Relational and Semantics Views over Documents

John Snelson, Principal Engineer, MarkLogic
The start of the data modelling process
Hypertext on the Internet
Documents and Links
Relational Model

Shredded Entities

http://flickr.com/photos/14804582@N08 [CC BY 2.0]
RDF Model
Everything is a relationship
Document Model
Hierarchy, Sequences, Schema-less

- element
  - library
    - element
      - book
        - element
          - title
        - element
          - author
      - element
        - issue
        - element
          - title
          - element
            - author
MarkLogic 9 – Easier Data Query

- SEARCH
- SQL
- SPARQL

OPTIC API

DOCUMENTS (JSON OR XML)

LENSES (TEMPLATES)
{  
  "name": "10K",
  "runs": [
    {
      "name": "John",
      "age": 36
    }
  ]
}

<template>
  ...
  <context>/runs/name</context>
  ...
  <schema-name>Race</schema-name>
  <view-name>Runs</view-name>
  <columns>
    <column>
      <name>age</name>
      <scalar-type>integer</scalar-type>
      <value>../age</value>
    </column>
    <column>
      ...
    </column>
  </columns>
  ...

select age
from Race.Runs

op:from-view("Race","Runs")
  =>op:select("age")
  =>op:result()
{  
"name": "10K",
"runs": [
  {
    "name": "John",
    "age": 36
  }
]
}

<template>
  <context>/runs/name</context>
  <schema-name>Race</schema-name>
  <view-name>Runs</view-name>
  <triples>
    <triple>
      <subject>
        <val>sem:iri(xdmp:node-uri(.))<val/>
      </subject>
      <predicate>
        <val>sem:iri($EX||"hasAge"
        <val/>
      </predicate>
      <object>
        <val>xs:integer(.)/age</val>
      </object>
    </triple>
    ...  
  </triples>
  select distinct ?p {
    ?s ?p ?o
  }
  op:from-triples(
    op:pattern(op:col("s"),
               op:col("p"), op:col("o"))
  ))
  =>op:select("p")
  =>op:where-distinct()
  =>op:result()
## Tables in the Triple Index

<table>
<thead>
<tr>
<th>Triple Index Permutation</th>
<th>Major Order</th>
<th>Minor Order</th>
<th>Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td>Column ID</td>
<td>Row ID</td>
<td>Column access ordered by row ID</td>
</tr>
<tr>
<td>SOP</td>
<td>Row ID</td>
<td></td>
<td>Row access</td>
</tr>
<tr>
<td>POS <em>new</em></td>
<td>Column ID</td>
<td>Value</td>
<td>Column access ordered by value</td>
</tr>
</tbody>
</table>
Optic Engine Architecture – MarkLogic 9

D-Node Optic Engine
- Lexicon Accessor
- Triple Accessor
- Row Accessor
- Joins
- Sorting
- Aggregates and Grouping

E-Node Optic Engine
- Query planning
- Joins
- Column selection
- Triple selection
- Filter constraints
- Aggregates and Grouping
- Stand based query planning

SQL
- SQL parsing

SPARQL
- SPARQL parsing

Optic API
- Plan builtins

Compressed row / column optimizations

Plan builtins

Optic API
Templates - Demo
Benefits

Multiple Views, One Source

- Happy Business Analysts, Ontologists, etc.
- One document to update or delete
- One database to query
- Transactional
- Backups
- Replication
Benefits
Multiple Sources, One View

• Not just for different views of a single document type
• Create one view over different documents
• Unify linking styles
• Same semantics extracted from multiple document formats
Benefits

NoETL

- Keep the original content (no extract)
- Active rather than one time only transform
- No load-reload cycle
- Progressive enhancement
- Lossless modelling
  - No extract means nothing is thrown away
- Data provenance
Benefits
Data in Context

• Relational rows and triples maintain a link to their source document
• Fetch source and display or query
• Pre-filter a SQL/SPARQL/Optic query based on document context
• Provenance – check source, verify confidence, etc.
• Inherit (bi-)temporal validity of source document
• Inherit access control of source document
BI Tools: connect and extract

Reports and Analytics over relational data (tables)

- Interactive SQL/ODBC
  - Ad hoc SQL via drag and drop
  - Most general: requires only ad hoc SQL

- Data Extract
  - Batch extract of data into a relational dump
  - Most efficient, powerful; requires some programming

- Web Data Connector
  - Web page / form to define a real-time extract
  - Compromise: efficient/powerful; requires less programming
Early Access 4 – Publically available now!