Contents

• We integrate through a set of tools in our triple store
• A three minute introduction to triple stores
• Data integration with Linked Open Data [Demo]
• Can we do this integration in the RDB world?
  – Pfizer, BC
• Our current process for organic data integration
  – Vocabularies, Thesauruses, Taxonomy, Ontologies
  – Schema Spaces
  – RDFy-ing your data (kind of ETL)
  – Matching your data and building an inverted metadata instance store
  – Querying
Graphs, triples, triple-store?

createTripleStore("seminar.db")

addTriple (Person1 first-name Steve)
addTriple (Person1 isa Organizer)
addTriple (Person1 age 52)
addTriple (Person2 first-name Jans)
addTriple (Person2 isa Psychologist)
addTriple (Person2 age 50)
addTriple (Person3 first-name Craig)
addTriple (Person3 isa SalesPerson)
addTriple (Person3 age 32)

addTriple (Person1 colleague-of Person2)
addTriple (Person1 colleague-of Person3)

addTriple (Person1 likes Pizza)
Keep adding New Relationships

addTriple ( Person3 neighbor-of Person1)
addTriple ( Person3 neighbor-of Person2)

addTriple ( Person3 !o:lives-in !o:Place1111)
addTriple ( Place1111 !o:name !"Moraga")
addTriple ( Place1111 !o:latitude !"37.12223")
addTriple ( Place1111 !o:longitude !"-122.4325")
Apply Logic – Infer New Relationships

addTriple (first-name domain Person)

Every thing that has a first name must be a person
SELECT ?x ?y WHERE {
  ?x simple:family-name 'Steve'.
}

Enter a SPARQL SELECT or DESCRIBE query to the left and press the Do Query button. All known namespace abbreviations will be in effect. Or press the Prolog radio button and enter a Prolog query instead (perhaps with additional lisp forms as well).

Click a node to visit that resource or literal in the table view and add the node to the graph view, connecting it to other nodes by the current predicates. Shift-click a node to only add the node to the graph. Control-click a node to only visit the resource in the table view. Control-shift-click a URL to visit it in your web browser. Control-click a predicate cell to toggle whether that predicate is a current predicate. Right-click anywhere to go back. Control-right-click a cell to copy a URL to the clipboard. Click a column header cell to sort the table by that column. Shift-click a column header cell to lock its column.

Explicit Nodes from Query
**Steve**

Explicit Predicates from Query
Colleague-of
First-name

Query COMPLETED with two results.
Edit query

Query language: SPARQL

select ?x ?p ?o where {
}

Result

<table>
<thead>
<tr>
<th>?x</th>
<th>?p</th>
<th>?o</th>
</tr>
</thead>
<tbody>
<tr>
<td>1127</td>
<td>scis_described_in</td>
<td>11685242</td>
</tr>
<tr>
<td>1127</td>
<td>rdfs:label</td>
<td>&quot;pGEX-2T-NM&quot;</td>
</tr>
<tr>
<td>1127</td>
<td>rdfs:subClassOf</td>
<td>scis:synthetic_plasmid</td>
</tr>
<tr>
<td>1127</td>
<td>sciraries_sequence_described_by</td>
<td>651752</td>
</tr>
<tr>
<td>1127</td>
<td>scis:availability_described_by</td>
<td>pgve1?f=c&amp;attag=b&amp;cmd=findp&amp;identifier=1127</td>
</tr>
<tr>
<td>1394</td>
<td>scis_described_in</td>
<td>7592789</td>
</tr>
</tbody>
</table>
The Web Applications Working Group invites implementation of the Candidate Recommendation of Widget Packaging and Configuration. This specification standardizes a packaging format for software known as widgets. Widgets are client-side applications that are authored using Web standards, but whose content can also be embedded into Web documents. The packaging format acts as a container for files used by a widget. The configuration document is an XML vocabulary that declares metadata and configuration parameters for a widget. The steps for processing a widget package describe the expected behavior and means of error handling for runtimes while processing the packaging format, configuration document, and other relevant files. The group plans to track implementations in an implementation report. Learn more about the Rich Web Client Activity.
Semantic Web

In addition to the classic “Web of documents” W3C is helping to build a technology stack to support a “Web of data,” the sort of data you find in databases. The ultimate goal of the Web of data is to enable computers to do more useful work and to develop systems that can support trusted interactions over the network. The term “Semantic Web” refers to W3C’s vision of the Web of linked data. Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data. Linked data are empowered by technologies such as RDF, SPARQL, OWL, and SKOS.

Linked Data
The Semantic Web is a Web of data — of dates and titles and part numbers and chemical properties and any other data one might conceive of. RDF provides the foundation for publishing and linking your data. Various technologies allow you to embed data in documents (RDFa, GRDDL) or expose what you have in SQL databases, or make it available as RDF files.

Vocabularies
All this data is important or valuable to organize data. Using OWL (to build vocabularies, or “ontologies”) and SKOS (for designing knowledge organization systems) it is possible to enrich data with additional meaning, which allows more people (and more machines) to do more with the data.

Inference
Near the top of the Semantic Web stack, one finds inference — reasoning over data through rules. W3C works on rules, primarily through RIF and OWL, is focused on translating between rule languages and

Vertical Applications
W3C is working with different industries — for example in Health Care and Life, scientific, Government, and Energy — to improve collaboration, research and development, and innovation adoption
Demoing Data Integration over a federation of 11 linked data sets

• We took 5 public databases: Drugbank, Dailymeds, Clinical trials, Diseasesome, and Sider. Entities are mostly linked together through same-as relationships.

• And using some entity extraction created some more databases
  – CT-discusses-drug,
  – CT-discusses-side-effect
  – CT-discusses-target,
  – CT-discusses-disease

• With some help from Alitora entity extraction on Rheumatoid Arthritis
  – CT-mentions-genes

• And to facilitate search through schema space: Schema-connections
Interesting queries

- **Sparql**
  - Give me the title of all clinical trials that discuss the drug Lipitor and the side-effect “Diabetes type 2”
  - Give me clinical trials that discuss Rheumatoid Arthritis and give me the genes and drugs discussed

- **Prolog**
  - Find all clinical trials that resemble clinical trial NCT00130091 given diseases, drugs, targets, and side-effects
Can we do this kind of integration in the Relational Database World?
Knowledge Sharing using Semantic Technologies

February 25, 2010

Vijay K. Bulusu
Sr. Manager
R&D Informatics
Knowledge Sharing via imprecise connections

- **Goal**
  - Identify and aggregate data from various sources in the absence of unique identifiers and lack of referential integrity

- **Challenges**
  - Incompatible databases
    - Same name; different meaning (Batch / Lot Information)
  - Imprecise Connections
    - Lack of controlled vocabulary for key fields
    - One identifier mapped to multiple entities
Business Problem #1

- **Compound Purity Verification**
  - For release results in LIMS for a clinical batch with x% specified/unspecified impurity, FDA wants confirmation on the integration of the peak in the CDS and the calculations of standards and samples to get the final result.
  - For an impurity value recorded in LIMS (Certificate of Analysis or Stability Report), find the corresponding impurity value in Empower.

- **Challenges**
  - No common identifiers between LIMS and Empower
  - Limited data stored in Empower (Historical data archived to storage)
Business Problem #1

1. Data from LIMS and CDS are transformed via ontologies in context of multiple imprecise connections.

2. SPARQL queries across linked data provide rich pattern-matching capabilities.

3. CDS datasets are identified that provide ranked matching to LIMS Reports, for reproducible report verification.
ETL Overview

- Relational DBMS/Data Sources
  - Basic Transformation
  - Query-Based Transformation
  - Stored Procedures + Web Services
  - SPARQL against RDBMS

- Provenance
- Versioning
- Governance
- Meaning

- Federated Semantic Datastore(s)
A Common Pattern

- You have multiple Business Units (Hardware, Software, Services, Applications) that sell all to the same customers.
- Each BU ‘result responsible’, so has most efficient set of databases to support own business:
  - Customers, contracts, software/hardware versioning, configurations, inventory.
- Only few cross company databases:
  - ERP for accounting and to track sales
  - Customer Care and Trouble Tickets Databases
  - SLA
Common problems

- Same customer might be in 40 different databases with different customer contacts and account managers, different location addresses and billing addresses.
- Same hardware and software product referenced in many databases, sometimes with different names
- Customers use collections of hardware and software products with different configuration (parameters)
- Inventories discoupled from bill of materials discoupled from customer demand discoupled from problem tickets discoupled from SLA contracts.
• CFO Citigroup: how much do I spend in total with you?
  – yes, he has the same problem 😊
• Sales person: I’m going to sell this video equipment to this company, the customer already has this software/hardware/services configuration, can we expect problems
  – well, après nous le deluge 😊
• How much do I have to keep in stock given the current rate of problems and the customers that have this in their configuration.
  – Currently we keep *10, just to be sure 😞
1. Semantified Schema’s  
2. Vocabularies, Thesauri, Taxonomies  
3. Product and customer ontologies  
4. Customer -> DB links  
5. Product -> DB links  
6. Customer/product aggregations
Traditional Approach: Top Down

- Master Data Management
- Virtual or Federated Database Management
- Think it all out beforehand,
- Heavy Weight,
- Changes are very costly
Semantic Tech Approach: Bottom up

- Use vocabularies, thesauruses, taxonomies, ontologies
- Translate data into triple stores
- Or query original DB with SPARQL

- Lazy, Late binding
- Organic, Evolving
- Very flexible
- Better suited to ad hoc
Step 1: Vocabularies, Thesauri, Taxonomy, Ontologies

- **Vocabularies**: the heart of linking
  - `bc:Citi rdf:type bc:VocabularyEntry`

- **Thesauri**: linking variants to Vocabulary
  - `bc:Citi bc:hasAlternativeName 'Citi Group'`

- **Taxonomy**: finding the hierarchy in your data
  - `bc:Banamex bc:part of bc:Citi`

- **Ontology**: types, subtypes, constraints
  - `bc:Citi rdf:type bc:Bank`
  - `bc:Bank rdf:type owl:Class`
  - `bc:Bank rdfs:subClassOf bc:Company`
  - `bc:Company rdfs:subClassOf bc:Organization`
Step 2: Schema Spaces

- Create Schema Connection Spaces
  - Take original RDB schemas and syntactically transform to RDF and RDFS
    - `bc:customer1 rdf:type bc:table`  
    - `bc:customerID1 rdf:type bc:columnName`  
    - `bc:customerID1 bc:dataType bc:long`  
  - Annotate with origin
    - `bc:customer1 bc:fromDB bc:ERP1`  
  - Annotate with connections to other schema
    - `bc:customer1 bc:relatesTo bc:customer2`
Specify database connection

- Database: employees
- User: agraph
- Password: ********
- Host: localhost

OK  Cancel
Table employees

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
<th>Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>row:employees/employees.emp_no</td>
<td>rel:employees/birth_date</td>
<td>&quot;employees.birth_date&quot;^^xsd:date</td>
<td>[default graph]</td>
</tr>
<tr>
<td>row:employees/employees.emp_no</td>
<td>rel:employees/first_name</td>
<td>&quot;employees.first_name&quot;^^xsd:string</td>
<td>[default graph]</td>
</tr>
<tr>
<td>row:employees/employees.emp_no</td>
<td>rel:employees/last_name</td>
<td>&quot;employees.last_name&quot;^^xsd:string</td>
<td>[default graph]</td>
</tr>
<tr>
<td>row:employees/employees.emp_no</td>
<td>rel:employees/gender</td>
<td>&quot;employees.gender&quot;^^xsd:string</td>
<td>[default graph]</td>
</tr>
<tr>
<td>row:employees/employees.emp_no</td>
<td>rel:employees/hire_date</td>
<td>&quot;employees.hire_date&quot;^^xsd:date</td>
<td>[default graph]</td>
</tr>
</tbody>
</table>
Please provide a definition for the namespaces rel and row.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>xs</td>
<td><a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2001/XMLSchema#</a></td>
</tr>
<tr>
<td>row</td>
<td><a href="http://www.example.com/empdb#">http://www.example.com/empdb#</a></td>
</tr>
<tr>
<td>owl</td>
<td><a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a></td>
</tr>
<tr>
<td>rel</td>
<td>...</td>
</tr>
<tr>
<td>err</td>
<td><a href="http://www.w3.org/2005/xqt-errors#">http://www.w3.org/2005/xqt-errors#</a></td>
</tr>
<tr>
<td>fn</td>
<td><a href="http://www.w3.org/2005/xpath-functions#">http://www.w3.org/2005/xpath-functions#</a></td>
</tr>
<tr>
<td>rdfs</td>
<td><a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a></td>
</tr>
<tr>
<td>xsd</td>
<td><a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2001/XMLSchema#</a></td>
</tr>
<tr>
<td>rdf</td>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a></td>
</tr>
</tbody>
</table>
Subject
Row ID
Table: titles

Predicate
Column Name
Column: titles.emp_no

Object
Row ID
Table: employees

Graph
Default graph
Step 3: RDFy data
ETL Overview

Relational DBMS/Data Sources

- Basic Transformation
- Query-Based Transformation
- Stored Procedures + Web Services
- SPARQL against RDBMS

Provenance  Versioning  Governance  Meaning

Federated Semantic Datastore(s)
Step 4: match entities

Entity Resolution

- Is this the same address
- Find same products
- Is this the same company
- Is this the same person
Step 5: inverted database

- bc:Citi hasPart bc:Banamex
- bc:Banamex bc:inDB bc:ERP/customer/name
Thanks!