

XQuery, SQL/XML, and the Semantic Web



Jim Melton
USA: Oracle Corp.

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, and
forthcoming “Querying XML”
from Morgan Kaufmann**

Part I
XQuery: An Introduction



What is XQuery?

- A language for querying XML
- A human-readable syntax for an XML query language (contrast: XQueryX)
- A functional programming language
- The product of the W3C's XML Query Working Group

Historical Information

- XML Query Working Group established September, 1999
- Requirements posted 31 January, 2000
- First public Working Drafts posted early 2000
- Candidate Recommendation published November 3, 2005

Typical W3C Process



- Workshop?
- Working Group established
- Requirements and Use Cases
- Internal and Public Working Drafts
- Last Call Working Draft
- Candidate Recommendation
- Proposed Recommendation
- Recommendation

XQuery 1.0 & XPath 2.0

XQueryX 1.0

XQuery 1.0

XSLT 2.0

XPath 2.0

XQuery 1.0 & XPath 2.0
Formal Semantics

XQuery 1.0 & XPath 2.0
Functions & Operators

XQuery 1.0 & XPath 2.0
Data Model (XDM)

XML Schema 1.0

XQuery 1.0 & XPath 2.0
Requirements & Use Cases

XML 1.0

Namespaces 1.0

XQuery “Model”

- FLWOR expressions (formerly “FLWR”):
for, let, where, order by, and return
- Operates only on abstract XML (“instance of XDM”); produces only XDM instances
- Depends on XML Schema (but can deal with DTD-validated XML or XML with no metadata)

Sample Data

```
<?xml version="1.0"?>
<bib>
  <book>
    <ISBN>1-12345-123-12</ISBN>
    <title>A Fictional Work</title>
    <author>M. O. Whistler</author>
  </book>
  <book>...</book>
  <book>...</book>
</bib>
```

Possible DTD for Sample Data

```
<?xml version="1.0"?>  
<!ELEMENT ISBN (#PCDATA)>  
<!ELEMENT author (#PCDATA)>  
<!ELEMENT bib (book)>  
<!ELEMENT book  
      (ISBN, title, author)>  
<!ELEMENT title (#PCDATA)>
```

Possible Schema for Sample Data

```
<?xml version="1.0"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="ISBN" type="xs:string"/>
  <xs:element name="author"
    type="xs:string"/>
```

Possible Schema for Sample Data (continued)

```
<xs:element name="bib">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="book"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Possible Schema for Sample Data (continued)

```
<xs:element name="book">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ISBN"/>
      <xs:element ref="title"/>
      <xs:element ref="author"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Possible Schema for Sample Data (continued)

```
<xs:element name="title"  
            type="xs:string" />  
</xs:schema>
```

An XPath “query”

`/bib/book[1]/author`

→ `<author>M. O.
Whistler</author>`

`/bib//author`

→ `<author> M. O. Whistler
</author> <author>... </author>
<author>... </author>`

An XQuery

**Find all books whose author wrote another book;
return the titles of those books in order by ISBN**

```
for $b1 in /bib/book
for $b2 in
  /bib/book[author=$b1/author]
order by $b1/ISBN
return $b1/title
```

What does an XQuery return?

- A sequence of elements or values — as a “data model instance”
- Not necessarily well-formed, much less *valid*
- Problems arise only if *outermost* query returns non-well-formed results

XQuery Validates

- XQuery expressions may include *explicit validation* — query author may choose between strict, lax, or skip validation
- Validation based on “pre-known” XML schemas or imported schemas

XQuery Analogs in SQL

- Joins, including outer joins
- Collations for string comparisons
- Casts and “treat as” operations
- Optimization often depends on lack of explicit order of data

Part II
SQL/XML: The Basics



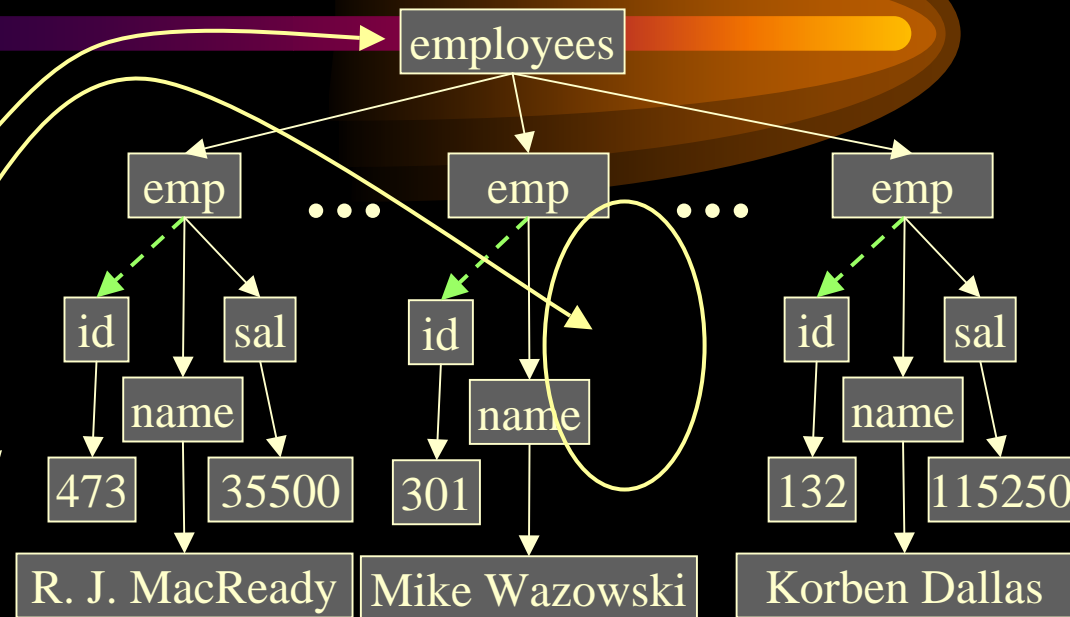
SQL—Structured Data

- Based on relational model
- Tables of rows and columns
- Every “cell” has a value (possibly null)
- SQL Query Language extremely powerful (probably too big)
- Products extremely well-developed: scalable, robust, manageable, *etc.*

EMP_ID	NAME	SALARY
473	R. J. MacReady	35500.00
921	Sam Loomis	26350.00
301	Mike Wazowski	(null)
17	David Kessler	22600.00
1284	Laurie Strode	14000.00
132	Korben Dallas	115250.00

XML—Semi-Structured Data

- Foundation for “User” Markup Languages
- Tree-structured, semi-structured
- Descended from SGML
- DTD, XML Schema, RELAX (NG), *etc.*
- Rapid Acceptance, Tools Widely Available
- “We Make Your Data Fatter” (binary XML???)



XML Schema & the XPath/XQuery Data Model

- XML Schema
 - Structures
 - Types
 - Complex, Poorly Understood, Poorly Written?
- XPath 2.0 & XQuery 1.0 Data Model (XDM)
 - Subset of Superset of XML Schema
 - Adds New Types, Nearly Ignores Others
 - Less Complex, More Expressive

XQuery 1.0 & XPath 2.0

- XQuery 1.0 \subset XPath 2.0 (same sources)
- XQuery 1.0: FLWOR expressions
XPath 2.0: FR expressions
- XPath 2.0 optimized for use in other languages, such as XSLT 2.0
- XQuery 1.0: Modules, prologs, (external) variables, construction, user-defined functions, validation
- Both: Strong typing, optional static typing, static and dynamic contexts

Relational → XML Publishing Functions

- XMLParse
- XMLSerialize
- XMLQuery
- XMLTable
- XMLDocument
- XMLElement
 - XMLAttribute
- XMLForest
- XMLAgg
- XMLConcat

- XMLComment
- XMLPI
- XMLText

Relational → XML

Example: XMLElement

```
XMLELEMENT ( NAME = "emp"  
  XMLCOMMENT ( "Example 1" ),  
  XMLATTRIBUTES  
    ( EMP_ID AS "id" ),  
  XMLELEMENT ( NAME = "name",  
    NAME ),  
  XMLELEMENT ( NAME = "sal",  
    SALARY ) )
```

In serialized form:

```
<emp id="473">  
  <!-- Example 1 -->  
  <name>R. J.  
    MacReady</name>  
  <sal>35500.00</sal>  
</emp>
```

Relational → XML Mappings

- SQL identifiers to XML names
- SQL data types to XML names and Schema data types (& SQL values to XML)
- SQL tables, schemas, catalogs to XML Schema data types, XML element(s), and XML (documents)+ XML Schema documents)

XML → Relational Mappings

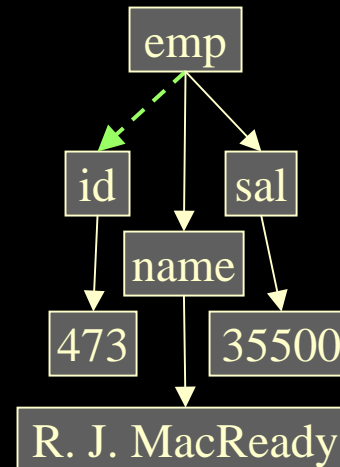
- XML names to SQL identifiers
- Unicode to SQL character sets
- XQuery atomic values to SQL values

XML → Relational Storage Considerations

- VARCHAR
- CLOB
- BLOB

```
<emp id="473">  
  <!-- Example 1 -->  
  <name>R. J.  
    MacReady</name>  
  <sal>35500.00</sal>  
</emp>
```

- “Native” XML



The XML Type & Modifiers

- XML
- Primary Modifiers
 - CONTENT
 - DOCUMENT
 - SEQUENCE
- Secondary Modifiers (CONTENT & DOCUMENT)
 - UNTYPED
 - ANY
 - XMLSCHEMA

DEPT_ID	Department
ENG-1	<pre><?xml version 1.0?> <dept id="ENG-1"> <name="Engineering">...</name> ... </dept></pre>
HR-23	<pre><?xml version 1.0?> <dept id="HR-23"> <name="Human Resources">...</name> ... </dept></pre>
PAY-7	<pre><?xml version 1.0?> <dept id="PAY-7"> <name="Payroll">...</name> ... </dept></pre>

XML (DOCUMENT (UNTYPED))

XML (CONTENT (XMLSCHEMA))

Parsing & Serialization

- **XMLParse**: Parses a string value using an XML parser; produces value whose specific type is `XML(DOCUMENT(ANY))`, or `...CONTENT...`, or `...UNTYPED...`
- **XMLSerialize**: transforms an XML value into a string value (`CHAR`, `VARCHAR`, `CLOB`, or `BLOB`)

XQuery Within SQL

- Problem: How can SQL applications locate and retrieve information in XML documents stored in an SQL database cell?
- Answer: Invoke XQuery from SQL statements (retrieve—in SELECT list; locate—in WHERE clause)!

XQuery Within SQL

- **XMLQuery**: A new SQL expression, invoked as a pseudo-function, whose data type can be an XML type—such as `XML(CONTENT(ANY))`—or an ordinary SQL type
- **XMLExists**: A new SQL predicate, invoked as a pseudo-function, returning *true* when the contained XQuery expression returns anything other than the empty sequence (*false*) or SQL null value (*unknown*)

XQuery Within SQL

- **XMLValidate**: Validates an XML value against an XML Schema (or target namespace), returning new XML value with type annotations
- **IS VALID**: Tests an XML value to determine whether or not it is valid according to an XML Schema (or target namespace); return *true/false* without altering the XML value itself

XQuery Within SQL

- Other new predicates:
 - IS DOCUMENT: determines whether an XML value satisfies the (SQL/XML) criteria for an XML document
 - IS CONTENT: determines whether an XML value satisfies the (SQL/XML) criteria for XML content

XMLTable

- Provides an *SQL view* of XML data
- Evaluates an XQuery “row pattern” with optional arguments (as with XMLQuery)
- Element/attribute values mapped to columns using XQuery “column patterns”
- Names & types of columns required; default values optional

Namespaces

- Completely supported per XML+Namespaces:
 - XMLElement, XMLForest, XMLTable
 - Default namespace, explicit namespace (prefix)
 - Declare namespace within scopes of WITH clause, column definitions, constraint definitions, insert/delete/update statements, compound statements
- Built-in namespaces: xs:, xsi:, xdt:, and sqlxml:

Validation & Registered Schemas

- XML Schemas may be *registered* with the SQL-server
 - Implementation-defined mechanism
 - Known by SQL name & by target namespace URI
 - Used by XMLValidate(), IS VALID, and to restrict values of XML(DOCUMENT-or-CONTENT(XMLSCHEMA))

Validation & Registered Schemas

- Registered because of security issues
 - Schemas cannot “disappear” without SQL-server knowing about it
 - Schemas cannot be “hijacked” (altered in inappropriate ways) without SQL-server knowing about it
 - Documents cannot be marked “valid” against schemas unless SQL-server knows about them

The SQL/XML Standard


- ISO/IEC 9074-14:2003
 - Mappings and Publishing Functions
- ISO/IEC 9075-14:2005 (“almost completed”)
 - Adds XQuery, including Data Model, Validation
- ISO/IEC 9075-14:2007 (planned)
 - Full-Text?
 - Update?
 - Something else?

Summary

- SQL/XML:2003 plus
 - Additional publishing functions
 - XQuery data model
 - More precise XML type (modifiers)
 - XMLQuery, XMLTable
 - XMLValidate, IS VALID
 - XMLExists, IS DOCUMENT, IS CONTENT
 - Casting between XML type and SQL types

Part III

*SQL, XQuery, and SPARQL:
What's Wrong With This Picture?*



Query Languages: SQL (SQL Query Language)

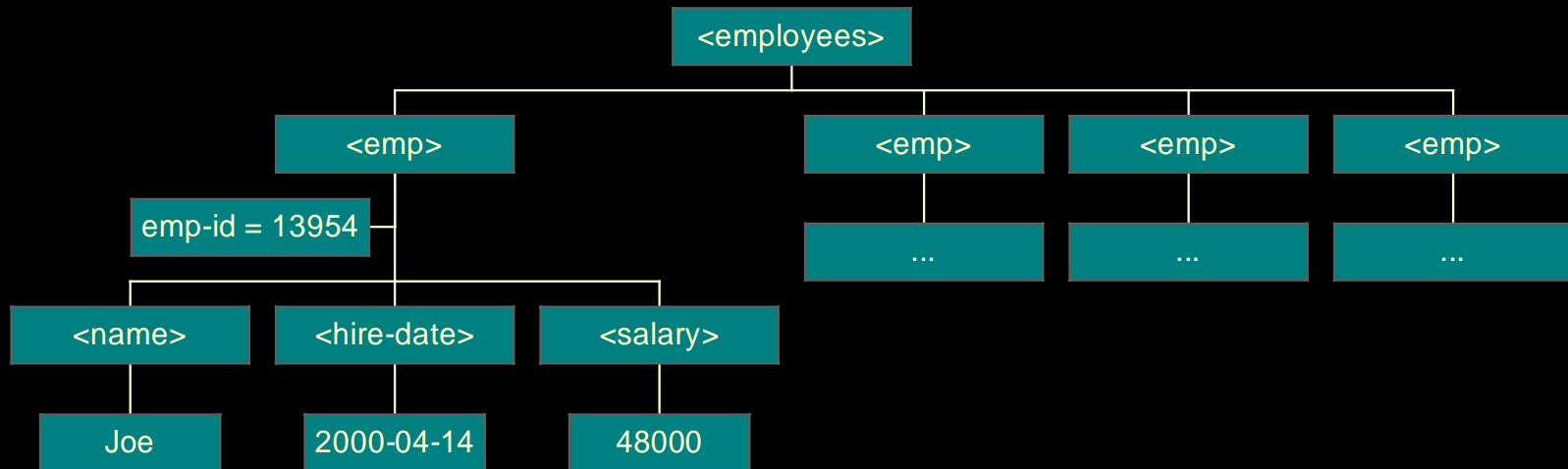
- A language for querying collections of tuples:

```
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

Query Languages: XQuery (XML Query)

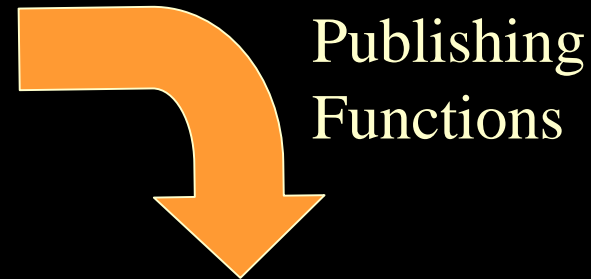
- A language for querying trees of XDM nodes:
`for $e in document(my_employees.xml)`
`where $emp/emp/@emp-id = 13954`
`return $emp/emp/salary`



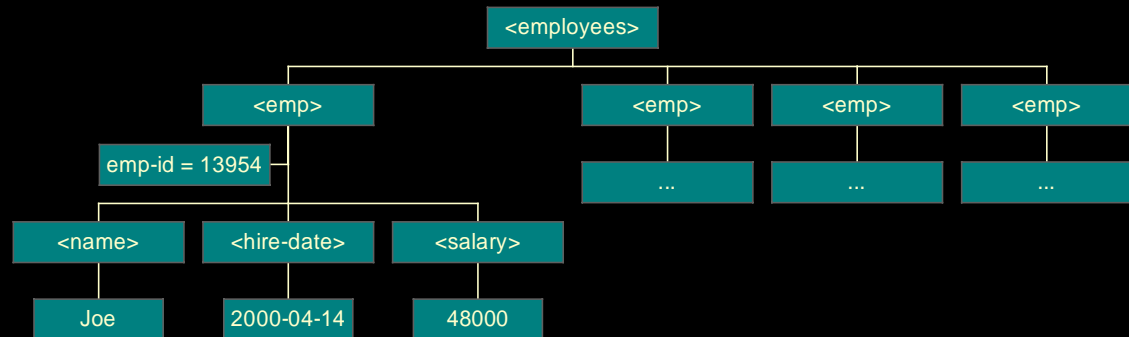
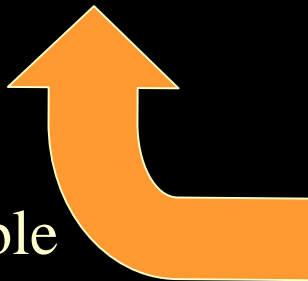
Crossing Data Model Boundaries

- 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



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:

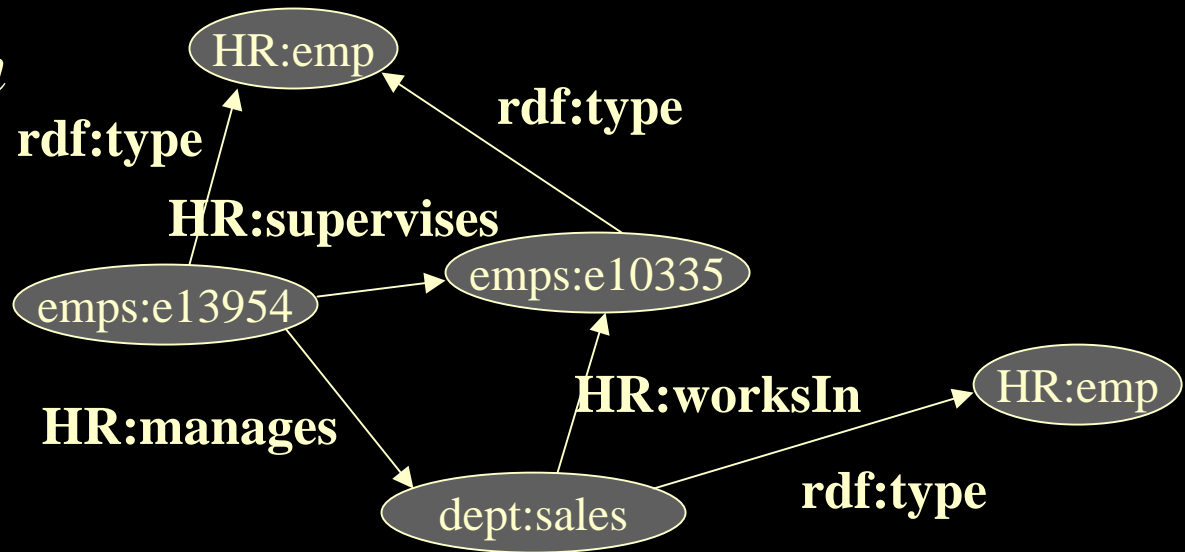
Subject	Predicate	Object
emps:e13954	HR:name	'Joe'
emps:e13954	HR:hire-date	2000-04-14
emps:e13954	HR:salary	48000

- Trivial SQL statement:

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

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



(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)

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

```
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
```

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!

*XQuery, SQL/XML, and
the Semantic Web*



Questions?