

SQL

SQL, XQuery, and SPARQL

XQuery

What's Wrong With This Picture?

SPARQL

Jim Melton, Oracle Corp.

Editor: all parts of SQL standard

Editor: XQuery F&O, XQueryX

Co-chair: W3C XML Query WG

Author: 5 SQL books,

forthcoming “Querying XML”

from Morgan Kaufmann

Query Languages: SQL (SQL Query Language)

- A language for querying collections of tuples:

```
SELECT SALARY, HIRE_DATE  
FROM EMPS  
WHERE EMP_ID = 13954
```

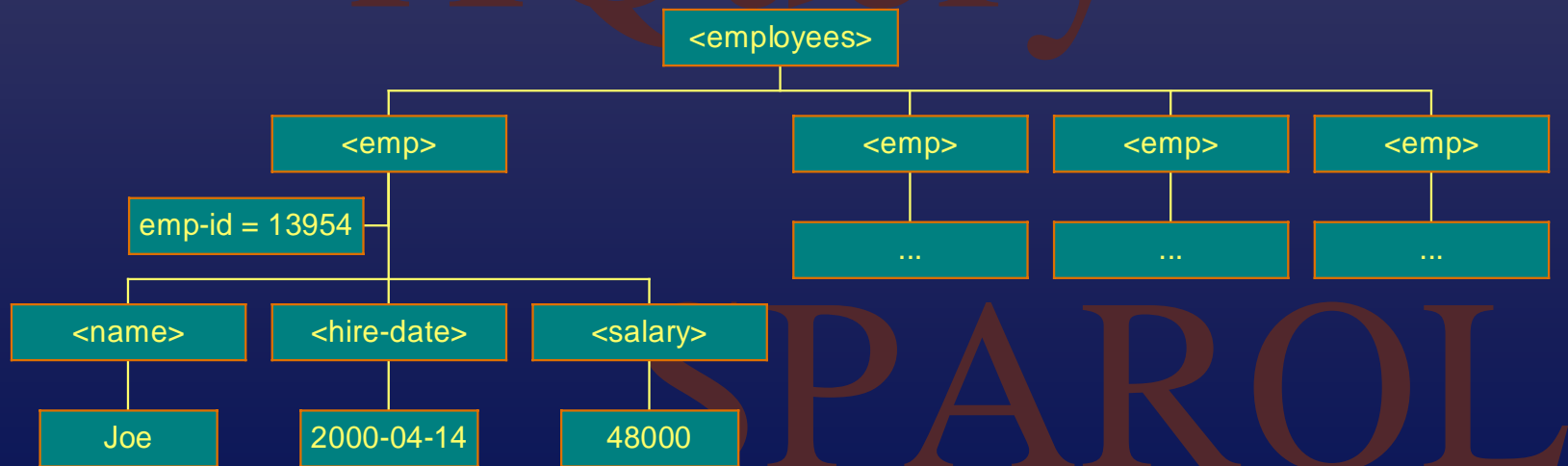
EMP_ID	NAME	HIRE_DATE	SALARY
13954	Joe	2000-04-14	48000
10335	Mary	1998-11-23	52000
...
04182	Bob	2005-02-10	21750

SQL (XML Query)

Query Languages: XQuery

- A language for querying trees of XDM nodes:

```
for $e in document(my_employees.xml)
where $emp/emp/@emp-id = 13954
return $emp/emp/salary
```

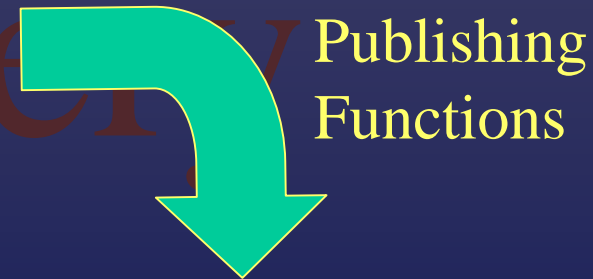


SQL/XML Boundaries

Crossing Data Model

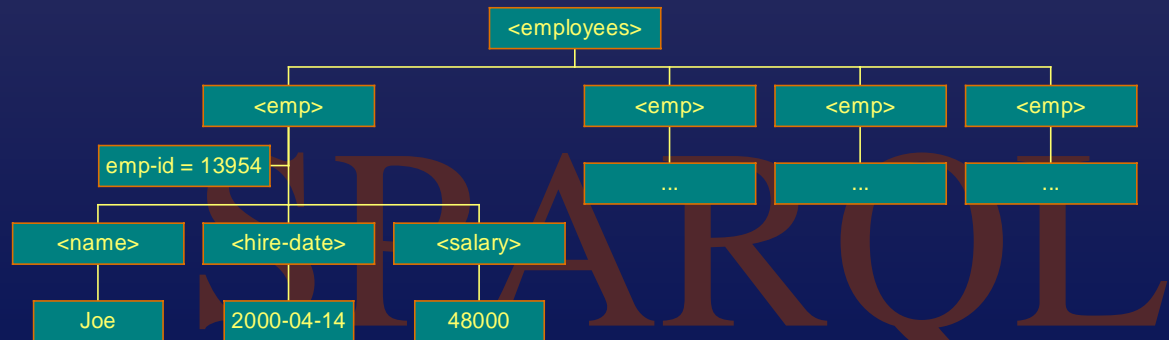
- SQL/XML

EMP_ID	NAME	HIRE_DATE	SALARY
13954	Joe	2000-04-14	48000
10335	Mary	1998-11-23	52000
...
04182	Bob	2005-02-10	21750



XMLTable

2005-11-16



Copyright © 2005 Oracle Corp.

RDF: Collections of Tuples (Resource Description Framework)

- 3-tuples: subject, predicate, object

```
emps:e13954 HR:name 'Joe'  
emps:e13954 HR:hire-date 2000-04-14  
emps:e13954 HR:salary 48000
```

- RDF in a table:

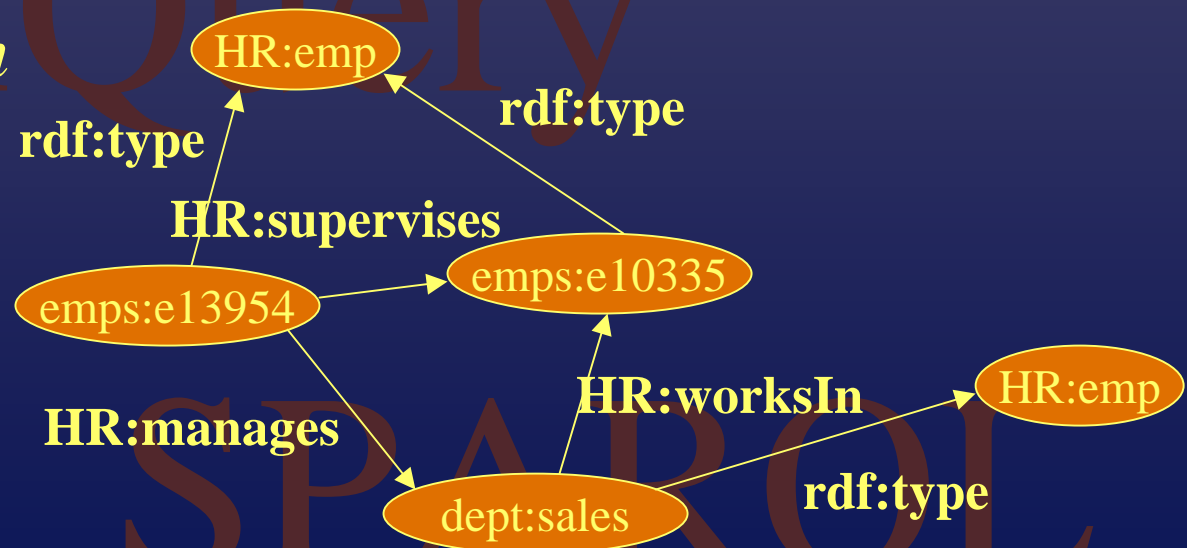
Subject	Predicate	Object
emps:e13954	HR:name	'Joe'
emps:e13954	HR:hire-date	2000-04-14
emps:e13954	HR:salary	48000

- Trivial SQL statement:

```
SELECT object  
FROM RDFtable  
WHERE subject="emps:e13954"
```

SQL RDF: Not Quite That Simple

- RDF can indicate membership in classes
(`emps:e13954 rdf:type HR:employee`)
- RDF prefixes are shorthand for full URIs
- RDF is a *graph* data model



OWL (Web Ontology Language)

- A particular vocabulary of RDF
- Represents meanings of terms and relationships between terms: an *ontology*
- OWL adds to RDF:
 - Relations between classes
 - Cardinality
 - Equality
 - More typing of and characteristics of properties
 - Enumerated classes

RDF vs The Relational Model

- Relational
 - Flat, tabular, implicit typing (column definition)
 - Joins used to combine information from tables
 - Foreign keys: semantics and graph-like structure
 - Each table: many columns = many attributes of object
- RDF
 - May be *viewed* as flat; explicit typing common
 - Explicit relationships *via* predicates
 - Inherent graph structure violates “flatness”
 - Triples form E-R model (similar to a table w/2 columns)

SQL RDF vs XDM

- XDM
 - Tree-structured plus sequences of items
 - No support for explicit relationships (references)
 - No tuples, not limited by tuples
- RDF
 - Network of objects; more general than trees
 - Relationships/references are the *point* of RDF
 - Triple nature creates plethora of tiny data

Query Languages: SPARQL (SPARQL Protocol And RDF Query Language)

- Designed to query collections of triples...
- ...and to easily traverse relationships
- Vaguely SQL-like syntax (SELECT, WHERE)
- “Matches graph patterns”

```
SELECT ?sal
```

```
WHERE { emps:e13954 HR:salary ?sal . }
```

SQL SPARQL vs SQL

- SPARQL

```
SELECT ?sal
```

```
WHERE { emps:e13954 HR:salary ?sal . }
```

- SQL

```
SELECT salary
```

```
FROM employees
```

```
WHERE emp_id = 'e13954'
```

SQL SPARQL vs SQL

- SPARQL

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

- SQL

```
SELECT emp_id, salary  
FROM employees
```

SPARQL

SQL SPARQL vs SQL

- 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
```

SQL SPARQL vs SQL

- 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
```

SQL Conclusions

- SQL: Great for finding data from tabular representations, can get complex when many tables are involved in a given query
- XQuery: Great for finding data in tree representations, can get complex when many relationships have to be traversed
- SPARQL: Good pattern matching paradigm, especially when relationships have to be used to answer a query
- Surprising conclusion: SPARQL can be translated to SQL and possibly to XQuery!

SQL

Q & A

XQuery

?

SPARQL

I just learned about:

A relational algebra for SPARQL

Richard Cyganiak

Digital Media Systems Laboratory

HP Laboratories Bristol

HPL-2005-170

September 28, 2005*

SQL, XQuery, and SPARQL

What's Wrong With This Picture?

Consulting Member of Technical Staff

Author JimMelton Oracle Corp.
SandyUTUSA

Copyright: 2005

Does the world really need Yet Another Query Language? A new language for querying RDF, named SPARQL, is emerging from the W3C. Some observers say that the W3C's own XQuery is sufficient for querying RDF, at least in its XML incarnations, while others suggest that SQL is a more mature, widely-implemented language for querying tuples. This presentation explores these issues and positions the three languages.

Keywords: XMLXQuerySQLquery languagesSPARQLsemantic webRDF

Introduction

The World-Wide Consortium (W3C) has recently published two Last Call Working Drafts [<Xref to SPARQL-Language>](#)[<Xref to SPARQL-Results>](#) and one Working Draft [<Xref to SPARQL-Protocol>](#) defining a new query language named SPARQL. This new language is described as “a query language for getting information from ... RDF graphs” (that is, an RDF query language), which seems on the surface to be a new technology requirement.

But is it? RDF is described [<Xref to RDF-Concepts>](#) as “a collection of triples, each consisting of a subject, a predicate and an object”. Triples are, of course, 3-tuples, and the well-known relational model of data is explicitly designed to represent tuples and collections of them. Similarly, SQL is a language expressly designed for the identification and retrieval of information from collections of tuples.

On the other hand, RDF can be (and frequently is) represented in XML [<Xref to RDF-Syntax>](#). Another language currently nearing completion within the W3C, XQuery [<Xref to XQuery>](#), has been carefully designed for the location and retrieval of information from XML documents.

A naïve user could be excused for wondering just why SPARQL is justified, given that two existing query languages—one of them developed in the same organization—appear to cover the requirements that led to the new language. In this paper, we examine the relationships between SQL, XQuery, and SPARQL to determine whether a new query language for RDF is justified.

Data Models

Query languages are typically designed to be applied to data corresponding to a particular data model. For example, SQL [<Xref to SQL2003>](#) is used to retrieve, create, modify, and delete data represented in (a variation of) the relational model of data. Similarly, XQuery is used to locate and retrieve (but not yet update) data that is represented in the XPath data model [<Xref to DataModel>](#).

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 language's data model. One important example is a recent addition to the SQL standard, SQL/XML [<Xref to SQLXML>](#). SQL/XML allows relational data to be *published* in an XML form (XPath data model instance) that can then be queried using XQuery. Additionally, it provides a facility called XMLTABLE that allows XML to be viewed as through it were ordinary SQL tabular data. Naturally, such mappings run into the famous “impedance mismatch” caused by factors such as the collections of data types differing amongst query languages and their corresponding data models.

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 of [<Xref to RDF-Concepts>](#) shows subtle differences between collections of RDF triples and multisets of rows in SQL tables of three columns. For example, SQL tables are defined to comprise one or more columns, each having a particular declared data type (such as INTEGER, TIMESTAMP, or some user-defined 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.

For columns of a user-defined type, values may have a *most-specific type* that is a subtype of that user-defined type. None of SQL's built-in types are defined to have subtypes (or supertypes), so this notion does not apply to columns declared to be of those types.

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 a number of “system tables”, which (if actually provided by a given SQL implementation) 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 [<Xref to OWL-Language>](#) constructs that specify the *class* to which a given RDF entity belongs. We are currently investigating whether the use of SQL's user-defined types would offer some way to map such class information from RDF into the SQL model.

Strengths and Weaknesses

Each of these data models and corresponding query languages has advantages and disadvantages. SQL and the relational model are well designed to represent highly regular (“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 (but not the pure relational model!) supports a special value—called the *null value*—to represent data that is missing, unknown, or inapplicable. The possibility of null values complicates the definition of and queries written in the SQL language, which makes SQL awkward for use in dealing with less structured information. The syntax of 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. In particular, combining data from two or more tables is specified by explicit SQL operators such as JOIN or UNION.

XPath, XQuery, and the XPath Data Model are all directed at support of less regular data. These languages and their data model function well when presented with data in which most or all of the values are present, but they also function well when applied to data in which many values are not represented at all. Such data, often called “semi-structured data”, is quite common in the XML tree-structured world in which elements and attributes may be optional and omitted

entirely from instance data. It has been argued that the XPath Data Model represents a superset of, and could thus supersede, the relational model. We reject that conclusion because of the inherent overhead required by the XPath Data Model to self-identify each piece of data *versus* SQL's regular structures. That is, each datum in an SQL table takes its “name” from the name of the column in which it appears, while the XPath Data Model requires that the name of each datum be given explicitly—on every instance—as the name of the element or attribute of which it is the value. XQuery syntax is optimized for building new XML documents from one or more inherently semi-structured XML documents, easily accomodating the complete absence of some data. Combining data from two or more (source) XML documents is specified through the use of explicit operators such as a comma in a for expression or the keyword union.

To which of these camps does RDF belong? Every RDF triple comprises a subject, a predicate (or *property*), and an object, which—even though the subject or the object may not be explicit (that is, they may be represented by “blank nodes”) in some triples—implies that RDF is structured data. However, RDF defines a graph-based (in years past, the term “network” was often used) data model, which—like tree-based data models such as XML—supports the concept of optional data. In fact, [<Xref to RDF-Concepts>](#) states that “it is not assumed that complete information about any resource is available”. We conclude that the RDF model is structured in the same sense that the relational model is: every provided assertion—SQL row or RDF triple—is complete (with the occassional missing datum represented by an SQL null value or an RDF blank node), but there may be assertions missing from the table or graph. As we show in [<Xref to query-examples>](#), SPARQL's syntax has been designed to combine information taken from one or more RDF graphs without the need for query authors to explicitly identify the mechanisms by which the graphs are combined. That is, operations such as joins are implicit rather than explicit in the language's syntax.

Transformations

In many ways, RDF corresponds to an 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” (for example, “Srinivas salary 125000.00”). In fact, this observation provides a (trivial) method of transforming SQL tables into RDF graphs. Similarly, XPath Data Model instances could also be decomposed into entity-relationship assertions of the form “parent-element-node-id has-childor-attribute child-element-or-attribute-value-or-node-id” (e.g., “node-for-Yungmin attribute-named-birthdate 1975-06-15”). Again, this provides a trivial method of transforming XML documents into RDF collections.

We mentioned earlier ([<Xref to data-models>](#)) that SQL/XML provides the ability to publish relational data into an XML form. That standard also provides a default mapping from SQL tables into XML. We call such mappings “lenses” because they permit applications to “see” XML when they “look at” relational data. The default mapping is necessarily highly generalized and does not account for applications' interpretation of the data being mapped. The preceding paragraph illustrates how such lenses could be built for mapping relational data and XML data into RDF. Those mappings are similarly naïve and do not necessarily capture the essence of the data's meaning. In all such cases, customized lenses can be constructed by application architects to perform mappings that capture more of the semantics of the data. Whether all semantics of the data can be transformed is an open question.

In the same way that data can be mapped, or transformed, between data models, queries written in the language associated with each such model can often be translated into the language associated with another model. For example, we know of at least one XQuery implementation that transforms XQuery expressions into SQL expressions to operate on XPath Data Model instances that have been transformed into SQL tabular data (through an operation commonly called “shredding”). Our exploration of this notion has demonstrated—but not yet *proved*—that it is similarly possible to translate (most?) SPARQL queries into SQL syntax. We suspect that SPARQL queries can also be translated into XQuery syntax, but have not yet investigated that possibility. It may also be possible to transform SQL expressions and/or XQuery expressions into SPARQL syntax, but we have not even started to look at that.

In [<Xref to query-examples>](#), we illustrate an SQL query and the corresponding XQuery and SPARQL queries.

Query Examples

To illustrate the different approaches taken by SQL, XQuery, and SPARQL, we provide “the same” data in three forms: relational tables ([<Xref to depts-table>](#) and [<Xref to depts-table>](#)), two XML documents ([<Xref to emps-doc>](#) and [<Xref to depts-doc>](#)), and a collection of RDF triples ([<Xref to RDF-collection>](#)), then write “the same” query in each language. The query we have chosen, expressed in natural language, is: What is the salary of the manager of each department?

First, we provide abbreviated versions of two SQL tables describing employees and departments, as seen in [<Xref to emps-table>](#) and [<Xref to depts-table>](#), respectively.

EMPLOYEES Table

ID	NAME	SALARY	DEPARTMENT
105	Jung-Shin Park	105000.00	Software
...
29	Ralph Swinden	91500.00	Marketing

DEPARTMENTS Table

ID	NAME	MANAGER
21	Software	105
...
83	Marketing	105

The SQL query expression corresponding to our question is:

```
SELECT UNIQUE E.SALARY
FROM EMPLOYEES AS E JOIN DEPARTMENTS AS D
USING E.ID = D.MANAGER
```

Next, we illustrate abbreviated XML documents that describe employees and departments, as seen in [<Xref to emps-doc>](#) and [<Xref to depts-doc>](#), respectively.

Example: Employees Document

```
<?xml version="1.0" encoding="UTF-8"?>
<employees>
  <employee id="105">
    <name>Jung-Shin Park</name>
    <salary>105000.00</salary>
    <department>Software</department>
  </employee>
  <employee id="...">
    <name>...</name>
    <salary>...</salary>
    <department>...</department>
  </employee>
  <employee id="29">
    <name>Ralph Swinden</name>
```

```

    <salary>91500.00</salary>
    <department>Marketing</department>
  </employee>
</employees>

```

Example: Departments Document

```

<?xml version="1.0" encoding="UTF-8"?>
<departments>
  <department id="21" name="Software">
    <manager ref="105"</>
  </department>
  <department id="..." name="...">
    <manager ref=""</>
  </department>
  <department id="83" name="Marketing">
    <manager ref="105"</>
  </department>
</departments>

```

The XQuery expression that states our question is:

```

for $e in fn:doc("http://employees.example.com/emps-
doc.xml")/employees/employee,
    $d in fn:doc("http://employees.example.com/depts-doc.xml")
departments/department
where $e/@id eq $d/manager/@ref
return $e/salary

```

Finally, we present an abbreviated RDF collection that describes employees and departments, as seen in [<Xref to RDF-collection>](#).

In [<Xref to RDF-collection>](#), we presume that the following namespace declarations are in effect:

```

emps:      http://emps.example.org/employees#
depts:     http://depts.example.org/departments#
rdb:       http://databases.example.org/

```

RDF for Employees and Departments

Subject	Predicate	Object
emps:id105	rdb:employees/column#name	Jung-Shin Park
emps:id105	rdb:employees/column#salary	105000.00
emps:id105	rdb:employees/column#department	depts:id21
emps:id29	rdb:employees/column#name	Ralph Swinden
emps:id29	rdb:employees/column#salary	91500.00
emps:id29	rdb:employees/column#department	depts:id83
emps:id...	rdb:employees/column#...	...
depts:id21	rdb:departments/column#name	Software

depts:id21	rdb:departments/column#manager	emps:id105
depts:id21	rdb:departments/column#name	Marketing
depts:id83	rdb:departments/column#manager	emps:id105
depts:id...	rdb:departments/column#...	...

Our question expressed in SPARQL (assuming the same namespace declarations) is:

```
select ?salary
where
{
  ?e rdb:employees/column#salary ?salary .
  ?d rdb:departments/column#manager ?e .
}
```

So, What Is Wrong With This Picture?

Perhaps the most important thing that is “wrong with this picture” is the confusion that is likely to arise among potential users of the technologies discussed in this paper. We have already started to get questions from smart, well educated application developers asking for an explanation—a set of simple rules, if possible—about the circumstances under which they should choose to use a relational database and SQL, XML documents and XQuery, or RDF and SPARQL.

Under the assumption that people understand the tradeoffs between relational databases, XML documents, and RDF collections, it is somewhat easier to explain the justification for each of the three languages. But we continue to encounter people who ask why RDF isn't, or shouldn't be, stored in a relational database and then queried using SQL. Surely, they say, those “reasoning engines” that operate on RDF and OWL constructs could just as easily operate on them in the context of a relational database as in a new kind of collection. In that case, why wouldn't SQL be the language of choice for answering questions before and after those engines have done their jobs?

The best answer we have been able to provide is this: SPARQL syntax makes virtually all join operations implicit, while SQL syntax usually makes them explicit. A consequence of this design decision is that the SQL expressions to answer typical questions that will be asked against RDF collections tend to be much larger and somewhat more difficult to create (correctly!) because of the need to write explicit join operations and the explicit join conditions. Because typical questions made of RDF involve several, sometimes many, join operations, SPARQL provides a more compact notation that is perhaps easier to get right with less debugging time spent.

Similarly, we answer questions asking whether XQuery wouldn't be more appropriate, since RDF is typically serialized as XML, with a similar answer: the number of explicit join operations in `for` clauses and the number of join conditions required in the `where` clauses probably makes the XQuery expressions more tedious to write and to debug than the corresponding SPARQL queries.

Conclusions

We have, somewhat reluctantly, concluded that the design goals of SQL and SPARQL are sufficiently different that there is adequate justification for the creation of a special-purpose language for querying RDF collections. We are comforted by the belief that it is possible to translate SPARQL expressions into SQL expressions, allowing users to

store their RDF collections in relational databases if they wish to do so, and to write their queries in either SQL or in SPARQL, as they see fit. While predicting that it will be similarly possible to serialize RDF collections into XML documents and transform SPARQL expressions into XQuery expressions, we do not believe that most users would take that direction.

The author would like to thank Ashok Malhotra, Dana Florescu, and Jeff Mischkin for their help in researching topics related to the semantic web, including RDF and SPARQL.

[SPARQL Language][*SPARQL Query Language for RDF.*](#)

[SPARQL Results][*SPARQL Query Results XML Format.*](#)

[SPARQL Protocol][*SPARQL Protocol for RDF.*](#)

[RDF Concepts][*Resource Description Framework \(RDF\): Concepts and Abstract Syntax.*](#)

[RDF Syntax][*RDF/XML Syntax Specification \(Revised\).*](#)

[XQuery][*XQuery 1.0: An XML Query Language.*](#)

[SQL2003]ISO/IEC 9075:2003, *Information technology — Database languages — SQL*, International Organization for Standardization2003.

[DataModel][*XPath 2.0 and XQuery 1.0 Data Model*](#)

[SQLXML]ISO/IEC 9075-14:2003*Information technology — Database languages — SQL — Part 14: XML-Related Specifications (SQL/XML)*, International Organization for Standardization2003.

[OWL Language][*OWL Web Ontology Language Reference.*](#)