Semantic Integration of XML Using a RDF Global Mediator
Feihong Hsu
Masters Thesis Defense
March 10, 2004

Why Semantic Integration? (1)

• XML documents are scattered throughout the web—but there is a lot of heterogeneity in terms of their schemas!
• We want to use the information contained within them, but we don’t have the time to translate each and every document to “our” format.
Why Semantic Integration? (2)

- We would like to be able to take advantage of common data among documents.
- We would like to be able to ask higher-level queries.

The Semantic Web Stack – Where We Fit In
Key Problems in Semantic Integration

• Schematic Heterogeneity
• Semantic Heterogeneity
• Semantic Relationships
• Object Identity

Schematic Heterogeneity

<actors>
  <actor name="B. del Toro">
    <films>
      <film title="21 Grams"/>
      <film title="Traffic"/>
    </films>
  </actor>
</actors>

<films>
  <film title="21 Grams">
    <actor name="B. del Toro"/>
  </film>
  <film title="Traffic">
    <actor name="B. del Toro"/>
  </film>
</films>

Documents can contain the same element and attribute names but have different nested structures.
### Semantic Heterogeneity (1)

<table>
<thead>
<tr>
<th>Employees</th>
<th>Workers</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;employees&gt;</td>
<td>&lt;workers&gt;</td>
</tr>
<tr>
<td>&lt;employee&gt;</td>
<td>&lt;worker&gt;</td>
</tr>
<tr>
<td>&lt;name first=&quot;Feihong&quot; last=&quot;Hsu&quot;/&gt;</td>
<td>&lt;name&gt;Feihong Hsu&lt;/name&gt;</td>
</tr>
<tr>
<td>&lt;role&gt;Janitor&lt;/role&gt;</td>
<td>&lt;job&gt;Janitor&lt;/job&gt;</td>
</tr>
<tr>
<td>&lt;salary&gt;90000&lt;/salary&gt;</td>
<td>&lt;comp&gt;90000&lt;/comp&gt;</td>
</tr>
<tr>
<td>&lt;/employee&gt;</td>
<td>&lt;/worker&gt;</td>
</tr>
<tr>
<td>&lt;/employees&gt;</td>
<td>&lt;/workers&gt;</td>
</tr>
</tbody>
</table>

Documents can have the same semantics but have different names for elements and attributes.

---

### Semantic Heterogeneity (2)

<table>
<thead>
<tr>
<th>Stars</th>
<th>Stars</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;stars&gt;</td>
<td>&lt;stars&gt;</td>
</tr>
<tr>
<td>&lt;star name=&quot;Betelgeuse&quot;&gt;</td>
<td>&lt;star name=&quot;Eva Gardner&quot;&gt;</td>
</tr>
<tr>
<td>&lt;distance&gt;425 light years &lt;/distance&gt;</td>
<td>&lt;born&gt;1922-12-24&lt;/born&gt;</td>
</tr>
<tr>
<td>&lt;luminosity from=&quot;40000&quot; to=&quot;100000&quot;/&gt;</td>
<td>&lt;died&gt;1990-01-25&lt;/died&gt;</td>
</tr>
<tr>
<td>&lt;/star&gt;</td>
<td>&lt;/star&gt;</td>
</tr>
<tr>
<td>&lt;/stars&gt;</td>
<td>&lt;/stars&gt;</td>
</tr>
</tbody>
</table>

Documents can have the same names for elements and attributes but have different semantics.
Semantic Relationships (1)

What if you wanted to do a search for information involving Automobiles (a hypernym of Car and Truck)?

Semantic Relationships (2)

What if you wanted to do a search for Coupes?

(Coupe is a hyponym of Car—it's a Car that has only 2 doors.)
Object Identity

How do we figure out that the two XML snippets describe the same person?

Architecture of the Semantic Integration Framework (1)
Architecture of the Semantic Integration Framework (2)

Layers:
- RDF Global Mediator – provides view of the data as a conceptual model
- XML Repository – provides view of homogeneous documents as a single document
- XML Local Data Source – provides the actual information

RDF Global Mediator
- Simulates an RDF repository
- Accepts RDQL queries, and returns RDQL result tables
- Provides a global ontology in the form of RDF Schema
- Keeps track of mappings between the global ontology and the local schema through mapping structures
Mapping Structures

- Owned by the RDF Global Mediator
- Bridge the gap between the global ontology (RDFS) and the local schemas (XMLS)
- Not a separate layer because they are static data structures
- Currently have to be constructed by hand

XML Repository

- Simulates a single, monolithic XML document
- Accepts XQuery expressions
- Returns DOM trees
- Handles distributed XQuery processing
- Provides a schema for its local data sources
XML Local Data Source

- Does not simulate anything; it’s the source of the data
- Can run XQuery expressions on it
- Results of XQuery (DOM tree) sent back to XML Repository
- Conforms to the schema of its XML Repository

Semantic Integration Process

- Query Translation:
  RDQL → XQuery → Distributed XQuery
- Result Transformation:
  DOM Tree → Merged DOM tree → RDQL Result Table + RDF Model
Query Translation

1. Accept RDQL query
2. Translate to XQuery
3. Process distributed XQuery for multiple local sources

Anatomy of an RDQL Query

Clauses:
- SELECT – list of variables for output
- WHERE – RDF subgraph constraints
- AND – boolean expression constraints
- USING – namespace prefixes
A sample RDQL Query

SELECT ?title, ?pub
WHERE (?book dc:title ?title),
    (?book dc:creator ?author),
    (?book dc:publisher ?pub)
    (?author foaf:name ?name)
AND   ?name eq "Neil Gaiman"
USING dc AS <http://purl.org/dc/elements/1.1/>,
    foaf AS <http://xmlns.cm/foaf/0.1/>

Graph of WHERE clause

- ?book
  - dc:title
  - dc:creator
  - dc:publisher
  - ?author
    - foaf:name
  - ?pub

- ?title
- ?name
Anatomy of an XQuery Expression

Clauses:
• let – bind variables
• for – bind variables, iterate over nodes
• where – boolean expression constraints
• return – list of variables to output

A Sample XQuery Expression

let $authors := doc("authors.xml")/authors
for $author in $authors, $name in $author/@name
  for $book in $author/book
    for $title in $book/@title,
      $pub in $book/@publisher
where $name = "Neil Gaiman"
return ($title, $pub)
Mapping of Clauses

Conceptually, the algorithm can proceed by mapping an RDQL clause with its equivalent XQuery clause(s):

- SELECT → return
- WHERE → for (multiple)
- AND → where
- USING → [none]
Mapping the WHERE Clause to For Clauses

- Need to map a graph to a tree
- Can break down an RDF graph into triples
- Can break down an XML tree into path expressions
- Map triples to path expressions using a pattern-matching technique!

Break RDF Graph into Triples
Break XML Tree into Path Expressions

A/B/C
A/B/D
A/E

Pattern Matching with the Mapping Structure

- We want to map triples to path expressions
- But we must respect the class hierarchy and the property hierarchy
- Therefore, do sub-triple matching
What Is a Sub-Triple?

• A sub-triple is a specialization of another triple.
• So (A, b, C) is a sub-triple of (X, y, Z) iff
  – A is subclass of X
  – b is subproperty of y
  – C is subclass of Z
• Example: (Painter, paints, Painting) is a subclass of (Artist, creates, Work).

Pattern Matching Example (1)

SELECT ?name, ?title, ?year
WHERE (?artist foaf:name ?name),
  (?artist art:creates ?work),
  (?work art:finishedIn ?year),
  (?work art:title ?title)
USING art AS <http://example.org/art/>
  foaf AS <http://xmlns.com/foaf/0.1/>
Perform type resolution using the global ontology (in RDF Schema)
Pattern Matching Example (4)

let $novelists := doc("novelists.xml")/novelists
for $novelist in $novelists/novelist,
    $name in $novelist/@name
for $book in $novelist/book,
    $title in $book/@title,
    $year in $book/@year
return ($name, $title, $year)

Pattern Matching Example (5)
Result Transformation

- XQuery produces DOM trees
- XML repository merges DOM trees from each local source
- Merged DOM tree is converted to RDF graph
- RDF graph from each XML repository is added to the result RDF graph

Converting DOM Tree to RDF Model

- We can reuse the Mapping Structure
- Map path expressions to RDF triples
- Based on same principles as query translation
Advantages (1)

- Layered architecture provides modularization, separation of concerns
- Query translation is fast
- Takes advantage of high-level query languages

Advantages (2)

- Provides end-to-end solution
- Extensible, more layers can be added on top
- Uses currently-available languages and tools
Disadvantages

- Result transformation is slow
- Cannot deal with semantic relationships that are not length-one paths
- Mapping structure limits the types of schemas that can be handled

Related Work (1)

- Camillo, Heuser, & Mello:
  - Global ontology uses ER variant
  - CXPath to XPath
  - Mapping views
- Amann, Beeri, Fundulaki, & Scholl:
  - Global ontology is generic ontology model
  - OQL to XQuery
  - Mapping rules
Related Work (2)

• Lakshmanan & Sadri:
  – Global ontology is generic ontology model
  – XQuery to XQuery
  – Mapping catalog

• Patel-Schneider & Siméon:
  – Global ontology is RDF Schema
  – XQuery to XQuery
  – Mapping rules

Future Work

• Complete the implementation that deals with conversion of XML data to RDF
• Use a tree regular expression structure for the mapping structure instead of a table
• Add the OWL layer on top of the current framework
What the OWL Layer Would Give Us

- OWL has more ways to express axioms, such as disjoint, union, etc.
- OWL properties can be symmetric, transitive, functional, etc.
- OWL has the sameIndividualAs property, which gives us a means to make statements about object identity

What Is It Good For? Potential Applications

- Publishing Framework
- Sensor Network
- Software Agents
- Multimedia Integration