the complete Simplex.

To achieve the complete simplexes graph we follow the below mentioned steps.

1. First Simplex method scans the entire data set and finds all the vertices (high frequency item names) for building the Simplex. And discard the low frequency item names since they never be part of Simplex (high frequent item set).

2. Sort the high frequency items in descending order. (High count value to low count value)

3. Read each transaction and keep the list of items whose value is set to 1 and sort the items according to the high frequency items order.

4. Take each transaction sorted items and perform the AND operation with each item. For example A, B, C, D, E is first transaction sorted items. First perform A AND B and if the count > support value, we consider that as a one of the association rule. Then perform (A AND B) AND C and check the count value. If the end result is less than support value discard C item. And continue performing AND operation with next item from the transaction items. i.e. (A AND B) AND D and (A AND B AND D) AND E. [We will use the already computed end result (A AND B), (A AND B AND D) while doing other AND operations.]

5. Continue the same process from step3-4 for all transactions and find all the association rules.

6. We will find all possible connections (lines) from one vertex to another vertex. Let us consider association rule from table ABCDE (**A** AND **B** AND **C** AND **D** AND **E**).

The possible 2- dimensional simplexes are AB, AC, AD, AE, BC, BD, BE, CD, CE, DE, 3-dimensional simplexes are ABC, ABD, ABE, ACD, ACE, ADE, BCD, BCE, BDE, CDE, 4- dimensional simplexes are ABCD, ABCE, BCDE, 5- dimensional simplex is ABCDE . Remaining all association rules are derived from the ABCDE rule. This means we generate the maximum possible dimension simplex for vertex A.

7. Continue the same method for all association rules to build the whole simplex.

5.3 Maximal Simplex Method Simplex Construction

A dataset contains nine transactions and eleven items named as A, B, C, D, E, F, G, H, X, Y, and Z. The below mentioned table is shown in the form of bit pattern and the last row represents the frequency of each attribute (vertex).

Item Name	A	B	C	D	E	F	**G**	**H**	X	Y	Z
1	1	1	1	1	1	0	**0**	**0**	0	0	0
2	1	1	1	1	1	0	**0**	**0**	0	0	0
3	1	1	1	1	0	0	**0**	**0**	0	0	0
4	1	1	1	1	0	0	**0**	**0**	0	0	0
5	1	1	0	0	1	1	**0**	**0**	0	0	0
6	0	0	0	0	1	1	**0**	**0**	1	0	0
7	0	0	0	0	0	1	**0**	**0**	1	1	1
8	0	0	0	0	0	1	**0**	**0**	0	1	1
9	0	0	0	0	0	0	**0**	**0**	1	1	1

Count	5	5	4	4	4	4	**0**	**0**	3	3	3

Table 6: Data set and Frequency count of each item

Only high frequent items will be considered while building simplexes of the attributes. The items G and H will be discarded from the frequent item list since those two items are infrequent.

Transaction Number	**Items Order**	**Sorted items based on high frequency order**	**Association Rules**
1	A, B, C, D, E	A, B, C, D, E	{(A, B), (A, C), (A, D),(A, E),(A, B, C), (A, B, D), (A, B, E), (B, D, E), (B, C, D), (B, C, E), (C, D, E), (A, C, D), (A, C, E), (A, D, E), (A, B, C, D), (A, B, C, E),(A, B, D, E), (B, C, D, E), (B, C), (C, D),

			(D, E), (B, D), (B, E), (C, E),(A, B, C, D, E)}
2	A, B, C, D, E	A, B, C, D, E	{(A, B), (A, C), (A, D),(A, E),(A, B, C), (A, B, D), (A, B, E), (B, D, E), (B, C, D), (B, C, E), (C, D, E), (A, C, D),(A, C, E), (A, D, E), (A, B, C, D), (A, B, C, E),(A, B, D, E), (B, C, D, E), (B, C), (C, D), (D, E), (B, D), (B, E), (C, E),(A, B, C, D, E)}
3	A, B, C, D	A, B, C, D	{(A, B), (A, C), (A, D),(A, B, C), (A, B, D) , (A, C, D), (B, C, D), (A,

			B, C, D), (B, C), (C, D), (B, D)}
4	A, B, C, D	A, B, C, D	{(A, B), (A, C), (A, D),(A, B, C), (A, B, D) , (A, C, D), (B, C, D), (A, B, C, D), (B, C), (C, D), (B, D)}
5	A, B, E, F	A, B, E, F	{(A,B)}
6	E, F, X	E, F, X	{(E, F)}
7	F, X, Y, Z	F, X, Y, Z	{(F, X)}
8	F, Y, Z	F, Y, Z	{(F, Y), (Y, Z), (F, Z),(F, Y, Z)}
9	X, Y, Z	X, Y, Z	{(X, Y), (Y, Z), (X, Z), (X, Y, Z)}

Table 7: Association rules with respect to each transaction

While constructing the simplex, first we find all the vertices of the simplex. After that we choose each transaction one by one and build the simplexes. From the above

association rule table start with first transaction and build the maximum possible dimension simplex. For example first transaction, first build 2- dimension simplex and then move to 3-dimension simplex … till n-dimension simplex. While building top dimension simplex, we recursively find the lower dimension simplexes. After building the complete simplex graph, we can easily find star neighborhood of vertex A for finding frequent items for vertex A.

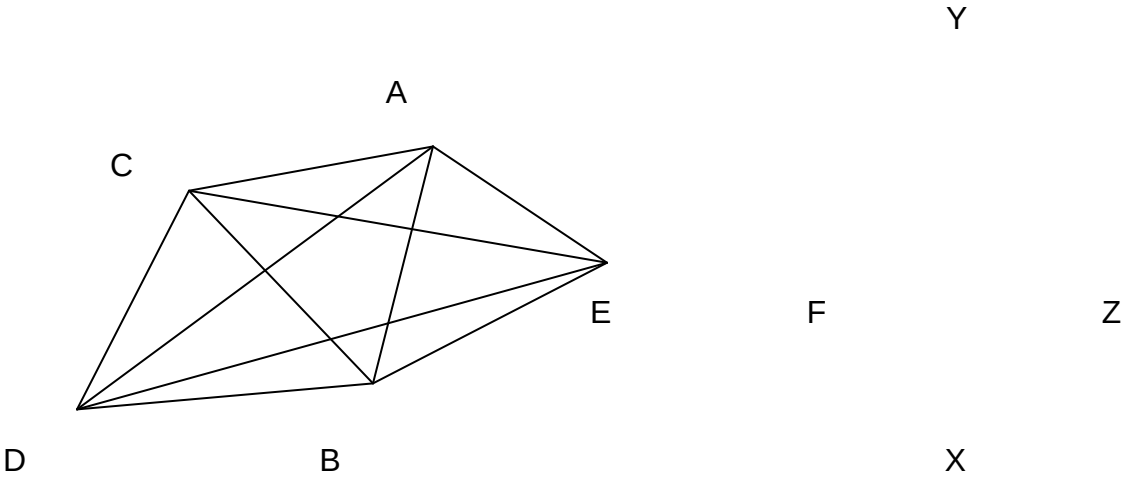

Figure 8: First transaction Simplex

The below mentioned Complete simplex has been formed after building simplexes for each vertex (only high frequency items). By calculating the star neighborhood any vertex gives the all possible frequent item names. For example for vertex F, the star

neighborhood will be E, X, Y, Z that means these all vertices are directly connected to vertex F.

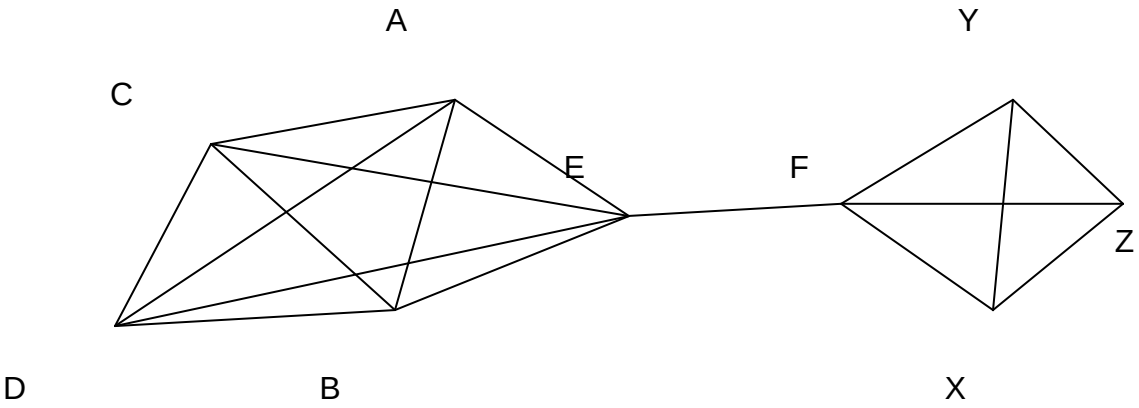

Figure 9: Complete Simplex for all transactions

5.4 Maximal Simplex Method Implementation Details

The following java file is the main file of Maximal Simplex Method Algorithm.

File Name: Maximal_Simplex_Method.java

Method Names:

1. LoadCylinderData()

This method is used to read the required cylinder data into main memory. By using this method we are accessing data set data from disk files.

2. cylinderCount()

This method decides whether the particular cylinder (column) meets the threshold value or not. If the cylinder count is above the threshold value that cylinder is considered as a high frequency item.

3. findAssocRules_Largedataset(Map<Integer,Integer> highFreqItems)

This method finds the all dimensional simplexes for each vertex. For finding simplexes this method reads the required cylinder data into main memory and performs fast bit count operations.

4. findLines()

This method finds all lines between the vertices by using already computed association rules (frequent itemsets).

5. sortItems()

This method sorts the hashmap of high frequency items based on their count value and return the sorted the hash map.

5.5 Bit count Techniques

In this paper, we have experimented with three different techniques for counting the bits set of a 32-bit integer.

5.5.1 Counting bits set using 16-operations

This method follows divide and conquer technique for finding the bit count value. It uses five steps and 16 operations. In each step, it adds the neighboring bits together and stores the result in two or four or eight etc segment registers.

5.5.2 Counting bits set using 12-operations

This method uses 12 operations to find the bit count. This method is as similar as lookup table method. The counting of the bits that are set to 1 in the bytes is computed in parallel. The total sum of bits set count is calculated by multiplying with 0x1010101 and by performing right shift of 24 bits.

5.5.3 Integer.bitCount(int) method Java utility

Without using user defined bit count method, directly uses the java utility method for finding the number of set bits in an integer.

By using second and third method, bit set count computation time is same but for first technique it takes little more time for computation.

4 Experiment Results and Analysis

6.1 Program Computation Results

The Maximal Simplex Algorithm is computed with different minimum support value varies in between 75% to 0.01%. In this implementation, we are reading required data set data from disk files and keeping in main memory to avoid without scanning the database. The tests are conducted using Lenovo T400 PC with 2.4 GHz CPU, 6 GB RAM memory under Windows 7. The simplex program is coded in Java. To check the correctness of the algorithm and association rules, we experiment on different set of data.

The first experiment is small example which has 9 attributes and 9 transactions and minimum support value is 2. This program takes few seconds to give the association rules. We got 35 association rules from this example.

Item Name	A	B	C	D	E	F	**G**	**H**	X	Y	Z
1	1	1	1	1	1	0	**0**	**0**	0	0	0

2	1	1	1	1	1	0	**0**	**0**	0	0	0
3	1	1	1	1	0	0	**0**	**0**	0	0	0
4	1	1	1	1	0	0	**0**	**0**	0	0	0
5	1	1	0	0	1	1	**0**	**0**	0	0	0
6	0	0	0	0	1	1	**0**	**0**	1	0	0
7	0	0	0	0	0	1	**0**	**0**	1	1	1
8	0	0	0	0	0	1	**0**	**0**	0	1	1
9	0	0	0	0	0	0	**0**	**0**	1	1	1
Count	5	5	4	4	4	4	**0**	**0**	3	3	3

Convert database table data into data warehouse by using Build_DataWarehouse.java file, the transactions data will be stored in disk file. i.e. {496, 496,480, 480, 408, 30, 13, 7, 7}.

Output Association Rules:

A B, A C, B C,A B C,A D, B D, C D, A B D, A C D, B C D, A B C D, A E, B E, C E, D E, A B E, A C E, B C E, B D E, C D E, A C D E, A D E, A B C E, A B D E, B C D E, A B C D E, E F, F X, F Y, X Y, X Z, Y Z, F Z, X Y Z, F Y Z.

The second experiment data set is having 1257 columns and 65536 transactions, around 70.0MB. The computation time for finding association rules is 147 seconds for 0.01% support value. We have converted the data set into data warehouse and store the data in three multiple disks to suitable for parallel computing. The part of association rules are listed in the appendix.

6.2 Results Analysis

We need to consider additional time for building the simplexes of each association rule. If the number of high frequency columns set is large (sup = 0.01), Maximal simplex algorithm takes little more time to build the complete simplex for all association rules.

(1) Converting database data into data warehouse and store data in 32-bit pattern. While performing bit operations, we will perform bit computations on 32 bits at a time.

(2) Maximal Simplex Method and FP Tree read the each transaction and find the frequent items set. In Maximal Simplex Method, using frequent item set it finds the association rules by performing fast bit operations on attributes. While FP Tree builds frequent item set as a branch of the FP tree in main memory.

(3) FP Growth Tree algorithm requires high frequency items order otherwise the FP tree size grows in memory. For large databases, FP Tree may not fit in the main memory.

7 Conclusion and Future Work

In this paper, first build the data warehouse system using translated bit pattern database table and then develop a new geometric algorithm called Maximal Simplex Method for finding all possible frequent item sets using association rule mining. Maximal Simplex Method avoids the costly database scans and instead reading data from disk files (Data Warehouse). Finding high frequency items and find the frequent items set for each transaction is essentially same as FP tree algorithm. And then determine association rules by performing fast bit operations (AND) to reduce the computation time. This step will be faster than FP Tree algorithm by performing AND operations on attributes data. We believe that this new algorithm will give same association rules as FP Tree algorithm. Building a data warehouse system makes the algorithm faster. And we organize data in multiple disk files which is useful to achieve parallelism. By using above mentioned techniques, Maximal Simplex Method is giving promising results. Comparing this algorithm with FP Growth Tree algorithm will appear in future.

8 References

[1] Hilton, P.J, and Wylie, S.*HOMOLOGY THEORY: An Introduction to Algebraic Topology* .1962.

[2] Hector Garcia-Molina , Jeffrey D. Ullman, and Jennifer Widom. *Database Systems: The Complete Book.*

[3] Lin, T.Y., Xiaohua Hu, & Louie, E.(July 2003). A Fast Association Rule Algorithm Based On Bitmap and Granular Computing*. Fuzzy Systems FUZZ '03. The 12th IEEE International Conference, 1*, 671-683.

[4] Christian Borgelt . An Implementation of the FP-Growth Algorithm. *Workshop Open Source Data Mining Software (OSDM'05, Chicago, IL)*, 1-5.
ACM Press, New York, NY, USA 2005

[5] Lin, T. Y. (2000, September).Data Mining and Machine Oriented Modeling: A Granular Computing Apporach. *Journal of Applied Intelligence, 13,*113-124.

[6] *Association Rule Mining* – Wikipedia (n. d.). Retrieved Sep 20 2011 from Wikipedia web page: http://en.wikipedia.org/wiki/Association_rule_learning.

[7] *Association Analysis: Basic Concepts and Algorithms*. Retrieved Sep 25 2011 from: http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf.

[8] *Bit Set counting*. Retrieved Oct 5 2011 from:
http://graphics.stanford.edu/~seander/bithacks.html#CountBitsSetParallel

Appendix – Running Example

Maximal Simplex Method running example (0.01 % support value)

In below mentioned example the database attribute names starts from 1 to 1257.

Attribute (Item Name) Count values:

0=44688, 1=13, 2=20370, 3=29, 4=21, 5=86, 6=51, 7=16891, 8=145, 9=6207, 10=931, 11=178, 12=199, 13=32343, 14=113, 15=151, 17=186, 16=16707, 19=1056, 18=6037, 21=114, 20=146, 23=454, 22=22543, 25=1035, 24=303, 27=1152, 26=328, 29=10039, 28=3260, 31=6746, 30=3697, 34=24294, 35=4054, 32=371, 33=210, 38=2536, 39=4458, 36=2687, 37=11304, 42=342, 43=30972, 40=3643, 41=9042, 46=5700, 47=5880, 44=3013, 45=17899, 51=259, 50=169, 49=860, 48=345, 55=1192, 54=11383, 53=4470, 52=42385, 59=411, 58=4145, 57=189, 56=826, 63=3540, 62=427, 61=3665, 60=252, 68=47962, 69=4523, 70=211, 71=178, 64=830, 65=139, 66=3090, 67=447, 76=244, 77=54150, 78=5968, 79=121, 72=560, 73=2078, 74=251, 75=956, 85=233, 84=141, 87=56763, 86=184, 81=158, 80=191, 83=178, 82=161, 93=4152, 92=26421, 95=4476, 94=5079, 89=216, 88=7298, 91=581, 90=1493, 102=141, 103=128, 100=30172, 101=5017, 98=748, 99=302, 96=17639, 97=2778, 110=685, 111=123, 108=4428, 109=1495, 106=3975, 107=9553, 104=175, 105=16935, 119=5523, 118=108, 117=307, 116=2854, 115=19074, 114=10412, 113=32302, 112=150, 127=85, 126=141, 125=157, 124=1830, 123=3430, 122=153, 121=3251, 120=16744, 137=128, 136=195,

139=169, 138=179, 141=150, 140=178, 143=159, 142=171, 129=143, 128=139, 131=237, 130=209, 133=152, 132=179, 135=350, 134=594, 152=167, 153=164, 154=169, 155=436, 156=6770, 157=4927, 158=137, 159=106, 144=136, 145=39394, 146=6731, 147=191, 148=427, 149=119, 150=113, 151=113, 171=15075, 170=3486, 169=7987, 168=4349, 175=112, 174=131, 173=468, 172=5362, 163=132, 162=138, 161=171, 160=159, 167=4298, 166=1594, 165=1337, 164=2794, 186=156, 187=130, 184=9211, 185=6953, 190=150, 191=91, 188=5963, 189=7255, 178=189, 179=162, 176=159, 177=191, 182=90, 183=96, 180=142, 181=136, 205=2412, 204=2633, 207=4637, 206=5237, 201=5167, 200=5665, 203=580, 202=199, 197=265, 196=324, 199=297, 198=202, 193=1594, 192=1733, 195=2473, 194=179, 220=223, 221=427, 222=1434, 223=1217, 216=525, 217=13592, 218=6118, 219=714, 212=876, 213=1086, 214=534, 215=1064, 208=30943, 209=10413, 210=10834, 211=3282, 239=216, 238=4514, 237=9214, 236=5654, 235=3926, 234=344, 233=1039, 232=535, 231=1318, 230=933, 229=999, 228=942, 227=6398, 226=16511, 225=7971, 224=799, 254=550, 255=11856, 252=189, 253=161, 250=25216, 251=5653, 248=6293, 249=2849, 246=2782, 247=2405, 244=14468, 245=6894, 242=236, 243=13254, 240=322, 241=355, 275=1142, 274=1794, 273=179, 272=164, 279=154, 278=173, 277=175, 276=181, 283=148, 282=192, 281=182, 280=181, 287=162, 286=183, 285=170, 284=175, 258=152, 259=150, 256=34164, 257=17898, 262=11175, 263=5788, 260=210, 261=188, 266=152, 267=143, 264=428, 265=157, 270=28327, 271=18864, 268=232,

269=163, 305=220, 304=230, 307=127, 306=138, 309=142, 308=135, 311=97, 310=102, 313=140, 312=144, 315=756, 314=159, 317=834, 316=1482, 319=120, 318=181, 288=118, 289=296, 290=676, 291=618, 292=200, 293=127, 294=118, 295=114, 296=162, 297=117, 298=133, 299=127, 300=161, 301=1461, 302=1199, 303=234, 343=114, 342=115, 341=165, 340=154, 339=136, 338=138, 337=183, 336=170, 351=115, 350=159, 349=151, 348=153, 347=126, 346=186, 345=175, 344=166, 326=126, 327=130, 324=114, 325=121, 322=153, 323=163, 320=141, 321=151, 334=141, 335=125, 332=172, 333=117, 330=215, 331=155, 328=507, 329=391, 373=159, 372=156, 375=20761, 374=24807, 369=139, 368=156, 371=136, 370=133, 381=298, 380=279, 383=114, 382=269, 377=225, 376=2686, 379=353, 378=330, 356=147, 357=152, 358=153, 359=161, 352=158, 353=170, 354=175, 355=180, 364=193, 365=123, 366=119, 367=107, 360=1245, 361=1373, 362=552, 363=214, 410=191, 411=141, 408=191, 409=190, 414=143, 415=120, 412=166, 413=164, 402=183, 403=154, 400=238, 401=228, 406=134, 407=139, 404=162, 405=143, 395=631, 394=1267, 393=1518, 392=344, 399=162, 398=180, 397=176, 396=307, 387=143, 386=162, 385=155, 384=144, 391=164, 390=156, 389=154, 388=145, 440=164, 441=127, 442=148, 443=126, 444=156, 445=150, 446=155, 447=861, 432=138, 433=136, 434=126, 435=130, 436=150, 437=143, 438=114, 439=110, 425=2322, 424=2969, 427=459, 426=1332, 429=167, 428=306, 431=191, 430=195, 417=174, 416=170, 419=171, 418=180, 421=179, 420=156, 423=2189, 422=316,

478=2150, 479=4207, 476=161, 477=278, 474=188, 475=132, 472=175, 473=161, 470=130, 471=129, 468=137, 469=138, 466=162, 467=167, 464=190, 465=205, 463=157, 462=165, 461=144, 460=165, 459=136, 458=173, 457=160, 456=223, 455=271, 454=315, 453=432, 452=470, 451=236, 450=595, 449=1747, 448=2095, 508=776, 509=326, 510=429, 511=246, 504=139, 505=481, 506=2029, 507=2186, 500=181, 501=198, 502=150, 503=154, 496=166, 497=181, 498=168, 499=178, 493=141, 492=193, 495=121, 494=153, 489=167, 488=178, 491=156, 490=182, 485=168, 484=166, 487=131, 486=132, 481=783, 480=3180, 483=216, 482=225, 550=129, 551=1748, 548=145, 549=123, 546=165, 547=162, 544=168, 545=157, 558=156, 559=139, 556=402, 557=185, 554=1106, 555=710, 552=4094, 553=3170, 567=142, 566=163, 565=194, 564=199, 563=174, 562=241, 561=211, 560=197, 575=162, 574=209, 573=195, 572=204, 571=176, 570=204, 569=187, 568=184, 516=176, 517=168, 518=170, 519=171, 512=211, 513=152, 514=156, 515=173, 524=163, 525=143, 526=158, 527=139, 520=204, 521=165, 522=142, 523=134, 533=265, 532=222, 535=197, 534=214, 529=198, 528=194, 531=172, 530=182, 541=120, 540=138, 543=113, 542=137, 537=222, 536=207, 539=157, 538=242, 610=160, 611=173, 608=161, 609=158, 614=436, 615=1144, 612=102, 613=102, 618=5350, 619=1916, 616=17484, 617=36731, 622=165, 623=417, 620=179, 621=141, 627=155, 626=193, 625=253, 624=1061, 631=213, 630=123, 629=179, 628=159, 635=130, 634=173, 633=199, 632=203, 639=108, 638=181, 637=147, 636=172,

576=115, 577=101, 578=701, 579=2082, 580=2958, 581=2703, 582=1598, 583=813, 584=282, 585=1788, 586=3822, 587=1057, 588=277, 589=139, 590=140, 591=138, 593=184, 592=176, 595=163, 594=125, 597=143, 596=131, 599=117, 598=129, 601=148, 600=173, 603=105, 602=182, 605=122, 604=139, 607=122, 606=147, 687=664, 686=2937, 685=1094, 684=338, 683=1047, 682=2680, 681=1493, 680=490, 679=151, 678=144, 677=154, 676=297, 675=193, 674=969, 673=399, 672=1757, 702=189, 703=170, 700=443, 701=823, 698=203, 699=252, 696=441, 697=918, 694=197, 695=192, 692=862, 693=698, 690=160, 691=359, 688=396, 689=214, 653=30314, 652=12698, 655=4374, 654=3923, 649=1887, 648=169, 651=1128, 650=4207, 645=2748, 644=4554, 647=177, 646=169, 641=148, 640=169, 643=38348, 642=15732, 668=11942, 669=31828, 670=6596, 671=3340, 664=1070, 665=1700, 666=3975, 667=1585, 660=6454, 661=9107, 662=2298, 663=1064, 656=4920, 657=1242, 658=12650, 659=30993, 747=1403, 746=306, 745=2583, 744=5812, 751=6995, 750=3749, 749=1105, 748=3470, 739=31471, 738=7696, 737=4683, 736=3771, 743=2320, 742=37505, 741=18588, 740=3425, 762=213, 763=342, 760=202, 761=230, 766=175, 767=111, 764=216, 765=151, 754=6365, 755=8839, 752=38435, 753=1504, 758=6287, 759=118, 756=46229, 757=1369, 713=7796, 712=204, 715=3742, 714=30446, 717=372, 716=4705, 719=177, 718=325, 705=154, 704=157, 707=394, 706=166, 709=1389, 708=1338, 711=315, 710=336, 728=1752, 729=4110, 730=181, 731=121, 732=160, 733=1315, 734=4242, 735=3277, 720=169,

721=148, 722=120, 723=119, 724=4488, 725=14416, 726=6972, 727=32020,
821=173, 820=158, 823=149, 822=132, 817=162, 816=49668, 819=142,
818=7763, 829=1003, 828=250, 831=249, 830=188, 825=180, 824=184,
827=460, 826=209, 804=149, 805=153, 806=136, 807=147, 800=176,
801=155, 802=164, 803=166, 812=161, 813=111, 814=131, 815=5110,
808=222, 809=162, 810=145, 811=130, 791=171, 790=167, 789=185,
788=172, 787=174, 786=186, 785=196, 784=189, 799=116, 798=153,
797=122, 796=152, 795=114, 794=168, 793=141, 792=165, 774=151,
775=150, 772=161, 773=184, 770=154, 771=170, 768=158, 769=156,
782=181, 783=144, 780=193, 781=153, 778=262, 779=163, 776=192,
777=148, 881=54728, 880=1182, 883=5882, 882=159, 885=162, 884=2025,
887=200, 886=134, 889=51322, 888=4885, 891=5252, 890=737, 893=4360,
892=2216, 895=52160, 894=309, 864=252, 865=6541, 866=1411, 867=305,
868=5484, 869=50494, 870=1416, 871=5273, 872=1525, 873=205,
874=3709, 875=52416, 876=524, 877=5942, 878=1752, 879=550, 851=170,
850=877, 849=6614, 848=261, 855=1451, 854=22478, 853=1584, 852=217,
859=863, 858=3224, 857=613, 856=34848, 863=52543, 862=3666, 861=257,
860=186, 834=50643, 835=381, 832=185, 833=4030, 838=51823, 839=176,
836=7515, 837=4128, 842=186, 843=223, 840=7275, 841=654, 846=2871,
847=48224, 844=716, 845=5009, 956=2168, 957=3629, 958=5775,
959=52175, 952=4128, 953=184, 954=410, 955=53432, 948=188,
949=53576, 950=1842, 951=3715, 944=124, 945=190, 946=181, 947=724,
941=1386, 940=53195, 943=3745, 942=3947, 937=4388, 936=1104,

939=189, 938=4796, 933=167, 932=210, 935=50054, 934=5180, 929=4031, 928=928, 931=193, 930=2921, 926=195, 927=48290, 924=4613, 925=11423, 922=51141, 923=271, 920=113, 921=5213, 918=115, 919=102, 916=4836, 917=3013, 914=55309, 915=112, 912=1311, 913=89, 911=2742, 910=5054, 909=68, 908=55377, 907=3106, 906=4328, 905=895, 904=51366, 903=81, 902=6031, 901=96, 900=130, 899=135, 898=2310, 897=5060, 896=316, 1016=23549, 1017=4361, 1018=4074, 1019=30995, 1020=3782, 1021=21383, 1022=6361, 1023=1720, 1008=25732, 1009=3588, 1010=18147, 1011=19570, 1012=22706, 1013=4590, 1014=12060, 1015=23597, 1001=22863, 1000=9917, 1003=1929, 1002=3006, 1005=21338, 1004=2864, 1007=24869, 1006=8810, 993=2460, 992=4485, 995=198, 994=5494, 997=24873, 996=29806, 999=4641, 998=3105, 986=50829, 987=4173, 984=5107, 985=1279, 990=225, 991=50956, 988=2718, 989=5782, 978=3590, 979=2907, 976=998, 977=51618, 982=3803, 983=2874, 980=5918, 981=51280, 971=51342, 970=6172, 969=3359, 968=2950, 975=152, 974=4778, 973=3190, 972=3201, 963=51883, 962=5861, 961=3545, 960=2426, 967=51665, 966=5923, 965=3418, 964=2772, 1100=152, 1101=177, 1102=156, 1103=176, 1096=182, 1097=216, 1098=172, 1099=159, 1092=7849, 1093=174, 1094=187, 1095=175, 1088=116, 1089=44652, 1090=10626, 1091=150, 1117=43024, 1116=1086, 1119=190, 1118=12418, 1113=12163, 1112=43382, 1115=7186, 1114=227, 1109=178, 1108=159, 1111=375, 1110=177, 1105=202, 1104=152, 1107=175, 1106=206, 1134=292,

1135=5649, 1132=36323, 1133=12438, 1130=7258, 1131=2052, 1128=378, 1129=6568, 1126=42400, 1127=12667, 1124=378, 1125=8251, 1122=41948, 1123=12498, 1120=6944, 1121=2287, 1151=1926, 1150=5776, 1149=300, 1148=13664, 1147=38833, 1146=1185, 1145=2787, 1144=162, 1143=184, 1142=1618, 1141=6055, 1140=275, 1139=13189, 1138=39289, 1137=1131, 1136=4055, 1032=1205, 1033=6743, 1034=865, 1035=48145, 1036=7113, 1037=1173, 1038=6762, 1039=43282, 1024=50906, 1025=6541, 1026=1400, 1027=6425, 1028=242, 1029=795, 1030=48429, 1031=6948, 1049=8108, 1048=47614, 1051=7244, 1050=588, 1053=233, 1052=138, 1055=14385, 1054=141, 1041=5965, 1040=5063, 1043=191, 1042=9278, 1045=152, 1044=493, 1047=165, 1046=380, 1066=166, 1067=7786, 1064=45325, 1065=9111, 1070=150, 1071=186, 1068=169, 1069=209, 1058=2423, 1059=5564, 1056=35945, 1057=5704, 1062=229, 1063=148, 1060=266, 1061=1442, 1083=147, 1082=128, 1081=154, 1080=161, 1087=162, 1086=168, 1085=483, 1084=125, 1075=190, 1074=188, 1073=185, 1072=141, 1079=147, 1078=197, 1077=188, 1076=197, 1221=417, 1220=4047, 1223=301, 1222=273, 1217=16538, 1216=33809, 1219=4241, 1218=797, 1229=156, 1228=4272, 1231=1710, 1230=1932, 1225=18717, 1224=35501, 1227=3424, 1226=120, 1236=2597, 1237=1562, 1238=502, 1239=332, 1232=235, 1233=236, 1234=977, 1235=628, 1244=374, 1245=2647, 1246=1013, 1247=762, 1240=2809, 1241=1025, 1242=793, 1243=578, 1255=298, 1254=234, 1253=914, 1252=1557, 1251=386, 1250=475, 1249=391, 1248=520, 1256=32998, 1153=141, 1152=181,

1155=992, 1154=111, 1157=138, 1156=443, 1159=39749, 1158=253, 1161=172, 1160=14656, 1163=2258, 1162=5434, 1165=143, 1164=117, 1167=126, 1166=121, 1168=890, 1169=497, 1170=196, 1171=217, 1172=39034, 1173=15472, 1174=126, 1175=5106, 1176=2626, 1177=165, 1178=185, 1179=177, 1180=110, 1181=135, 1182=129, 1183=333, 1187=176, 1186=16688, 1185=38766, 1184=141, 1191=38406, 1190=418, 1189=3102, 1188=4861, 1195=7489, 1194=4572, 1193=195, 1192=16902, 1199=4445, 1198=399, 1197=15673, 1196=36257, 1202=174, 1203=361, 1200=7017, 1201=1790, 1206=180, 1207=120, 1204=617, 1205=242, 1210=156, 1211=176, 1208=119, 1209=157, 1214=2265, 1215=4835, 1212=33719, 1213=16577}

Association rules (separated by commas):

908 881 863 875 895 838 904 889 834 869 816 847 756 1126 752 643 742 617 1056 856 659 653 1012 854 375 1011 271 1225 741 257 1192 1186 1213 1217 1197 1173 1160 725 1148 1139 1127 1123 1133 1118 1113 1090 1042 1065 1049 88 1036 185 1031 245 1025 227 902 78 1057 172 201 101 1013 69 992 108 987 35 106 982 1020 978 972 44 968 964 960 425 956 950 941 55 936 928 23 99 197 76, 908 881 863 875 895 838 904 889 834 869 816 847 756 0 1122 752 643 742 617 856 659 653 1008 997 1012 375 1011 271 1225 257 1192 1186 1213 1217 1197 1173 1160 725 1139 1127 1123 1133 1118 1113 1090 1042 1065 1049 88 189 185 1031 1025 78 1057 201 101 1013 69 992 987 35 106 982 1020 978 972 44 1002 580 968 964 960 956 950 941 165 55 936 928 23 76, 908 881 863 875 895 838 904 889 834

816 847 756 1126 752 643 856 669 714 653 1008 997 375 1011 271 1225 257 1192 1186 1213 1217 1197 1173 1160 725 1148 1139 1127 1123 1133 1118 1113 1090 1042 1065 1006 1049 88 189 1036 185 1031 245 1025 227 78 1057 201 157 1013 69 992 53 108 650 987 106 982 1020 40 978 121 972 998 44 968 964 745 960 425 956 950 193 941 936 928 64, 908 881 863 875 895 838 904 889 834 869 816 847 756 0 752 643 742 617 856 727 669 659 714 653 997 375 1011 115 271 1225 257 1192 1186 1213 1217 1197 1173 1160 725 1139 1123 1133 1118 1113 1090 1065 661 1049 88 189 185 1031 1025 227 78 1057 172 101 1013 69 992 987 93 35 982 1020 736 978 170 121 972 44 968 964 581 960 956 950 193 941 55 936 928 62 74, 908 881 863 875 895 838 904 834 869 816 847 756 0 1112 752 643 742 617 727 659 653 1012 375 1011 271 1225 257 1192 1186 1213 1217 1197 1173 1160 725 1139 1127 1123 1133 1118 1113 210 1090 1042 1065 41 1049 88 189 1031 245 1025 218 78 1057 201 101 157 1215 207 69 992 53 108 1017 167 987 93 982 1020 978 28 972 44 968 249 964 960 956 950 193 941 708 936 928 317 42 939, 908 881 863 875 895 963 838 904 889 834 869 816 756 0 752 643 617 653 1008 1012 375 1011 115 271 1225 257 1192 1186 1213 1217 1197 1173 1160 1139 1127 1123 1133 1118 1113 1090 1042 1065 1006 1049 88 189 1036 185 1031 670 1025 78 263 1057 172 201 157 207 1013 69 992 53 108 1017 987 35 982 1020 978 972 998 44 1002 968 964 36 960 956 950 941 55 936 928, 908 881 863 875 895 963 838 904 889 834 869 816 847 756 1122 752 643 617 856 659 653 997 1012 375 1011 271 1225 257 1192 1186 1213 1217 1197 1173 1160 1139 1127 1123 1118 1113 210 1090 1042

1065 41 1049 88 185 1031 1025 227 218 78 1057 172 201 101 1013 69 992 53 1017 987 93 982 1020 874 978 1009 170 972 480 553 44 1002 968 97 964 960 423 956 1131 950 941 275 936 928 317, 908 881 863 875 895 838 904 889 834 869 816 847 756 0 1122 752 643 617 856 669 659 653 997 1012 375 1011 115 271 1225 741 257 1192 1186 1213 1217 1197 1173 1160 1139 1127 1123 1133 1118 1113 210 1090 1042 1065 1049 88 189 1036 1031 1025 227 78 1057 201 101 1013 69 992 53 1017 987 35 1220 982 1020 978 1009 972 998 44 968 97 964 960 956 950 941 1146 936 928 1253 59 74, 908 881 863 875 895 838 904 889 834 869 935 816 756 0 752 643 742 617 856 669 739 659 714 653 997 1007 1011 271 1225 257 1192 1186 1213 1217 1197 1173 1160 725 1127 1123 1133 1118 1113 1090 1042 1065 1049 88 189 1036 185 1031 1025 227 218 78 1057 868 101 1013 69 992 53 1017 987 982 1020 978 1009 170 972 998 44 1002 964 745 960 1131 950 941 1146 936 928 1253 67 99 74, 908 881 863 875 895 838 904 889 834 869 816 847 756 0 1122 752 643 256 669 739 659 714 653 1007 1012 854 271 1225 257 1192 1186 1213 1217 1197 1173 1160 725 1139 1127 1123 1133 1118 1113 1090 1042 1065 1006 1049 88 189 1036 1031 670 1025 227 78 1057 236 868 921 201 101 207 1013 69 992 108 1017 987 93 982 1020 978 1009 121 972 44 917 1002 968 964 911 960 205 950 1201 941 165 426 275 936 928 317 923 915 919 913