A Comparison between Query Languages for the Extraction of Association Rules

Marco Botta[1], Jean-Francois Boulicaut[2], Cyrille Masson[2], and Rosa Meo[1]

[1] Universitá di Torino, Dipartimento di Informatica,
corso Svizzera 185, 10149, Torino, Italy
[2] Institut National des Sciences Appliquées de Lyon,
69621 Villeurbanne cedex, France

Abstract. Recently inductive databases (IDBs) have been proposed to afford the problem of knowledge discovery from huge databases. With an IDB the user/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 pattern post-processing. In this paper we present a comparison between query languages (MSQL, DMQL and MINE RULE) that have been proposed for association rules extraction in the last years and discuss their common features and differences. We present them using a set of examples, taken from the real practice of data mining. This allows us to define the language design guidelines, with particular attention to the open issues on IDBs.

1 Introduction

Knowledge Discovery in Databases (KDD) is a complex process which involves many steps that must be done sequentially. When considering the whole KDD process, the proposed approaches and querying tools are still unsatisfactory. The relation among the various proposals is also sometimes unclear because, at the moment, a general understanding of the fundamental primitives and principles that are necessary to support the search of knowledge in databases is still lacking.

In the cInQ project[1] we want to develop a new generation of databases, called "*inductive databases*", suggested in [2]. This kind of databases integrates *raw data* with *knowledge* extracted from *raw data*, materialized under the form of patterns into a common framework that supports the KDD process. In this way, the KDD 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. A few query languages can be considered as candidates for inductive databases. Most of the proposals emphasize one of the different phases of the KDD process. This paper is a critical evaluation of three proposals in the light of the IDBs' requirements: `MSQL` [3,4], `DMQL` [6,7] and `MINE RULE` [8,9].

The paper is organized as follows. Section 2 summarizes the desired properties of a language for mining inside an inductive database. Section 3 introduces

[1] Project (IST 2000-26469) partially funded by the EC IST Programme - FET.

Y. Kambayashi, W. Winiwarter, M. Arikawa (Eds.): DaWaK 2002, LNCS 2454, pp. 1–10, 2002.
© Springer-Verlag Berlin Heidelberg 2002

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. Finally Section 5 draws some conclusions.

2 Desired Properties of a Data Mining Language

A *query language* for IDBs, 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 (e.g., via standard queries but also by means of sampling), to manipulate and to query data and views in the database. It must also provide support for multi-dimensional data manipulation.
  `DMQL` and `MINE RULE` allow the selection of data. None of them has primitives for sampling. All of them allow multi-dimensional data manipulation (because this is inherent to SQL).
- The specification of the type of patterns to be mined. Clearly, real-life KDD processes need for different kinds of patterns like various types of descriptive rules, clusters or predictive models.
  `DMQL` considers different patterns beyond association rules.
- The specification of the needed background knowledge (e.g., the definition of a concept hierarchy).
  Even though both `MINE RULE` and `MSQL` can treat hierarchies if the relationship `'is-a'` is represented in a companion relation, `DMQL` allows its explicit definition and use during the pattern extraction.
- 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 (e.g., using measures like frequency, generality, coverage, similarity, etc).
  All the three languages allow the specification of various kinds of constraints based on rule elements, rule cardinality and aggregate values. They allow the specification of support and confidence measures. `DMQL` allows some other measures like novelty.
- The satisfaction of the *closure property* (by storing the results in the database).
  All the three languages satisfy this property.
- 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.
  `MSQL` is richer than the other two languages in its offer of few post-processing primitives (it has a dedicated operator, `SelectRules`). `DMQL` allows some visualization options. However, the three languages are quite poor for rule post-processing.

3 Query Languages for Rule Mining

MSQL has been described in [4,5], and designed at the Rutgers University, New Jersey, USA. The main features of `MSQL`, as stated by the authors, are the