SQL, XQuery, and SPARQL

What’s Wrong With This Picture?
Jim Melton, Oracle Corp.
Co-chair: W3C XML Query WG
Editor: XQuery F&O, XQueryX
Editor: all parts of SQL standard
Author: 5 SQL books, as well as “Querying XML” (all from Morgan Kaufmann Publishers)
Genesis of this Talk

- Interested in Semantic Web, RDF, & OWL
- Discovered existence of SPARQL
- Employer implementing RDF/OWL
- Directed to make recommendation re: implementing SPARQL
- Research into implications of and relationships between RDF/SPARQL, relational/SQL, and XML/XQuery
**Query Languages: SQL (SQL Query Language)**

- A language for querying collections of tuples:

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

<table>
<thead>
<tr>
<th>EMP_ID</th>
<th>NAME</th>
<th>HIRE_DATE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>13954</td>
<td>Joe</td>
<td>2000-04-14</td>
<td>48000</td>
</tr>
<tr>
<td>10335</td>
<td>Mary</td>
<td>1998-11-23</td>
<td>52000</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>04182</td>
<td>Bob</td>
<td>2005-02-10</td>
<td>21750</td>
</tr>
</tbody>
</table>
Query Languages: XQuery (XML Query)

• A language for querying trees of XDM nodes:

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

- SQL/XML

<table>
<thead>
<tr>
<th>EMP_ID</th>
<th>NAME</th>
<th>HIRE_DATE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>13954</td>
<td>Joe</td>
<td>2000-04-14</td>
<td>48000</td>
</tr>
<tr>
<td>10335</td>
<td>Mary</td>
<td>1998-11-23</td>
<td>52000</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>04182</td>
<td>Bob</td>
<td>2005-02-10</td>
<td>21750</td>
</tr>
</tbody>
</table>

Publishing Functions

XMLTable

2006-03-01

Copyright © 2006 Oracle Corp.
RDF: Collections of Tuples (Resource Description Framework)

- 3-tuples: subject, predicate, object
  
  \[
  \begin{align*}
  &\text{emps:e13954 HR:name 'Joe'} \\
  &\text{emps:e13954 HR:hire-date 2000-04-14} \\
  &\text{emps:e13954 HR:salary 48000}
  \end{align*}
  \]

- RDF in a table:

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>emps:e13954</td>
<td>HR:name</td>
<td>'Joe'</td>
</tr>
<tr>
<td>emps:e13954</td>
<td>HR:hire-date</td>
<td>2000-04-14</td>
</tr>
<tr>
<td>emps:e13954</td>
<td>HR:salary</td>
<td>48000</td>
</tr>
</tbody>
</table>

- Trivial SQL statement:
  
  ```sql
  SELECT object
  FROM RDFtable
  WHERE subject="emps:e13954"
  ```
RDF: Not Quite That Simple

- RDF can indicate membership in classes
  \[(\text{emps:e13954} \ \text{rdf:type} \ \text{HR:employee})\]
- RDF prefixes are shorthand for full URIs
- RDF is a \textit{graph} data model
OWL (Web Ontology Language)

• A particular vocabulary of RDF
• Represents meanings of terms and relationships between terms: an ontology
• OWL is RDF, but adds:
  – 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 ≈ E-R model (representable as 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 }
SPARQL vs SQL

- SPARQL
  SELECT ?sal
  WHERE { emps:e13954 HR:salary ?sal . }

- SQL
  SELECT salary
  FROM employees
  WHERE emp_id = 'e13954'
SPARQL vs SQL

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

- **SQL**
  
  ```
  SELECT emp_id, salary
  FROM employees
  ```
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
  ```
SPARQL vs SQL

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

• **SQL**
  ```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
  ```
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!