ELSEVIER

Expert Systems with Applications 28 (2005) 395–407

Expert Systems with Applications

www.elsevier.com/locate/eswa

Data mining in deductive databases using query flocks

Ismail H. Toroslu[a], Meliha Yetisgen-Yildiz[b,*]

[a]Middle East Technical University, Department of Computer Engineering, Ankara, Turkey
[b]University of Washington, The Information School, Mary Gates Hall, Room 416, Box 352840, Seattle, WA 98195, USA

Abstract

Data mining can be defined as a process for finding trends and patterns in large data. An important technique for extracting useful information, such as regularities, from usually historical data, is called as association rule mining. Most research on data mining is concentrated on traditional relational data model. On the other hand, the query flocks technique, which extends the concept of association rule mining with a 'generate-and-test' model for different kind of patterns, can also be applied to deductive databases. In this paper, query flocks technique is extended with view definitions including recursive views. Although in our system query flock technique can be applied to a data base schema including both the intensional data base (IDB) or rules and the extensible data base (EDB) or tabled relations, we have designed an architecture to compile query flocks from datalog into SQL in order to be able to use commercially available data base management systems (DBMS) as an underlying engine of our system. However, since recursive datalog views (IDB's) cannot be converted directly into SQL statements, they are materialized before the final compilation operation. On this architecture, optimizations suitable for the extended query flocks are also introduced. Using the prototype system, which is developed on a commercial database environment, advantages of the new architecture together with the optimizations, are also presented.

Keywords: Data mining; Association rule mining; Query flock; Deductive databases; Recursive query evaluation

1. Introduction

1.1. Association rule mining

Since *data mining* is a process of finding trends and patterns in large data, it is usually applied on historical data, like organizational operations or customer transactions, in order to discover hidden regularities which can be used to improve the decision making process. In recent years, many organizations have begun to routinely capture huge volumes of data related to all levels of their operations. Nowadays, this huge historical data is used in decision making in many different ways, such as analyzing medical outcomes, detecting credit card fraud, predicting the personal interests of web users, and optimizing manufacturing processes.

One of the most popular data mining application is known as *market basket analysis* (Agrawal, 1994; Agrawal, Imielinski, & Swami, 1993). In this simple application, customers' purchase behaviors are tried to be predicted from customer transactions data. A market basket data is a set of items purchased by a customer in a single transaction. By using the basket data, one can make decisions, like how to place goods on the shelves or how to make sales promotions, in order to increase the profit of a supermarket. Therefore, the sets of items that are related must be determined from the market basket data. *Association rules* describe these kind of relations among the item sets (Agrawal, 1994; Agrawal et al., 1993; Fayyad, Piatetsky-Shapiro, Smyth, & Uthurusamy, 1996; Hand, Mannila, & Smyth, 2001; Ramakrishnan, 1997; Srikant and Agrawal, 1995; Tsur et al., 1998). Association rules are a class of simple but powerful regularities in large data. An association rule is in the form of $A \Rightarrow B$, where, in the case of market basket analysis, both A and B are sets of items. Such a rule means 'if every item in A is purchased then it is likely that items in B will also be purchased'. There are two important measures related to association rules:

- *Support* shows what percent of all the transactions should include ($A \cup B$). If support for a rule is low, the rule may

* Corresponding author. Tel.: +1 2062216402; fax: +1 2066163152.
E-mail address: melihay@u.washington.edu (M. Yetisgen-Yildiz).

0957-4174/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2004.12.001

have arisen purely by chance. For example, if support for rule *pencil* ⇒ *soap* is low, only a small percentage of the transactions involve both *pencil* and *soap*, which means we do not have enough evidence to draw conclusions about the correlations between *pencil* and *soap* purchases.
- *Confidence* is the probability of items in *B* to be in the basket, given that the ones in *A* are in the basket. It indicates the degree of correlation between purchases of these sets of items. For example, if the rule *pencil* ⇒ *soap* has a low confidence, then purchase of a *pencil* is not highly correlated with the purchase of *soap* in the given database.

In order to search the sets with high support *a-priory property* is used, which can be defined as follows:

A-priory Property. Every subset of a frequent item-set must also be a frequent item-set.

A-priory property uses the fact that if a set of items *S* appears in *c* baskets then any subset of *S* appears in at least *c* baskets. For example, let us consider a database containing information about market baskets. Each time a customer appears at the cash register, the set of items bought is entered at the database. The database is a relation *baskets*(*BID*,*Item*), giving pairs consisting of a basket ID and an item that appeared in the basket. If we are looking for pairs of items that appear in at least *c* baskets then we can start by finding those items that by themselves appear in at least *c* baskets. If *c* is large enough most of the tuples in the *baskets* relation can be eliminated before joining baskets with itself to count occurrences of pairs of items.

1.2. Query flocks

Query flock as presented in Tsur et al. (1998), is a generate-and-test system, in which a family of queries that are identical, except for the values of one or more *parameters*, are asked simultaneously. The answers to these queries are filtered and those that pass the filter test enable their parameters to become part of the answer to the query flock. Briefly, query flocks generalize the association rule mining for larger class of problems. The idea of flocks is to perform complex data analysis on relational database schemas including a number of relations. However, automatic generation of rules by pushing the constraints down into rule generation in ordinary association rule mining technique is lost in the query flock. Instead, the user has to specify the pattern of the rule and the system tests it, and determines if this association has enough support and confidence. On the other hand, in a query flock, complex rules including several relations can be generated. In order to define query flocks a language to express parameterized queries, and, a language to express filter conditions about the results of the queries are needed. *Datalog*, (Ramakrishnan, 1997; Tsur et al., 1998), is used for specifying the query part. In the query part, a-priory trick to arbitrary flocks can also be expressed easily in datalog. For the filter language, SQL aggregates can be used as in HAVING clauses in order to define the support and the confidence of the rule searched by the query flock. A query flock can be defined by specifying:

- One or more predicates representing the relations,
- A set of parameters appearing in these relations whose names beginning with $,
- A query expressed in a datalog which might include ordinary variables, parameters and constants,
- A filter condition that the results of the query must satisfy in order to specify the support and the confidence.

The query flock means a set of tuples that represent acceptable assignments of values for the parameters. The acceptable parameter assignments are determined by trying all such assignments in the query, evaluating the query, and checking whether the result passes the filter test. For example (from Tsur et al., 1998) on the market basket analysis for pairs of items $*I*1 and $*I*2 that satisfy the filter condition is given as:

```
QUERY:
  ans1(B):-baskets(B,$I1),
           baskets(B,$I2),
           $I1<$I2,
  ans2(B):-baskets(B,$I1),
FILTER1:
  COUNT(ans1.B)>N
FILTER2:
  COUNT(ans1.B)>COUNT(ans2.B)*c.
```

The first filter condition represents the support by specifying that the pairs of items should appear in at least *N* basket. Even though in most applications the support is specified in terms of the percentage of the total number of transactions, as in this example it is also possible to specify it as the count of the transactions as well. The second filter condition represents the confidence of the rule '$*I*1 implies $*I*2' by using a coefficient *c*. In this example, confidence is specified as the percentage of transactions including both items $*I*1 and $*I*2 to the number of transactions including only item $*I*1. Since query flock is a query about its parameters, the parameter values are the ones that satisfy minimum support and confidence conditions and they represent the rules about the related items.

Query flocks have been successfully applied to text mining applications. In the TopCat project Clifton (to appear); Clifton and Cooley (1999), documents have been viewed as market baskets of named entities and query flocks have been used in identifying topics that recur in documents of a text corpus.

1.3. Query flock architecture

In this paper, we define a *query flock compiler*, using tightly coupled approach with the database which was