Query Rewriting in Itemset Mining

Rosa Meo, Marco Botta, and Roberto Esposito
Dipartimento di Informatica, Università di Torino, Italy
{meo,botta,esposito}@di.unito.it

Abstract. In recent years, researchers have begun to study inductive databases, a new generation of databases for leveraging decision support applications. In this context, the user interacts with the DBMS using advanced, constraint-based languages for data mining where constraints have been specifically introduced to increase the relevance of the results and, at the same time, to reduce its volume.

In this paper we study the problem of mining frequent itemsets using an inductive database\(^1\). We propose a technique for query answering which consists in rewriting the query in terms of union and intersection of the result sets of other queries, previously executed and materialized. Unfortunately, the exploitation of past queries is not always applicable. We then present sufficient conditions for the optimization to apply and show that these conditions are strictly connected with the presence of functional dependencies between the attributes involved in the queries.

We show some experiments on an initial prototype of an optimizer which demonstrates that this approach to query answering is not only viable but in many practical cases absolutely necessary since it reduces drastically the execution time.

1 Introduction

The problem of mining association rules and, more generally, that of extracting frequent sets from large databases has been widely investigated in the last decade [1, 21, 15, 23, 4, 18, 24]. These researches addressed two major issues: on one hand, performance and efficiency of the extraction algorithms; on the other hand, the exploitation of user preferences about the patterns to be extracted, expressed in terms of constraints. For instance, Ng et al. [15] proposed a constrained frequent set mining framework within which the user can use a rich set of constraints that must be satisfied by the searched rules and that include SQL-style aggregate and non-aggregate predicates. These constraints can be exploited to guide the mining process, by pushing them deeply in the mining algorithms, in order to prune the search space of frequent itemsets as early as possible.

Constraints are widely exploited also in data mining languages, such as in [10, 14, 8, 23, 24] where the user specifies in each data mining query, not only the constraints that the items must satisfy, but also different criteria to create groups

\(^1\) This work has been funded by EU FET project cInQ consortium on discovering knowledge by Inductive Queries (IST-2000-26469).
of tuples from which itemsets will be extracted. Constraint-based mining languages are also the main key factor of inductive databases proposed by Mannila and Imielinski in [9], in order to leverage decision support systems. These new promising approaches to mining will become really effective only when efficient optimizers for the mining languages will be available, i.e., if it will be possible to execute a query exploiting the available information in the database, such as the constraints in the schema, the indices or the results of previously executed queries.

In this paper, we introduce a very generic constraint-based language for the extraction of frequent itemsets and study the conditions under which query rewriting in the constraint-based mining language is possible.

By query rewriting we mean the determination of a relational expression on a set of queries whose result is equivalent to the result of a given query for every database on the same schema. Query rewriting is usually performed by query optimizers because the execution plan of the DBMS for the query in the rewritten form is better in terms of execution costs than for the original query. In the past, query rewriting has been widely used in relational databases, in data warehouses and in statistical database systems [3, 7, 6, 11, 25, 17]. In these works, query rewriting of a query computing aggregate functions is performed in terms of union of other queries whose results have already been materialized. In particular, [25] suggests that the choice of the materializations (the summaries) should be made according to the user frequent requests. We also make this assumption for the choice of the data mining materializations. Interestingly, [6] introduces iceberg queries and observes that a query which searches for itemsets is an example of this kind of queries. Iceberg queries are generally very expensive to compute since they require several scans of huge relations. As a consequence, in order to speed up the execution time, it makes sense to try to factorize the effort already done by the DBMS. [13, 11] search for the conditions under which query rewriting of queries with aggregate functions is possible. In [16] the problem of recognizing equivalent queries in multidimensional databases has been addressed.

1.1 Storage and Exploitation of the Results of Other Queries

We imagine that we can store the result sets of some query in the database. We do this because our aim is to reduce as much as possible the computational times of the data mining engine since, nowadays, the storage space is critic to a lesser extent.

Furthermore, we suppose to work in an environment similar to a data warehouse, in which database content updates occur rarely and in known periods of time. Thus, previous results are considered up to date and can be usefully exploited to speed up the execution of current queries. Suppose, for instance, that the optimizer recognizes that the current query is equivalent to a previous one whose result is available in the database. This allows the system to completely avoid huge computational effort: the exploration of the lattice search space of the frequent itemsets, and several scans of the database which are needed to