AllegroGraph

a graph database

Gary King
gwking@franz.com
What we store

How we store it – the possibilities

Using AllegroGraph
Databases

- Put stuff in
- Get stuff out
- quickly
- safely
Stuff

- things with attributes *and* connections
- Reasoning, rules, inference
- Lots of change. all the time.
What stuff in?

- Modeling knowledge of assets in an Enterprise
- Modeling an extensive river network
- Representing 1000’s of different types of objects
- Managing biological knowledge
- Multimedia Metadata
- Bug and version tracking
- Collaborative Workspace for Analyst
NASA Constellation project...

- Deals with 1000s of different types of objects:
  - Machine parts
  - Processes
  - Software
  - People skills
  - Drawings
  - Documents
  - In 100s of distributed databases
  - Coordinated through registries
  - To provide meaningful search
A River Network

- Given the polluted segment S1 find all the upstream segments within 50 miles of City1200
- Given the polluted drainage D1 find all the schools in the rectangle \(<x_1, y_1, x_2, y_2>\) that might be influenced
Semantic Web...
What stuff out?

- Things like *this*
- Things like *this* only with *that*
- Things like *this* only with *that* and the other thing sorted by *that*
- Things like *this* linked to *that* linked to *that* linked to *that* and *that* and back to things like *this*
- Things like *this* where *that* can be inferred from *this* other stuff
In particular

- We want to ask for
  - What – Attributes
  - Where – geospatial
  - When – events and temporal logic
  - Whom – Social networks

Find the people I know that share my taste and have traveled to Hawaii during the last year?
Data – Dissected

- Documents (unstructured – mostly)
- Key / value
- subject / predicate / object
- Tuples (by row, by column)
System of Analysis

- The main data is stored safely away somewhere else
- Batch & Bulk oriented loads
- Materialize types and other inferences
- Do queries & analysis
- Few simultaneous users
System of Record

- Data changes on a second to second basis
- You care about the long time persistence of the data
- You care about transactions and recoverability
- You care about concurrent access
- You care about continuous querying and instant reasoning
How?

- Relational Database Systems
- Object Oriented Databases
- Key-value Databases
- Graph databases (Triple Stores)

Essentially equivalent; the devil is in the details.
RDBMS

- Tables
- Columns
- Indices
- Joins
RDBMS

- Mature and Standardized (SQL)
  - Robust, safe, scalable
    - Great for simple queries that touch only a few tables once
  - But...
    - Modeling the world in tables is hard
    - Table schema is inflexible; early design lock-in
    - One-to-many and many-to-many relationships add extra tables
    - Lousy for queries that follow transitive relationships across many tables (or the same table many times)
Graph DBMS

- Subject – Predicate – Object
GDBMS

- Easy to put stuff in
- No Schema, everything indexed

But...

- Young technology
- Less robust, less standardized
Our problem

- Continually accrue massive interconnected information with an evolving schema (or no schema) including text, events, relationships, locations

- Query this data using description logics, custom rule sets and ask for information on moving objects, events, and social networks, in real-time
In particular

Find another truck that can pick up package X at location Y so that I can pick up package A at location B so that we both will arrive at P before time T.

- We want to ask for
  - What – Attributes
  - Where – geospatial
  - When – events and temporal logic
  - Whom – Social networks
RDBMS is not the answer

- A graph database looks like an relational database with only one table so start with an RDBMS and add triple-store features
- The relational model is *too* complex for triple-stores
- The relational model is *too* simplistic for rapidly evolving schemas and massive transitive relations
Hadoop is not the answer

- Yes, it is a great way to store billions of triples
- Hadoop can be used for work that is batch-oriented rather than real-time, very data-intensive, and parallelizable.

- But what about
  - Deeply nested SPARQL or rule based queries (e.g., Prolog)
  - Graph & Social network analysis.
  - Reasoning and inference
Building a triple-store

- Start fresh and add *enterprise* features
- adding *triples* (with five parts)
- Emphasis is on addition (not updates, not deletion)
Really Simple Diagram

- Triples In
  - processes to
    - index
    - merge
    - text index
    - process strings
AllegroGraph 4.0

- ACID Transactions and Recoverability
  - page management
  - checkpointing every $x$-minutes or $y$-triples
- Read/write concurrency
  - 100% read concurrency at all times
- Dynamic and automatic indexing
  - with column based compression
- Resource management
  - Use all disks, all memory and all processors (one box)
  - Automatic, or user configurable
AllegroGraph 4.0

- Per-predicate *Lucene* style text indexing
- 2D and 3D geo-temporal indexing for moving objects
- Social networking toolkit with path finding, importance measures, etc.
- REST protocol for all client interaction
  - Franz supported: Sesame, Jena, Python,
  - Community supported: Ruby, Perl, C#
2D and 3D details
Performance: Input

- $5000 quad-core machine with 32 Gigabytes RAM

<table>
<thead>
<tr>
<th>dataset</th>
<th>Size (Billions)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUBM 8000</td>
<td>1.1</td>
<td>3:48</td>
</tr>
<tr>
<td>Billion Triples Challenge</td>
<td>1.15</td>
<td>5:13</td>
</tr>
<tr>
<td>2000 Census data</td>
<td>0.99</td>
<td>2:00</td>
</tr>
<tr>
<td></td>
<td>3.2</td>
<td>11:01</td>
</tr>
</tbody>
</table>

- with full-text indexing on all strings
Thanks

gwking@franz.com

http://www.franz.com