Relational-Style XML Query

Taro L. Saito    Shinichi Morishita
University of Tokyo

June 10th, SIGMOD 2008
Vancouver, Canada

http://www.xerial.org/
If Your Manager Says ...

- I decided to start a new XML project.
- Mastering XML is crucial to our company because it is completely a new data model.
- Everybody must start learning SAX, DOM, XPath, XQuery, DTD, XML Schema, Relax NG...

- It’s a kind of tragedy...
Benefits of using XML:
- XML is a portable text-data format
- Tree-structured XML can reduce redundancy of relational data.

<table>
<thead>
<tr>
<th>Company</th>
<th>Employee</th>
<th>Office</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>e1</td>
<td>NY</td>
</tr>
<tr>
<td>1</td>
<td>e2</td>
<td>NY</td>
</tr>
</tbody>
</table>
**Problem**

- Querying relational data translated into XML
- Q: Retrieve a node tuple \((\text{Co, Emp, Office})\) from the XML data
  - e.g. XPath, a path expression query
    \(/\text{Co/Emp/Office}\)

<table>
<thead>
<tr>
<th>Co</th>
<th>Emp</th>
<th>Office</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>e1</td>
<td>NY</td>
</tr>
<tr>
<td>1</td>
<td>e2</td>
<td>NY</td>
</tr>
</tbody>
</table>

Relational Data

XML Data
A Pitfall: Structural Variations

- Tree-representation of relational data is not unique.

<table>
<thead>
<tr>
<th>Co</th>
<th>Emp</th>
<th>Office</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>e1</td>
<td>NY</td>
</tr>
<tr>
<td>1</td>
<td>e2</td>
<td>NY</td>
</tr>
</tbody>
</table>

Relational Data

- The tree representation shows that despite the relational data being the same, the tree structure can be different due to structural variations.
Inconvenience of XPath Query

- User must know the entire XML structures to produce correct path queries.
Relation in XML

- A key observation:
  - Relation is simply embedded in XML

<table>
<thead>
<tr>
<th>Co</th>
<th>Emp</th>
<th>Office</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>e1</td>
<td>NY</td>
</tr>
<tr>
<td>1</td>
<td>e2</td>
<td>NY</td>
</tr>
</tbody>
</table>
To Retrieve Relations in XML...

WHY DO WE HAVE TO USE XPATH?
Relational-Style XML Query

- Query relations in XML
  - with an SQL-like syntax
  - **SELECT Co, Emp, Office** from (XML Data)

The query statement is stable for variously structured XML data.

SQL over XML!
Problem Definition

Convert an SQL query, `SELECT A, B, C`, into an XML structure query.

- There can be many structural variations of (A, B, C)
- For $N$ nodes, there exists $N^{N-1}$ structural variations.
Amoeba

- A node tuple \((A, B, C)\) is an amoeba iff one of the \(A\), \(B\) and \(C\) is a common ancestor of the others.

- Amoeba join retrieves all amoeba structures in the XML data.
Some amoeba structure may not form a relation.

Why this structure is not allowed?

Because there are functional dependencies (FD) implied in the XML structure.
Functional Dependencies (FD)

- **FD: X -> Y** (From a given X, Y is uniquely determined)
  - employee -> office (Each employee belongs to an office)
  - office -> company (Each office belongs to a company)

- Relation in XML must have an amoeba structure corresponding to each FD.

ER-diagram (Data Model)
If FDs are ignored....

- The company has \( M \) offices, and each office has \( N \) employees:
- # of (company, office, employee) tuples:
  - When \( M = 100 \), \( N = 5 \) \( 100 \times (100 \times 5) = 50,000 \)
- While, # of correct answers is only \( M \times N = 500 \)
FD-Aware Amoeba Join

- FDs: Emp -> Office, Office -> Company
- Bottom-up construction of query results
  1. Amoeba Join (Employee, Office)
  2. Amoeba Join (Office, Company)

- FD-aware amoeba join avoids invalid XML structures.
Query Performance

- FD-aware amoeba join scales well
  - For various sizes of XML data

![Diagram showing query performance](image)
Relational query into XML query

- SELECT Co, Office, Emp
  - (with FDs: Emp -> Office, Office -> Co)

XML structures of interest are automatically determined from a relation and functional dependencies
Detecting FDs

A type of FDs required to determine XML structures to query is one-to-many (or one-to-one) relationships:

- FD: Emp -> Office
  - Each employee belongs to an office
  - An office may have several employees (one-to-many)

We can observe these relationships by counting node occurrences or directory from the ER-diagram.
First, consider

- XML := Relations + their annotations

Steps

- 1. Detect relational part from XML data
- 2. Detect one-to-many(one) relationships (FDs)
- 3. Write relational queries
  - SELECT Co, Emp, Office

Note:

- It is also possible to include annotations in query statements.
Summary of Our Contributions

- **Relation in XML**
  - Defined using amoeba structure and FDs

- **Relational-Style XML Query**
  - Retrieves relations in XML with a SQL-like query syntax (SQL over XML)
  - Allows structural variations of XML data

- **Departure from path expression queries**
  - Target XML structures are automatically determined.
Applications of Relational-Style

- (see the paper for details)

- XML Algebra
  - Based on relational-semantics
    - selection, projection, etc.

- Keys for XML
  - A key is a special-case of FDs

- Database integration

- Schema evolution

- Managing relational data enhanced with XML syntax

- A lot more…
Conclusions

“"It’s Just SQL”"
- A large number of XML data and queries are still relational.

Before going deep into the XML world, Think in Relational-Style!!!