Querying Databases with Taxonomies

Davide Martinenghi\textsuperscript{1} and Riccardo Torlone\textsuperscript{2}

\textsuperscript{1} Dip. di Elettronica e Informazione
Politecnico di Milano, Italy
martinen@elet.polimi.it

\textsuperscript{2} Dip. di Informatica e Automazione
Università Roma Tre, Italy
torlone@dia.uniroma3.it

Abstract. Traditional information search in which queries are posed against a known and rigid schema over a structured database is shifting towards a Web scenario in which exposed schemas are vague or absent and data comes from heterogeneous sources. In this framework, query answering cannot be precise and needs to be relaxed, with the goal of matching user requests with accessible data. In this paper, we propose a logical model and an abstract query language as a foundation for querying data sets with vague schemas. Our approach takes advantages of the availability of taxonomies, that is, simple classifications of terms arranged in a hierarchical structure. The model is a natural extension of the relational model in which data domains are organized in hierarchies, according to different levels of generalization. The query language is a conservative extension of relational algebra where special operators allow the specification of relaxed queries over vaguely structured information. We study equivalence and rewriting properties of the query language that can be used for query optimization.

1 Introduction

There are today many application scenarios in which user queries do not match the structure and the content of data repositories, given the nature of the application domain or just because the schema is not available. This happens for instance in location-based search (find an opera concert in Paris next summer), multifaceted product search (find a cheap blu-ray player with an adequate user rating), multi-domain search (find a database conference held in a seaside location), and social search (find the objects that my friends like). In these situations, the query is usually relaxed to accommodate user’s needs, and query answering relies on finding the best matching between the request and the available data.

In spite of this trend towards “schema-agnostic” applications, the support of current database technology for query relaxation is quite limited. The only examples are in the context of semi-structured information, in which schemas and values are varied and/or missing \cite{1}. Conversely, the above mentioned applications can greatly benefit from applying traditional relational database technology enhanced with a comprehensive support for the management of query relaxation.
To this aim, we propose in this paper a logical data model and an abstract query language supporting query relaxation over relational data. Our approach relies on the availability of taxonomies, that is, simple ontologies in which terms used in schemas and data are arranged in a hierarchical structure according to a generalization-specialization relationship. The data model is a natural extension of the relational model in which data domains are organized in hierarchies, according to different levels of detail: this guarantees a smooth implementation of the approach with current database technology. In this model data and metadata can be expressed at different levels of detail. This is made possible by a partial order relationship defined both at the schema and at the instance level.

The query language is called Taxonomy-based Relational Algebra (TRA) and is a conservative extension of relational algebra. TRA includes two special operators that extend the capabilities of standard selection and join by relating values occurring in tuples with values in the query using the taxonomy. In this way, we can formulate relaxed queries that refer to attributes and terms different from those occurring in the actual database. We also present general algebraic rules governing the operators over taxonomies and their interactions with standard relational algebra operators. The rules provide a formal foundation for query equivalence and for the algebraic optimization of queries over vague schemas.

In sum, the contributions of this paper are the following: (i) a simple but solid framework for embedding taxonomies into relational databases: the framework does not depend on a specific domain of application and makes the comparison of heterogeneous data possible and straightforward; (ii) a simple but powerful algebraic language for supporting query relaxation: the query language makes it possible to formulate complex searches over vague schemas in different application domains; (iii) the investigation of the relationships between the query language operators and the identification of a number of equivalence rules: the rules provide a formal foundation for the algebraic optimization of relaxed queries.

Because of space limitation, we do not address the issue of implementing the formal framework proposed in this paper and we disregard the orthogonal problem of taxonomy design. Both issues will be addressed in forthcoming works.

2 A Data Model with Taxonomies

2.1 Partial Orders and Lattices

A (weak) partial order \( \leq \) on a domain \( V \) is a subset of \( V \times V \) whose elements are denoted by \( v_1 \leq v_2 \) that is: reflexive (\( v \leq v \) for all \( v \in V \)), antisymmetric (if \( v_1 \leq v_2 \) and \( v_2 \leq v_1 \) then \( v_1 = v_2 \)), and transitive (if \( v_1 \leq v_2 \) and \( v_2 \leq v_3 \) then \( v_1 \leq v_3 \)). If \( v_1 \leq v_2 \) we say that \( v_1 \) is included in \( v_2 \). A set of values \( V \) with a partial order \( \leq \) is called a poset.

A lower bound (upper bound) of two elements \( v_1 \) and \( v_2 \) in a poset \( (V, \leq) \) is an element \( b \in V \) such that \( b \leq v_1 \) and \( b \leq v_2 \) \( (v_1 \leq b \) and \( v_2 \leq b) \). A maximal lower bound (minimal upper bound) is a lower bound (upper bound) \( b \) of two elements \( v_1 \) and \( v_2 \) in a poset \( (V, \leq) \) such that there is no lower bound (upper bound) \( b' \) of \( v_1 \) and \( v_2 \) such that \( b' \leq b \) \( (b \leq b') \).