ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)

A Conceptual Approach of Data Warehouse Maintenance Systems using Materialized Data Mining Views

[1]P. R. Vishwanath, [2]Dr. Rajylakshmi, [3]Dr. M. Sreedharreddy, [4]Dr. P. V. Lakshmi
[1]Royal Institute of Technology and Science, Hyderabad, AP, India
[2,4]Information Technology, GITAM University, Vizag, AP, India
[3]Information Technology, GITS, Hyderabad, AP, India

I. Introduction

Data mining, also referred to as database mining or Knowledge Discovery In Databases (KDD), aims at discovery of useful patterns from large databases or warehouses. Currently we are observing the evolution of data mining environments from specialized tools to multi-purpose data mining systems offering some level of integration with existing management systems. From a user's point of view data mining can be seen as advanced querying a user specifies the source data set and the requested class of patterns, the system chooses the right data mining algorithm and returns discovered patterns to the user. The most serious problem concerning data mining queries is a long response time Current systems consume minutes or hours to answer simple queries.

Another important feature of data mining is that it is an interactive and interactive process. Users very often periodically perform the same data mining tasks to get the up-to-date information. We claim that data mining systems should provide support for such repetitive queries. It is desirable to store the results of a data mining query that will be repeated after some changes to the database because there are known algorithms for incremental data mining. In this paper we propose using periodically refreshed Materialized Data Mining Views (MDMVs) for repetitive data mining queries in the same manner as materialized views are used in relations database management systems to store results of complex and time consuming queries.

Benefits of using MDMVs to answer data mining quries where the query to be answered is the same as the query defining an existing MDMV are obvious. The question we try to answer in this paper is can we use MDMVs to efficiently answer a data mining query that is not equal but only similar to the query defining some MDMV? We consider two data mining queries similar, if they have the same scheme of source datasets and resulting patterns, and differ in selection predicates applied to the query on the source dataset and/or constraints concerning statistical strength and contents of patterns.

the concept of MDMVs and their application in the discovery of frequent item sets and association rules. Since it is straightforward to generate association rules from frequent itemsets, we focus on the frequent itemsets only. We illustrate our optimization rules with any examples expressed in Mine SQL, which is a declarative, multi-purpose SQL-like language for interactive and iterative data mining in relations databases, developed by us over the last couple of years.

A. Basic Definitions

1. Frequent Itemsets,

Let L = (l1, l2, ……. lm) be a set of literals, called items. Let a nonempty set of items T be called an itemset . Let D be a set of variable length itemsets, where each itemset TC L. We say that an itemset T supports an item xEL if x is in T. we say that an intemset T supports an itemset XCL if T supports every item n the set X. The support of the itemset X is the percentage of T in D that supports X. The problem of mining frequent itemsets in D consists in discovering all item sets whose support is above a user-defined support threshold.

2. Association Rules

An association rule is an implication of the form X-Y, where XCL , YCL, X Y= φ Each rule has associated measures of its statistical significance and strength called support and confidence. The X-Y holds in the set D with support s if s% of itemsets in D support XUY. The rule X-Y has confidence c if c% of itemsets in D that support X also support Y. The problem of mining association rules in D consists in discovering all associations rules whose support and confidence are above user-defined support thresholds for support and confidence respectively.

Data Mining Queries We have proposed a declarative language, called Mine SQL, for expressing data mining problems by means of data mining queries. Mine SQL is a SQL-based interface between a client application and a data mining system. It plays similar role to data mining applications as SQL does to traditional database applications. Mine SQL is declarative – the client application is separated from the data mining algorithm being used. Any modifications and improvements done to the algorithm do not influence the applications. Mine SQL follows the syntax philosophy of SQL language – data mining queries can be combined with SQL follows the syntax philosophy of SQL language – data mining queries can be combined with SQL queries, i.e. SQL results can be mined and Mine SQL results can be queried. Thus, existing database applications can be easily modified to use data mining methods. In this section usage are discussed.

The detailed syntax of Mine SQL language defines a set of new SQL data types, which are used to store and manage association rules and itemsets. The SET OF data types family (SET OF NUMBER, SET OF CHAR, etc) is used to represent sets of items, e.g. a shopping cart contents. In order to convert single item values into a SET OF value, we use a new SQL group function called SET.

The ITEMSET OF data types family is used to represent frequent itemsets. For an itemset its support is stored together with the set of items it contains. We define a set of SQL functions and operators that operate on rules: SIZE(x) returns the number of items in the itemset x, s CONTAINS q returns TRUE if the itemset s contains the set q, SUPPORT (x) returns support of the itemset x.

The RULE OF data type's family is used to represent association rules, containing body, head, and support and confidence values. We define a set of SQL functions and operators that operate on rules BODY(x) returns the SET OF value representing the body of the rule x, HEAD (x) returns the SET OF value representing the head of the rule x, SUPPORT (x) returns support of the rule x, CONFIDENCE (x) returns confidence of the rule x.

The central statement of the Mine SQL language is MINE. MINE is used to discover frequent item sets or association rules from the database. MINE also specifies a set of predicates to be satisfied by the returned rules or patterns. The following MINE statement

uses the PURCHASED_ITEMS table to discovery all frequent itemsets, whose support is greater than 0.1. We display the item sets and their supports.
MINI ITEMSET, SUPPORT (ITEMSET)
FOR X FROM (SELECT SET (ITEM) AS X
FRO PURCHASED_ITEMS GROUP BY T_ID)
WHERE SUPPORT (ITEMSET)>0.1

B. Data Mining Views

Relational databases provide users with a possibility of creating views and materialized views. A view is a virtual table presenting the results of the SQL query hidden in the definition of the view. Views are used mainly to simplify access to frequently used data sets that are results of complex queries. When a user selects data from a view, its defining query has to be executed but the user does not have to be familiar with its syntax.
Since data mining tasks are repetitive in nature and the syntax of data mining queries may be complicated, we propose to extend the usage of views to handle both SQL.

1. SQL Queries and Mine SQL Queries

The following statement creates the data mining view presenting the results of the data mining task discussed earlier.
CREATE VIEW BASKET_ITEMSETS
AS MINE ITEMSET
FOR X FROM (SELECT SET (ITEM) AS X FROM PURCHASED_ITEMS GROUP BY T_ID)
WHERE SUPPORT (ITEMSET)>01
In the defining statement of a data mining view, there are two classes of constraints; database constraints and mining constraints. Database constraints are located within the SELECT statement in the FROM clause of the MINE statement. Database constraints are used to apply selection conditions on the source dataset that is being mined. Mining constraints are located in the WHERE clause of the MINE statement and are used to specify selection conditions on the set of patterns to be discovered. An important advantage of data mining views is separation of applications processing results of data mining queries from predicates defining parameters of data mining algorithms. If applications access frequent patterns by means of data mining views, they do not have to be changed when only selection predicates (database or mining predicates) are changed in a data mining query. In such case only views have to be modified.
Any SQL query concerning the view presented above involves performing the data mining task according to the data mining query that defines the view. This guarantee access to up-to-date patterns but leads to long response times, since data mining algorithms are time consuming,

D. Materialized Data Mining Views

In database systems it is possible to create materialize views that materialize the results of the defining query to shorten response times. Of course, data presented by a materialized view may become invalid as the source data changes. One of the solutions minimizing effects of this problem is periodic refreshing of materialized views. In fact, in the area of data mining, changes to the source database should not be considered to be a serious problem because data mining tasks are usually performed on data warehouses rather than on operational databases. IN data warehouses, changes are applied in bulks and materialized data mining views should be refreshed only after a series of changes, together with other views existing in the data warehouse.
We introduce Materialized Data Mining Views (MDMVs) with the option of automatic periodic refreshing. A materialized data mining view is a database object containing patterns (association rules or frequent itemsets) discovered as a result of a data mining query. It contains rules and patterns that were valid at a certain point of time. MDMVs can be used for further selective analysis of discovered patterns with no need to re-run mining algorithms. They can be automatically refreshed according to a user-defined time interval. This might be useful when a user is interested in a set of rules of itemsets, whose specification does not change in time, but he or she always wants to have access to relatively recent information.
The following statement creates a MDMV containing all frequent itemsets with support greater than 0.1, discovered in the set of transactions from PURCHASED_ITEM table. The view is to be refreshed once a week.
CREATE VIEW BASKET_ITEMSETS
REFRESH 7 AS MINE ITEMSET
FOR X FROM (SELECT SET (ITEM) AS X FROM PURCHASED_ITEMS GROUP BY T_ID)
WHERE SUPPORT (ITEMSET)>01
In most cases when a MDMV is being refreshed, if can be refreshed efficiently with one of the algorithms for incremental mining. Moreover, it is desirable to store information about the time of last changes applied to the source objects, in order to detect situations when refreshing is not necessary, sine the source dataset has not changed.
Data Mining Query Rewriting with Materialized Data Mining Views
MDMVs can be also used to reduce execution time of data mining queries, which are not identical to those, on which the views were built. Consider the following example. We are given a MDMV defined over the following data mining query.
CRETIVE MATERIALIZED VIEW V1
AS MINE ITEMSET
FOR ITEMS FROM (SELECT ET (ITEM) AS ITEMS
FROM PURCHASED _ ITEMS GROUP BY T_ID)
WHERE SUPPORT (ITEMSET)-0.2
Assume that a user wants to discover frequent itemsets with the following data mining query
MINE ITEMSET
FOR ITEMS FROM (SELECT SET (ITEM) AS ITEMS
FROM PURCHASED ITEMS GROUP BY T_ID)
WHERE SUPPORT (ITEM SET)-0.2
AND ITEMSET CONTAINS TO_SET (A.D.)
Notice that in order to execute the query, we can simply filter the actual contents of the materialized data mining view V1, without running a data mining algorithm. Thus, MDMVs can play a similar role to data mining queries, an indexes or materialized views do to database queries. Application developers can create MDMVs to transparently decrease execution times of their applications' data mining queries. We need formal methods for determining data mining query execution plans, which use MDMVs to reduce time completely. First, we define four relations, which ay occur between two data mining queries, DMQ1 and DMQ2
We say that:

1. DMQ1 extends database constraints of DMQ2. If DMQ1 does one of the following

- appends a WHERE or HAVING clause of database constraints of DMQ2
- appends an additional ANDed condition to a WHERE or HAVING

clause of database constrants of DMQ2
- removes an ORed condition from WHERE or HAVING clause of database constraints of DMQ2

Example, the following data mining query DMQ1 extends database constraints of the data mining query DMQ2.

```
DMQ1:
MINE ITEMSET
FOR ITEMS
FROM (SELECT SET (ITEM) AS ITEMS
FROM PURCHASED_ITEMS
WHERE ITEM!=D' AND T_ID>100
GROUP BY T_ID)
WHERE SUPPORT (ITEMSET)>0.2
```

```
DMQ2.
MINE ITEMSET
FOR ITEMS
FROM (SELECT SET (ITEM) AS ITEMS
FROM PURCHASED_ITEMS
WHERE ITEM!=D
GROUP BY T_D)
WHERE SUPPORT (ITEMSET)>0.2
```

Intuitively, extension of database constraints means narrowing the mined dataset.

2. DMQ1 reduce database constraints of DMQ2. if DMQ1 does one of the following:
- removes a WHERE or HAVING clause of database constraints of DMQ2
- appends an additional ORed condition to a WHERE or HAVING clause of database
constraints of DMQ2
- removes an ANDed condition from a WHERE or HAVING clause of database constraints of DMQ2

Example: The following data mining query DMQ1 reduces database constraints of the data mining query DMQ2.

```
DMQ1
MINE ITEMSET
FOR ITEMS
FROM (SELECT SET (ITEM) AS ITEMS
FROM PURCHSED _ITEMS
GROUP BY T_ID)
WHERE SUPPORT (ITEMSET) >0.2
```

```
DMQ2
MINE ITEMSET
FOR ITEMS
FROM (SELECT SET (ITEM) AS ITEMS
FROM PURCHASED_ITEMS
GROUP BY T_ID
HAVING COUNT (*)>1/0)
WHERE SUPPORT (ITEMSET)> 0.2
```

Intuitively, reduction of database constraints means extending the mined data set.

3. DMQ1 extends mining constraints of DMQ2, if DMQ1 does one of the following.
- replaces SUPPORT(ITEMSET)>x with SUPPORT (ITESET)>y in DMQ2, where x<y
- replaces SUPPORT(ITEMSET)>x with SUPPORT (ITESET)<y in DMQ2, where x<y
- replaces ITEMSET CONTAINS X with ITEMSET CONTAINS Y in DMQ2, where XCY
- replaces ITEMSET CONTAINS X with ITEMSET CONTAINS Y in DMQ2, where XCY
- replace SIZE (ITEMSET)>x with SIZE(ITEMSET)>y in DMQ2, where x<y
- replace SIZE (ITEMSET)<x with SIZE(ITEMSET)< y in DMQ2, where x<y
- appends a WHERE or HAVING clause of mining predicates of DMQ2
- appends an additional AND ed condition a WHERE or HAVING clause of mining constraints of DMQ2
- removes an ORed condition from a WHERE or HAVING clause of mining constraints of DMQ2

Example : The following data mining query DMQ1 extends mining constraints of the data mining query DMQ2

```
DMQ1:
MINE ITEMSET
FOR ITEMS
FROM ( SELECT SET(ITEM) AS ITEMS
FROM PURCHASED_ITEMS
GROUP BY T_ID)
WHERE SUPPORT (ITEMSET)>0.5;
```

```
DMQ2
MINE ITEMSET
FOR ITEMS
FROM (SELECT SET (ITEM) AS ITEMS
FROM PURCHASED_ITES
GROUPED BY T _ID)
WHERE SUPPORT (ITEMSET)>0.2
```

Intuitively, extension of mining constraints means narrowing the resulting set of discovered patterns.

4. DMQ1 reduces mining constraints of DMQ2, if DMQ1 does one of the following.
- replaces SUPPORT(ITEMSET)>x with SUPPORT (ITESET)>y in DMQ2, where x<y
- replaces SUPPORT(ITEMSET)>x with SUPPORT (ITESET)< in DMQ2, where x<y
- replaces ITEMSET CONTAINS X with ITEMSET CONTAINS Y in DMQ2, where XCY
- replaces ITEMSET CONTAINS X with ITEMSET CONTAINS Y in DMQ2, where XCY
- replace SIZE (ITEMSET)>x with SIZE(ITEMSET)>y in DMQ2, where x<y
- replace SIZE (ITEMSET)<x with SIZE(ITEMSET)<y in DMQ2, where x<y
- appends a WHERE or HAVING clause of mining predicates of DMQ2
- appends an additional an ORed condition from a WHERE or HAVING clause of mining constraints of DMQ2
- removes an ANDed condition a WHERE or HAVING clause of mining constraints of DMQ2

Example : The following data mining query DMQ1 extends mining constraints of the data mining query DMQ2

```
DMQ1: MINE ITEMSET
FOR ITEMS
FROM ( SELECT SET(ITEM) AS ITEMS
FROM PURCHASED_ITEMS
GROUP BY T_ID)
WHERE SUPPORT (ITEMSET)>0.4;

DMQ2:
MINE ITEMSET
FOR ITEMS
FROM ( SELECT SET(ITEM) AS ITEMS
FROM PURCHASED_ITEMS
GROUP BY T_ID)
WHERE SUPPORT (ITEMSET)>0.2;
```

Intuitively, reduction of mining constraints means expanding the resulting set of discovered patterns. We also define four different mining methods, which will be used to execute data mining queries over MDMVs full mining, incremental mining complementary mining, and verifying mining. Full mining (FM) refers to executing a complete algorithm for discovering frequent itemsets (e.g. [2]). This method is used if MDMV contents are unusable to execute the data mining quary. Incremental mining (IM) refers to discovering frequent itemsets in an incremented data set (e.g. [4]). It can be used for data mining queries which reduce database constraints. Complementary mining (CM) refers to discovering frequent itemsets based on currently materialized itemsets, which will remain frequent (e.g. (10)]. This method can be used for data mining queries which reduce mining constraints. Finally, we have verifying mining (VM) that simply consists in pruning those materialized itemsets, which do not satisfy mining constraints. It is used for data mining queries, which extend mining constraints.

If two relations occur between a data mining query and a data mining query on which a MDMV is based, then we use the compatibility table (see table 1) to decide what mining method to use.

Table 1: Compatibility Table for Using Materialized Data Mining Views

	Reduction of database constraints	Extension of database constraints	--------
Reduction of mining constraints	CM, IM	FM	CM
Extension of mining constraints	VM, IM	FM	VM
	IM	FM	------

Example : We are given the following data mining query DMQ1 and the materialized data mining view MDMV1

```
DMQ1
MINE ITEMET
FOR ITEMS
FROM (SELECT SET(ITEM) AS ITEMS
FROM PURCHASED _ITEMS
GROPU BY T_ID)
WHERE SUPPORT/ITEM SET)>0.4;
```

```
MDMV1
MINE ITEMET
FOR ITEMS
FROM (SELECT SET (ITEM) AS ITEMS
FROM PURCHASED_ITEMS
GROUP BY T_ID
HAVING COUNT (*) >10)
WHERE SUPPORT( ITEMSET)>2;
```

Since DMQ1 extends mining constraints (higher minimum support) and reduces database constraints (removed HAVING clause) of the data mining query MDMV, we perform verifying mining (VM), and then incremental mining (IM). The verifying mining prunes all materialized itemsets, whose support value is not above 0.4 while the incremental mining discovers frequent itemsets using the information on frequent itemsets discovered in a subset of the mined data set.

References

[1] Gupta., H., Mumick, I.S,"Selection of views to materialized under a maintenance cost constraint", International Conference on Database Theory (ICDT). Jerusalem, Israel, 1999.

[2] Harinarayan, V., Rajaraman, A., Ullman, J.D.,"Implementing data cubes efficiently, ACM", SIGMOD International Conference on Management of Data (SIGMOD), Montreal, Canada, 1996.

[3] Kalnis, P., Mamoulis, N., Papadias, D.,"View selection using randomized search", Data & Knowledge Engineering, 42 (1), 89-111, 2002.

[4] Kotidis, Y., Roussopoulos, N,"A Case for dynamic. View Management. ACM Transactions on Database Systems", 26(4), 388-423, 2001.

[4] Labrinidis, A., Roussopoulos, N.,"Web View materialization. ACM SIGMOD sInternational Conference on Management of Data (SIGMOD)", Dallas, Texes, 2000.

[5] Lee, M., Hammer, J.,"Speeding up materialized view selection in data warehouses using a randomized algorithm", International Journal of Cooperative Information system (IJCIS), 10(3), 327-353, 2001.

[6] Liang, W,"Making multiple views self-maintainable in a data warehouse". Data & Knowledge Engineering. 30(2), 121-134, 1999.

[7] Quass, D., Gupta, A., Mumick, I.S., Widom, J.,"Making views self-maintainable for data warehousing", International Conference on Parallel and Distributed Information Systems (PDIS), Florida, 1996.

[8] Ross, K., Srinivasa, D., Sudarshan S,"Materialized view maintenance and integrity constraint checking", Trading space for time. ACM SIGMOD International Conference on Management of Data (SIGMOD), Montreal, Canada, 1996.

[9] Shukla, A – Deshpande, P., Naughton, J,"Materialized view selection for multidimensional datasets", International Conference on Very Large Data Bases (VLDB), New York, 1998.

[10] Theodoratos, D.,"Complex view selection for data warehouse self-maintainability", International Conference on Cooperative Information Systems (Coop IS), Eilat, Israel, 2000.

Viswanath Raghava .P is currently working as Associate professor at Royal Institute of technology and Science, Hyderabad. He is presently doing research in Data mining and data warehousing at GitamUniversity, Vizag. He has published 6 international and1 national journals.

Dr. D.RajyaLakshmi is working as Professor in the department of Information Technology, GITAM Institute of Technology, GITAM University, Andhra Pradesh, India. She was awarded Ph.D. from Jawaharlal Nehru Technological University in the area of Image Processing. She has 18 years of teaching and research experience. She has more than twenty six papers were presented and published in various conferences and journals respectively.

Dr. M. Sreedhar Reddy is currently working as professor and Head of Geetanjaliof Engineering and Technology, institute, JNTU, Hyderabad. He has published 15 international and nine national level conferences and journals. His Research areas softwareengineering, Datamining, Networks and ecommerce.

Dr. P. V. Lakshmi is working as Professor Head of the Department in the department of Information Technology, GITAM Institute of Technology, GITAM University, and Andhra Pradesh, India. . She has 18 years of teaching and research experience. She has more than twenty papers were presented and published in various conferences and journals respectively.