Q3.1 XMLInc

• Step1: Represent in RDF/XML notation
• Step2: Get rid of “semantics” and capture only hierarchy of elements
• Step3: Form required DTD
• Step4: Make relation tables and then evaluate the given query
RDF/XML notation of the RDF graph

```xml
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example1.org/common"
         xml:base="http://example.org/here">
  <s:Person rdf:about="#John">
    <s:employee-of rdf:resource="#XMLInc"/>
    <s:client-of rdf:resource="#DB.2.0"/>
  </s:Person>

  <s:Organization rdf:about="#XMLInc">
    <s:headqrts rdf:resource="#Paris"/>
    <s:branch rdf:resource="#London"/>
  </s:Organization>
</rdf:RDF>
```
DTD Schema:

<!ELEMENT rdf:RDF (s:Person*, s:Organization*) >
<!ELEMENT s:Person (s:employee-of?, s:client-of*) >
   <!ATTLIST s:Person rdf:about CDATA #REQUIRED>

<!ELEMENT s:employee-of EMPTY>
   <!ATTLIST s:employee-of rdf:resource CDATA #REQUIRED>

<!ELEMENT s:client-of EMPTY>
   <!ATTLIST s:client-of rdf:resource CDATA #REQUIRED>

<!ELEMENT s:Organization (s:headqrts?,s:branch*)>
   <!ATTLIST s:Organization rdf:about CDATA #REQUIRED>

<!ELEMENT s:headqrts EMPTY>
   <!ATTLIST s:headqrts rdf:resource CDATA #REQUIRED>

<!ELEMENT s:branch EMPTY>
   <!ATTLIST s:branch rdf:resource CDATA #REQUIRED>
DTD for compacted form

- RDF
  - Person
    - Employee-of
    - Client-of
  - Organisation
    - headqrts
    - branch
Tables

We present only the tables which are necessary for our query. Some tables are Simplified. These two tables model the Person node and its children

<table>
<thead>
<tr>
<th>Id_Person</th>
<th>Person_about</th>
<th>Person_employee_of</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>#John</td>
<td>#XML Inc</td>
</tr>
<tr>
<td>P2</td>
<td>#Anne</td>
<td>Null</td>
</tr>
<tr>
<td>P3</td>
<td>#Harry</td>
<td>#RDF Unlimited</td>
</tr>
<tr>
<td>P4</td>
<td>#Sue</td>
<td>Null</td>
</tr>
</tbody>
</table>

These two tables model the Organization node and its children

<table>
<thead>
<tr>
<th>Id_Organization</th>
<th>Organization_about</th>
<th>Organization_headqrts</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>#XML Inc</td>
<td>#Paris</td>
</tr>
<tr>
<td>O2</td>
<td>#RDF Unlimited</td>
<td>#London</td>
</tr>
<tr>
<td>O3</td>
<td>#DB2.0</td>
<td>#Munich</td>
</tr>
</tbody>
</table>

Client-of Table

<table>
<thead>
<tr>
<th>client-of</th>
<th>Parent Id Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>P1</td>
</tr>
<tr>
<td>RDF Unlimited</td>
<td>P2</td>
</tr>
<tr>
<td>XML Inc</td>
<td>P2</td>
</tr>
<tr>
<td>DB2</td>
<td>P3</td>
</tr>
<tr>
<td>Db2</td>
<td>P4</td>
</tr>
</tbody>
</table>

Branch table

<table>
<thead>
<tr>
<th>Branch</th>
<th>Parent_Id_Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>London</td>
<td>01</td>
</tr>
<tr>
<td>Geneva</td>
<td>02</td>
</tr>
<tr>
<td>Munich</td>
<td>02</td>
</tr>
</tbody>
</table>

Person table is in foreign key relation with Organization table on Organization_about.
Cost Analysis

• Find headqrts for all companies which have at least one employee
  – On RDF graph, start from Person node (the root node) and traverse till the leaf node where headqrts present
  – Model this traversal using SQL queries on the tables

• SELECT O.Organization_headqrts
  FROM Person P, Organization O
  WHERE P. Person_employee_of = O. Organization_about

• No. of tuple accesses = 4 + 2 = 6 accesses
  – 4 tuple accesses for scanning the Person table
  – For each non-Null value an (indexed) access to the Organization table (2)
  – Assumption: Organization_about is indexed

• For large DB: $p$ is the size of the Person table and $n$ is the number of non-null entries, the cost of the query is $p + n$
Q3.2 RDF Unlimited

RDF Reification

- Statements on RDF statements (commenting, disputing, ...)

Irma Müller created the statement

Urs Giger created http://www.doc.ch/
Reification on the given RDF graph
RDF/XML notation of the reified graph

```xml
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

<rdf:description rdf:ID ="statement1" />
  <rdf:type rdf:statement />
  <rdf:Subject "John">
    <rdf:Predicate "rdf:type">
      <rdf:Object "Person">
    </rdf:Object>
  </rdf:Predicate>
</rdf:Description>

....
...
...
...
...
...

</rdf:RDF>
```
DTD

```
<rdf:statement rdf:star=* rdf:subject=subject rdf:predicate=predicate rdf:object=object />
```
RDFTable

<table>
<thead>
<tr>
<th>Id_statement</th>
<th>Statement_subject</th>
<th>Statement_predicate</th>
<th>Statement_object</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>John</td>
<td>rdf:type</td>
<td>Person</td>
</tr>
<tr>
<td>S2</td>
<td>John</td>
<td>employee-of</td>
<td>XML Inc</td>
</tr>
<tr>
<td>S3</td>
<td>John</td>
<td>client-of</td>
<td>DB2.0</td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

No. of tuples = no. of edges = 24

- Find headqrts for all companies which have at least one employee
  - On RDF graph, start from Person node (the root node) and traverse till the leaf node where headqrts present
  - Model this traversal using SQL queries on the tables

- SELECT T3.object
  FROM RDFTable T1, RDFTable T2, RDFTable T3
  WHERE
    T1.subject = “Person” AND T1.predicate=“person”
  AND
    T1.object=T2.subject AND T2.predicate=“employee-of”
Cost analysis

• Assuming indices on all attributes:
  - Select person subject
    • 4 accesses
  - First join will access 11 tuples (all statements where a person is the subject) of which 2 will be selected (which have predicate « employee-of »)
    • 11 accesses
  - Second join will access 5 tuples (all statements where org is the subject) of which finally 2 will be selected (which have predicate « headqrts »)
    • 5 accesses
  - Total: 20 accesses

• With a growing number of persons: for \( p \) Persons \( p \) accesses, each person on average in \( c \) statements a subject: \( c*p \) accesses, for \( o \) Organizations being subject in \( b \) statements \( o*b \) accesses: \( p + c*p + b*o \)

• With no index:
  - 24 subjects checked, of which 4 selected
  - 4*24 join predicates of which 2 will selected
  - 2*24 join predicates of which 2 selected
  - Total: 24 * (1+4+2) = 24*7 accesses
Tables

DG graph table DGgraph

<table>
<thead>
<tr>
<th>DgnodeId</th>
<th>Label</th>
<th>ToDgnodeId</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Person</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Client-of</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Employee-of</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Branch</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Headquarters</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>Branch</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Headquarters</td>
<td>7</td>
</tr>
</tbody>
</table>

DG instance table DGinstance

<table>
<thead>
<tr>
<th>DgnodeId</th>
<th>Oid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Person</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
</tr>
<tr>
<td>2</td>
<td>Anne</td>
</tr>
<tr>
<td>2</td>
<td>Harry</td>
</tr>
<tr>
<td>2</td>
<td>Sue</td>
</tr>
<tr>
<td>3</td>
<td>XMLInc</td>
</tr>
<tr>
<td>3</td>
<td>RDFUnlimited</td>
</tr>
<tr>
<td>3</td>
<td>DB2.0</td>
</tr>
<tr>
<td>4</td>
<td>XMLInc</td>
</tr>
<tr>
<td>4</td>
<td>RDFUnlimited</td>
</tr>
<tr>
<td>5</td>
<td>London</td>
</tr>
<tr>
<td>5</td>
<td>Geneva</td>
</tr>
<tr>
<td>5</td>
<td>Munich</td>
</tr>
<tr>
<td>6</td>
<td>Paris</td>
</tr>
<tr>
<td>6</td>
<td>London</td>
</tr>
<tr>
<td>6</td>
<td>Munich</td>
</tr>
<tr>
<td>7</td>
<td>Paris</td>
</tr>
<tr>
<td>7</td>
<td>London</td>
</tr>
</tbody>
</table>

- Index on foreign key for DGgraph (= {DgnodeId})
Query

• Find headqrts for all companies which have at least one employee
  – On RDF graph, start from Person node (the root node) and traverse till the leaf node where headqrts present
  – Model this traversal using SQL queries on the tables

• We know that we start traversing from root node (where Dgnodeld = 1)
• Select I.Oid from DGgraph G1, DGgraph G2, DGgraph G3, DGinstance I where
  G1.label = “Person” AND 
  G1.Dgnodeld = 1 AND 
  G1.ToDgnodeld = G2.Dgnodeld AND 
  G2.label = “employee-of” AND 
  G2.ToDgnodeld = G3.Dgnodeld AND 
  G3.label = “headquarters” AND 
  G3.ToDgnodeld = I.Dgnodeld
Cost Analysis

• No of accesses
  – Root node access + all with label “person”
    • 1 access
  – First join
    • 2 accesses
  – Second join
    • 2 accesses
  – Finally retrieve all headqrts from table I
    • 2 accesses
  – **Total : 7 accesses**

• Size of the DG = $d$, the two joins will always cost $i$ (outdegree in DG), the lookup cost for result tuples is exactly the number of headqrts $n$
  – $d$ and $i$ will remain bounded, even with growing number of persons and organizations

• So for large DBs this is likely the most efficient organization
Summary

• For the given DB
  – Solution 1 > Solution 3 > Solution 2

• But for growing DB sizes (more persons and more organizations)?
  – Solution 3 is likely the by far most efficient.
  – It’s even more efficient if we have index on attribute *label*. 