AllegroGraph for Flexibility in the Enterprise and on the Web

Jans Aasman
Franz Inc
ja@franz.com
What is a triple store
Person Database Diagram

Table Person
- ID: 2
- First-Name: Rose
- Last-Name: Fitzgerald
- Middle-In.: E
- DoB: 1890
- DoD: 1995
- PlaceOB: 1
- Sex: F

Table Spouses
- ID1: 2
- ID2: 1

Table to-schools
- ID1: 2
- SchoolID: 3

Table Schools
- ID: 3
- Name: Harvard

Table has-profession
- ID1: 2
- ProfID: 3

Table Professions
- ID: 3
- Name: Home-maker

Table Has-Child
- ID1: 2
- ID2: 2: 2: 2: 2
- ID2: 17: 15: 14: 13

Table Place
- ID: 1
- Name: Boston
- State: MA
- Longitude: 42.3
- Latitude: -71.4
<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>person2</td>
<td>type</td>
<td>person</td>
</tr>
<tr>
<td>person2</td>
<td>first-name</td>
<td>Rose</td>
</tr>
<tr>
<td>person2</td>
<td>middle-initial</td>
<td>E</td>
</tr>
<tr>
<td>person2</td>
<td>last-name</td>
<td>Fitzgerald</td>
</tr>
<tr>
<td>person2</td>
<td>suffix</td>
<td>none</td>
</tr>
<tr>
<td>person2</td>
<td>alma-mater</td>
<td>Harvard</td>
</tr>
<tr>
<td>person2</td>
<td>birth-year</td>
<td>1890</td>
</tr>
<tr>
<td>person2</td>
<td>death-year</td>
<td>1995</td>
</tr>
<tr>
<td>person2</td>
<td>sex</td>
<td>female</td>
</tr>
<tr>
<td>person2</td>
<td>spouse</td>
<td>person1</td>
</tr>
<tr>
<td>person2</td>
<td>has-child</td>
<td>person17</td>
</tr>
<tr>
<td>person2</td>
<td>has-child</td>
<td>person15</td>
</tr>
<tr>
<td>person2</td>
<td>has-child</td>
<td>person13</td>
</tr>
<tr>
<td>person2</td>
<td>has-child</td>
<td>person11</td>
</tr>
<tr>
<td>person2</td>
<td>has-child</td>
<td>person9</td>
</tr>
<tr>
<td>person2</td>
<td>has-child</td>
<td>person7</td>
</tr>
<tr>
<td>person2</td>
<td>has-child</td>
<td>person6</td>
</tr>
<tr>
<td>person2</td>
<td>has-child</td>
<td>person4</td>
</tr>
<tr>
<td>person2</td>
<td>has-child</td>
<td>person3</td>
</tr>
<tr>
<td>person2</td>
<td>profession</td>
<td>home-maker</td>
</tr>
</tbody>
</table>

How is it different and why is it more flexible?

- No **Schema**.
  - Say whatever you want to say but
  - ontologies may constrain what you put in triple store
- No **Link Tables**
  - because you can do one-to-many relationships directly
- No **Indexing Choices**
  - add new data attributes (predicates) on-the-fly that will be real-time available for querying
- Takes **anything** you give it: it is trivial to consume
  - Rows and columns from RDB, XML, RDF(S), OWL, Text and Extracted Entities
And how is it the same

- AllegroGraph has all the enterprise features of a relational database
  - 24/7, robust, real time processing
  - ACID, transactional
  - Replication, On-line Backup, Point in time recovery
  - High Availability
  - Many kinds of full text indexing
  - REST Protocol
Why Triple Stores are here to stay.

- There is now a standard for **meta data or graphs**
  - data representation: RDF/RDFS
  - logic and inferencing: RDFS/OWL
  - querying: SPARQL
  - rules: RuleML, Swirl
- With vertical compression we can get this smaller than RDB
- Comparable to RDB and NoSQL wrt speed and scalability
- And an enormous impetus to share data via RDF
latest LOD cloud
LOD meets LOC

LINKED OPEN COMMERCE

Universal e-Commerce Model
http://purl.org/goodrelations/

Good Relations

Vertical Ontologies
For specialized e-commerce Domains

OS Commerce
Magento
Joomla Virtuemart

The Long Tail
Small shops, SMBs, Blogs, Facebook, etc.

Base LOC Product Catalogue

Product Catalogues
From vendors and aggregators

Best Buy
Amazon
Big Shops

CNet

O'Reilly

Tesco
eBay
Overstock

Native RDF
Via Linked Data Adapters
Just look at this
But what about NoSQL?

- And by that we mean
  - Cassandra
  - Hbase
  - CoucheDB
  - Etc..
Classes of applications and their natural database.

Regular Enterprise Applications.

Web Scale Shallow Objects

Complex Meta Data Applications.

Relational Databases

Hadoop, Hbase, Cassandra, etc

Triple Stores
Examples of things that work better in triple stores
A SPARQL query spanning 4 sources

```sparql
PREFIX go: <http://purl.org/obo/owl/GO#>  
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  
PREFIX owl: <http://www.w3.org/2002/07/owl#>  
PREFIX mesh: <http://purl.org/commons/record/mesh/>  
PREFIX sc: <http://purl.org/science/owl/sciencecommons/>  
PREFIX ro: <http://www.obofoundry.org/ro/ro.owl#>

SELECT ?genename ?processname
WHERE
{ graph <http://purl.org/commons/hcls/pubmesh> 
  { ?paper rdfs:subject mesh:0017966. }
  ?article sc:identified_by_pmid ?paper.
  ?gene sc:describes_gene_or_gene_product_mentioned_by ?article.
} graph <http://purl.org/commons/hcls/goa> 
  ?res owl:onProperty ro:has_function.
  ?res2 owl:onProperty ro:realized_as.
  ?res2 owl:someValuesFrom ?process.
} graph <http://purl.org/commons/hcls/ncis/20070416/classrelations> 
{ [ ?process rdfs:subClassOf go:GO_0007166 ]
  UNION
  { [ ?process rdfs:subClassOf go:GO_0007166 ]. }
  ?res3 owl:hasValue ?gene.
} graph <http://purl.org/commons/hcls/gene>
{ ?gene rdfs:label ?genename. }
graph <http://purl.org/commons/hcls/20070416>
{ ?process rdfs:label ?processname. }

Inference required
```

- Mesh: Pyramidal Neurons
- Pubmed: Journal Articles
- Entrez Gene: Genes
- GO: Signal Transduction
A Financial Application

- Find a cyclic path between two companies that are engaged in revenue boosting
Q1: A reasonable hard query for horizontally scaling stores and rdb, a straight forward query for vertical/parallel stores

```
where {
  Franz send-money ?a
  ?a send-money ?b
  ?b send-money ?c
  ?c send-money Cray
  Cray send-money ?d
  Not (?d = ?c)
  ?d send-money ?e
  Not (?e ?b)
  ?e send-money Franz}
```
Q2: A very hard query for horizontally scaling stores and rdb, a straight forward query for vertical/parallel store

Find a money trail from Franz to Cray that is two or more steps, find another money trail from Franz Cray that is also two or more steps where the two trails are completely different

(Select (?path1 ?path2)
  (path Franz Cray <send-money> >= 2 ?path1)
  (path Cray Franz <send-money> >= 2 ?path2)
  (empty (intersection ?path1 ?path2))
)
Why is this hard in SQL

- Relational databases very good at straight joins but less optimal for self-joins of unpredictable length

- Try writing this as a sql query 😊
Why is this hard in distributed key/value stores.

- Databases like Cassandra are extremely good at retrieving nested objects in a sea of billion of objects but are less optimal for joins.

- Relatively hard to write these as map reduce expressions. Every query has to be expressed as program, ad hoc is therefore discouraged’
Why is this easier in vertical scaling triple stores

- These queries can be easily parallelized
- Graph databases are made to do graph search
- Ad hoc queries easy to write in Sparql or Prolog
## Summarizing

<table>
<thead>
<tr>
<th>Feature</th>
<th>RDB</th>
<th>Cassandra, Hbase</th>
<th>AllegroGraph 4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schemaless</td>
<td>-</td>
<td>+/-</td>
<td>+</td>
</tr>
<tr>
<td>Transactions</td>
<td>+</td>
<td>+/-</td>
<td>+</td>
</tr>
<tr>
<td>ACID</td>
<td>+</td>
<td>+/-</td>
<td>+</td>
</tr>
<tr>
<td>Concurrent &amp; Dynamic</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Random Access</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>High Availability</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Complex graph search</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Structured + Unstructured</td>
<td>-</td>
<td>+/-</td>
<td>+</td>
</tr>
<tr>
<td>Scalability</td>
<td>+</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>General Flexibility</td>
<td>-</td>
<td>+/-</td>
<td>+</td>
</tr>
</tbody>
</table>
Is it also used in the enterprise?
Why do they say they use triple stores?

- **Flexibility**
  - In changing your data schemas on the fly and do real time processing with (meta) data
  - Add new predicates and use them in queries the next microsecond

- **Flexibility**
  - In writing ad-hoc queries
  - In finding complex patterns and working with rules

- **Flexibility**
  - Integrating various data sources without complicated masterdata management.

_and there is a standard_
An event example
A Simple Event Ontology

• **A type**
  – Meetings, communications event, financial transactions, visit, attack/truce, an insurance claim, a purchase order
  – RDFS++ reasoning

• **A list of actors**
  – Social Network Analysis

• **A place**
  – GeoSpatial Reasoning

• **A Start-time and possible an end-time**
  – Temporal Reasoning

• **Anything else that describes the event**
  – Goods that changed hands
Social Network Analysis
Answers 4 questions

• How far is P1 from P2 (and how strong is the relation?)
• To what groups does this person belong (ego groups, cliques?)
• How important is this person in the group?
• Does this group have a leader, how cohesive are they?
GeoSpatial

• Make the following super efficient
  – Where did something happen?
  – How far was event1 from event2?
  – Find all the events that occurred in a bounding box or radius of M miles?
  – Do these two shapes overlap?
  – Find all the objects in the intersection of two shapes
• On a very large scale
  – when things don’t fit in memory
  – millions of events and polygons
• Adhere to our convention to encode StartTimes and EndTimes and enjoy efficient temporal primitives

• Implementation of Allen’s interval logic primitives
Activity Recognition

- Our customers use AllegroGraph as an event database with social network analysis and geospatial and temporal reasoning.

Find all meetings that happened in November within 5 miles of Berkeley that was attended by the most important person in Jans’ friends and friends of friends.

```
(select (?x)
 (ego-group person:jans knows ?group 2)  
(actor-centrality-members ?group knows ?x ?num)  
(q ?event fr:actor ?x)  
(qs ?event rdf:type fr:Meeting)  
(interval-during ?event "2008-11-01" "2008-11-06")  
(geo-box-around geoname:Berkeley ?event 5 miles)
!)
```
Thank you
And questions?