ISSN:0975-9646
Vipin Kumar.N et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 (4) , 2011, 1706-1710

A Comprehensive Comparative study of SPARQL and SQL

Vipin Kumar.N[#], Archana P. Kumar[*], Kumar Abhishek[*]
[#]*Education and Research Division, Infosys Technologies, Mangalore*
[*]*Department of Computer Science and Engineering, MIT Manipal-576104*

Abstract: **With development of Semantic Web, much research focuses on various technologies about Web ontology. Especially, for management and searching of data in Web ontology, various storages based on RDB and query languages (e.g., SPARQL, RDQL and RQL) have been developed with activity. SPARQL cannot search data in RDB model because RDB model use SQL as query language. The Resource Description Format (RDF) is used to represent information modeled as a "graph": a set of individual objects, along with a set of connections among those objects. In that role, RDF is one of the pillars of the so-called Semantic Web. RDF Data represents a graph. Graphs are natural way to represent things and the relationships between them. RDF data stores are optimized to efficiently to recognize graph sub-patterns and there is a standard query language SPARQL that is used to query these data stores. This paper is intended to show the differences between the two query languages. i.e. SPARQL and SQL.**

Keywords: **Semantic Web, Resource Description Framework, Web ontology language, SQL, SPARQL.**

I. INTRODUCTION

As Web information increases exponentially, the current Web faces a limit to find exact information which users want. Semantic Web has been proposed as a solution to resolve the aforementioned problem[2]. Currently, much research is focused on development of various technologies about web ontology on Semantic Web.

In the Semantic Web environment, description languages for Web ontology (e.g., RDF[3], RDF-S[4], and OWL[5]) and query languages for Web ontology (e.g., RQL[6], RDQL[7], and SPARQL[8]) have been proposed. SPARQL recommended by W3C is used as the most representative description language.

Query languages are typically applied to data corresponding to particular data model[9]. SQL is used to retrieve, create, modify and delete data represented in the relational model of data. Similarly, XQUERY is used to locate and retrieve data that is represented in the XPath data model. It is sometimes possible to use one language to query data represented in a data model other than that for which the language was designed. This may be accomplished by mapping the data from its native data model into the query's language data model.

RDF is presented as yet another data model, distinct from the XPath data model and from the SQL data model. It is tempting to reject that assertion because of the tuple nature of RDF entities. However, a close examination shows subtle differences between collections of RDF triples and multisets of rows in SQL tables. For example SQL tables are defined to comprise one or more columns, each having a particular data type. Every row in that table has exactly that number of columns and the value of each column in each such row must be of the column's declared type. Notably missing from the definition of SQL tables is the idea that rows in a given table contain information about the data types of any of the (other) data in that table. SQL's metadata is recorded in s number of "system tables", which could be combined in some way with the data in the table- although the criteria for such combinations to be made meaningful are unclear. By contrast , a given RDF collection can be augmented by RDF triples expressed using OWL (Web Ontology Language) constructs that specify the class to which a given RDF entity belongs.

II. SPARQL- SPARQL PROTOCOL AND QUERY LANGUAGE

SPARQL is a language that lets users query RDF graphs by specifying "templates" against which to compare graph components[10]. Data which matches or "satisfies" a template is returned from the query. A triple template will contain variables that represent triplet components (e.g., a subject, predicate, or object within a triplet). For example the template:

?person <example:age> "21"^^example:age .

identifies a list of triplet subjects that have an example:age property of "21", and is analogous to asking "Who has age 21?" The SparQL query engine will return an exhaustive list of the subject component of triples that satisfy each query through value substitution. This is basically "query by example" (QBE) where the user defines an example pattern that the query engine will attempt to match using components from the data store. This process is reasonably intuitive, and similar to QBE approaches applied to relational data and pattern matching within regular expressions or SQL. SparQL is implemented in Jena through the ARQ package, and queries may be made from within Java scripts or via a SparQL client distributed with Jena.

SPARQL is one of a number of query languages designed to query formal representations of data such as XML (eXtensible Markup Language), Topic Maps and RDF which consist of data models represented as trees, topics and associations, and directed graphs respectively[11]. Similar to other query languages, SPARQL allows users to declaratively specify the conditions required for data to be retrieved rather than explicitly describing the individual steps required to return the data. SPARQL provides definitions for:

- Simple matching of RDF data,
- The ability to combine multiple matches together,
- Matching data types such as integers, literals, etc. based on conditions such as greater than, equal to, etc.,

- Optionally matching data – that is, if certain data does exist it must meet a certain criteria but the query does not fail if the data doesn't exist,
- Combining RDF data sets together to query at the same time, and
- Ordering and limiting matched data.

Further it is highlighted that the design of an RDF query language should support :

- The RDF abstract data model,
- Formal semantics and inference,
- XML schema data types and
- The ability to handle incomplete or contradictory information.

SPARQL - query language for getting information from RDF graphs. It provides facilities to:

- Extract information in the form of URIs, blank nodes, plain and typed literals.
- extract RDF subgraphs.
- construct new RDF graphs based on information in the queried graphs

The beneficial properties of a query language for the Semantic Web include:

- Referentially transparent - "within the same scope, an expression always means the same",
- Strong answer closure - the result of a query can be used as the input for further querying,
- Set-oriented functional – also known as a backtracking-free logic programming,
- Incomplete queries and answers - support for data on the Web that may not have defined schemas,
- Multiple serialisation aware - able to serialise data to various formats including XML, OWL, RDF and Topic Maps, and
- Queries that support reasoning capabilities - the ability to query different data sources and infer new statements.

Here is an example SparQL query that simply asks for a list of up to 10 of the subject and object portions of the triples in the file specified in the FROM clause:

```
PREFIX rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX example
<http://fake.host.edu/example-schema#>
select $s $o
from
<http://myhost.edu/rdf-example-1.rdf>
where
{
$s $p $o .
}
LIMIT 10
```

$s, $p, and $o are variable names that will each be assigned a value as the query is "satisfied," and the triplet pattern "$s, $p,$o" will match any triple that has 3 parts, so all triples should be displayed. Note that variable names may also start with "?", and may be full words. A subset of the basic syntax of a SparQL select query is shown below:

```
BASE < some URI from which relative FROM and
PREFIX entries will be offset >
PREFIX prefix_abbreviation: < some_URI >
SELECT
some_variable_list
FROM
< some_RDF_source_URL >
WHERE
{
{ some_triple_pattern .
another_triple_pattern . }.
}
```

Notes:

- the "<" and ">" characters are required literals,
- the BASE and PREFIX entries are optional and BASE applies to relative URIs appearing in either PREFIX or FROM clauses,
- other commands that can appear in place of SELECT are: CONSTRUCT, ASK and DESCRIBE,
- * is a valid variable list, specifying any variable returned by the query engine, and may be preceded by DISTINCT, which will omit duplicate triples from the resulting list,
- there may be multiple FROM clauses, whose targets will be combined and treated as a single store,
- a "." separating multiple triple patterns is intuitively similar to an "and" operator,
- the term WHERE is optional, and may be omitted.

III. SQL –STRUCTURED QUERY LANGUAGE

The relational model is an existing model that could be used to provide a compatible set based, formal model. This model has long been used as the basis for database management. Date defines it as consisting of three components: structure, integrity and manipulation. It has been extended to support rules and inferencing ,support for XML schema data types and other data types, to query hierarchical data and to support merging data, potentially incomplete or contradictory information, through the use of outer joins and other techniques. The relation model supports answer closure and referential transparency (for read-only queries). The set of relational operators combined with the relational model are collectively called the relational algebra. The operations on relations originally defined by Codd include: set operations, projection, join, Cartesian product, and restriction. It is from these original operators that other relational operations have been derived including: restrict, project, join, and union.

An alternative to the relational model is SQL (Structured Query Language). SQL is often seen as an implementation of the relational model even though it has numerous incompatibilities with the relational model such as bag instead

of set semantics, column ordering, duplicates and handling of nulls. SQL has been formally reconstructed using bags (a collection of values that allow duplicates) rather than sets (a collection of values that allows no duplicates). From this work it's shown that operations such as DISTINCT and aggregate functions are only applicable for bags and not sets. SQL's use of duplicate values can also cause problems with both optimisation and query processing.

SQL also has other problems such as, "…no one really knows what SQL is, since there are many different versions, it is widely accepted that any version of SQL has at least two features which are not present in the relational algebra: aggregate operators [and it] allows a limited form of nesting by using the GROUP-BY construct…one needs bag semantics for the correct evaluation of aggregate functions." Software that depends on SQL frequently has to adapt to each vendor specific implementation due to these differences.

Similarly, SQL's UNION operator has a number of problems in that it relies on a column ordering being used to match values rather than the columns being the same type (as defined by relational algebra). Date claims, "…given any two SQL tables, there are typically many distinct tables that can all be regarded as a union between two given SQL tables".

IV. Difference between SPARQL and SQL

The distinct differences between SQL and RDF are the reason why SQL is not a natural choice as the basis of an RDF query language. It is clear than any formal SPARQL definition should abstract away any dependence from SQL and be solely based on the data model it is querying, RDF.

SQL does have a large industry following so it is crucial that a mapping from SPARQL to SQL exists. Work on this mapping has already occurred, but further work, especially using known SQL optimisation techniques, has yet to occur.

Previous work has highlighted specific limitations of the current SPARQL specification and subsequent implementations. To overcome these issues an underlying formal model should be established. However, little work has been done in developing and evaluating an RDF query language that is built on formal set-based models while maintaining a focus on SPARQL.

As both the RDF model and the relational model are both propositional and set-based it is likely that a compatible model for querying RDF can be provided. This should lead to two direct advantages for users and implementers:

- It provides a formal model that unambiguously outlines a consistent set of principles to create a coherent foundation for the formulation of queries. This provides a stable set of fundamentals that remain constant as implementations or syntaxes evolve over time.
- It allows the continuing work being done on the relational model to be applied to querying RDF.

RDF corresponds to Entity-Relationship model. SQL tables at least those with an explicit key of n columns could be decomposed into n-1 entity relationship assertions for each row, each such assertion having the form "key-value column-name column-value". In fact this observation provides a trivial method of transforming SQL tables into RDF graphs.

This syntax resembles SQL, and it has similar semantics. In particular, SQL semantics revolve around joining tables together and then looking through every row to see if the contents of row
fields meet specified conditions. If one thinks of a collection of triples containing the same predicate as a (distributed) table named by the triplet predicate and containing 2 columns, the
triplet subject and object, then the "." operator in SparQL queries is similar to a join, in which shared SparQL variables within triple patterns essentially define a join condition specifying equality. Here is a SparQL query that can be used to search 4 files holding "live" data

```
PREFIX rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX example:
<http://fake.host.edu/example-schema#>
select *
from <http://myhostname.edu/smith>
from <http://myhostname.edu/jones>
from <http://myhostname.edu/george>
from <http://myhostname.edu/blake>
where
{
$s $p $o .
}
```

If the data were all in one file, only one FROM clause would have been required. Here is a representation of the query results:

```
-----------------------------------------------
| s | p | o |
===============================================
| <myname/blake> | example:fav | myname/blake |
| <myname/blake> | example:age | "12" |
| <myname/blake> | example:name| "Blake" |
| <myname/blake> | rdf:type | example:Person |
| <myname/jones> | example:fav | myname/smith |
| <myname/jones> | example:age | "35" |
| <myname/jones> | example:name| "Jones" |
| <myname/jones> | rdf:type | example:Person |
| <myname/george>| example:fav | myname/smith |
| <myname/george>| example:age | "21" |
| <myname/george>| example:name| "George" |
| <myname/george>| rdf:type | example:Person |
| <myname/smith> | example:fav | myname/jones |
| <myname/smith> | example:age | "21" |
| <myname/smith> | example:name| "Smith" |
| <myname/smith> | rdf:type | example:Person |
-----------------------------------------------
```

where "myname" is an abbreviation for "http://myhostname.edu". In this query all 4 files were searched as if they were in a single file. (Note that the URI contents are different in this live
example.)

V. DIFFERENCE OF SYNTAX IN SQL AND SPARQL

SQL
SELECT UNIQUE E.SALARY
FROM EMPLOYEES AS E JOIN DEPARTMENTS AS D
WHERE E.ID=D.MANAGER;

SPARQL
SELECT ?salary
WHERE
{
?e rdb:employees/column#salary ?salary.
?d rdf:departments/column#manager ?e.
}

SPARQL
SELECT ?id, ?sal
WHERE { ?id HR:salary ?sal }

SQL
SELECT emp_id, salary
FROM employees

SPARQL
SELECT ?hdate
WHERE { ?id HR:salary ?sal .
?id HR:hire_date ?hdate .
FILTER ?sal >= 21750 }

SQL
SELECT hire_date
FROM employees
WHERE salary >= 21750

SPARQL
SELECT ?hdate
WHERE { ?id HR:salary ?sal .
?id HR:hire_date ?hdate .
FILTER ?sal >= 21750 }

SQL
SELECT v.hire_date
FROM emp_vars AS v, emp_consts AS c
WHERE v.salary >= 21750
AND v.emp_id = c.emp_id

It is possible to translate SPARQL expressions into SQL expressions thus allowing users to store RDF collections in relational database if required.
SQL: Great for finding data from tabular representations, can get complex when many tables are involved in a given query
SPARQL: Good pattern matching paradigm, especially when relationships have to be used to answer a query.

TABLE I
DIFFERENCE BETWEEN SPARQL AND SQL

Component Name	**Description**	**SQL**	**SPARQL**
Type Name	A data type	integer, char, sno, name	subject, predicate, object, uri, literal and bnode
Attribute Name	A distinct, descriptive name	status, city, sno, sname	Variables: ?s, ?city Node Postions: subject, predicate, object
Attribute	A combination of type name and attribute name	status:integer, char:city, sno:sno, sname:name	?s:subject, p1:predicate, ?city:object
Tuple or Tuple Value	A set of attribute and value pairs	sno sno(‘s1’), sname name(‘smith’), status 20, city ‘london’	?s:subject(#s1), p1:predicate(#name), o1:object(‘smith’)
Heading	A set of attributes	sno sno, sname name, status integer, city char	?s subject, p1 predicate, o1 object

VI. CONCLUSION:

SPARQL is a query language used to query RDF data stores. While SPARQL may initially look like SQL you will see that there are important differences because the data is graph-based so queries match graph patterns instead SQL’s relational matching operations. So the syntax is similar but SPARQL queries graph data and SQL queries relational data in tables.
Each of the query language has advantages and disadvantages. SQL and the relational model are well designed to represent highly regular or structured data such as that used by many business processes. Widely used examples include personnel and departments, students and classes, and manufacturing components. Such data usually includes a value for every column of every table. SQL also supports a special value- a null value- to represent data that is missing, unknown or inapplicable. The possibility of null values complicated the definition of and queries written in the SQL language focuses on identifying data for which most or all components are available and combining data based on the values of those components, by using SQL operators such as JOIN or UNION.

REFERENCES

[1] Pratical Semantic Web and Linked Data Applications by Java, JRuby, Scala and Clojure Edition.
[2] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web", Scientific American, May 2001, Vol. 284,No. 5, pp. 34-43.
[3] W3C, Resource Description Framework, http://www.w3.org/RDF/, 2004.
[4] W3C, RDF Vocabulary Description Language 1.0: RDF Schema, http://www.w3.org/TR/rdf-schema/, 2004.
[5] W3C, Web Ontology Language,http://www.w3.org/2004/OWL/, 2004.
[6] The RDF Query Language (RQL), http://139.91.183.30:9090/RDF/RQL/
[7] RDQL - A Query Language for RDF, W3C Member Submission,http://www.w3.org/Submission/2004/SUBMRDQL-20040109/, 9 January 2004.
[8] SPARQL Query Language for RDF, W3C Working Draft, http://www.w3.org/TR/2006/WD-rdf-sparql-query- 20061004/, 4 October 2006.
[9] SQL, XQUERY and SPAQL by Jim Melton.
[10] Michael Grobe : RDF, Jena, SparQL and the "Semantic Web"
[11] Querying the Semantic Web using a Relational Based SPARQL by Andrew Newman.