Lecture 11:  
Relational Algebra,  
XML and Semistructured Data  
Wednesday, October 23, 2002

**Relational Algebra**

- Five operators:
  - Union: \( \cup \)
  - Difference: \( \setminus \)
  - Selection: \( \sigma \)
  - Projection: \( \Pi \)
  - Cartesian Product: \( \times \)
- Derived or auxiliary operators:
  - Intersection, complement
  - Joins (natural, equi-join, theta-join, semi-join)
  - Renaming: \( \rho \)

**Renaming**

- Changes the schema, not the instance
- Notation: \( \rho_{B_1,...,B_n}(R) \)
- Example:
  - \( \rho_{\text{LastName}, \text{SocSocNo}}(\text{Employee}) \)
  - Output schema:
    Answer(\text{LastName}, \text{SocSocNo})

**Natural Join**

- Notation: \( R_1 \bowtie R_2 \)
- Meaning: \( R_1 \bowtie R_2 = \Pi_A(\sigma_C(R_1 \times R_2)) \)
- Where:
  - The selection \( \sigma_C \) checks equality of all common attributes
  - The projection eliminates the duplicate common attributes

**XML: Outline**

XML
- Relational algebra (5.2, 5.3, 5.4)
- XML (4.6, 4.7)
  - This lecture: syntax, semistructured data
  - Next lectures: DTDs, XPath, XQuery
- Additional readings for XML:
  - www.w3.org/xml/XMLTutorial/General/book_en.html
- Main source: www.w3.org (but hard to read)
**Natural Join Example**

<table>
<thead>
<tr>
<th>Employee</th>
<th>Dependents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SSN</td>
</tr>
<tr>
<td>John</td>
<td>9999999999</td>
</tr>
<tr>
<td>Tony</td>
<td>7777777</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>Dname</th>
</tr>
</thead>
<tbody>
<tr>
<td>999999999</td>
<td>Emily</td>
</tr>
<tr>
<td>7777777</td>
<td>Joe</td>
</tr>
</tbody>
</table>

Employee \( \bowtie \) Dependents = \( \Pi_{\text{Name, SSN, Dname}}(\sigma_{\text{SSN=SSN}}(\text{Employee} \times \text{Dependents})) \)

**Natural Join**

- Given the schemas \( R(A, B, C, D) \), \( S(A, C, E) \), what is the schema of \( R \bowtie S \) ?
- Given \( R(A, B, C) \), \( S(D, E) \), what is \( R \bowtie S \) ?
- Given \( R(A, B) \), \( S(A, B) \), what is \( R \bowtie S \) ?

**Theta Join**

- A join that involves a predicate
- \( R_1 \bowtie_{\theta} R_2 = \sigma_{\theta}(R_1 \times R_2) \)
- Here \( \theta \) can be any condition

**Eq-join**

- A theta join where \( \theta \) is an equality
- \( R_1 \bowtie_{A=B} R_2 = \sigma_{A=B}(R_1 \times R_2) \)
- Example:
  - Employee \( \bowtie_{\text{SSN}=\text{SSN}} \) Dependents
- Most useful join in practice

**Semijoin**

- \( R \bowtie S = \Pi_{A_1, \ldots, A_n}(R \bowtie S) \)
- Where \( A_1, \ldots, A_n \) are the attributes in \( R \)
- Example:
  - Employee \( \bowtie \) Dependents
Semijoins in Distributed Databases

- Semijoins are used in distributed databases

### Semijoins in Distributed Databases

<table>
<thead>
<tr>
<th>Employee</th>
<th>SSN</th>
<th>Name</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Dependents</th>
<th>SSN</th>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
</table>

Employee \( \bowtie \text{Dependents} \) \( \sigma_{\text{age} > 71} \) (Dependents)

\[ R = \text{Employee} \bowtie T \]

\[ T = \Pi_{\text{ssn}} \sigma_{\text{age} > 71} \] (Dependents)

Answer = \( R \bowtie \text{Dependents} \)

---

Complex RA Expressions

\[ \Pi_{\text{name}} \]

\[ \Pi_{\text{ssn}} \]

\[ \Pi_{\text{pid}} \]

Person Purchase Person Product

\( \sigma_{\text{name} = \text{fred}} \)

\( \sigma_{\text{name} = \text{gizmo}} \)

---

Operations on Bags

A **bag** = a set with repeated elements

All operations need to be defined carefully on bags

- \( \{a,b,c\} \cup \{a,b,b,c,f,f\} = \{a,a,b,b,b,b,c,c,f,f\} \)

- \( \{a,b,b,c,c\} - \{b,c,c,c,d\} = \{a,b,b,d\} \)

- \( \sigma_c(R) \): preserve the number of occurrences

- \( \Pi_c(R) \): no duplicate elimination

- Cartesian product, join: no duplicate elimination

Important! Relational Engines work on bags, not sets!

Reading assignment: 5.3 – 5.4

---

Finally: RA has Limitations!

- Cannot compute “transitive closure”

<table>
<thead>
<tr>
<th>Name</th>
<th>Name2</th>
<th>Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>Mary</td>
<td>Father</td>
</tr>
<tr>
<td>Mary</td>
<td>Joe</td>
<td>Cousin</td>
</tr>
<tr>
<td>Mary</td>
<td>Bill</td>
<td>Spouse</td>
</tr>
<tr>
<td>Nancy</td>
<td>Joe</td>
<td>Sister</td>
</tr>
</tbody>
</table>

- Find all direct and indirect relatives of Fred

- Cannot express in RA !!! Need to write C program

---

XML

- eXtensible Markup Language

- XML 1.0 – a recommendation from W3C, 1998

- Roots: SGML (a very nasty language).

- After the roots: a format for sharing *data*

---

Why XML is of Interest to Us

- XML is just syntax for data
  
  - Note: we have no syntax for relational data
  
  - But XML is not relational: *semistructured*

- This is exciting because:
  
  - Can translate *any* data to XML
  
  - Can ship XML over the Web (HTTP)
  
  - Can input XML into any application
  
  - Thus: data sharing and exchange on the Web
XML Data Sharing and Exchange

From HTML to XML

HTML describes the presentation

XML

Bibliography

HTML

<book title="Foundations of Databases"
      author="Abiteboul, Hull, Vianu"
      publisher="Addison Wesley"
      year="1995"></book>

<book title="Data on the Web"
       author="Abiteboul, Buneman, Suciu"
       publisher="Morgan Kaufmann"
       year="1999"></book>

XML describes the content

More XML: Attributes

attributes are alternative ways to represent data

XML Terminology

- tags: book, title, author, ...
- elements: <book>...</book>, <author>...</author>
- elements are nested
- empty element: <red></red> abbrv. <red/>
- an XML document: single root element

well formed XML document: if it has matching tags
More XML: Oids and References

```xml
<person id="a555"> <name> Jane </name> </person>
<person id="a456"> <name> Mary </name> <children idref="a123 a555"> </children> </person>
<person id="a123" mother="a456"> <name> John </name> </person>
```

oids and references in XML are just syntax

More XML: CDATA Section

- Syntax: `<![CDATA[ .....any text here...]]>`
- Example:

```xml
<example>
 <![CDATA[ some text here </notAtag> <>]]> 
</example>
```

More XML: Entity References

- Syntax: `&entityname;`
- Example:
  ```xml
  <element> this is less than &lt; </element>
  ```
- Some entities:

<table>
<thead>
<tr>
<th>Entity</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;</td>
</tr>
<tr>
<td>'</td>
<td>’</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>©</td>
<td>©</td>
</tr>
<tr>
<td>°</td>
<td>°</td>
</tr>
</tbody>
</table>

More XML: Processing Instructions

- Syntax: `<?target argument?>`
- Example:

```xml
<product> <name> Alarm Clock </name>
  <?ringBell 20?>
  <price> 19.99 </price>
</product>
```

- What do they mean?

More XML: Comments

- Syntax `<!-- .... Comment text... -->`
- Yes, they are part of the data model !!!

XML Namespaces

- [http://www.w3.org/TR/REC-xml-names](http://www.w3.org/TR/REC-xml-names)
- name := [prefix:]localpart

```xml
<book xmlns:isbn="www.isbn.org/def"> 
  <title> ... </title>
  <number> 15 </number>
  <isbn:number> ... </isbn:number>
</book>
```
**XML Namespaces**

- syntactic: `<number>`, `<isbn:number>`
- semantic: provide URL for schema

```xml
<tag xmlns:mystyle="http://...">
  ...
</tag>
```

**From Relational Data to XML Data**

<table>
<thead>
<tr>
<th></th>
<th>name</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td></td>
<td>3634</td>
</tr>
<tr>
<td>Sue</td>
<td></td>
<td>6343</td>
</tr>
<tr>
<td>Dick</td>
<td></td>
<td>6363</td>
</tr>
</tbody>
</table>

**XML Data**

- XML is self-describing
- Schema elements become part of the data
  - Relational schema: persons(name, phone)
  - In XML `<persons>`, `<name>`, `<phone>` are part of the data, and are repeated many times
- Consequence: XML is much more flexible
- XML = semi-structured data

**Semi-structured Data Explained**

- Repeated attributes

```xml
<person>
  <name>Mary</name>
  <phone>2345</phone>
</person>
```

- Impossible in tables:

<table>
<thead>
<tr>
<th>name</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>2345</td>
</tr>
<tr>
<td></td>
<td>3456</td>
</tr>
</tbody>
</table>

**Semi-structured Data Explained**

- Attributes with different types in different objects

```xml
<person>
  <name>John</name>
  <phone>1234</phone>
</person>
```

- Nested collections (no 1NF)
- Heterogeneous collections:
  - `<db>` contains both `<book>`s and `<publisher>`s