Semantic-based query processing for relational data integration

Miao Zhuang Zhang Yafei Wang Jinpeng Lu Jianjiang Zhou Bo

(Institute of Command Automation, PLA University of Science and Technology, Nanjing 210007, China)

Abstract: To solve the query processing correctness problem for semantic-based relational data integration, the semantics of SAPRQL (simple protocol and RDF query language) queries is defined. In the course of query rewriting, all relative tables are found and decomposed into minimal connectable units. Minimal connectable units are joined according to semantic queries to produce the semantically correct query plans. Algorithms for query rewriting and transforming are presented. Computational complexity of the algorithms is discussed. Under the worst case, the query decomposing algorithm can be finished in $O(n^2)$ time and the query rewriting algorithm requires $O(n^m)$ time. And the performance of the algorithms is verified by experiments, and experimental results show that when the length of query is less than 8, the query processing algorithms can provide satisfactory performance.

Key words: data integration; relational database; simple protocol and RDF query language (SPARQL); minimal connectable unit; query processing

doi: 10.3969/j.issn.1003 – 7985.2011.01.005

With the rapid development and extensive applications of information technology, the explosive growth of data resources is a new, disturbing trend[1]. Considering the vast amounts of data stored in heterogeneous relational databases, it is difficult to achieve interoperability among them[2]. Integrating and querying data from heterogeneous sources is a hot research topic in the field of databases[3]. The goal of data integration is to provide users uniform access to multiple heterogeneous data sources[4]. Due to the lack of semantic description capabilities, traditional data integration systems cannot resolve the semantic heterogeneity of data.

The semantic Web has provided several new approaches for data integration. One of the most important applications is the ontology-based integration of heterogeneous relational databases, which has recently received special attention in the database community[5-6]. However, one of the primary obstacles is how to convert a global query into correct SQL query plans for accessing the relational databases. Systems like Virtuoso[7] rewrite SPARQL[8] queries to SQL. The MySQL SPASQL[9] module compiles SPARQL queries directly into an evaluation structure to be executed by a relational engine. However, these works only focus on accessing conventional relational databases using SPARQL, which do not take the integration into account. A few early works on integration use ontology for relational databases and construct ontologies from relational databases schema. This approach cannot fully capture the semantics behind the relational schema and only reflects the structure of databases, which makes it hard to integrate heterogeneous relational databases.

This paper focuses on the query processing problem for ontology-based relational data integration. In our proposal, RDF (resource description framework) ontology is used as a mediated schema for the explicit description of the data source semantics, providing a shared vocabulary for the specification of the semantics. The semantic of query rewriting is further discussed and a query rewriting algorithm is presented to reformulate an SPARQL query into SQL queries, so that SPARQL can access heterogeneous relational databases without converting the data into physical triples.

1 Architecture of Semantic-Based Query Processing

The architecture of semantic-based query processing for relational data consists of five layers (see Fig. 1). The application layer provides a unified portal based on global schema for users and other applications. The semantic layer is responsible for the establishment and maintenance of global schema, storage of the semantic descriptions of heterogeneous relational databases, and providing clear definitions of concepts. The mediator layer mainly consists of three parts: query decomposition, query rewriting, and query transformation. Query decomposition is in charge of the semantic

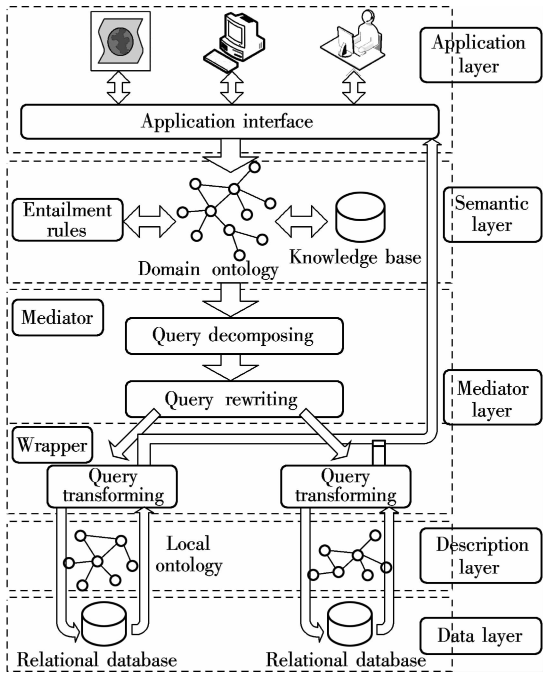

Fig. 1 Architecture of semantic-based query processing for relational data

Received 2010-07-08.

Biography: Miao Zhuang (1976—), male, doctor, associate professor, emiao_beyond@163.com.

Foundation items: Weaponry Equipment Pre-Research Foundation of PLA Equipment Ministry (No. 9140A06050409JB8102), Pre-Research Foundation of PLA University of Science and Technology (No. 2009JSJ11).

Citation: Miao Zhuang, Zhang Yafei, Wang Jinpeng, et al. Semantic-based query processing for relational data integration[J]. Journal of Southeast University (English Edition), 2011, 27(1): 22 – 25. [doi: 10.3969/j.issn.1003 – 7985.2011.01.005]

analysis, standardization and reorganization of the global query; query rewriting is in charge of rewriting the global query into sub-queries over the underlying databases; the query transformation converts SPARQL queries over global schema into local SQL queries. The description layer describes the semantics of heterogeneous relational databases using global schema. The data layer consists of several heterogeneous relational databases. Our work focuses on the query processing from the semantic layer to the description layer and mining the semantics of relational data. The architecture is shown in Fig. 1.

2 Semantics of SPARQL Queries

RDF describes things by making statements about an entity's properties. Thus, the term RDF graph is defined as a set of RDF triples. In this paper, we propose query processing algorithms over an RDF graph model. The SPARQL query language is based on matching graph patterns. Pérez et al. introduced the formal semantics of SPARQL in Ref. [10] and we use SAPRQL as the query language.

Definition 1 (SPARQL query) A SPARQL query Q = SELECT vars WHERE gp, where vars = $\{v_1, v_2, \ldots, v_n\}$, lists all the return variables in Q, gp is a set of RDF triples that Q should satisfy. Let var(gp) be all variables in gp, obviously vars $\subseteq$ var(gp).

Definition 2 (SPARQL query result) The query result of an SPARQL query Q = SELECT vars WHERE gp above an RDF graph G is a finite set of mappings $\Omega_Q = (\mu_1, \mu_2, \ldots, \mu_n)$, where μ_i maps v_i to an RDF term in G.

Definition 3 (query containment) Given two SPARQL queries Q_1 and Q_2, $\forall \mu_i \in \Omega_{Q_2}$, $\exists \mu_j \in \Omega_{Q_1}$, where $\mu_i(\text{vars}(Q_2)) \subseteq \mu_j(\text{vars}(Q_1))$ and $\mu_j(\text{gp}_{Q_1}) \subseteq \mu_i(\text{gp}_{Q_2})$, then Q_1 contains Q_2, denoted by $Q_1 \subseteq Q_2$ or $Q_2 \supseteq Q_1$.

Definition 4 (query equivalent) Given two SPARQL queries Q_1 and Q_2, if $Q_1 \subseteq Q_2$ and $Q_2 \subseteq Q_1$, then Q_1 is equivalent to Q_2, denoted by $Q_1 = Q_2$.

Definition 5 (SPARQL containment mapping) Given two SPARQL queries Q_1 and Q_2, mapping τ from vars(Q_1) to vars(Q_2) is a containment mapping, if the mapping meets the following conditions:

1) If $\tau(x) \in \text{vars}(Q_2)$, then $x \in \text{vars}(Q_1)$, namely, vars(Q_2) $\subseteq$ vars(Q_1);

2) $\tau(\text{gp}_{Q_1}) \subseteq \text{gp}_{Q_2}$.

Lemma 1 Given two SPARQL queries Q_1 and Q_2, $Q_1 \supseteq Q_2 \Leftrightarrow$ there is a containment mapping from Q_1 to Q_2.

3 Query Processing Algorithms

In a semantic-based heterogeneous data integration system, users are faced with a global model, and put forward queries based on it. The integration system decomposes global queries into sub-queries according to their content, rewrites the sub-queries to be consistent with the semantic of underlying relational databases, and then transforms the rewriting result to SQL queries, so that the queries can be executed on relational databases. Query processing includes query decomposing, rewriting and transforming.

3.1 Query decomposing

According to the semantics of SPARQL queries, a complicated graph pattern is composed of base graph patterns joined with operators and variables, and the base graph pattern is the minimum unit of the query graph pattern. Query decomposing is to decompose complicated graph patterns into base ones. The main idea of query decomposing is as follows. If the query graph pattern is the basic graph pattern then the query is returned; and if the query graph pattern is the group graph pattern, the alternative graph pattern or the optional graph pattern, then sub-queries are constructed. The method of constructing sub-queries is to combine queries with AND, UNION, OPTION and recursively call the query decomposing course until the query cannot be decomposed any further.

3.2 Query rewriting

A semantically correct query plan amounts to finding proper relational tables whose sub-graph covers the SPARQL query. We present a query rewriting algorithm to generate semantically correct and executable query plans under the RDF graph model. Our algorithm proceeds in two stages. In the first stage, we find all the relative tables and decompose them to minimal connectable units (MCU) according to source descriptions. In the second stage, we join MCUs to produce the query plans.

3.2.1 Generating the MCUs

An MCU is a subset of RDF triples in source description of a relational table that can be joined with other MCUs and executed on heterogeneous databases. Intuitively, an MCU represents a fragment of semantic mapping from the query to the rewriting of the query.

Formally, we define MCUs as follows.

Definition 6 (minimal connectable units) Given a SPARQL query Q, a relational table P and its table subgraph G_P, an MCU m for P is a tuple of the form (Y, μ') where Y is a subset of triple patterns in G_P and μ' is a partial mapping from variables appearing in G_P to corresponding columns in P.

Then the algorithm of finding the MCUs is as follows:

```
find MCUs($Q$, $G_P$)
Input: SPARQL query $Q$.
Output: $G_P$.
Initialize $M = \Phi$;
For each triple pattern $t$ in $Q$, do
  for each triple pattern $t'$ in $G_P$, do
    If there exists a mapping $\tau$ that map $t$ to $t'$, then
      find the minimal subset (denoted by $Y$) of triple patterns of $G_P$ that is connectable.
      find the subset (denoted by $\mu'$) of $\mu$ that is relative to $Y$.
    $M = M \cup \langle Y, \mu' \rangle$;
  end for
end for
return $M$
```

The algorithm for creating the MCDs is shown in Fig. 2. We say a set of triple patterns Y is connectable if the following conditions hold: 1) Each return variable of Q is also a return variable in Y. 2) If x is not a return variable in Q, then for every triple pattern t that includes x, $t \in Y$.

3.2.2 Joining the MCUs

In this phase we consider combinations of MCUs, and for each valid combination we create a conjunctive rewriting of the query. The final result is a union of conjunctive queries. Given a set of MCUs M, the actual rewriting is constructed as follows:

```
joinMCUs(Q, M, A)
Input: an SPARQL query $Q$, a set of MCUs $M$, $M = \{m_1, m_2, \cdots, m_n\}$, where $m_i = (Y_i, \mu_i')$.
Output: a set of rewritings $A$.
Initialize $A = \Phi$;
For each minimal subset $\{m_1, m_2, \cdots, m_k\}$ of $M$ such that
    $Y_1 \cup Y_2 \cup \cdots \cup Y_k$ cover all triple patterns of $Q$.
    Create the conjunctive rewriting $Q'$ containing all the relative tables to $\{m_1, m_2, \cdots, m_k\}$
    Add $Q'$ to $A$
end for
return $A$
```

3.3 Query transforming

After rewriting, the sub-queries are consistent with the semantics of underlying relational databases. But they still cannot execute on relational databases. The rewriting results should be transformed into SQL queries for accessing the relational databases. The query transforming algorithm is shown as follows:

```
queryTransformation(gp)
Input: SPARQL query Q = SELECT vars WHERE gp.
Output: SQL query plans.
For each variable $v_i$ in vars
    add $v_i$ in varlist
    SQL = "SELECT DISTINCT" + varlist + "FROM" + graphPatternTranslation(gp)
return SQL
```

Here, graphPatternTranslation(gp) is the graph pattern translating algorithm, which translates the given graph pattern gp into SQL queries and the detail of the algorithm is omitted.

4 Evaluation

The proposed algorithms in this paper are fully implemented on a Pentium Dual 1.79 GHz computer with Java 1.6. We construct 10 different relational databases, containing 100 heterogeneous relational tables. We also construct 100 different global queries with length from 1 to 10 to simulate various user queries.

4.1 Computational complexity

In the query decomposing algorithm, given an SPARQL query, if it contains some nested operations, the query decomposing algorithm will be called repeatedly. Let n stand for the number of triples in a query pattern; and $k(1 < k < n-1)$ stands for the number of nested operations in the query. Under the worst case, the query decomposing algorithm can be finished in $O(n^2)$ time. Its time complexity is polynomial.

As for the query rewriting algorithm, given a set of data source $V = \{v_1, v_2, \cdots, v_n\}$ and query Q = SELECT vars WHERE gp, $|\text{gp}| = m$, the query rewriting algorithm needs to find all the possible subsets of V that can form an MCU, which requires $O(n^m)$ time. However, as the first step of query rewriting already rules out the nonrelevant data sources, significantly cutting the unnecessary search branches, it can be expected that the actual operating efficiency of the query rewriting algorithm will be more satisfactory.

4.2 Experimental results

We conducted experiments to verify the performance of the proposed query processing algorithms and the results are shown in Fig. 2. Fig. 2(a) shows the performance of query processing scalable in terms of the length of queries and the runtime is almost exponential to the length of queries. An interesting finding is that the runtime is not sensitive to the length of queries when the number of data sources is less than 50. Fig. 2(b) shows the performance of query processing scalable in terms of the number of data sources and the runtime is almost exponential to the number of data sources. But the runtime is not sensitive to the length of queries when the length of queries is less than 8.

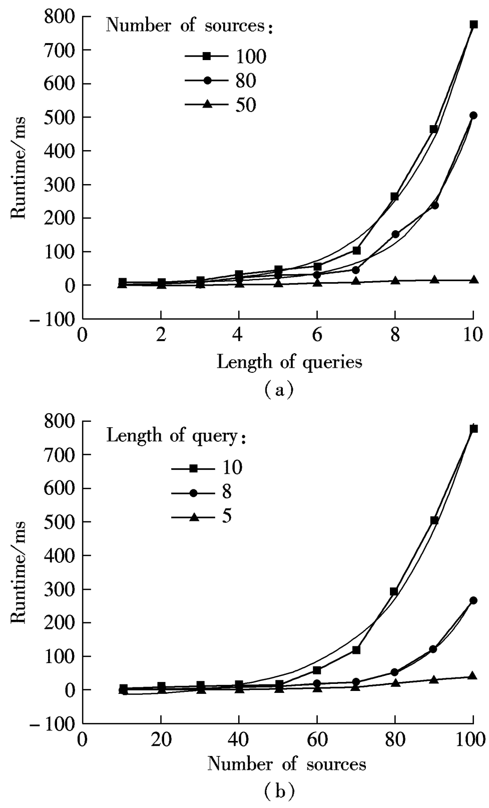

Fig. 2 Performance test of query processing

4.3 Discussion

The experimental results are consistent with the analysis of time complexity. When the length of queries is less than 8, but there are lot of data sources (more than 80), the query processing algorithms can provide nice performance (within 500 ms). This is also consistent with the practical circumstances in integration systems, where the query length is usually short.

5 Conclusion and Future Work

In this paper we consider the query processing problem in heterogeneous data integration systems. We use ontology as the mediated schema for integration. The semantics of query rewriting is defined. Algorithms for query rewriting and transforming are presented, so that SPARQL queries over global schema can be rewritten into local SQL queries that can be executed on heterogeneous relational databases. The approach and algorithms provided in this paper can be extended for other types of data sources, but some of the complex semantics (i. e. CONSTURCT, ASK) are not discussed. We leave these problems for future work.

References

[1] Gantz J. The diverse and exploding digital universe[R]. Framingham, MA, USA: International Data Corporation, 2008.
[2] Bell G, Hey T, Szalay A. Beyond the data deluge[J]. *Science*, 2009, **323**(5919): 1297－1298.
[3] Halevy Y A, Rajaraman A, Ordille J J. Data integration: the teenage years[C]//*Proc of the 32nd International Conference on Very Large Data Bases*. Seoul, Korea, 2006: 9－16.
[4] Bernstein A P, Hass M L. Information integration in the enterprise[J]. *Communications of the ACM*, 2008, **51**(9): 72－79.
[5] Dou D, LePendu P. Ontology-based integration for relational databases [C]//*Proc of ACM Symposium on Applied Computing*. Dijon, France, 2006: 461－466.
[6] Zhu H, Madnick S. A lightweight ontology approach to scalable interoperability[C]//*Very Large Data Bases Workshop on Ontology-Based Techniques or DataBases and Information Systems*. Seoul, Korea, 2006: 45－54.
[7] Erling O, Mikhailov I. RDF support in the virtuoso DBMS [C]//*Proc of the 1st Conference on Social Semantic Web*. Leipzig, Germany, 2007: 59－68.
[8] Hommeaux E, Seaborne A. SPARQL query language for RDF[EB/OL]. (2008-01)[2010-01-10]. http://www.w3.org/TR/rdf-sparql-query/.
[9] Prudhommeaux E. SPASQL: SPARQL support in MySQL [EB/OL]. (2006)[2009-12-20]. http://xtech06.usefulinc.com/schedule/paper/156.
[10] Pérez J, Arenas M, Gutierrez C. Semantics and complexity of SPARQL[C]//*Proc of the 5th International Semantic Web Conference*. Athens, USA, 2006: 30－43.

关系数据集成中的语义查询处理技术

苗　壮　张亚非　王进鹏　陆建江　周　波

(解放军理工大学指挥自动化学院,南京 210007)

摘要:为解决基于语义的关系数据集成中的查询处理正确性问题,形式化定义了 SPARQL 查询语句的语义. 在查询重写过程中,发现查询相关的数据表并将其分解为最小可连接单元,再根据查询语义连接最小可连接单元来产生正确的查询. 给出了基于语义的查询重写和查询转换算法. 对算法复杂性进行了讨论,在最坏情况下,查询分解算法可在 $O(n^2)$ 时间内完成,查询重写的时间复杂度为 $O(n^m)$. 通过实验验证了算法的性能,实验结果表明当查询长度小于 8,而数据源较多时,查询处理算法具有较好的效果.

关键词:数据集成;关系数据库;SPARQL;最小可连接单元;查询处理
中图分类号:TP182