Scaling the Semantic Wall with AllegroGraph and TopBraid Composer

A Joint Webinar by TopQuadrant and Franz

Dean Allemang – Chief Scientist, TopQuadrant Inc.

Jans Aasman – CTO, Franz Inc.
This Seminar

- Part 1 (45 min) – Dean Allemang
  - The basics of RDF and Triples
  - RDFS and Classes, Properties
  - AllegroGraph as a triple-store
  - Integration of TopBraid Composer with AllegroGraph

- Part 2 (15) Jans Aasman
  - Scalable deployment with AllegroGraph
What is RDF? Distribution of data

<table>
<thead>
<tr>
<th>ID</th>
<th>Model No.</th>
<th>Division</th>
<th>Product Line</th>
<th>Manufacturer location</th>
<th>SKU</th>
<th>In Stock</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ZX-3</td>
<td>Manufacturing support</td>
<td>Paper machine</td>
<td>Sacramento</td>
<td>FB3524</td>
<td>23</td>
</tr>
<tr>
<td>2</td>
<td>ZX-3P</td>
<td>Manufacturing support</td>
<td>Paper machine</td>
<td>Sacramento</td>
<td>KD5243</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>ZX-3S</td>
<td>Manufacturing support</td>
<td>Paper machine</td>
<td>Sacramento</td>
<td>IL4028</td>
<td>34</td>
</tr>
<tr>
<td>4</td>
<td>B-1430</td>
<td>Control Engineering</td>
<td>Feedback Line</td>
<td>Elizabeth</td>
<td>KS4520</td>
<td>23</td>
</tr>
<tr>
<td>5</td>
<td>B-1430X</td>
<td>Control Engineering</td>
<td>Feedback Line</td>
<td>Elizabeth</td>
<td>CL5934</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>B-1431</td>
<td>Control Engineering</td>
<td>Active Sensor</td>
<td>Seoul</td>
<td>KK3945</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>DBB-12</td>
<td>Accessories</td>
<td>Monitor</td>
<td>Hong Kong</td>
<td>ND5520</td>
<td>100</td>
</tr>
<tr>
<td>8</td>
<td>SP-1234</td>
<td>Safety</td>
<td>Safety Valve</td>
<td>Cleveland</td>
<td>HI4554</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>SPX-1234</td>
<td>Safety</td>
<td>Safety Valve</td>
<td>Cleveland</td>
<td>OP5333</td>
<td>14</td>
</tr>
</tbody>
</table>
### Distribute by rows?

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ZX-3</td>
<td>Manufacturing support</td>
<td>Paper machine</td>
<td>Sacramento</td>
<td>FB3524</td>
</tr>
</tbody>
</table>

Needs common schema - which column is which?

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>B-1430</td>
<td>Control Engineering</td>
<td>Feedback Line</td>
<td>Elizabeth</td>
<td>KS4520</td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>DBB-12</td>
<td>Accessories</td>
<td>Monitor</td>
<td>Hong Kong</td>
<td>ND5520</td>
</tr>
<tr>
<td>Model No.</td>
<td>In Stock</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZX-3</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZX-3P</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZX-3S</td>
<td>34</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B-1430</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B-1430X</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B-1431</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBB-12</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP-1234</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPX-1234</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Needs to reference entities – which thing are we talking about?
Distribute by cells!?

<table>
<thead>
<tr>
<th>Division</th>
<th>Accessories</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID7</td>
<td></td>
</tr>
</tbody>
</table>

Needs to reference both schema and entities

Most flexible – can distribute data in any way at all!

<table>
<thead>
<tr>
<th>Product Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Feedback</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ZX-3</td>
</tr>
</tbody>
</table>
Distribute by cells!?

Subject

ID7 | Division
--- | ---
Accessories

Predicate

Object

URI’s

• Store, Index, and Federate these triples

AllegroGraph
Type information for data? e.g.,
- Sacramento – Capital
- Control Engineering – Division
- Monitor – Product Line

Related types?
- Capital < City < Location
- Division < Profit Center < Organizational Unit
Defining types
  - rdfs:Class

Defining relationships
  - rdf:type, rdfs:subClassOf, rdfs:subPropertyOf

Relating Classes to Properties
  - rdfs:domain
  - rdfs:range
### Type information as Triples

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sacramento</td>
<td>rdf:type</td>
<td>Capital</td>
</tr>
<tr>
<td>Control Engineering</td>
<td>rdf:type</td>
<td>Division</td>
</tr>
<tr>
<td>Monitor</td>
<td>rdf:type</td>
<td>Product</td>
</tr>
<tr>
<td>Capital</td>
<td>rdfs:subClassOf</td>
<td>City</td>
</tr>
<tr>
<td>City</td>
<td>rdfs:subClassOf</td>
<td>Location</td>
</tr>
<tr>
<td>Division</td>
<td>rdfs:subClassOf</td>
<td>Profit Center</td>
</tr>
<tr>
<td>Profit Center</td>
<td>rdfs:subClassOf</td>
<td>Org Unit</td>
</tr>
</tbody>
</table>
The marriage between
- Object oriented type system
- Well understood Description logic
- Web languages like XML and RDF

Typical reasoning
- Class membership
- Equivalence of classes
- Consistency
- Classification
TopBraid Composer

- Environment for Viewing, Managing, Editing RDFS and OWL graphs
- Features (among others):
  - View class hierarchy as outline
  - Graph view
  - Instance counts
  - Integrated inferencing
  - Source management (e.g., version control)
Kennedy family information
Genealogy, schools, posts, etc. Represented as triples
Merge information from multiple sources
Utilize a variety of inferencers
Federated queries
Fetch information from the spreadsheet (colleges1:), combine it with information from the AllegroGraph (?kenu), and construct a triple that combines the information. Two universities match if they have the same name (?nn=?kn).

CONSTRUCT {?kenu colleges1:state ?state}
WHERE {?kenu rdfs:label ?kn .
    ?newu colleges1:_name ?nn .
    ?newu colleges1:state ?state .
FILTER (xsd:string (?nn) = xsd:string (?kn))}
Find kennedys (include first name and last name) who went to school in NY

```
SELECT ?kennedy ?fn ?ln
WHERE {?kennedy simple:alma-mater ?u .
  ?u colleges1:state "NY"^^xsd:string}
```
AllegroGraph is

- A scalable persistent triple store
  - 1.1 Billion triples in 23 hours on a $5000 dollar box
  - 20 to 40,000 triples per second,
  - Record query performance on LUBM benchmark queries.
- Based on standards
  - RDF, RDFS, OWL, SPARQL, Named Graphs
- Two modes of working
  - Standalone for analytics
  - Client/Server for real time services
- Accessible from any language
  - Java: we adhere to Sesame and Jena remote repository APIs
  - .Net, Python, Ruby, Lisp, C through REST interface
- Reasoning
  - Prolog, RDFS++ and Description Logics (direct connection with Racer)
- GUI & Ontology Management
  - TopBraid Composer, RacerPorter
AllegroGraph Unique Features

- RDFS++ Reasoner
- Direct reification
  - Triples point to triples
- Named Graphs fully supported
  - But slot can also be used for weights, trust factors, provenance, distance, etc.
- Native data types and efficient range queries
  - Existing triple stores store all data as strings, range queries inefficient
  - AllegroGraph supports most xml schema types (dates, times, longitudes, latitudes, durations, telephone numbers, etc)
- Basic geospatial and temporal primitives
- Social Network Analysis library
- Combine it all with Prolog & Sparql
Why an AllegroGraph reasoner?

- Full description logics
  - Good at handling (complex) ontologies
  - Complete but unpredictable time complexity when the number of individuals increase beyond millions

- AllegroGraph does
  - All of RDFS
  - Most of OWL
  - Nearly complete but predictable, fast performance