International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 8 (2018) pp. 6424-6432

Pattern generation based Reverse engineering on Database Relation scheme using Enhanced Association rule mining

J. M. Dhayashankar[1], A. V. Ramani[2]

[1] *Department of Computer Applications, Sri Ramakrishna Mission Vidyalaya College of Arts and Science, Coimbatore, India.*

[2] *Department of Computer Science, Sri Ramakrishna Mission Vidyalaya College of Arts and Science, Coimbatore, India.*

Abstract

In most of the companies the maintenance of documentation of the application software is not considered very important which leads to more technical problem in usage of software and degrades the performance of the organization process on the whole. Hence maintenance of such legacy databases is very challenging task when it is poorly written or the data is missing. To overwhelm this problem in this paper the concept of database reverse engineering is used which is an attempt to recover high level conceptual design from the existing instances of the database. This work discovers the conceptual schema by adapting pattern generation in association rule mining which is different from the traditional association rule mining process. Once the un-normalized table is given to this algorithm it will perform normalization at the third norm form because many of the business organization maintain their database till 3NF. The resultant rules are pruned to solve the issue of producing more uninteresting rules innate with the association mining technique.

***Keywords*:** legacy database, conceptual, relational, normalization, association rule.

INTRODUCTION

In this current era maintenance of Legacy databases of organizations becomes one of most valuable resource. The existing databases are outdated because they are developing using older programming language which is not fit to update for the current needs of modernized companies. Incase suppose if all existing system had been developed using a transformation system recording the design and its rationale there would be no need for reverse engineering. However, in reality a huge amount of conventionally developed system exists and these systems have errors and continual demand for the enhancement of their functional and performance requirements. Modification is a necessity. This means that there is a fundamental need to understand them. As they are often badly designed and have an incomplete, nonexistent, or, even worse, wrong documentation without any design information, which is a challenging task.

The process of extracting the domain semantics of the existing database structures like integrity constraints, keys and functional dependencies is known as database reverse engineering. [6, 13]. This process retrieves the design details of the legacy system and it performs the reverse operation of converting the logical schema to conceptual schema of a given database.

RELATED WORK

Since the introduction of a famous association technique known as Apriori algorithm [1, 2], there have long been immense attempts to integrate this technique to improve database design, consistency checking, and querying. Han et al. [3] improved the DBMiner system to work with relational databases and data warehouses. DBMiner can do many data mining tasks such as classification, prediction and association. Sreenath, Bodagala, Alsabti, and Ranka [4] adopted Apriori algorithm to work with relational database system. They created Fast UPdate algorithm to search association data when the system has new transaction. Tsechansky, Pliskin, Rabinowitz and Porath [5] applied Apriori to find association data from many relations in the database. Berzal, Cubero, Marín and Serrano [6] used Tree-Based Association Rule mining (TBAR) to find association data in relational database.

They kept large item set in tree structure format to reduce time cost in association process. Hipp, Güntzer and Grimmer [7] implemented Apriori algorithm with C++ programming language to work on DB2 database system. They used the program to find association data in Daimler-Chrysler Company database. In parallel to the attempts of applying learning techniques to existing large databases, researchers in the area of database reverse engineering have proposed some means of extracting conceptual schema. Lee and Yoo [8] proposed a method to derive a conceptual model from object-oriented databases. The derivation process is based on forms including business forms and forms for database interaction in the user interface. The final products of their method are the object model and the scenario diagram describing a sequence of operations. The work of Perez et al. [9] emphasized on relational object-oriented conceptual schema extraction. Their reverse engineering technique is based on a formal method of term rewriting.

They use terms to represent relational and object-oriented schemas. Term rewriting rules are then generated to represent the correspondences between relational and object-oriented

elements. Output of the system is the source code to migrate legacy database to the new system. Recent work in database reverse engineering has not concentrated on a broad objective of system migration. Researchers rather focus their study on a particular issue of semantic understanding. Lammari et al. [10] proposed a reverse engineering method to discover inter-relational constraints and inheritances embedded in a relational database. Chen et al. [11] also based their study on entity relationship model. They proposed to apply association rule mining to discover new concepts leading to a proper design of relational database schema. They employed the concept of fuzziness to deal with uncertainty inherited with the association mining process.

In Chiang et al.'s Method [12, 13] requires, as input, the data instances and relation-schemes, including primary keys. Optionally, the user may insert some inclusion dependencies. The assumptions made are 3NF, consistent naming of attributes and no error on key attributes values. The output given is an Extended Entity-Relationship (EER) model.

The method introduced by Johannesson's [14] requires as input, the relation-schemes, functional dependencies and inclusion dependencies. Relations are assumed to be in 3NF. The output is a conceptual schema described as a pair containing a language L and a set IC of typing, mapping and generalization constraints. In Markowitz et al.'s Method [15] requires, as input, the relation schemes, key dependencies and key-based inclusion dependencies, i.e. referential integrity constraints. Relations are assumed to be in Boyce-Codd normal form (BCNF), considering only the "relevant" functional dependencies (i.e. allowing hidden-objects. The output is an EER model. In Navathe and Awong's Method [16] only requires as input the relation schemes, although with several assumptions: 3NF or BCNF relations are required; coherency on attribute names: there are no ambiguities in foreign keys and there are no homonyms; and all candidate keys must be specified. Petit et al.'s Method [17] requires, as input, the relation schemes, with unique and not null constraints, data instances and code. The assumptions identified in our framework are not mandatory. The output is an EER model. Premerlani and Blaha's Method [18, 20] requires, as input, the relation schemes and data. The assumptions identified in our framework are not mandatory. The output is an Object Modeling Technique (OMT) model. Signore et al.'s Method [19] requires, as input, the relation-schemes and code. The assumptions identified in our framework are not mandatory. The output is an ER model. This paper aims at developing a mining based legacy database reverse engineering. The proposed method incorporates correlation based association rule mining for database reverse engineering by finding the dependency among the attributes and converting the un-normalized database to the normalized format.

METHODOLOGY

Association Rules:

Association rule mining is the process of finding patterns, associations and correlations among sets of items in a database. The association rules generated have an antecedent and a consequent. An association rule is a pattern of the form $X \& Y \Rightarrow Z$ *[support, confidence]*, where X, Y, and Z are items in the dataset. The left hand side of the rule X & Y is called the antecedent of the rule and the right hand side Z is called the consequent of the rule. This means that given X and Y there is some association with Z. Within the dataset, confidence and support are two measures to determine the certainty or usefulness for each rule. Support is the probability that a set of items in the dataset contains both the antecedent and consequent of the rule or P ($X \cup Y \cup Z$). Confidence is the probability that a set of items containing the antecedent also contains the consequent or P ($Z| X \cup Y$). Typically an association rule is called strong if it satisfies both a minimum support threshold and a minimum confidence threshold that is determined by the user.

Some basic terms and concepts need to be understood in order to further understand data mining on large datasets. An itemset is defined as a set of items. An itemset containing *k* items is a *k*-itemset. Therefore the set {A, B} is a 2-itemset. The occurrence frequency of an itemset is simply the number of transactions that contain the itemset. It is sometimes referred to as the frequency, support count, or count of the itemset. The minimum support count is defined as the number of transactions required for the itemset to satisfy minimum support. The minimum support count is equal to the product of the total number of transactions in the dataset and the user-defined minimum support. Any itemsets satisfying the minimum support is considered a frequent itemset and the set of *k*-itemsets is commonly denoted by L_k.

PROPOSED WORK

This proposed methodology of database reverse engineering encomposes of designing and improving the process of normalization with correlation based association analysis technique. This work uses the normalization concept and association analysis technique to create a new algorithm called Database Reverse Engineering with pattern generation based Association Rule Mining. The given unstructured database is converted into a normalized database using the mining technique which has the slight modification in the conventional association rule mining. In general to generate the association rule it mainly depends on the support and confidence of the rule. But this work additionally takes into the account the correlation existing among the attributes of database.

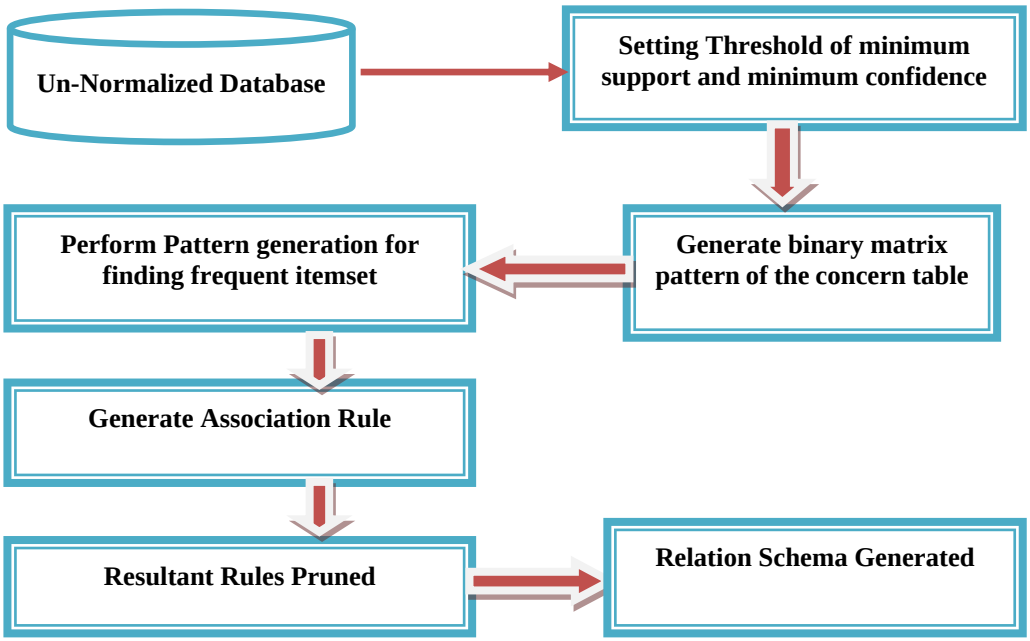


Figure 1: Workflow of the proposed method

This proposed method persuades conceptual schemas from the selected database instances with the basic assignment that the database design documents are missing. So this work adapts the concept of normalization and fuzzy correlation based association rule generation to find out the missing database design. Transforming the unstructured dataset into separate relation is the process of normalization. The objective of this process is to eliminate the redundant data occurred in the dataset and considerably lessens the anomaly. There are different levels of normalization available but most commonly third normal form are also known as Boyce-Codd norm form is constantly well adequate for most of the business requirements. In the reverse engineering process the situation is quite complex to handle because process of business and requirements are unknown, so the task of dependency analysis in such database is tough even for the experienced ones. Thus this proposed approach is designed to use the fuzzy correlation based association rule mining.

In this proposed method like traditional Apriori algorithm, initially transaction database D is represented by bit matrix in which each row corresponds to an item set for transaction and the columns correspond to the items. This transaction database D and user defined minimum support threshold are used as input to this algorithm. First, frequent item set with length one i.e. L_1 is obtained from transaction database D. Then the pattern table is derived by using items in L_1. The frequency for each pattern from pattern table is counted as logical AND of pattern with transactions in the database which gives output as true.

The patterns having frequency count equal to zero are pruned. Next, k item sets (k >= 1) are used to generate (k+1) item sets. For each item set the support is calculated as sum of all frequency count of matched patterns from the pattern table and check if it is frequent. The procedure is repeated until no more frequent item set can be found.

Figure 2. Algorithm for proposed method of association rule mining

Input: Database Instances in table format D, min_sup : user defined minimum support .

Output: Association Rules

Phase 1:

Begin

Step 1: For each item, count the occurrence from transactional database and add it to frequent one item set L1 if it satisfies minimum support constraint.

Step 2: P = Generate_ Pattern_Table (L1)

Step 3: For each i in P do

Assign N = 0

For each j in D do

flag = Pi AND Dj

If flag is True Then N++

End for

Assign Pi.Count = N

If Pi.Count = 0 then ignore Pi

End for

Step 4: k = 2

Step 5: Lk+1 = Ø;

Step 6: Ck+1 = Generate_Candidate_Itemset(Lk);

Step 7: For each j in Ck+1 do

Count = 0;

For each i in P do

If match found (Pi. pattern, j) is True then

Count = Count + Pi.Count

End for

If Count >= min-sup then Lk+1 = {j}

End for

Step 8: Set k= k +1; Repeat steps 5 to 8 until no more frequent item set is found i.e. Lk+1 = null

Step 9: L = Ù Lk

Step 10: R = Ø

Step 11: For each l from Lk do

For each x in l do

R = R U {x => (l - x)}

End for

End for

Phase 2:

Input: Association Rules

Output: Relation Scheme

Step 11 : For each rule i= 1 to n

Step 12: Select one cause rule with max support value

Step 13: Select primary key of relation (2NF)

Step 14: Find corresponding attribute

Step 15: Select primary key of relation (3NF)

Step 16: Compare non-key attribute in (15) with all attributes in (12) then cut it off to to get one 3NF relation

Step 17: Compare all non-key attributes with 1-NF relation and cut it off

Step 18: Combine all primary keys of 2NF with the remaining attribute in (17) to become the 3NF

WORKING EXAMPLE

To illustrate the working of proposed method here used transactional database D shown in "Table 1". The transactional database contains 30 transactions with an item set I = {A, B, C, D, E, F} of six items.

Table 1: Example of Transaction Database

Tid	Items
1	{ A , B }
2	{A, B , D }
3	{ B ,C }
4	{ B , C , D ,E}
5	{ A , E}
6	{ B , C ,E}
7	{ B , E}
8	{ A , B , D,E}
9	{ A , B , C}
10	{ B , D,E}
11	{ B ,E }
12	{ B , C }
13	{ A , B , D, E }
14	{A,D}
15	{ A, C,D }
16	{B,D,E}
17	{B,C}
18	{B,C,E}
19	{B,C,D}
20	{B,D,E}
21	{C,D}
22	{C,D,E}
23	{D,E}
24	{E,F}
25	{D,E,F}
26	{C,D,F}
27	{A,B,C,D,E}
28	{A.B,C,D,E,F}
29	{D,E,F}
30	{E,F}

Table 2: Representation of Bit matrix of the table 1

Tid	A	B	C	D	E	F
1	1	1	0	0	0	0
2	1	1	0	1	0	0
3	0	1	1	0	0	0
4	0	1	1	1	1	0
5	1	0	0	0	1	0
6	0	1	1	0	1	0
7	0	1	0	0	1	0
8	1	1	0	1	1	0
9	1	1	1	0	0	0
10	0	1	0	1	1	0
11	0	1	0	0	1	0
12	0	1	1	0	0	0
13	1	1	0	1	1	0
14	1	0	0	1	0	0
15	1	0	1	1	0	0
16	0	1	0	1	1	0
17	0	1	1	0	0	0
18	0	1	1	0	1	0
19	0	1	1	1	0	0
20	0	1	0	1	1	0
21	0	0	1	1	0	0
22	0	0	1	1	1	0
24	0	0	0	0	1	1
25	0	0	0	1	1	1
26	0	0	1	1	0	1
27	1	1	1	1	1	0
28	1	1	1	1	1	1
29	0	0	0	1	1	1
30	0	0	0	0	1	1

The bit matrix for the table 1 is given in the table 2. In the items columns, all the input attributes are marked with 0 as its absence and 1 as its presence according to the pattern.

Initially, consider the candidate item set of size one C1 and scan the database to determine the support i.e. occurrence count of each item of C1. Take item A as an example. Support of A is 9 as occurrence count of A is 9 in transaction database.

The support of each item is compared with predefined minimum support threshold i.e. min-supp. Here min-supp is considered as 2.Since the support of items A, B, C, D, E and F are respectively larger than the predefined min-supp, all these six items are considered as frequent and added in the frequent 1-itemset L1 shown in Table 3.

Table 3: Frequent 1-ItemSet L_1

Feature	Count
A	9
B	16
C	8
D	9
E	10
F	4

Table 4. Pattern Generated Table

PID	Pattern	Items						TID	Count
		A	B	C	D	E	F		
pid1	{A,B}	1	1	0	0	0	0	1	1
pid2	**{A,C}**	**1**	**0**	**1**	**0**	**0**	**0**	**Nil**	**0**
pid3	{A,D}	1	0	0	1	0	0	14	1
pid4	{A,E}	1	0	0	0	1	0	5	1
pid5	**{A,F}**	**1**	**0**	**0**	**0**	**0**	**1**	**Nil**	**0**
pid6	{B,C}	0	1	1	0	0	0	3,12,17	3
pid7	**{B,D}**	**0**	**1**	**0**	**1**	**0**	**0**	**Nil**	**0**
pid8	{B,E}	0	1	0	0	1	0	7,11	2
pid9	**{B.F}**	**0**	**1**	**0**	**0**	**0**	**1**	**Nil**	**0**
pid10	{C,D}	0	0	1	1	0	0	21	1
pid11	**{C,E}**	**0**	**0**	**1**	**0**	**1**	**0**	**Nil**	**0**
pid12	**{C,F}**	**0**	**0**	**1**	**0**	**0**	**1**	**Nil**	**0**
pid13	{D,E}	0	0	0	1	1	0	23	1
pid14	**{D,F}**	**0**	**0**	**0**	**1**	**0**	**1**	**Nil**	**0**
pid15	{E,F}	0	0	0	0	1	1	24,30	2
pid16	{A,B,C}	1	1	1	0	0	0	9	1
pid17	{A,B,D}	1	1	0	1	0	0	2	1
pid18	**{A,B,E}**	**1**	**1**	**0**	**0**	**1**	**0**	**Nil**	**0**
pid19	**{A,B,F}**	**1**	**1**	**0**	**0**	**0**	**1**	**Nil**	**0**
pid20	{A,C,D}	1	0	1	1	0	0	15	1
pid21	**{A,C,E}**	**1**	**0**	**1**	**0**	**1**	**0**	**Nil**	**0**
pid22	**{A,C,F}**	**1**	**0**	**1**	**0**	**0**	**1**	**Nil**	**0**
pid23	**{A,D,E}**	**1**	**0**	**0**	**1**	**1**	**0**	**Nil**	**0**
pid24	**{A,D,F}**	**1**	**0**	**0**	**1**	**0**	**1**	**Nil**	**0**
pid25	**{A,E,F}**	**1**	**0**	**0**	**0**	**1**	**1**	**Nil**	**0**
pid26	{B,C,D}	0	1	1	1	0	0	19	1
pid27	{B,C,E}	0	1	1	0	1	0	18	1
pid28	**{B,C,F}**	**0**	**1**	**1**	**0**	**0**	**1**	**Nil**	**0**
pid29	{B,D,E}	0	1	0	1	1	0	10,16,20	3
pid30	**{B,D,F}**	**0**	**1**	**0**	**1**	**0**	**1**	**Nil**	**0**
pid31	{C,D,E}	0	0	1	1	1	0	22	1
pid32	{C,D,F}	0	0	1	1	0	1	26	1
pid33	{D,E,F}	0	0	0	1	1	1	25,29	2
pid34	**{A,B,C,D}**	**1**	**1**	**1**	**1**	**0**	**0**	**Nil**	**0**
pid35	**{A,B,C,E}**	**1**	**1**	**1**	**0**	**1**	**0**	**Nil**	**0**
pid36	**{A,B,C,F}**	**1**	**1**	**1**	**0**	**0**	**1**	**Nil**	**0**
pid37	{A,B,D,E}	1	1	0	1	1	0	8,13	2
pid38	**{A,B,D,F}**	**1**	**1**	**0**	**1**	**0**	**1**	**Nil**	**0**
pid39	**{A,B,E,F}**	**1**	**1**	**0**	**0**	**1**	**1**	**Nil**	**0**
pid40	**{A,C,D,E}**	**1**	**0**	**1**	**1**	**1**	**0**	**Nil**	**0**
pid41	**{A,C,D,F}**	**1**	**0**	**1**	**1**	**0**	**1**	**Nil**	**0**
pid42	**{A,D,E,F}**	**1**	**0**	**0**	**1**	**1**	**1**	**Nil**	**0**
pid43	{B,C,D,E}	0	1	1	1	1	0	4	1
pid44	**{B,C,D,F}**	**0**	**1**	**1**	**1**	**0**	**1**	**Nil**	**0**
pid45	**{B,C,E,F}**	**0**	**1**	**1**	**0**	**1**	**1**	**Nil**	**0**
pid46	**{B,D,E,F}**	**0**	**1**	**0**	**1**	**1**	**1**	**nil**	**0**
pid47	**{C,D,E,F}**	**0**	**0**	**1**	**1**	**1**	**1**	**Nil**	**0**
pid48	{A,B,C,D,E}	1	1	1	1	1	0	27	1
pid49	**{A,B,C,D,F}**	**1**	**1**	**1**	**1**	**0**	**1**	**Nil**	**0**
pid50	**{A,C,D,E,F}**	**1**	**0**	**1**	**1**	**1**	**1**	**Nil**	**0**
pid51	**{B,C,D,E,F}**	**0**	**1**	**1**	**1**	**1**	**1**	**Nil**	**0**
pid52	{A,B,C,D,E,F}	1	1	1	1	1	0	28	1

Additionally pattern table as shown in table 4 is generated by extending all items in L1 with all other items from L1. The frequency for each pattern from pattern table is counted as logical AND of pattern with transactions in the database which gives output as true. The patterns having frequency count equal to zero are pruned. Hence patterns pid2, pid5, pid7, pid9, pid11, pid12, pid14, pid18, pid19, pid21, pid22, pid23, pid24, pid25. Pid28, pid30, pid34, pid35, pid36, pid38, pid39, pid40, pid41, pid42, pid44, pid45, pid46m, pid47, pid49, pid50 and pid51 from table 4 are pruned.

Table 5: Frequent 2-itemset

Item Set	Occurrence	Count
{A,B}	pid1+pid16+pid17+pid37+pid48+pid52	7
{A,D}	pid3+pid17+pid20+pid37+pid48+pid52	7
{A,E}	pid4+pid37+pid43	4
{B,C}	pid6+pid16+pid26+pid27+pid43+pid48 +pid52	9
{B,E}	pid27+pid29+pid37+pid43+pid48+pid52	9
{C,D}	pid10+pid20+pid26+pid31+pid32+pid43 +pid48+pid52	7
{D,E}	pid13+pid29+pid31+pid33+pid37+pid43 +pid48+pid52	12
{E,F}	pid15+pid33+pid52	2

Next, frequent -2 item set L2 is obtained using L1 and pattern table. For each item set from L2 the support is calculated as sum of all frequency count of matched patterns from the pattern table and check if it is frequent. Matched patterns are found as logical AND of item set with each patterns in the pattern table which gives output as the item set itself. The item set having support count greater than user defined minimum support threshold (min-supp) are considered as frequent item set. Taking item set {A B} as an example, pid1, pid16, pid17, pid37, pid48 and pid52 are found as matched patterns. Therefore sum of support counts of all these patterns is counted as support value of item set {A B} which is frequent item set as it satisfies min-supp condition.

Table 6: Frequent 3-itemset

Item Set	Occurrence	Count
{A,B,C}	pid16+Pid48+pid52	3
{A,B,D}	pid37+pid48+pid52	4
{A,C,D}	pid37+pid48+pid52	4
{B,C,D}	pid26+pid43+pid48+pid52	4
{C,D,E}	pid31+pid43+pid48+pid52	4

Table 7: Frequent 4-itemset

Item Set	Occurrence	Count
{A,B,C,D}	pid48+Pid52	2
{A,C,D,E}	pid48+Pid52	2
{B,C,D,E}	Pid43+Pid48+pid52	3

Table 8: Frequent 5-itemset

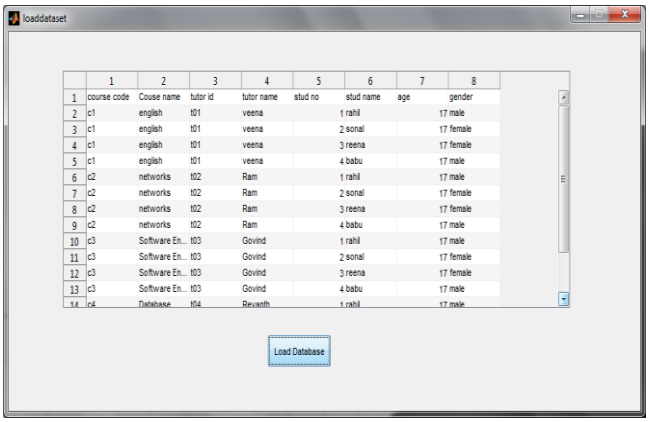
Item Set	Occurrence	Count
{A,B,C,D,E}	pid48+pid52	2

The Same procedure is repeated to obtain frequent 3-item set L3 using L2 and pattern table. Item sets {A B C} {A B D} {A C D} { B C D} and {C D E} are found to be frequent hence added to L3. The frequent 4-itemset is shown in the table 7 and frequent 5-itemset is shown in the table 8. The frequent 5 itemset is null and it cannot be generated so the process is terminated.

EXPERIMENTAL RESULT

This proposed algorithm starts when user enter query to define dataset to normalize. This algorithm will find the association rules by calling pattern generation algorithm and save resulting a form of association rules in the database. Then finally it will select some rules to use in normalization process. Finally, use the selected rules to generate the 3NF table in relational schema form. The input of proposed algorithm is the un-normalize table. The experimentation is done using matlab software with four datasets as shown in Table 9.

The un-normalized data as shown in figure 2 which then it will be analyzed by the algorithm, and then its schema in 3NF is generated.



The running example is student database. This dataset is originally un-normalized and its structure is as follows

Student_details (STUDENT_NO, STUDENT_NAME, TUTOR_ID, TUTOR_NAME, SUBJECT_CODE, SUBJECT_NAME)

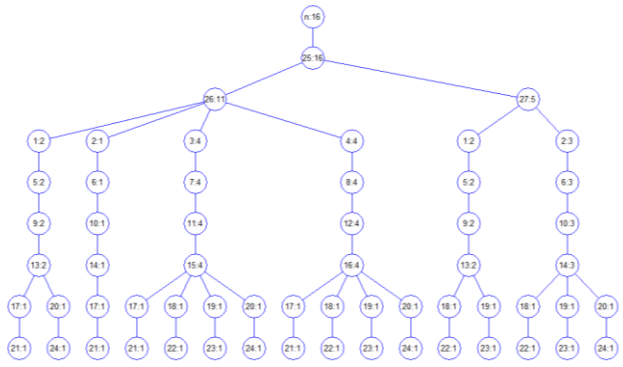
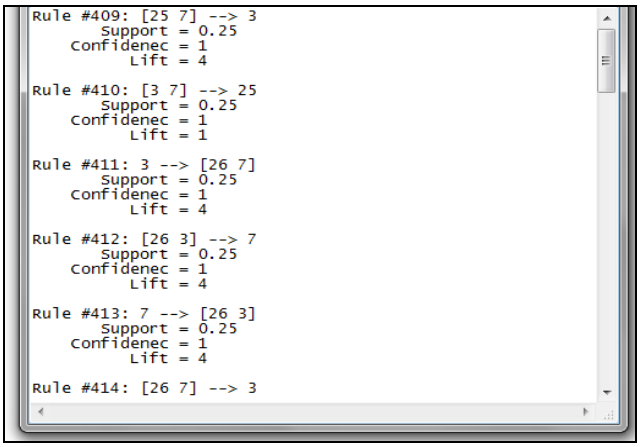


Figure 3: Tree structure Representation of Pattern Generation

The figure 3 shows the pattern generation of given dataset with its corresponding support values for finding the frequent item sets. In FP tree base pattern generation was constructed to represent each transaction in the form of corresponding branches. The new branch is created every time when a unique set of items are presented in the transaction. If same sequences are presented then the corresponding nodes count is incremented by one. Thus the above tree represents the transaction details of each student details.



The figure 4 shows the sample rules generated by the pattern generation based association rule mining

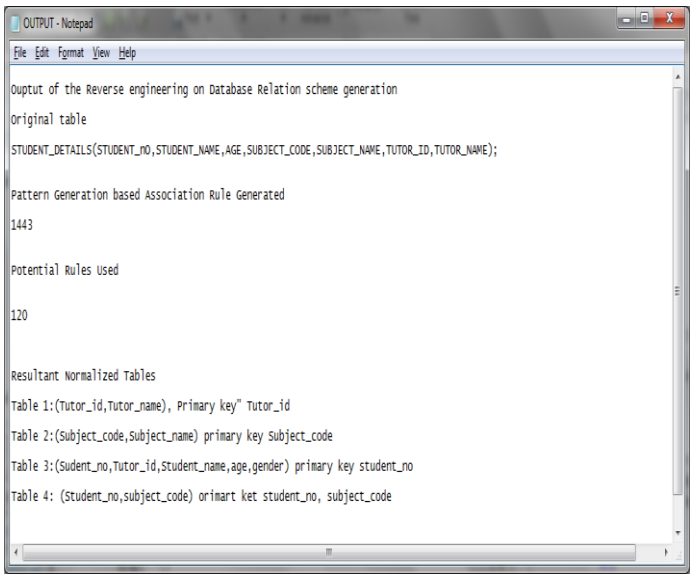


Table 9: Performance of finding potential rules and its comparison

	Total no. of rules generation		Potential rules generation	
	Apriori	**Pattern generation**	**Apriori**	**Pattern generation**
Student details	1800	1500	120	100
House rental details	2200	1800	210	150
Sales invoice details	2500	2000	260	210
Vehicle color details	1600	1145	150	90

Table 10: Accuracy

	Apriori	**Pattern generation**
Student details	75%	92%
House rental details	78%	85%
Sales invoice details	82%	90%
Vehicle color details	85%	89.60%

Based on their rules filtered the resultant accuracy is also shown in the table 10. The pattern generation method produces more accuracy in process of database reverse engineering with the values 92% for student details, 85% for house rental details, 90% for sales invoice details and 89.6% for Vehicle color details to construct the normalized database.

Due to the performance enhancement of the traditional association rule with the concept of pattern generation the number of scans done on each iterations is considerably reduced and the accuracy of the resultant output is also much better compared to the existing apriori based association rule mining to normalize the given database to achieve the goal of Database Reverse Engineering in generation relational scheme.

The time taken by the proposed pattern generation based association rule is smaller because it doesn't generates the candidate item set during each iteration it directly produces the frequent itemset with the concept of bit matrix operation among the generated patterns. Thus the apriori algorithm due its more number of scanning the whole dataset it takes more time in which it is not needed by the proposed work.

Table 11. Performance Evaluation- Apriori Vs Proposed Algorithm

No of records	Apriori Algorithm (Time in Sec)	Proposed Algorithm (Time in Sec)
100	42	12
150	58	15
200	75	18
250	92	21
300	110	23

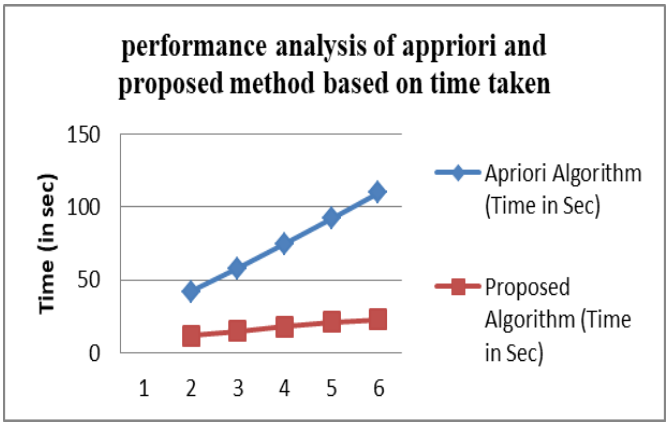


Figure 6: Performance Analysis based on time taken

The table 11 and figure 6 show the comparison of the apriori algorithm with proposed algorithm when the number of records increases the time taken is also increased. But the proposed algorithm takes optimum time for processing the given number of records.

Table 12. Performance analysis of apriori algorithm and improvised association rule mining based on the number of iterations

Minimum Support	No. of iterations required (Apriori Algorithm)	No. of iterations required (Proposed Algorithm)
15%	320	280
30%	195	148
60%	170	112
75%	110	70
80%	98	50

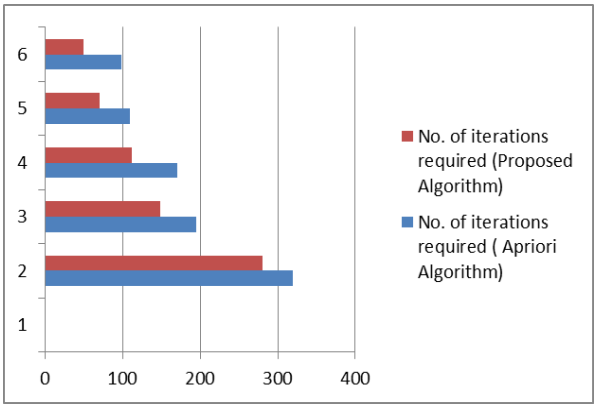


Figure 7. Performance comparison based on number of iterations

The drawback of the apriori algorithm is that it takes more number of scanning process because first it generates the candidate itemset and then only it will finds the frequent itemset for each number of itemset it is followed the same. Hence the number of iteration performed by the apriori is much higher while comparing the proposed work. The proposed algorithm only generates the frequent itemset with the help of the bit pattern and the support count. To generate next set of items it considers only those patterns which have frequent count alone is considered and their performance are shown in table 13 and figure7.

CONCLUSION

A forward engineering approach to the design of a complete database starts from the high-level conceptual design to capture detail requirements of the enterprise. Common tool normally used to represent these requirements is the entity-relationship, or ER, diagram and the product of this design phase is a conceptual schema. Typically, the schema at this level needs some adjustments based on the procedure known as normalization in order to reach a proper database design. Then, the database implementation moves to the lower abstraction level of logical design in which logical schema is constructed in a form of relations, or database tables.

In legacy systems that design documents are incomplete or even missing, the system maintenance or modification is a difficult task due to the lack of knowledge regarding high level design of the system. To tackle this problem, a database reverse engineering approach is essential. This paper adapts pattern generation based association rule mining method to discover conceptual schema from the database instances, or relations. The discovering technique is based on the association mining incorporated with some heuristic to produce a minimal set of association rules. Transformation rules are then applied to convert association rules to database dependencies. Normalization is the principal concept of our heuristic and transformation. To deduce repeating group, insert anomaly, delete anomaly and update anomaly. The simulation result shows that the proposed method outperforms the conventional association rule mining in database reverse engineering.

REFERENCE

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between set of items in large databases", in Proceedings of ACM SIGMOD International Conference on Management of Data, 1993, pp. 207-216.

[2] R. Agrawal, and R. Srikant, "Fast algorithms for mining association rules in large database", in Proceedings of 20th International Conference on Very Large Data Base, 1994, pp. 487-499.

[3] J. Han, et al., "DBMiner: System for data mining in relational databases and data warehouses", in Proceedings of CASCON'97: Meeting of Minds, 1997, pp. 249-260.

[4] S. T. Sreenath, S. Bodogala, K. Alsabti, and S. Ranka, "An efficient algorithm for the incremental updation of association rules in large databases", in Proceedings of 3rd International Conference on KDD and Data Mining, 1997, pp. 263-266.

[5] S. Tsechansky, N. Pliskin, G. Rabinowitz, and A. Porath, "Mining relational patterns from multiple relational tables", Decision Support Systems, Vol.27, No.1-2, 1999, pp. 179-195.

[6] F. Berzal, J. Cubero, N. Marin, and J. Serrano, "TBAR: An efficient method for association rule mining in relational databases", Data & Knowledge Engineering, Vol.37, No.1, 2001, pp. 47-64.

[7] J. Hipp, U. Guntzer, and U. Grimmer, "Integrating association rule mining algorithms with relational database systems", in Proceedings of 3rd International Conference on Enterprise Information Systems, 2001, pp. 130-137.

[8] H. Lee, and C. Yoo, "A form driven object-oriented reverse engineering methodology", Information Systems, Vol.25, No.3, 2000, pp. 235-259.

[9] J. Perez, I. Ramos, V. Anaya, J. M. Cubel, F. Dominguez, A.Boronat, and J. A. Carsi, "Data reverse engineering for legacy databases to object oriented conceptual schemas", Electronic Notes in Theoretical Computer Science, Vol.72, No.4, 2003, pp. 7-19.

[10] N. Lammari, I. Comyn-Wattiau, and J. Akoka, "Extracting generalization hierarchies from relational databases: A reverse engineering approach", Data & Knowledge Engineering, Vol.63, 2007, pp. 568-589.

[11] G. Chen, M. Ren, P. Yan, and X. Guo, "Enriching the ER model based on discovered association rules", Information Sciences, Vol.177, 2007, pp. 1558-1556.

[12] R. Chiang, T. Barron, and V. Storey. Reverse engineering of relational databases: Extraction of an EER model from a relational database. Data & Knowledge Engineering, 12:107– 142, 1994.

[13] R. Chiang. A knowledge-based system for performing reverse engineering of relational databases. Decision Support Systems, 13:295–312, 1995.

[14] P. Johannesson. A method for transforming relational schemas into conceptual schemas. In Rusinkiewicz, editor, Proc. of the 10th Int. Conf. on Data Engineering, pages 115–122, Houston, 1994. IEEE Press.

[15] V. Markowitz and J. Makowsk. Identifying extended entity relationship object structures in relational schemas. IEEE Transactions on Software Engineering, 16(8), Aug. 1990.

[16] C. Batini, S. Ceri, and N. Shamkant. Conceptual Database Design - An Entity-Relationship Approach. Benjamin/ Cummings, 1992.

[17] J.-M. Petit, F. Toumani, J.-F. Boulicaut, and J. Kouloumdjian. Towards the reverse engineering of denormalized relational databases. In Proc. of the 12th Int. Conf. on Data Engineering, New Orleans, USA, Feb. 1996. IEEE Press.

[18] W. Premerlani and M. Blaha. An approach for reverse engineering of relational databases. Communications of the ACM, 37(5), May 1994.

[19] O. Signore, M. Loffredo, M. Gregori, and M. Cima. Using procedural patterns in abstracting relational schemata. In Proc. of the 13th Int. Conf. on Entity-Relationship Approach,volume 881 of LNCS, Dec. 1994.

[20] M. Blaha and W. Premerlani. Object-Oriented Modelling and Design for Database Applications. Prentice-Hall, 1998.