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.

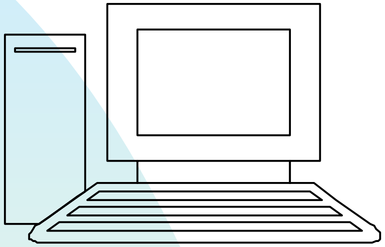
This Seminar

- 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

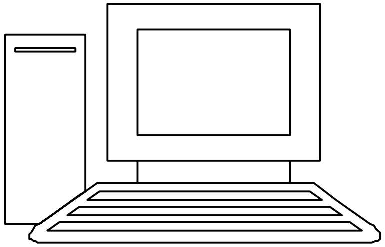
ID	Model No.	Division	Product Line	Manufacture location	SKU	In Stock
1	ZX-3	Manufacturing support	Paper machine	Sacramento	FB3524	23
2	ZX-3P	Manufacturing support	Paper machine	Sacramento	KD5243	4
3	ZX-3S	Manufacturing support	Paper machine	Sacramento	IL4028	34
4	B-1430	Control Engineering	Feedback Line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

Distribute by rows?

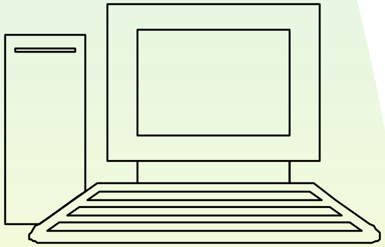


1	ZX-3	Manufacturing support	Paper machine	Sacramento	FB3524	23

Needs common schema - which column is which?



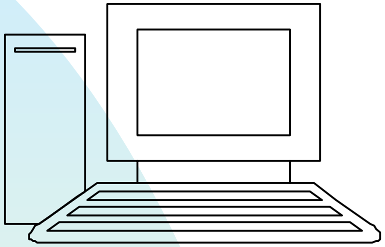
4	B-1430	Control Engineering	Feedback Line	Elizabeth	KS4520	23



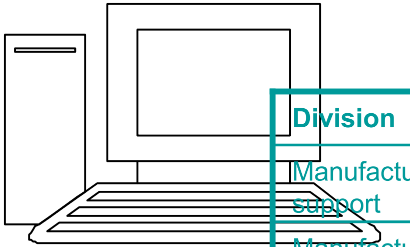
7	DBB-12	Accessories	Monitor	Hong Kong	ND5520	100

Distribute by columns?

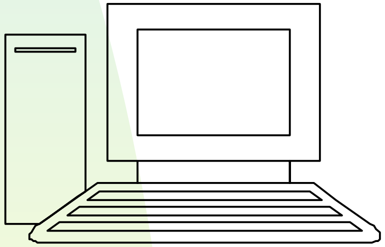
Model No.
ZX-3
ZX-3P
ZX-3S
B-1430
B-1430X
B-1431
DBB-12
SP-1234
SPX-1234



Needs to reference entities – which thing are we talking about?



Division
Manufacturing support
Manufacturing support
Manufacturing support
Control Engineering
Control Engineering
Control Engineering
Accessories
Safety
Safety



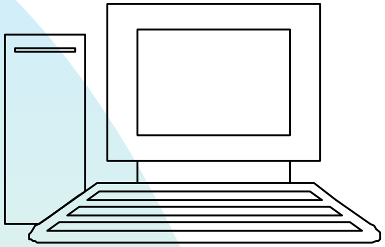
In Stock
23
4
34
23
14
0
100
4
14

Distribute by cells!?

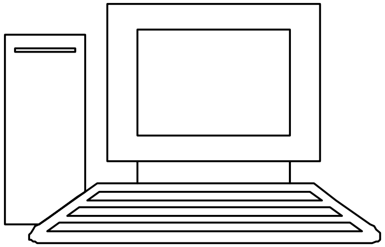
	Division
ID7	Accessories

Needs to reference both schema and entities

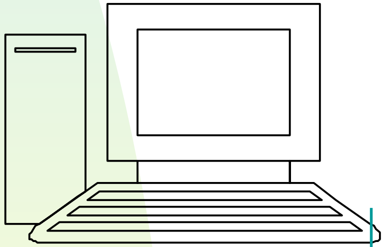
Most flexible – can distribute data in any way at all!



	Division
ID7	Accessories

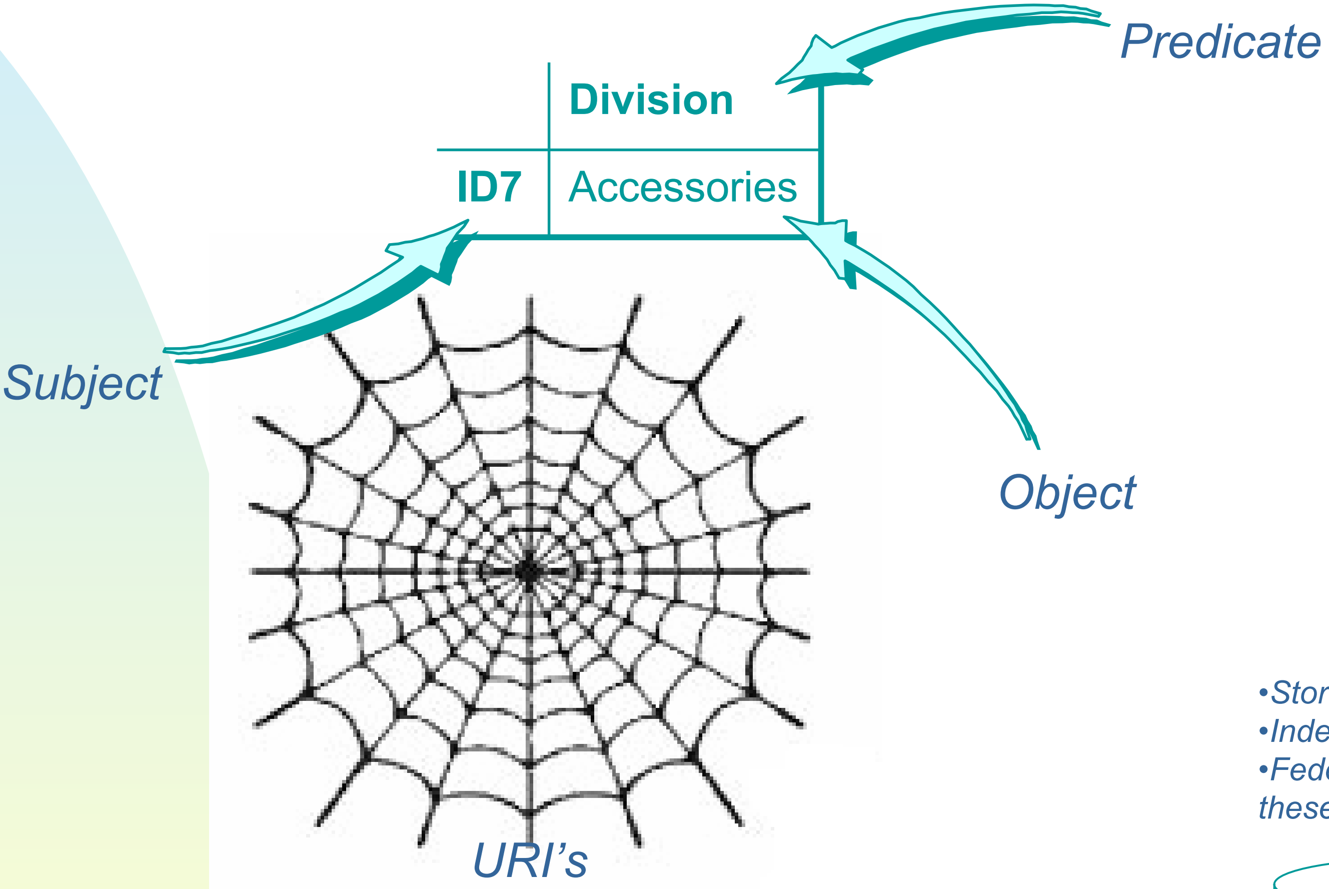


	Product Line
4	Feedback Line



	Model
1	ZX-3

Distribute by cells!?



	Division
ID7	Accessories

•Store,
•Index, and
•Federate
these triples

RDFS – Classes & Properties

- Type information for data? e.g.,
  - Sacramento – Capital
  - Control Engineering – Division
  - Monitor – Product Line
- Related types?
  - Capital < City < Location
  - Division < Profit Center < Organizational Unit

RDFS

- Defining types
  - rdfs:Class
- Defining relationships
  - rdf:type, rdfs:subClassOf, rdfs:subPropertyOf
- Relating Classes to Properties
  - rdfs:domain
  - rdfs:range

Type information as Triples

Subject	Predicate	Object
Sacramento	rdf:type	Capital
Control Engineering	rdf:type	Division
Monitor	rdf:type	Product
Capital	rdfs:subClassOf	City
City	rdfs:subClassOf	Location
Division	rdfs:subClassOf	Profit Center
Profit Center	rdfs:subClassOf	Org Unit

OWL

- 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)

Demo

- Kennedy family information
- Genealogy, schools, posts, etc. Represented as triples
- Merge information from multiple sources
- Utilize a variety of inferencers
- Federated queries

Demo SPARQL Query 1

- 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).

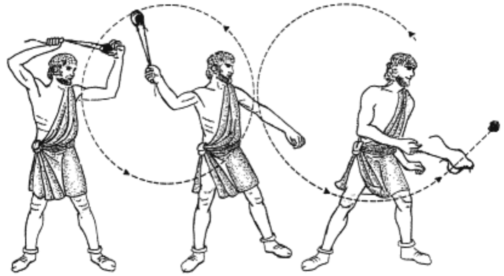
```
CONSTRUCT {?kenu colleges1:state ?state}
WHERE {?kenu rdfs:label ?kn .
       ?newu colleges1:_name ?nn .
       ?newu colleges1:state ?state .
FILTER (xsd:string (?nn) = xsd:string (?kn))}
```

Demo SPARQL Query 2

- Find kennedys (include first name and last name) who went to school in NY

```
SELECT ?kennedy ?fn ?ln
WHERE {?kennedy simple:alma-mater ?u .
              ?kennedy simple:first-name ?fn .
              ?kennedy simple:last-name ?ln .
              ?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

ssears@franz.com