IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 3, June-July, 2013
ISSN: 2320 - 8791
www.ijreat.org

Frequent Pattern Mining for Multi -Relational Databases Using Candidate Generation

Janvi Modi[1], Narendra Chauhan[2]

[1] **Department of Computer Engineering, Birla Vishvakarma Mahavidyalaya, Anand, India**

[2] **Department of Information Technology, D. Patel Institute of Technology, Anand, India**

Abstract

This paper presents the process of mining frequent patterns from relational database using the concept of candidate generation. The concept of generating frequent itemsets from transaction database (similar to Apriori algorithm) is extended on relational database considering multiple relations. The paper illustrates how the relational candidate itemsets are constructed and how their support is counted for multiple relations. Based on the criteria of minimum support , frequent itemsets are extracted. In the paper, the need of multi – relational data mining is justified and its categories are also explained in brief.

1. Introduction

Data mining is a field that aims at developing methods for exploring or analyzing large databases in order to extract some knowledge or innovative information from the existing data [5].Most of the data mining methods are focusing on extracting knowledge from transactional database (a single table) and this limitation of data mining methods hinders the use of an algorithm that mines patterns or knowledge from multiple relations [6]. Due to the presence of multiple relations in the relational database and dependency among these relations based on primary key – foreign key attributes, it is difficult to extend the same concepts and algorithms of frequent itemset mining on relational database. Transaction data are those data which resides in a single table and are not connected to tuples in other tables whereas relational data resides in multiple tables or relations and the tuples in one relation are connected to tuples in other relations. Data mining algorithms apply intelligent methods to mine data or patterns only from a single table. Mining in a relational database often requires mining across multiple interconnected relations. Such kind of mining across multiple data relations is well-known as multi relational data mining (MRDM), and it has been defined and elaborated by many researchers [1][2]. The need of MRDM is further justified in the following subsection.

1.1 Why MRDM?

Multi – Relational Data Mining (MRDM) is a field which has emerged to solve this limitation of data mining methods. The fact that drives this field is the increasing amount of data storage in various real time applications such as banks, sports, superstores, etc. There are many data mining algorithms available which looks for patterns but, this algorithms look for patterns in a single transaction table whereas the real – time databases needs to be stored in multiple tables [3][4][5]. This limitation induces the need of multi-relational data mining algorithms which can help in mining patterns from a relational database which consists of several tables which are semantically related [6]. MRDM aims at developing methods for extracting valuable information or knowledge from multiple tables from a relational database.

Similar to Data Mining functionalities, we can also classify MRDM functionalities into 3 main categories like multi – relational classification , multi – relational clustering and multi – relational frequent pattern mining. Multi – relational classification is used to develop a classification model that utilizes information in different relations. Multi – relational clustering is used to group tuples both from their own relation and from different relations into clusters. Multi – relational frequent pattern mining is used to find patterns that occurs frequently in the relations [2].

In this paper, the focus is given on multi – relational frequent pattern mining, for which an algorithm that mines frequent patterns from multiple relations by using the approach of candidate generation is presented and elaborated with example.

Candidate can be defined as a set of item or items and candidate generation is a method to form a set of item or items by mapping each item present in the relation on various keys. Candidate Generation produces a set of items known as Itemset which consists of all the combinations of all the items present in multiple relations [1].

2. The Algorithm

In this section, we present an algorithm to mine frequent itemsets from relational database using the candidate generation based approach. Candidate Generation approach is known among data mining researchers, which is used to mine frequent patterns from transactional data also [1]. The well-known algorithm that is used to mine frequent itemsets from transactional database using candidate generation approach is Apriori Algorithm [1]. In this paper, the same concept has been extended for achieving the task of multi – relational frequent pattern mining. A concept of finding interesting itemsets from relational database is also proposed by Goethals, et al. [7]. The algorithm presented in this work is simplified from and with some modification to the Naïve algorithm presented in [7]. The algorithm is a combination of the above two approaches which mines frequent patterns from multiple relations by generating multi – relational candidate itemsets and then finding frequent itemsets among them based on the user defined threshold. This presented algorithm also takes into consideration the mapping of items on different key attributes from different relations, which is not considered in [7]. The mapping on different keys while generating relational candidate itemsets is demonstrated using example later in this paper. The algorithm for finding frequent patterns from relational database using candidate generation based approach is shown in Fig.1.

Input: Set of relations and user defined minimum support count (min_sup).

Output: Set of frequent itemsets F generated from relational database.

Step 1: Perform full outer join on all the available relations and get a join table J.

$J = R_1 \infty R_2 \infty \ldots\ldots\ldots \infty R_n$

Step 2 : Generate relational candidate itemsets $RC_1, RC_2, \ldots, RC_n$ and Frequent Itemset $RL_1, RL_2, \ldots, RL_n$ using the following steps.

Repeat until RC_n = NULL.

2.1 To generate first relational candidate itemset RC_1 by mapping all the unique values of non-key attributes on all the key attributes.

2.2 Find the support of candidate itemsets by counting their number of occurrences in join table J.

2.3 Generate first frequent relational itemset RL1 by comparing their support counts with min_sup and discard those itemsets that are less than min_sup.

2.4 Generate next level of candidate relational itemset RC_k from RL_{k-1} by performing a self join operation on RL_{k-1} and calculate their support count from join table J.

2.5 Identify frequent itemset RL_k from RC_k by pruning those itemsets in RC_k which have support less than min_sup.

Step 3 :- The itemsets residing in RL_k , k=1..n are the Frequent Itemsets.

Figure 1 – Algorithm for finding frequent items for relational database using Candidate generation based approach

A relational database consists of multiple tables which are connected with each other based on primary key foreign key relations. The interconnections between relations (tables) of a relational database can be well understood by looking at their entity – relation diagram (ER – Diagram). The relations that used for illustration of the presented concept are shown in Tables 1(a-c) and the ER – Diagram corresponding to these relations is shown in Fig. 2.

Table 1 : Relation for Professor , Courses and Teaches

Table 1(a)

Professor		
PID	**Name**	**Surname**
A	Jan	P
B	Jan	H
C	Jan	VDB
D	Piet	V
E	Erik	B
F	Flor	C
G	Gerrit	DC
H	Patrick	S
I	Susan	S

Table 1(b)

Teaches	
PID	**CID**
A	1
A	2
B	2
B	3
C	4
D	5
D	6
E	7
F	8
G	9
G	10
G	11
I	11

Table 1(c)

Courses		
CID	**Credits**	**Project**
1	10	Y
2	10	N
3	20	N
4	10	N
5	5	N
6	10	N
7	30	Y
8	30	Y
9	10	N
10	10	N
11	10	N

Table 2 : Join Table J

Full Outer Join J					
P.PID	**P.Name**	**P.Surname**	**C.CID**	**C.Credits**	**C.Project**
A	Jan	P	1	10	Y
A	Jan	P	2	10	N
B	Jan	H	2	10	N
B	Jan	H	3	20	N
C	Jan	VDB	4	10	N
D	Piet	V	5	5	N
D	Piet	V	6	10	N
E	Erik	B	7	30	Y
F	Flor	C	8	30	Y
G	Gerrit	DC	9	10	N
G	Gerrit	DC	10	10	N
G	Gerrit	DC	11	10	N
H	Patrick	S	NULL	NULL	NULL
I	Susan	S	11	10	N

There are 3 tables shown in Table 1 out of which Table 1(a) 'Professor' and Table 1(c) 'Courses' are the entitiy tables and the Table 1(b) Teaches is the Relation table which connects the entities with each other. The ER –Diagram of the Relations shown in Table 1 is shown in Figure 2 which consists of two entities *Professor* and *Courses* which are connected to each other via a relation Teaches.

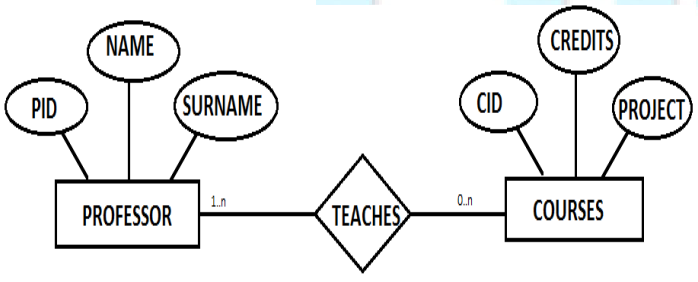

Figure 2 : ER – Diagram For Relations shown in Table 1.

In the first step of the algorithm, merging or joining of all the available relations or tables takes place into a single table using all entities and relation. Join operation cannot be performed using Equi Join [7] because if a tuple in one relation is not connected to a tuple in another relation it will not be included in the join table, which means that we loose some information and so to create a Join Table we use a *full outer join* [7]. Full outer join is said to combine the effects of both right and left outer join. The reason to choose Full outer join is that it connects all the tuples in all the relations with each other even if they are not actually connected with each other. It also correlates the non-connected tuples, if any with the help of NULL operator [7]. The join of all the available relations is performed and stored in a Join table named J.

The join table J created for relations shown in Table 1 is shown in Table 2. This join table is created using full outer join operation. This join table consists of all the combination of all the tuples given in the relation.

After performing the join operation, the join table J is used to generate candidate itemsets. *Relational Candidate itemsets* can be defined as the set of all possible combinations of items which are usually generated by mapping each non – key attribute value to each key attribute value. Each relational candidate itemset RC_k consists of k items in it's itemset.

Next, the support count of all the relational itemsets in RC_k is calculated. To calculate the support count of each relational itemset, we need to count the unique occurrences of that item from the join table J. The number of unique occurrences of that item in the join table will reflect the support count of that relational itemset.

In the next step pruning is carried out by finding out which relational itemsets from the candidate sets are frequent. This is done by generating frequent itemset RL_k. RL_k is generated from RC_k by removing those items from RC_k that are not frequent i.e. those items that have their support counts less than the minimum support threshold min_sup. The minimum support threshold is selected depending upon the requirements of the application.

Similarly more candidate and frequent itemsets are constructed until RC_n = NULL.

3. Illustration using Example

In this section, the generation of relational candidate itemsets and relational frequent Itemsets is illustration following the steps of the algorithm presented in section II.

To generate first relational candidate itemset RC_1, consider the relations shown in Table 1. We map each non – key attribute i.e. Name, Surname, Credits and Project onto it's key attributes i.e. pid and cid in our case. The relational candidate-1 itemsets RC_1 along with their support counts are shown in Table 3. The support count of each itemset is calculated from the Join Table J.

Table 3: A part of relational candidate-1 itemset (RC_1) and their support counts

RC_1 : Relational Candidate -1 Itemset	
Itemset	**Support**
$\{ Name = Jan \}_{P.PID}$	3
$\{ Name = Jan \}_{C.CID}$	4
$\{Credits=5\}_{C.CID}$	1
$\{Credits=10\}_{C.CID}$	7
$\{Surname=S\}_{P.PID}$	2
……	…….

The sementic meaning of the relational itemsets shown in Table 3 can be given as follow :

$\{ Name = Jan \}_{pid}$ => " How many unique professors (Pid) do we have whose name is Jan?"
$\{ Name = Jan \}_{cid}$ => " How many unique courses (Cid) do we have which are taught by professor's named Jan?"

To calculate the support counts of the relational itemsets shown in Table 3, the unique occurrences of these itemsets from Table 2 i.e. the Join Table are counted. The support count calculation of first relational itemset shown in Table 3 is as follows:

$\{ Name = Jan \}_{pid}$ => " How many unique professor's do we have whose name is Jan?"

Looking at Table 2, we see that we have 3 unique PID's which has Name = Jan. Those PID's are A, B and C and so the support count of this relational itemset is 3. Similarly, the support counts of all the other relational itemsets are calculated.

To generate relational frequent-1 itemset RL_1, we need to compare the support count of each relational candidate itemset in RC_1 with the minimum support count min_sup. Minimum support count can be defined as a threshold value, and the itemset having support counts below this value is not considered as frequent. The selection of the value of minimum support count or threshold is application dependent. For our example, let the minimum support min_sup be 3. Following calculations and comparison are applied on the relational candidate itemsets of Table 3.

$\{ Name = Jan \}_{pid}$ => Support = 3 ≥ min_sup. So, Accepted
$\{ Name = Jan \}_{cid}$ => Support = 4 ≥ min_sup. So, Accepted
$\{ Credits = 5 \}_{cid}$ => Support = 1 < min_sup. So, Rejected
$\{ Credits = 10 \}_{cid}$ => Support = 7 ≥ min_sup. So, Accepted
$\{ Surname = S \}_{pid}$ => Support = 2 < min_sup. So, Rejected

The part of relational frequent-1 itemsets RL_1 formed from RC_1 after comparison with min_sup are shown in Table 4.

Table 4: A part of relational frequent-1 itemsets (RL_1)

RL_1 : Relational Frequent-1 Itemset	
Itemset	**Support**
$\{ Name = Jan \}_{P.PID}$	3
$\{ Name = Jan \}_{C.CID}$	4
$\{Credits=10\}_{C.CID}$	7
…….	…….

Now, we will see how relational candidate-2 itemsets RC_2 are generated from relational frequent-1 itemsets RL_1. RC_2 can be generated by performing a self join operation [1][7] on RL_1. Self join operation means to join a table to itself.

There are number of combinations which can be formed by self joining RL_1, one of which is stated below:

$\{(Name = Jan), (Credits = 10)\}_{Pid,Cid}$ => " How many unique Professor's named *Jan* teaches a Course having Credits 10?"

A part of relational candidate-2 itemsets RC_2 are shown in Table 5. It can be observed from above stated relational candidate-2 itemset that it has been formed by mapping two keys from two relations simultaneously. This type of generation is not included in [7].

Table 5: A part of relational candidate-2 itemset (RC_2) and their support

RC_2 : Relational Candidate-2 Itemset	
Itemset	**Support**
$\{(Name = Jan), (Credits = 10)\}_{P.PID,C.CID}$	4
$\{(Name = Jan), (Credits = 10)\}_{C.CID}$	3
…….	…….

Now, to generate relational frequent-2 itemset RL_2 from RC_2, similar calculations are performed as in generating RL_1. With min_sup = 3, the combinations which comprises RL_2 are shown in Table 6.

Table 6: A part of relational frequent-2 itemsets (RL_2)

RL_2 : Relational Frequent-2 Itemset	
Itemset	**Support**
$\{(Name = Jan), (Credits = 10)\}_{P.PID,C.CID}$	4
$\{(Name = Jan), (Credits = 10)\}_{C.CID}$	3
…….	…….

Similarly, further relational candidate and relational frequent itemsets can be generated until empty candidate itemset i.e. RC_n = NULL is found.

The candidate generation and obtaining frequent itemset process demonstrated in the above example illustrates briefly the process of mining frequent itemsets from multiple relations using the candidate generation approach. Similar to the drawback of Apriori algorithm, the major drawback with this approach is that a large number of relational candidate itemsets are generated which is a computationally expensive process. The computations and number of candidate itemsets increases with the increase in size of relational databases. Hence, this candidate generation approach is more suitable for a relational database having small number of relations as compared to the database having more relations.

4. Summary and Conclusion

This paper briefly explained the need of multi-relational data mining for real world applications. In spite of much recognition and need of this field, it yet requires much exploration. In this paper, an attempt has been made to derive the process of finding frequent itemsets from small size relational database. The same theoretical concept is demonstrated using an illustrative example. The algorithm follows an extended procedure similar to candidate generation based Apriori algorithm which is applicable to transactional database. In the paper, the concepts of relational candidate itemsets and relational frequent itemsets are given and a procedure to calculate the support count as well as procedure to obtain next level of candidate itemsets based on previous frequent itemsets are also introduced.

Though using the concept of candidate generation, it is possible to extract frequent itemsets from relational database, this process generates large number of relational candidate itemsets. Even with increase in size of relational database, the number of relational candidate itemsets increases drastically. Hence, this process is computationally expensive. With modifications to the presented approach, and yet applying tree based approaches may lead to increase in efficiency and efficacy of the goal.

References

[1] M Kamber, J Han, "Data Mining: Concepts and Techniques", Morgan Kaufmann publishers, 2006, Pp (1-36),(243-276).

[2] M Kamber, J Han, "Data Mining: Concepts and Techniques", Morgan Kaufmann publishers, 2006, Pp (571-582).

[3] S.Dzeroski, "Multi-Relational Data Mining : An Introduction" , ACM SIGKDD Explorations, Volume 5 Issue 1, July 2003, Pp (1-16)

[4] S. Dzeroski ,L D Raedt , "Multi–Relational Data Mining : The Current Frontiers", ACM SIGKDD Explorations , Volume 5 Issue 1, July 2003 , Pp(100-101).

[5] P. Domingos, "Prospects and Challenges for Multi – Relational Data mining", SIGKDD Exploration, Volume 4, Issue 2, Pp (1-3).

[6] C. R Valencio , F.T Oyama, P.S Neto ,A. C. Colombini, A. M. Cansian, R.C.G Sooza & P.L.P Correa , " MR-Radix: A Multi-Relational data mining algorithm" Human-Centric Computing & Information Science A Springer Open Journal 2012, Pp (2-17).

[7] B. Goethals , W. L Page , M. Mampaey , "Mining Interesting Sets and Rules in Relational Databases.", SAC' 10 March 22-26, 2010, Sierre, Switzerland.