Chapter 1
Incremental Query Rewriting with Resolution

Alexandre Riazanov and Marcelo A. T. Aragão

Abstract  We address the problem of semantic querying of relational databases (RDB) modulo knowledge bases using very expressive knowledge representation formalisms, such as full first-order logic or its various fragments. We propose to use a resolution-based first-order logic (FOL) reasoner for computing schematic answers to deductive queries, with the subsequent translation of these schematic answers to SQL queries which are evaluated using a conventional relational DBMS. We call our method incremental query rewriting, because an original semantic query is rewritten into a (potentially infinite) series of SQL queries. In this chapter, we outline the main idea of our technique – using abstractions of databases and constrained clauses for deriving schematic answers, and provide completeness and soundness proofs to justify the applicability of this technique to the case of resolution for FOL without equality. The proposed method can be directly used with regular RDBs, including legacy databases. Moreover, we propose it as a potential basis for an efficient Web-scale semantic search technology.

1.1 Introduction.

1.1.1 Settings and motivation.

Consider the following scenario. Suppose we have a relational database (RDB) and some expressive knowledge bases (KB) for domains to which the data in the RDB is related, e.g., rule bases in expressive sublanguages of RuleML [8, 2] and/or ontologies in OWL. We would like to work with arbitrary (reasonably well designed)
RDBs, and, consequently, the database relations are not assumed to directly correspond to concepts and relations described by the KBs. So, optionally, we may also have some mapping between the RDB schema and the logical language of the domains, i.e., a logical description of the relations in the RDB, to link them to the concepts and relations defined by the KBs. In these settings, we would like to be able to formulate queries logically, and answer them w.r.t. the KBs and the RDB treated virtually as a collection of ground atomic facts, e.g., by viewing each table row as a separate ground fact. To make this process efficient, we would like to use the modern RDB technology as much as possible by delegating as much work as possible to the RDBMS hosting the database.

We propose a novel method to implement this scenario, based on the use of resolution for incremental transformation of semantic queries into sequences of SQL queries that can be directly evaluated on the RDB, and whose results provide answers to the original queries.

We envisage three main applications for the proposed technology.

Enhancing the interface to conventional relational databases  Ad hoc self-service querying of conventional RDBs by non-programmer users is very problematic because real-life enterprise databases often have complex designs. Writing correct queries requires good understanding of technical details of the DB schema, such as table and attribute names, foreign key relationships, nullable fields, etc., not to mention the knowledge of query languages like SQL. So most of RDB querying by non-programmer users is done with preprogrammed parameterised queries, usually represented as forms of various kinds.

Even when special methodologies are used, like Query-by-Example (see, e.g., [26]), that hide some of the complexities of SQL and database designs from the end users, one important inherent limitation remains in force. Whereas mapping some domain concepts to the RDB schema elements may be easy, many other concepts may be much more difficult to map. For example, it is easy to select instances of the concept “student” if there is a table explicitly storing all students, but if the user wants to extract a list of all members of a department in a university, he may have to separately query different tables storing information about students, faculty and support staff (assuming that there is no table specifically storing members of all these kinds), and then create a union of the results.

This example exposes well the root of the problem: mapping some domain concepts to the data is difficult because it requires application of the domain knowledge. In the example, the involved piece of domain knowledge is the fact that students, faculty and support staff are all department members, and the user has to apply it manually to obtain the required results.

Semantic querying is based on automatic application of domain knowledge written in the form of, e.g., rules and ontological axioms. In this approach, DB programmers “semantically document” their DB designs by providing an explicit mapping between the RDB schemas and domain terminologies, e.g., in the form of logical axioms. This alone allows an end user to formulate queries directly in the terminology of the domain, without even a slightest idea about how the underlying RDBs