Semantics for Data Integration

Stephen Buxton
Product Manager, MarkLogic
@stephenbuxton

John Snelson
Principal Engineer, MarkLogic
@jpcs
The World's Best Database for Integrating Data from Silos

RELATIONAL DATABASES

MAINFRAMES

FILE SYSTEMS

ANY OTHER SOURCE

FLEXIBLE DATA MODEL

BUILT-IN SEARCH

ACID TRANSACTIONS

Documents + Graphs

TRANSACTIONAL APPS

OPERATIONAL APPS

DOWNSTREAM SYSTEMS
Agenda

*Using Documents and Graphs for Data Integration:*

- **Data Models** – Documents + Graphs
- **7 Steps to Nirvana** – From load *as is* to a 360° view of governed data
- **Demo**
The Document Model

- Natural way to model an entity
- Self-describing / Human-readable
- Handles:
  - hierarchical data
  - repeating elements
  - sparse data
- Schema is flexible within/across documents
  - Add an element/property/column easily
  - Insert/update/delete documents in a single transaction – *even if it changes the schema*
**The Graph Model**

- Record and query relationships
  - Entity to Entity
  - Entity to Concept
  - Concept to Concept

- Most flexible data model:
  - Just add/delete a triple

- Enable Graph-style queries
  - Show me a suspect in X which is Y which is Z …
Semantics: The Graph Model

Data and Query: A Semantic Model

**Data is stored in Triples, expressed as:**

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Smith</td>
<td>livesIn</td>
<td>London</td>
</tr>
<tr>
<td>London</td>
<td>isIn</td>
<td>England</td>
</tr>
</tbody>
</table>

**QUERY with SPARQL, gives us a simple look up... and more!**

Find people who live in (a place that’s in) England

Based on W3C standards, triples are *written* using RDF (Resource Description Framework), *queried* using SPARQL, and *partitioned* into Named Graphs
Semantics Use Cases

- Semantic Search
- Metadata Hub: Managing (Digital) Assets
- Content Delivery
- Intelligence, Fraud Detection
- Data Integration
Semantics Use Cases

- Semantic Search
- Metadata Hub: Managing (Digital) Assets
- Content Delivery
- Intelligence, Fraud Detection
- **Data Integration**
Why Data Integration?

- **Query**: Provide a 360 view of a customer/patient/drug trial and related entities
  - Find what you're looking for
  - Produce accurate, insightful reports
  - Provide clean, curated data to operational processes
- **Governance**: Trust that the data you are querying is reliable
  - From a trusted source
  - Up-to-date
  - Secure
  - Fit-for-purpose
0: Out-Of-The-Box Search
Search

Data: Load as is

Index: Universal Index

Search: Show me all documents that contain "Paul"

- Load data as-is and search anywhere in any entity
- Zero Modeling, No Schema (yet)
- cf: SELECT * FROM every table WHERE some column [contains|equals] "Paul"
What if "Paul" has variants?
1: Variations in Vocabulary
Variations in Vocabulary

- "Paul" may also be "Pablo" (or "Paolo", or an alias, or …)
- Variants are recorded as Triples in a Graph (ontology)
- Show me all documents that contain "Paul" or a variant
  - Query the Triples to find all variants of "Paul"
  - Use the result to expand the query
  - cf:
  SELECT *
  FROM every table
  WHERE some column [contains|equals] some form of "Paul"

```json
{  "ID" : 1001,
  "Fname" : "Paul",
  "Lname" : "Jackson",
  "Phone" : "415-555-1212",
  "SSN" : "123-45-6789",
  "Addr" : "123 Avenue Road",
  "City" : "San Francisco",
  "State" : "CA",
  "Zip" : "94111"  }
```

```json
{  "Customer_ID" : 2001,
  "Given_Name" : "Karen",
  "Family_Name" : "Bender",
  "Shipping_Address" : {  
    "Street" : "324 Some Road",
    "City" : "San Francisco",
    "State" : "CA",
    "Postal" : "94111",
    "Country" : "USA"  }
}
```
Semantic Search
Context from a graph, for better search over WHAT

- Synonyms: Another word for "trust deed" is "trust agreement"
- SubClasses: Every Henley is a Shirt
- Other:
  - Joe Smith alias Fred Bloggs
  - John Matthews member of TEHC
  - John Cleese similar to Peter Cook
  - Catheter requires a sterile environment
- Domain-specific:
  - Use one or more graphs to provide a domain-specific lens
  - E.g. “LA Basin” is the same as “LA Area” for some purposes
Semantic Search
Why do this with graphs?

- Flexible: Easy to add/remove a single triple (no modeling required!)
- Relationships between values are a graph (an artifact)
  - Centrally manage, query, visualize
  - Graph-style queries: A catheter is an embeddable device which requires a sterile environment
- Data is sacrosanct: Iteration means changing the graph, not the data
  - Original data is always available; no updates/re-indexing required
- Iterative: Define triples just for the integration you need, and iterate
- Graphs can be layered to present domain- or task-based lenses
- Separation of concerns: Ontology builder is separate from query builder
How do I search just in zip code?
2: Variations in Structure
Semantic Query Layer: Example

{  
  "ID" : 1001 ,
  "Fname" : "Paul" ,
  "Lname" : "Jackson" ,
  "Phone" : "415-555-1212" ,
  "SSN" : "123-45-6789" ,
  "Addr" : "123 Avenue Road" ,
  "City" : "San Francisco" ,
  "State" : "CA" ,
  "Zip" : 94111  
}  

{  
  "Customer_ID" : 2001 ,
  "Given_Name" : "Karen" ,
  "Family_Name" : "Bender" ,
  "Shipping_Address" : {
    "Street" : "324 Some Road" ,
    "City" : "San Francisco" ,
    "State" : "CA"
  } ,
  "Postal" : "94111" 
  "Country" : "USA" }  

Semantic Query Layer: Example

Triples

CANONICAL
IS THE SAME AS
CANONICAL
IS THE SAME AS
CANONICAL
IS THE SAME AS
CANONICAL ZIP
IS THE SAME AS

Named Graphs

:sourceA

ZIP

:sourceB

POSTAL

{   "ID" : 1001 ,
    "Fname" : "Paul" ,
    "Lname" : "Jackson" ,
    "Phone" : "415-555-1212" ,
    "SSN" : "123-45-6789" ,
    "Addr" : "123 Avenue Road" ,
    "City" : "San Francisco" ,
    "State" : "CA" ,
    "Zip" : 94111 }

{   "Customer_ID" : 2001 ,
    "Given_Name" : "Karen" ,
    "Family_Name" : "Bender" ,
    "Shipping_Address" : {
        "Street" : "324 Some Road" ,
        "City" : "San Francisco" ,
        "State" : "CA" ,
        "Postal" : "94111" ,
        "Country" : "USA" } }
Semantic Query Layer: Description

Data mapping information is in a graph; better search over WHERE

- Recall Semantic Search: Better search over WHAT (values)
- Semantic Query Layer: Better search over WHERE (locations, columns)
- Same Field: Look in "Zip" and "Postal" and …
  - Optionally tie field names to sources via collections
- More complex mappings are possible
  - e.g. include a function call in the graph
    "fullname maps to (firstname||lastname)"
Semantic Query Layer: Benefits

- Flexible: To add a new source, add to the graph and the next query will reflect the change
  - You can have layered graphs in place; choose one or more at query-time
- The Semantic Query Layer is a graph (an artifact)
  - Centrally manage, query, visualize
- Data is sacrosanct: Data Integration means only changing the graph, not the data
  - Original data is always available; no updates/re-indexing required
- Iterative: Define triples just for the integration you need, and iterate
How do I record the canonical model of an entity? (such as a Customer)
3: Canonical Model of an Entity
Entity Type Model

- Entities
  - A Customer is something that exists as part of my business/mission

- Properties
  - Customer entities have a Name that is of type string and is required

- Relationships
  - Customers place Orders
Extending the Entity Type Model

- Governance
- Provenance
- …anything else

Finite core

Extend with Semantics
The Entity Type Model: Description

Double-click on Entity Services: The Entity Model

- Create an Entity Model Descriptor (a document)
  - Manual: create a JSON document and insert with a magic collection
  - GUI: create an Entity in the Data Hub Framework
- Model Descriptor is automagically indexed as triples
  - Query the model as a graph; link to other graphs
- Once you have a model, you can (auto-)generate artifacts
  - Indexes
  - Transformation code
  - Security
The Semantic Model: Example

Entity Model Descriptor (JSON) + Derived Graph

```json
{
  "info": {
    "title": "Customer",
    "version": "0.0.1",
    "baseUri": "http://marklogic.com/SemIntegrationDemo",
    "description": "Customer contact details"
  },
  "definitions": {
    "Customer": {
      "properties": {
        "id": {
          "datatype": "string",
          "description": "Customer ID"
        },
        "familyName": {
          "datatype": "string",
          "description": "Family name"
        },
        "givenName": {
          "datatype": "string",
          "description": "Given name"
        }
      }
    }
  }
}
```

```xml
@prefix xs: <http://www.w3.org/2001/XMLSchema#> .
@prefix p1: <http://marklogic.com/entity-services#> .
@prefix p2: <http://marklogic.com/Demo/Customer-0.0.1/Address/> .
@prefix p3: <http://marklogic.com/Demo/Customer-0.0.1/PhoneNumber/> .

p2:zip a p1:Property ,
  p1:RequiredProperty ;
  p1:description "Postal / zip code"^^xs:string ;
  p1:datatype xs:string ;
  p1:title "zip"^^xs:string .

p3:number a p1:Property ,
  p1:RequiredProperty ;
  p1:description "Phone number"^^xs:string ;
  p1:datatype xs:string ;
  p1:title "number"^^xs:string .
```
The Semantic Model: Example
Entity Model Descriptor (JSON) + Derived Graph

```json
{
  "info": {
    "title": "Customer",
    "version": "0.0.1",
    "baseUri": "http://marklogic.com/SemIntegrationDemo",
    "description": "Customer contact details"
  },
  "definitions": {
    "Customer": {
      "properties": {
        "id": {
          "datatype": "string",
          "description": "Customer ID"
        },
        "familyName": {
          "datatype": "string",
          "description": "Family name"
        },
        "givenName": {
          "datatype": "string",
          "description": "Given name"
        }
      }
    }
  }
}
```
The Semantic Model: Querying the Model

Because the Model is available as a Graph, you can:

- Apply Graph-style queries
  - In the Customer model, show me the name and description of each property* that is Range Indexed
- Link to other graphs and apply graph-style queries
  - For all entities that are a sub-type of Person, show me the name and description of each property* that is Range Indexed, and is subject to Business Rule X or Security Policy Y
- Easily extend the Model
  - Add Triples
  - Link to another Graph
The Semantic Model: Benefits

- Flexible: the Model is easy to change and extend
- The Model for each Entity Type is "written down" in one place (an artifact)
  - Model is managed, versioned, secured, queried, visualized
- Model can be used to manage the creation of entity instances
  - … and other artifacts
- Model can be used to robustly create Services
I want to query over transformed data, but I don’t want to change the data
4: Querying Transformed Data
Transformed Data
Transforming data in the index

- Data transformation/manipulation *in the index*
  - Pull out a phone number from a complex field
  - Source has firstname and lastname; I want to query for fullname

- Data manufacturing *in the index*
  - Fill in default values for a missing field
  - Add pre-calculated fields such as sum, average
Templates: Index-level Integration
Faster Semantic Data Layer without changing the data

Query-level integration:
Query the Model, expand the query

Data-level integration:
Harmonize the data
Templates: Index-level Integration
Faster Semantic Data Layer without changing the data

Query-level integration:
Query the Model, expand the query

Index-level integration:
Project out canonical, transformed data into the index

Data-level integration:
Harmonize the data

Most Agile

Fastest Queries
- Rich hierarchy
- Flexible schema
- Natural fit for messages, forms, objects

Template Driven Extraction

- Project out of trees into
  - Triples
  - Tuples
- Transactionally at index time
- Declarative mapping, functional transform

Triple Index

- Index optimized for joins, aggregates
- Shared by SPARQL, SQL, and Optic
Parts of my model are stable. I want to push down to the document.
5. Harmonize Documents
Harmonize Documents

- Most flexible – transformations in code
- Faster indexing
  - Transformations are pre-calculated
- Simpler queries
  - Query directly (only) against the document
- Visibility: All the data is together
Data Harmonization

- Harmonize data with different schemas in order to create a common view:
  E.g., Standardizing “male” and “female” versus, “m” and “f” or “M” and “F”
- Only harmonize what you need to
- Preserve and query data and metadata as is
- Update the model later without re-ingesting
- New data and schemas are okay
- See also: Data Hub Framework
How do I know I can trust the data I'm querying? Where did it come from?
6: Provenance
Provenance
Add a Provenance Graph to an Entity

- Provenance tells me where this entity came from
  - … and where that came from, and where that came from...
- Provenance data is naturally Graph-shaped
- Queries over Provenance are generally Graph-style queries
Provenance Graph

2018-01-10 11:45
RAN AT

GENERATED BY
Harmonize-1

APPROVED BY

CONSUMED

CONSUMED

FROM

FROM

Source-1

Source-2

DEPT
Eng-1

ROLE
Sr. Data Scientist

FROM

ENVELOPE PATTERN

SCHEMA 1

SCHEMA 2

SCHEMA 3

© 2018 MARKLOGIC CORPORATION
So much for a single Entity. How can I query across Entities?
7: Linking Entities
The (Extended) Semantic Model

- Use the Entity Type Model to generate:
  - Code to transform from raw source to an instance of the model
  - TDE Templates to expose data as triples, views
  - Config settings (such as indexes)
  - REST endpoints for APIs

- Use the extended Entity Type Model to create:
  - Provenance
  - Links to other entity instances
  - Entity usage
  - Security settings
  - Special-purpose materializations
SLIDE: 51

- **LOAD DATA AS IS**
  - **ONT/OLOGY**
    - **SEMANTIC QUERY LAYER**
      - **ENTITY TYPE MODEL**
        - **TEMPLATE**
          - **HARMONIZE DOCUMENTS**
            - **ENTITY TYPE MODEL EXTENSIONS**

- **OUT-OF-THE-BOX SEARCH**
  - **QUERY-BASED INTEGRATION**
    - **CENTRALLY-MANAGED, QUIERIABLE MODEL**
      - **DOCUMENT-BASED INTEGRATION**
        - **GOVERNANCE: SECURITY, PROVENANCE**
          - **SPECIAL-PURPOSE MATERIALIZATION**
            - **ENTITY TYPE MODEL EXTENSIONS**
Semantics for Data Integration

Stephen Buxton
Product Manager, MarkLogic
@stephenbuxton

John Snelson
Principal Engineer, MarkLogic
@jpcs