

Information Retrieval from Relational Databases using Semantic Queries

Anand Ranganathan
IBM T.J. Watson Research Center, NY 10601
arangana@us.ibm.com

Zhen Liu
IBM T.J. Watson Research Center, NY 10601
zhenl@us.ibm.com

ABSTRACT

Relational databases are widely used today as a mechanism for providing access to structured data. They, however, are not suitable for typical information finding tasks of end users. There is often a semantic gap between the queries users want to express and the queries that can be answered by the database. In this paper, we propose a system that bridges this semantic gap using domain knowledge contained in ontologies. Our system extends relational databases with the ability to answer semantic queries that are represented in SPARQL, an emerging Semantic Web query language. Users express their queries in SPARQL, based on a semantic model of the data, and they get back semantically relevant results. We define different categories of results that are semantically relevant to the users' query and show how our system retrieves these results. We evaluate the performance of our system on sample relational databases, using a combination of standard and custom ontologies.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – *retrieval models, search process, query formulation.*

General Terms

Algorithms, Design

Keywords

Semantic Queries, Ontologies, Semantic Relevance, Information Retrieval, Relational Databases, Knowledge Management

1. INTRODUCTION

Relational databases are widely used today as a mechanism for providing access to structured data. However, there are a number of limitations of such databases that make them unsuitable for typical information finding tasks of end-users. There is often a mismatch between the user's intended query and the database's schema. For example, assume there is a database containing information about disaster aid workers located in different parts of the United States. A user may have a query about which aid workers are located in the Gulf Coast region of the United States;

however the database tables may only have information about which towns and states aid workers are located in. Thus, although the database may contain the desired information, the user cannot get an answer to his query since the database does not know which towns and states are located in the Gulf Coast. Thus, the user is forced to re-frame his query based on the actual schema and contents of the database.

The Semantic Web envisions a world where loosely coupled, independently evolving ontologies provide a common understanding of terms between heterogeneous agents, systems, and organizations. In the past few years, different ontologies have been developed in various domains to capture relevant knowledge. In this paper, we present a system that uses both the data in a relational database and domain knowledge in ontologies to return semantically relevant results to a user's query. Users specify their queries using SPARQL [1], a popular Semantic Web language for querying RDF graphs. Our system takes as input a SPARQL query and uses a Description Logic (DL) reasoner to help infer semantically relevant results – this allows returning more results than what is explicitly mentioned in the databases. Our system returns three kinds of semantically relevant results: direct results, inferred results and related results.

Depending on the specific relational database and the applications it is used in, different sets of ontologies may be used by our system to help answer semantic queries. In order to support the creation of new ontologies and the reuse of existing ontologies, our system provides an ontology engineering framework. The framework is used by application developers to pull together an appropriate set of ontologies that can provide the required semantic information about terms used in the database. They may reuse existing domain ontologies or define new ones to capture application-specific semantic information. The framework also helps them specify mappers that can convert data from the relational model defined in the database to the semantic model defined in the ontologies.

2. SEMANTIC QUERIES

We use the term "semantic query" to refer to database queries that are based on concepts, properties and instances defined in an ontology and that return semantically relevant results. Such queries are based on the semantic model of the data expressed in the ontologies rather than the E-R model of the data present in a relational database.

The result of a semantic query is semantically relevant results. In order to return semantically relevant results, we first have to define which results qualify as semantically relevant. We define 3 types of semantically relevant results based on how these results are obtained and their relationship to the semantic query:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM'06, November 5–11, 2006, Arlington, Virginia, USA.
Copyright 2006 ACM 1-59593-433-2/06/0011...\$5.00.

2.1.1 Direct Results

Direct results are obtained directly from the database tables; i.e. they consist of only results that are explicitly listed in the database tables. For instance, let us again consider the database containing information about disaster aid workers located in different parts of the United States. The direct results of a semantic query for aid workers in New Orleans contain only those aid workers who are explicitly listed as being located in New Orleans in the database. Most standard database management systems and query languages support the return of direct results.

2.1.2 Inferred Results

Inferred results are those that can be inferred using the information that is explicitly listed in the database and the domain knowledge in the ontologies. The inference is done using Description Logic reasoning. For example, the inferred results of a semantic query for aid workers in the Gulf Coast contains all aid workers in Louisiana, Mississippi, Alabama and other states that form part of the Gulf Coast. Thus, although, the database itself had no information about "Gulf Coast", the semantic query can return relevant results. In this case, the inference is made possible with the help of location ontologies that describe which regions a state is located in.

2.1.3 Related Results

Related results are obtained using data in the database tables and a definition of similarity of concepts and individuals based on the data model in the ontologies. It includes results that do not strictly match the user's query, but may still be similar to the actual answers and hence, may also be semantically relevant. For example, the related results of a query for aid workers in New Orleans may also return results from other towns in the state of Louisiana close to New Orleans. Such an expanded result set is especially useful if there were no or very few aid workers in New Orleans itself.

Our system uses various strategies to generate more general queries that can produce results that are similar to those produced by the original query. These strategies make use of different kinds of information in the ontologies, including the concept hierarchy, transitive properties defined between different instances and membership of instances in a concept. The query generalization algorithm works by repeatedly applying these strategies to generate more and more general queries until a certain pre-specified number of results are obtained.

3. ARCHITECTURE

We have developed a system that allows posing semantic queries in SPARQL on a relational database. The reasoning is done using Minerva[2], which is a scalable, efficient RDBMS-based DL reasoner. A key feature of Minerva is that it precomputes all inferences and stores the basic asserted facts as well as the inferred facts in a backend relational database. Hence, queries to Minerva are fast since no description-logic reasoning is done at query time. In DLs, there are two kinds of facts: "TBox" (which has sentences describing concepts) and "ABox" (which has sentences describing instances). Minerva allows efficient assertion and retraction of ABox facts; however, it cannot efficiently handle changes to the TBox. Any such change requires a recomputation of all inferences and can take a few minutes for large ontologies.

Fig 1 shows the overall architecture of our system. It contains two Minerva reasoning systems (a primary and a standby) along with the databases they use for storing the asserted and inferred facts. It also contains the original relational database storing the data of interest. There are ontologies that describe the semantic model of the data as well as provide additional domain knowledge.

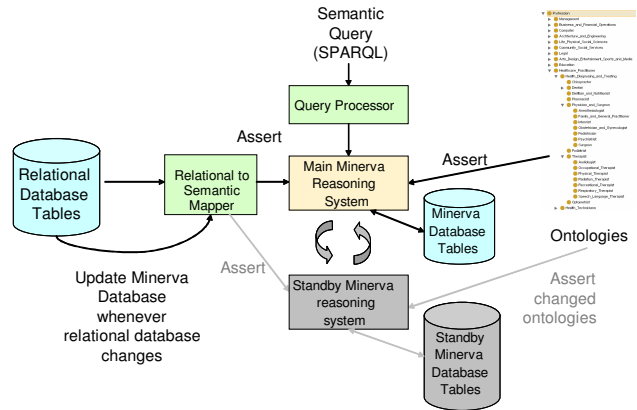


Figure 1. Architecture of the semantic query system

At the beginning, all the relevant ontologies are loaded into the primary Minerva system. Next, all the entries in the original relational database tables are asserted into the primary Minerva system through the Mapper, which translates the rows in the database into DL assertions that can be processed by Minerva. The system also ensures that the Minerva system is kept up-to-date with the relational database. It creates triggers on the original database, so that any changes in the tables result in a notification to the Mapper, which then updates the Minerva system.

The Query Processor takes in queries in SPARQL and poses them to the Minerva system to get direct and inferred results. It also does query generalization and obtains a certain pre-specified number of related results.

In order to prevent the system from being unresponsive to queries when the TBox changes, we make use of a standby Minerva system for performing the recomputation of inferences. The standby system is initialized with the facts in the ontologies and the database. Once the standby Minerva system is initialized, the two Minerva systems are switched. The standby system now becomes the primary one and answers all semantic queries.

We tested the performance of the system and found that the time for initializing the system and for answering semantic queries grows linearly with the size of the relational database. The tests were performed using a sample disaster relief ontology containing about 400 concepts, 120 properties and 500 individuals. The results give a good indication of the scalability of the system. We are investigating ways of reducing the time for answering queries by using better indices in the Minerva database, as well as using other strategies for obtaining related results.

4. REFERENCES

- [1] Prudhommeaux, E., and Seaborne, A. *SPARQL query language for RDF*. Working draft, W3C (2005)
- [2] Ma, L. et al. 2006. IBM Integrated Ontology Development Toolkit. <http://www.alphaworks.ibm.com/tech/semanticstck>