QUERY LANGUAGES FOR ASSOCIATION RULES

s.l. drd. ing.ec. Mirela Danubianu

« Stefan cel Mare » University Suceava
mdanub@eed.usv.ro

Abstract. Knowledge discovery from huge databases is one of the most actual problem. To solve it we can use an inductive database. With an inductive database the analyst performs a set of very different operations on data using a special-purpose language, powerful enough to perform all the required manipulations, such as data preprocessing, pattern discovery and post-processing. In this paper we present two query languages (MSQL and MINE RULE) that have been proposed for mining the association rules and discuss their common features and differences.

Keywords: association rules, query languages, data mining

1 INTRODUCTION

Knowledge Discovery in Databases (KDD) (Han and Kamber 2001) is a complex process which involves many steps that must be done sequentially. If we use an inductive database, that integrates raw data with knowledge extracted from raw data, materialized under the form of patterns into a common framework that supports the KDD process, then, this process consists essentially in a querying process, enabled by an ad-hoc, powerful and universal query language that can deal both with raw data or patterns.

There are some query languages that can be considered adequate for inductive databases. Most of this languages emphasize one of the different phases of the KDD process. This paper is a evaluation of two products: MSQL (Imielinski et al.,1996; Imielinski and Virmani. 1999) and MINE RULE (Meo et al., 1996). In Section 2 we mention the desired properties of a language for mining an inductive database, Section 3 introduces the main features of the analyzed languages, whereas in Section 4 some real examples of queries are discussed, so that the comparison between the languages is straightforward.

2 REQUIRED PROPERTIES OF A DATA MINING LANGUAGE

A query language for inductive databases, is an extension of a database query language that includes primitives for supporting the steps of a KDD process, that are:
–***the selection of data to be mined.*** The language must offer the possibility to select, to manipulate and to query data and views in the database. It must also provide support for multidimensional data manipulation.
–***the specification of the type of patterns to be mined***.
–***the specification of the needed background knowledge***
–***the definition of constraints that the extracted patterns must satisfy***. This implies that the language allows the user to define constraints that specify the interesting patterns.
–***the post-processing of results.*** The language must allow to browse the patterns, apply selection templates, cross over patterns and data, e.g., by selecting the data in which some patterns hold, or aggregating results.

3 QUERY LANGUAGES FOR ASSOCIATION RULES MINING

MSQL has been described in (Imielinski et al.,1996; Imielinski and Virmani. 1999). The main features of MSQL, are the following: ability to nest SQL expressions such as sorting and grouping in a MSQL statement and allowing nested SQL queries, cross-over between data and rules with operations allowing to identify subsets of data satisfying or violating a given set of rules, distinction between rule generation and rule querying. As the volume of generated rules might explode, rules might be extensively generated only at querying time, and not at generation time.

The language comprises five basic statements: *GetRules* that generates rules into a rule base; *SelectRules* that queries the rule base; *Create Encoding* that efficiently encodes discrete values into continuous valued attributes; *satisfies* and *violates* that allow to cross-over data and rules, and that can be used in

MINE RULE is an extension of SQL, and its description can be found in (Meo et al., 1996) and [Meo et al., 1997). This operator extracts a set of association rules from the database and stores them back in the database in a separate relation. Its main features are the following: selection of the relevant set of data for a data mining process, definition of the structure of the rules to be mined and of constraints applied at different granularity levels, definition of the grouping condition that determines which data of the relation can take part to an association rule, definition of rule evaluation measures (i.e., support and confidence thresholds).

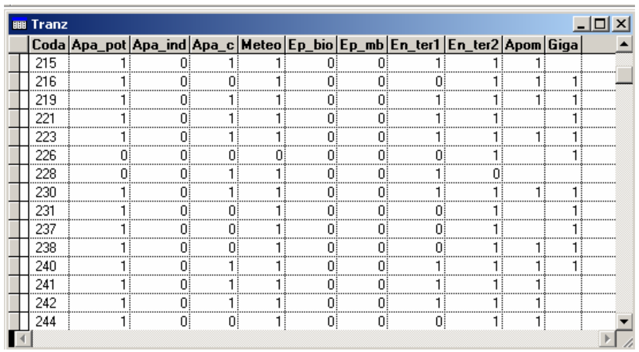
The selection above mentioned as the first feature of MINE RULE is applied at different granularity levels, that is at the row level or at the group level. The second feature defines unidimensional association rule (i.e., its elements are the different values of the same dimension or attribute), or multidimensional one (each rule element involves the value of more attributes). Furthermore, rules constraints belong to two categories: the former ones are applied at the rule level (mining conditions), while the second ones (cluster conditions), are applied at the body or head level (i.e., the sets of rule elements that compose each rule).

4 EXAMPLE

We describe here an analysis problem that will serve as example for a comparison between the three languages: we are considering information of Figure 1 and we are looking for association rules between the items and the measurement systems.

The data derives from a relational database which emphasizes the contracted services supplied by producer of public utilities by different clients identified by a code (coda). The considered data are: the agent's code (coda), the provided services and the water and thermic energy measurement.

We are considering a complete KDD process. We will consider one manipulation at the pre-processing step (selection of the items that are measured by various instruments), crossing-over between extracted rules and original data (selecting tuples of the source table that violate all the extracted rules of size 3),and two post-processing operations (selection of rules with 2 items in the body and selection of rules having a maximal body).



Coda	Apa_pot	Apa_ind	Apa_c	Meteo	Ep_bio	Ep_mb	En_ter1	En_ter2	Apom	Giga
215	1	0	1	1	0	0	1	1	1	
216	1	0	0	1	0	0	0	1	1	1
219	1	0	1	1	0	0	1	1	1	1
221	1	0	1	1	0	0	1	1		1
223	1	0	1	1	0	0	1	1	1	1
226	0	0	0	0	0	0	0	1		1
228	0	0	1	1	0	0	1	0		
230	1	0	1	1	0	0	1	1	1	1
231	1	0	0	1	0	0	0	1		1
237	1	0	0	1	0	0	0	1		1
238	1	0	0	1	0	0	0	1	1	1
240	1	0	1	1	0	0	1	1	1	1
241	1	0	1	1	0	0	1	1	1	
242	1	0	1	1	0	0	1	1	1	
244	1	0	0	1	0	0	0	1	1	

Fig. 1 Sample of source file used for the comparation of the two query languages

MSQL Table presented in Figure1 corresponds to the source data encoded in the input format used by MSQL. There are as many boolean attributes as there are possible items.

Pre-processing :selection of the subset of data to be mined.We are interested only in services measured with an instrument. MSQL requires that we
make a selection of the subset of data to be mined, before the extraction task.

The relation on which we will work is supposed to have been correctly selected from a pre-existing set of data, by means of a view, named VContr.

Rules extraction. We want to extract rules associating between the product contracted and the measurement system having a support over 25 % and a confidence over 80 %.

```
GETRULES (VContr) INTO VContrR
        WHERE BODY has {(apa_pot=1) OR
        (apa_ind=1)OR (apa_c=1) OR (meteo=1)
        OR (ep_bio=1) OR (ep_mb=1) OR
        (en_ter1=1)OR (en_ter2=1)}AND
        Consequent is {(Apom =1)} AND
         support>=0.25 AND confidence>=0.8
```

This example puts in evidence a limit of MSQL :if the number of items is high, the number of predicates in the WHERE clause increases correspondingly!

Crossing-over: looking for exceptions in the original data. Finally, we select tuples from *VContr* that violate all the extracted rules of size 3.

```
SELECT *FROM VContr
       WHERE VIOLATES ALL
       (SELECTRULES(VContrR)WHERE length=3)
```

Post-processing step 1:manipulation of rules. We select rules with 2 items in the body. As MSQL is designed to extract rules with one item in the head and as it provides access only to the extracted rules (not to the originating itemsets), we must specify that the total size of the rule is 3.

```
SelectRules(VContrR)where length=3
```

Post-processing step 2: extraction of rules with a maximal body is equivalent to require that there is no couple of rules with the same consequent, such that the body of one rule is included in the body of the other one.

```
SELECTRULES (VContrR) AS R1
       WHERE    NOT    EXISTS    (SELECTRULES
(VContrR)AS R2
       WHERE R2.body has R1.body
         AND NOT (R2.body is R1.body)
         AND R2.consequent is R1.consequent)
```

Clearly, the main advantage of MSQL is that it is possible to query knowledge as well as data, by using *SelectRules* on rule-bases and *GetRules* on data (and it is possible to specify if we want rules to be materialized or not). Another good point is that MSQL has been designed to be an extension of classical SQL, making the language quite easy to understand. But MSQL should be extended, particularly with a better handling of pre-and post-processing steps.

MINE RULE We include the same information of Figure 1 into a normalized relation *Contracts* over a schema *(coda, tip, tip_m).*

Pre-processing :selection of the subset of data to be mined. In contrast to MSQL , MINE RULE does not require to apply some pre-defined view on the original data. As it is designed as an extension to SQL, it perfectly nest SQL, and thus, it is possible to select the relevant subset of data to be mined by specifying it in the WHERE clause of the query.

Rules extraction. In MINE RULE, we specify that we are looking for rules associating one or more services (rule's body) and measurement system (rule's head):

```
MINE RULE ContrR AS
       SELECT DISTINCT 1..n item AS BODY,
       1..1 tip_m AS head,
       SUPPORT, CONFIDENCE
       FROM Contracts WHERE tip_m="A"
       GROUP BY coda
       EXTRACTING RULES WITH SUPPORT:0.25,
       CONFIDENCE:0.8
```

Extracted rules are stored into the table *ContrR (rid, bid, hid, supp, conf)* where *rid ,bid*, *hid* are the identifiers assigned to rules, body itemsets and head itemsets. The body and head itemsets are stored in tables *ContrR_B (bid,bodySchema)* and *ContrR_H(hid,headSchema).*

Crossing-over: looking for exceptions in the original data. We want to find tuples of the original relation violating all rules with 2 items in the body. As rule's components (bodies and heads) are stored in relational tables, we use an SQL query to manipulate itemsets. The correspondent query is presented here:

```
SELECT * FROM Contracts AS S1
WHERE NOT EXISTS
    (SELECT * FROM ContrR AS R1
    WHERE (SELECT tip_m FROM ContrR_H
       WHERE hid=R1.hid)=S1.tip_m
    AND (SELECT COUNT(*)FROM ContrR_B
       WHERE R1.bid=ContrR_B.bid)=2
    AND NOT EXISTS
       (SELECT *FROM ContrR_B AS I1
       WHERE I1.bid=R1.bid AND NOT EXISTS
              (SELECT *FROM Contracts AS S2
              WHERE    S2.coda=S1.coda    AND
              S2.tip=I1.tip )));
```

This query is hard to write and to understand. It aims at selecting tuples of the original table such that there are no rules of size 3 that hold in it. To check that, we verify that the consequent of the rule occurs in a tuple associated to a transaction and that there are no items of the rule 's body that do not occur in the same transaction.

Post-processing step 1: Once again, as itemsets corresponding to rule's components are stored in tables (ContrR_B ,ContrR_H), we can select rules having two items in the body with a simple SQL query.

```
SELECT * FROM ContrR AS R1 WHERE 2=
       (SELECT COUNT(*)FROM ContrR_B R2
        WHERE R1.bid=R2.bid);
```

Post-processing step 2:selection of rules with a maximal body.

We select rules with a maximal body for a given consequent. As rules' components are stored in relational tables, we use again a SQL query to perform such a task.

```
SELECT * FROM ContrR AS R1
       WHERE NOT EXISTS
       (SELECT * FROM ContrR AS R2
       WHERE R2.hid=R1.hid
       AND NOT R2.bid=R1.bid
       AND NOT EXISTS
              (SELECT *
              FROM ContrR_B AS B1
              WHERE R1.bid=B1.bid AND
               NOT EXISTS
                      (SELECT *
                     FROM ContrR_B AS B2
                     WHERE     B2.bid=R2.bid
              AND B2.tip=B1.tip)))
```

The first advantage of MINE RULE is that it has been designed as an extension to SQL. Moreover, as it perfectly nests SQL, it is possible to use classical statements to pre-process the data, and, for instance, select the subset of data to be mined. Like MSQL, data pre-processing is limited to operations that can be expressed in SQL: it is not possible to sample data before extraction, and the discretization must be done by the user. Like MSQL, MINE RULE allows the user to specify some constraints on rules to be extracted (on items belonging to head or body, on their cardinality as well as more complex constraints based on the use of a taxonomy). Also, like MSQL, MINE RULE is essentially designed around the extraction step, and it does not provide much support for the other KDD steps (e.g., post-processing tasks must be done with SQL queries).

5 CONCLUSIONS

In this paper, we have considered various features of two languages, MSQL and MINE RULE, that extract association rules from a relational database.

Then, we have compared them with the desired properties of an ideal query language for inductive databases, and we have presented a set of queries taken from data mining practice and have discussed the suitability of these languages for querying inductive databases.

The outcome is that no language presents all the desired properties: MSQL seems the one that offers the larger number of primitives tailored for post-processing and an on-the-fly encoding, specifically designed for efficiency, MINE RULE is the only one that allows to dynamically partition the source relation into groups from which the rules will be extracted; a second level of grouping, the clusters, from which more sophisticated rule constraints can be applied, is also possible. Furthermore, it looks as the only one with an algebraic semantics, what could become an important positive factor when query optimization issues will be addressed.

However, one of the main limits of all the languages is the insufficient support of post-processing issues. Whatever the language is, the user must use one of the predefined built-in options. This problem becomes crucial when considering user-defined post-processing operations involving something else than rule 's component, support and confidence.

This study allows us to conclude that the path to reach the maturity in inductive database technology is still far to be reached. However, the limits and the merits of the current query languages to give support to the knowledge discovery process have been already identified.

REFERENCES

Imielinski,T.,Virmani,A.,Abdulghani,A.(1996) :DataMine: Application Programming Interface and Query Language for Database Mining.Proc.of the 2nd Int.Conf.on Knowledge Discovery and Data Mining, KDD '96.3, p256 – 261.

Imielinski,T.,Virmani,A.:MSQL (1999) :A QueryLanguage for Database Mining.Data Mining and Knowledge Discovery.3 , p.373 – 408.

Han,J.,Kamber,M. (2001):Data Mining –Concepts and Techniques. Morgan Kaufmann Publishers .

Meo,R.,Psaila,G.,Ceri,S. (1996) :A New SQL-like Operator for Mining Association Rules. Proc.of the 22nd Int.Conf.of Very Large Data Bases.Bombay, India

Meo,R.,Psaila,G.,Ceri,S. (1997) :An Extension to SQL for Mining Association Rules. Data Mining and Knowledge Discovery.9:4 .