Triple Stores?
Use cases for Semantic Technology

Jans Aasman
Franz Inc
Contents

- Franz and customers
- 2 slides about triples
- A quick demo
- Two Use Cases for an RDF Graph database
  - **Ontology based data integration:**
    Amdocs: a real time semantic platform for telecom that knows everything about everyone in real time
  - **Intelligent Text Analysis and Semantic Search:**
    A platform for for real time news + entity extraction + Linked Open Data and social network analysis
- Integration with Open Source: MongoDB/Solr
- Scalability?
Franz Inc – Who We Are

- Private, founded 1984
- We are an AI and Semantic Technology company
- Out of Berkeley
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: 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 from an RDB 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**
  - Can add new data attributes (predicates) on-the-fly that will be real-time available for querying, because everything is automatically indexed.
- **Takes anything** you give it: it is trivial to consume
  - Rows and columns from RDB, XML, RDF(S), OWL, Text and Extracted Entities
AllegroGraph: RDF Graph Store

- Java: Sesame / Jena
- Python
- C#
- Ruby
- Clojure / Scala
- Lisp
- Perl

Backup / Restore
- Replication
- Warm Failover
- Security
- Management

Rest
- SPARQL
- Prolog
- Rules Clif++
- Geo
- SNA
- Time
- RDFS++
- Stored Procedures (Javascript, Lisp)

Session Management, Query Engine, Federation

Storage Layer (Compression, Indexing, Freetext, Transactions)
Getting more linked into the NoSQL world

- Java: Sesame / Jena
- Python
- C#
- Ruby
- Clojure / Scala
- Lisp
- Perl

Backup / Restore
- Replication
- Warm Failover
- Security
- Management

- SPARQL
- Prolog
- Rules
- Clojure++
- Geo
- SNA
- Time
- RDFS++
- Stored Procedures (Javascript, Lisp)

- Session Management, Query Engine, Federation

- Storage Layer (Compression, Indexing, FreeText, Transactions)

mongoDB
{name: "mongo", type: "DB"}

Apache Solr
DoD and Intelligence Customers
ANNOUNCING AMDOCS CES 8
OUR NEWEST PORTFOLIO RELEASE.

Hear Dov Baharav, Amdocs CEO, and Emily Green, President and CEO of Yankee Group, along with other Amdocs executives discuss Amdocs CES 8.

View this online event now >
Use Case Amdocs

Build a **semantic** platform
that knows **everything**
about **everyone**
in **real time**.
Amdocs Overview

Communication Service Provider Focus
Leading BSS, OSS & Service Delivery Products
Strategic Services

- $3B in revenue
- $367M operating income
- $1.2B in cash
- ~18,000 employees
- Over 60 countries

Amdocs is #1 in Worldwide OSS and Billing with a 31.2% Market Share.
(January 2009)

Amdocs named the #1 Telecom Operations Management Systems Vendor.
(May 2009)

“Gartner
Amdocs benefits from a strong managed services business...”
(February 2009)
Contextually Aware Business

Deliver Consistent Contextually Aware Service, Anywhere, Anytime

Empower Touch Points to Be Self Sufficient and Proactive

Seize Every Revenue Opportunity

Streamline And Cut Costs

Harmonizing business capabilities to work together to drive growth while delivering a great cost efficient customer experience
The Customer Contact Opportunity

Traditional IVR

- No personalization of IVR
  - 35% Abandonment Rate
- No context based call routing
  - 22% transfers
- No customer context for Agents
  - 28% of call time “understanding”
- No root cause assessment
  - 25% caller repeat rate

Large Service Oriented Business
- 50,000 Agents
- 500,000,000 calls annually
- $7.50 per call
- Contextual based savings = $575M/year
The Customer Churn Opportunity

Retention Solution

1. Recognize churn scenarios in real time
   - Price, value, and usage of product
   - Service Quality
   - Balance

2. Action personalized retention measures at time of detection

3. Focus on profitable and savable customers

N.A. Mid-Tier Carrier Annual Savings ($MM)
Revenue Generation Opportunity
Targeting drives revenues

Understanding customer’s interests enables delivery of highly relevant offers and service...

With a connected social media profile, customer’s interests are tracked as they change over time.

Based on interests, relevant marketing campaigns are offered during interactions with that customer.

Targeting the customer’s social network (SNA) results in an average of 6-10 friends clicking through to the offer.

Selling to the social group influencer is 11X more likely to result in 2 additional sales.

If the offer is recommended by a friend, customers are 4X more likely to make a purchase.

...net increase marketing campaign revenue by over 200%
Amdocs Intelligent Decision Automation Platform (AIDA)

Real time customer experience state, intelligence and history

What’s happening
- Billing
- CRM
- Ordering
- Web & Social Media
- Data Warehouse
- Network

What Action
- Proactive
- Reactive

Trends
- Patterns
- Relationships
- States
- Distance
- Timing

Absence of occurrence
- Historical patterns
- Subjective Concepts
- Probability of occurrence

- Data Service not used
- High Churn Risk
- DVR education needed
- frequent adjustment pattern
- Inside schedule window
- Long time Customer
- Voice mail not configured
- Emerging outage in area
- Case closed after outside work
- Voice Mail not configured
AIDA Commercial Differentiators

> History and real time state of the customer experience
  > Systemic Memory across channels & interactions*
  > Optimized for decision making*
  > Extreme Volume at low cost

> Built in Social, Semantic, Temporal, and Geospatial reasoning capabilities
  > Shrinks the gap between human and machine decision making

> Proactive and reactive processing models
  > Business concepts enable decisions that are out of reach for traditional rules engines*
  > Concepts extendable and manageable by the Business*

> One platform for all touch points, processes, customer facing and back office scenarios
  > Care, Revenue, M2M, New Services

* = Patents Pending
Events from many source systems are transformed into a set of related business concepts.
AIDA Maps Events to Concepts

Events from many source systems are transformed into a set of related business concepts.

**Many events**
- Interactions
- Orders
- Bills
- Payments
- Collections
- Charge dispute
- Customer
- Pay instructions
- Individual
- Device Activated
- Device heartbeat
- Subscriptions
- Device changes

**Subjective**
- Patterns
- Trends
- Geospatial
- Time
- Probability
- Absence of occurrence
- Relationship between

“good payer"
“always pays 2 days late"
“improving payer"
“within 5 miles of the tower"
“within 5 minutes of an outage"
“probably will call about the bill"
“missed payment"
“friend of a friend"
Architecture

Events

- Amdocs Event Collector

Event Data Sources

- RM
- CRM
- OMS
- NW
- Web 2.0

Decision Engine

- SBA Application Server
- Inference Engine (Business Rules)
- Bayesian Belief Network
- AllegroGraph Triple Store DB

Scheduled Events

- “Sesame”

Actions

- Operational Systems
  - CRM

Operational Systems

- CRM
• Events are collected from many heterogeneous, configured event sources
  – Phone calls, texting, video upload, roaming, etc.
  – iTune download, web site interaction, media upload
  – Emails, support calls
  – Bill payment or non-payment
  – Phones stop working or disconnect

• All fused and mapped into a single event knowledge base
AIDA Semantic Inference

- Define rules to operate to create higher level concepts
  - Event (mapping) rules - Map event data into the domain ontology
  - Automatic rules – Compute new properties defined by the ontology
  - On-demand rules - perform inference for the services
- Rules triggered upon event ingestion, service request or schedule
- Semantic rule inference generates **new triples** from existing ones
Semantic Inference – Using Business Rules to generate high level concepts

- AIDA provides **Workbench** for business rule construction
- Utilizes a sophisticated magnetic block GUI for business analysts
- Rules triggered to infer and generate **new business concepts**

```java
rule PaymentDetails.timeliness {
    if date within EarlyPeriod days after customerBill.billDate
    then timeliness = Early;
    else if date not within LatePeriod days after customerBill.billDate
    then timeliness = Late;
    else timeliness = OnTime;
}
```

Each business rule defines an attribute. This rule defines an attribute of the PaymentDetails class called timeliness.

All classes and their attributes are defined in the application ontology.

“Late Payment” defined in Workbench
Decisioning – Probabilistic Assessment

- AIDA incorporates also **Bayesian Belief Networks** (BBN)
- These are graphical models for reasoning under uncertainty
- Important part of decision making – the likelihood of something happening estimated by how often it occurred in the past (primarily used in medical research until recently)
- **Evidence** consists of **observations** on certain nodes leading to **conclusions**
Presenting insight to the CSR

Prediction on reason for the call – ranked by probability

Presentation of recent interactions and events

Prioritized Recommended treatment and script

Process opens relevant screen for reference and action
First application: CRM
Amdocs Guided Interaction Advisor

First Call Resolution
• Increase up to 15%

Average Handling Time
• Reduce up to 30%

Training Costs
• Reduce up to 25%
So why a triple store

- Flexibility, flexibility and flexibility
  - Change the schema on a daily basis
  - Customers create new policies which in turn will create new schemas on the fly
- Needed to work with meaning
  - Rdf describes data
- Needed to be declarative for everything
  - Most RTBI is a combination of data in the DB and java variables in the application.
### So why a triple store?

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Triple Store (AG)</th>
<th>Relational</th>
<th>No-SQL</th>
<th>Multi-dimensional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic data model</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inference</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scalability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic Capabilities &amp; M:M:M</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unstructured data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCO</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We can use this all over the place

- HR for the DOD
- Check an Insurance Card
- Fraud detection for Credit Cards
- Hospital Visits and knowing your patient (shouldn’t be there at least one agent knowing everything about you?)
- Online Learning
- Financial
Federated Unified Event Pipeline

Federated Event Collector

Decision Engine

Ingestion

Machine learning

Scheduled events

Inference Engine

Semantic Repository

Reporting Tools

Web based Views of Single Individual
A DBA uploads a **CSV** file every day with all the updated and new information. The FEC polls SFTP site and turns csv into triples.

Person P was admitted to Hospital H on day D for reason R.
The FEC does a set of standard SQL queries every day and turns output into triples.

Person P started course C in topic T at day D.
The FEC does a SPARQL Query against relational database using R2RML and turns output into triples.

Person P gets promotion to rank R on day D in pay scale S.
A DBA does a SPARQL Update query to the FEC to add triples directly.

Person P is deployed in Country A starting at Day D.
The RDB owner wrote **web services** and the FEC does a web service call every x minutes and turns output into triples.

Father F of Person P retired at time T.
Or anything else you can come up with.
Pull a new event for Person P and retrieve all existing triples for person P.
Hundreds of Business Rules are applied to turn basic events into Higher Level Concepts. For example: the fact that Person P got wounded means that he is currently **unavailable for missions**.
Old and New knowledge is fed into a machine learning system to generate predications about possible future individual’s behavior. Predications are again encoded as RDF triples. 

**Prediction:** She most likely will sign up for a new tour.
Triggers on time or conditions can be set that will execute rules, queries and actions in the real world.
All knowledge about individuals is stored in a Semantic Repository (AllegroGraph).
Give me all soldiers that have experience with country C, that have been on at least two missions, are currently well rested, and have taken a course in crowd control.
Reporting
How much money did we spend on the families of soldiers with children?
A view on an individual from a certain perspective. Realize that this view can be based on literally hundreds of source databases.
Integration with open source

- Mongo/solr
Two questions we get

• Can you make inserting and querying JSON documents as easy as MongoDB

&

• Can you make the freetext engine of AllegroGraph as powerful as Solr/Lucene
Agile and Scalable

MongoDB (from "humongous") is a scalable, high-performance, open source, document-oriented database. Written in C++, MongoDB features:

- **Document-oriented storage**
  JSON-style documents with dynamic schemas offer simplicity and power.

- **Full Index Support**
  Index on any attribute, just like you’re used to.

- **Replication & High Availability**
  Mirror across LANs and WANs for scale and peace of mind.

- **Auto-Sharding**
  Scale horizontally without compromising functionality.

Newsletter Signup
Keep up to date with MongoDB.

In Production
- Disney
- Craigslist
- MTV Networks
More Production Users

About
- Use Cases
- Philosophy
A Tiny MongoDB Browser Shell (mini tutorial included)
Just enough to scratch the surface

2. Documents
MongoDB is a document database. This means that we store data as documents, which are similar to JavaScript objects. Here below are a few sample JS objects:

```javascript
var a = {age: 25};
var n = {name: 'Ed', languages: ['c', 'ruby', 'js']};
var student = {name: 'Jim', scores: [75, 99, 87.2]};
Create some documents, then enter 'next'
```

> next

3. Saving
Here's how you save a document to MongoDB:

```javascript
db.scores.save({a: 99});
```

This says, "save the document '{a: 99}' to the 'scores' collection."

Go ahead and try it. Then, to see if the document was saved, try

```javascript
db.scores.find();
```

Once you've tried this, type 'next'.

> |
So why do we want a MongoDB API?

- Sparql
- JOINS
- OWL Formalism
- Prolog & Rules
- Social Network Analysis
- Reasoning
Agraph already ‘does’ Javascript

• An in-server JavaScript compiler
• Used for stored procedures
  – Write you own complex algorithms
  – Or query language
  – Or fine tune procedures for you client application
• Turns stored procedures into REST calls
The AllegroGraph RDF server can be scripted using, apart from Common Lisp, the JavaScript language. The easiest way to get started with this is to open a repository in WebView and select the 'Script' page from the navigation bar.

This document describes the programming interface available to JavaScript programs running in an AllegroGraph server.

Overview

The simplest way to run scripts (and this is what the WebView interface uses), is the /eval HTTP service, using a text/javascript content-type for your request. This will cause the body of the request to evaluated as JavaScript. If an error is raised, it will be returned as an HTTP error. If not, the result of evaluating the code is returned. Usually, you'll want to specify that you accept application/json responses, and receive the result in that form. If your code returns a cursor, you can also specify one of the formats suitable for that kind of
Parts and Triples

The `Part` function, when given a string, tries to parse it using N-Triples syntax, and returns a part value. This constructor has the following properties, used for constructing parts in other ways:

- `literal(string [, language])`
  Create a literal.

- `literalTyped(string, type)`
  Create a typed literal.

- `resource(string [, namespace])`
  Create a resource. If namespace is given, it is looked up (see `namespaces`) and prepended to the resource name.

- `fromInt(int), fromUnsignedInt(int), fromLong(long), fromUnsignedLong(long)`
  Create an encoded part from an integer value.

- `fromFloat(float), fromDouble(double)`
  Create an encoded part from a floating-point value.

- `fromSeconds(seconds)`
  When given an amount of seconds relative to 1900, encodes the time as a part.

- `fromDate(milliseconds)`
  Like `fromSeconds`, but divides its argument by 1000, and uses 1970 as reference point. This can be used to convert JavaScript `Date` objects to parts, since those (when converted to number, which this function implicitly does) will return an amount of milliseconds since 1970.

Part values themselves have the following properties:

- `toString()`
  Returns an N-Triples representation of the part.

- `value`
  The main value of the part. For resources, this is the URI, for literals, the string, for encoded values, usually a number.

- `type`
  A string, such as "literal" or "resource", which identifies the type of the part.

- `language`
  The language of a literal, or `null` if no language is specified.

- `datatype`
  The datatype of a literal, or `null` if no language is specified.

`Triple` objects, as returned by, for example, `store.getTriples`, have `object`, `predicate`, `subject`, and `graph` properties, which will return parts, and an `id` property, which will return an integer.
The Triple Store

The top-level store variable provides an interface to the current triple store (as determined by the URL from which the request is made). It is an instance of the Store type, and has the following properties:

name
- The name of the store.

getTriples(subject, predicate, object, graph)
- Returns a triple cursor containing the triples in the store that match the given query. Each of the arguments can be left as null (or not given) to specify a wildcard.

getTriplesArray(subject, predicate, object, graph [, limit])
- Like getTriples, but returns an array instead of a cursor. If limit is given, it provides a maximum length for the returned array.

deleteTriples(subject, predicate, object, graph)
- Deletes all matching triples. Again, null and undefined count as wildcards.

addTriple(subject, predicate, object [, graph])
- Adds a new triple. Graph is optional.

size
- The amount of triples in the store.

commit(), rollback()
- Commit or roll back the store.

newBlankNode()
- Returns a newly allocated blank node.

runSparql(query)
- Evaluates the given SPARQL query, and returns, depending on the type of query, a boolean, a triple cursor, or a row cursor. SPARQL is evaluated by the user has write access).

runProlog(query)
- Evaluates the given Prolog query. Will typically return an array of arrays of parts, but the exact format returned depends on the format of the Prolog query.

textIndices
- A list of names, corresponding to the full text indices present in the store.

createTextIndex(name [, options])
- Create a text index. options, if given, is an object containing options for the store. The properties predicates, indexFields, indexResolvers, minimumWordSize, stopWords, and wordFilters are supported, and accept values similar to those in the Lisp API.

dropTextIndex(name)
- Drop a text index.

textSearch(query [, index])
- Search the given text index (or all of them, when index is left off) for the given query string. Returns a triple cursor.

indices
- A list of indices present in the store.

addIndex(type)
- Add a new index. Type must be a postgreSQL-style specifier.

dropIndex(type)
- Delete an index.
Graph/Network Operations

All graph operations are modeled around 'generators', which, conceptually, are functions that take a node, and produce a set of 'neighbors', by some definition of neighbor. The Generator constructor can be called in two ways. In both, it takes a store as its first argument. When its second argument is a function, it wraps that function as a generator. The function should take a single part as an argument, and return an array of parts. When the second argument is not a function, the signature of the constructor is (store, objects, subjects, undirected), where each of the last three arguments can be null, a predicate part, or an array of predicates. The set of predicates indicated by the objects parameter causes relations with those predicates to be followed from the starting node to any objects. The subjects parameter does the reverse—it follows relations from object to subject. Finally, the undirected parameter causes both of these to happen for the given predicates.

Generator instances have the following methods:

\[ \text{asMatrix}(\text{group}, \text{maxdepth}) \]
Converting a generator to a matrix returns a new generator which is a pre-computed version of the original generator. The group parameter should be an array of parts, and maxdepth an integer, indicating how many 'jumps' (starting from this group) should be pre-computed. The return value is again an instance of Generator, and supports all the methods listed below.

\[ \text{breadthFirstPath}(\text{from}, \text{to} [, \text{maxdepth}]) \]
\[ \text{depthFirstPath}(\text{from}, \text{to} [, \text{maxdepth}]) \]
\[ \text{bidirectionalPath}(\text{from}, \text{to} [, \text{maxdepth}]) \]
These three methods try to compute and return a path between two parts, giving up after maxdepth jumps (or not giving up if no maximum depth is given). The return value will be an array of parts, or null if no path was found.

\[ \text{breadthFirstPaths}(\text{from}, \text{to} [, \text{maxdepth}]) \]
\[ \text{bidirectionalPaths}(\text{from}, \text{to} [, \text{maxdepth}]) \]
These work just like the methods described above, but will return an array containing all the shortest paths found, or the empty array if no path was found.

\[ \text{neighbors}(\text{node} [, \text{maxdepth}]) \]
Returns an array containing all the parts that can be reached within maxdepth jumps from node. If not given, maxdepth defaults to 1.

\[ \text{cliques}(\text{node} [, \text{minsize} [, \text{maxsize}]]) \]
Finds and returns any cliques (completely connected subgraphs) that node is part of. minsize defaults to 3.

\[ \text{isClique}(\text{nodes}) \]
Given an array of parts, returns a boolean that indicates whether this group is a clique.
UPI Hash Tables

Hash tables specialized for UPIs are exposed through the `UPIHash` constructor. These are much faster than using regular JavaScript objects to keep a mapping on triple parts. The constructor takes an optional size argument. Instances have the following methods:

- `set(key, value)`: Store value under the part given as key.
- `get(key)`: Retrieve the value stored under this key.
- `del(key)`: Remove the given key from the table.
- `keys()`: Returns an array holding all the keys in the table.
- `forEach(func)`: Calls `func` with `(key, value)` arguments for every entry in the table.
Custom Service Definitions

The `server` variable exposes a method `defineService(method, name, func)` which can be used to define a JavaScript custom service. Its first argument should be a string (one of "get", "post", "put", and "delete"), or an array of such strings. `name` is a string that will name the service (for example, "magic" will create a service under `/repositories/[x]/custom/magic`), and finally, `func` should be a JavaScript function of one argument.

When the service is called, this function will be applied to a request object, and its return value will be returned to the user, using the same content-negotiation process that was used for `/eval` (as described before).

The request object can be used to get information about the request. It has the following properties:

- `url`  
  The URL to which this request was made.

- `method`  
  The method used for the request, as a lower-case string.

- `param(name)`  
  Retrieves the value of a parameter. This includes url-query ("get") parameters, and parameters found in the request body if it has a content-type of `application/x-www-form-urlencoded`. Returns `null` if the parameter was not given.

- `paramArray(name)`  
  Works like `param`, but returns an array, allowing you to see whether a parameter has been given multiple times.

- `params`  
  An object with the parameter names as property names, their values as property values.

- `body`  
  The body of the request, as a string, or `null` if no body was given.

- `header(name)`  
  Retrieves the value of a request header.
function findPredicates(cls) {
    var map = new UPIHash;
    store.getTriples(null, Part.resource("type", "rdf"), cls).forEach(function(triple) {
        store.getTriples(triple.subject).forEach(function(triple) {
            map.set(triple.predicate, true);
        });
    });
    return map.keys();
}

server.defineService("get", "findPredicates", function(req) { return findPredicates(req.param("cls")); });
Server Scripting

Userscripts: None

Sitescripts: None

store.getTriplesArray(null, null, null, null, null, 10)

Language: JavaScript

Run (view as JSON)

Result

0: ["<http://www.franz.com/entity/Nokia>","<http://www.franz.com/SYNCON>","\"278236\""]
And now about MongoDB API

- Turns JSON objects into triples and store the entire JSON object too.

```json
{ a: 10, b: 30, c: [ 1,2, {name: 'jans', age: 52} ] }
```

turns into

```xml
<l a 10 store1>
<l b 30 store1>
<l c 1 store1>
<l c 2 store1>
<l2 name jans store1>
<l2 age 52 store1>
<l c 2 store1>
<l id 1 store1>
<l mongoblob { a: 10, b: 30, c: [ 1,2, {name: 'jans', age: 52} ] } store1>
```
Use Case: Text Intelligence

• Problem:
  – 80% of my business intelligence is not in relational databases but unstructured: documents and emails and spreadsheets and drawings and contract.
  – I also want to know how to relate information in the world (business news) and in social networks to things that are important to my business

• Solution:
  – Create ontologies for the important things in your business (incl ontologies, vocabularies, thesauri, taxonomies)
  – Spider your internal network and the web
  – Entity extract the results and link to ontologies and then link to LOD

• Where:
  – Intelligence agencies, banks, insurance, retail.
How would you do this with your standard search engine

- Give me a newspaper text with a republican and a democrat that serve on two subcommittees that have the same parent committee.

- Which [democrat|republican] is most vocal in the oil spill disaster

- Given this text, find all the other texts that have the same people and the same main topics but not democrats in the text.

- Which newspaper favors [democrats|republicans]

- Which [democrate|republican|senator|representative] get most of the attention in the last week.

- Give me the distribution of the most important topics yesterday
The process

• We spider daily > 300 on-line newspapers and thousands of blogs

• And search specifically for all the member of the senate and house of representatives and the executive branch

• Apply entity extractor to the text and extract main concepts
  – About 150 triples per text...

• Hook up these concepts with a detailed database of each politician and with information from the linked open data cloud
Dianne Feinstein urges Ed Lee to run for SF mayor

San Francisco Chronicle - John Sebastian Russo - Jul 26, 2011

Dianne Feinstein is calling on Mayor Ed Lee to run for a full four-year term, saying she believes "San Francisco needs his steady leadership and unifying ..."  

Feinstein presses SF interim mayor to run in fall

San Jose Mercury News

San Francisco mayoral race heats up as Ed Lee weighs run

Los Angeles Times

Feinstein Urges Lee to Run for SF Mayor

NBC Bay Area

SF Weekly (blog)

all 69 news articles »


Tahoe clarity is second-lowest ever measured

Fresno Bee - Tom Knudson - 1 day ago

Dianne Feinstein – stood at the podium. "Oh my goodness," Feinstein said, as beams of silvery light rippled across the lake and her audience at the West ...

Guest Column: Lake Tahoe gives up one of its deadly secrets

North Lake Tahoe Bonanza

all 13 news articles »


Social Scene: Tahoe Fund dinner raises more than $200K; seats big ...

Tahoe Daily Tribune - Vicki Kahn - 6 hours ago

Photographers and art auction contributors Elizabeth and Olof Carmel met Senator Dianne Feinstein and her husband Dick Blum at the cocktail ...


Dianne Feinstein backs debt-ceiling agreement

Los Angeles Times - Shane Goldmacher - Aug 1, 2011

Dianne Feinstein said Monday that she supported the deficit-reduction package agreed upon by congressional leaders and President Obama, ...

Sen. Diane Feinstein on Debt Ceiling Debate

The State Column

Feinstein: Current Debt Ceiling Plan Is A 'Very Different Sausage ...

TPMDC

all 8 news articles »
Dems Regrouping After Putting All Their Eggs In Brown's Basket
Comments (1,019)

Former Bush Press Secretary Goes Back To College
Comments (817)

Reid: Republicans Want Economy To Tank
Comments (1,459)

Rand Paul Says Foreign Policy Is 'A Complete Non-Issue'

If we do not resuscitate many "dead on arrival" deficit-reduction ideas as part of a larger stimulus compromise, then our economy will be "dead on arrival" -- and distressingly soon.

Read Post | Comments (205)
Harry Reid: Republicans Want Economy To Tank So Democrats Will Be Battered In Midterms

First Posted: 07-14-10 02:43 PM | Updated: 07-14-10 02:43 PM

WHAT'S YOUR REACTION?

Important Funny Typical Scary Outrageous Amazing Innovative Finally

Senate Majority Leader Harry Reid (D-Nev.) said on Wednesday that Republicans want the economy to tank and hope jobs legislation fails so that Democrats will lose the midterm elections.

"They're betting on failure," Reid said of Republicans at a press conference Wednesday focused on measures to help small businesses. "They think that the worse the economy is come November, the better they are going to do election-wise."

Reid says the Republican approach "was indicated very loudly in the health care bill when one Republican Senator said that he wanted this to be Obama's Waterloo -- another Senator said..."
From News Article to

- People (has-people)
  - And their roles
- Places (has-places)
  - And the county, state, country they are in
- Organizations (has-organizations)
  - Government departments, company names, etc.
- Main Categories (has-domains)
  - Politics, sports, ministries, energy, finance, economics, ecology, oil, mining industry, etc..
- Main Concepts (has-main-groups)
  - Other important nouns and phrases in a text
## Information on Senator Dianne Feinstein of California

### Extended Contact Information

**DC Address:** The Honorable Dianne Feinstein  
United States Senate  
331 Hart Senate Office Building  
Washington, D.C. 20510-0504

**DC Phone:** 202-224-3841  
**DC Fax:** 202-228-3954

**Contact Form:**  

**WWW Homepage:**  
http://feinstein.senate.gov/public/

**Twitter:** No Known Twitter Account

### District Offices:

- **750 B Street, Suite 1030**  
  San Diego, CA 92101-8126  
  **Voice:** 619-231-9712  
  **FAX:** 619-231-1108

- **2500 Tulare Street, Suite 4290**  
  Fresno, CA 93721  
  **Voice:** 559-485-7430  
  **FAX:** 559-485-9689

### Political Profile

**Party:** Democrat  
**Leadership:** None  
**Freshman:** No

**First Elected To Office:** November 3, 1992  
**Year of Next Election:** November 6, 2012

**Previous Political Work:** Mayor of San Francisco  
San Francisco Board of Supervisors

**Committee Membership:**
- Senate Committee on Appropriations - 7  
- Subcommittee on Agriculture, Rural Development, Food and Drug Administration and Related Agencies - 3  
- Subcommittee on Commerce, Justice, and Science, and Related Agencies - 5  
- Subcommittee on Defense - 5  
- Subcommittee on Energy and Water Development - Chair  
- Subcommittee on Interior, Environment, and Related Agencies - 2  
- Subcommittee on Transportation, Housing and Urban Development, and Related Agencies - 7  
- Senate Select Committee on Intelligence - Chair
latest LOD cloud
(select-distinct (?text ?person1 ?person2 ?faith) 
  (solr-text ?text "obama healthcare hate" 200) 
  (q ?text !franz:has-gov-person ?person1) 
  (q ?text !franz:has-gov-person ?person2) 
  (q ?person1 !gov:hasParty !gov:Democrat) 
  (q ?person2 !gov:hasParty !gov:Republican) 
  (q ?person1 !gov:hasFaith ?faith) 
  (q ?person2 !gov:hasFaith ?faith))

<table>
<thead>
<tr>
<th>?text</th>
<th>?person1</th>
<th>?person2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Herger getting national</td>
<td>Doris Matsui</td>
<td>Wally Herger</td>
</tr>
<tr>
<td>attention for his push to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>repeal 'Obamacare'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Herger getting national</td>
<td>Doris Matsui</td>
<td>Dan Lungren</td>
</tr>
<tr>
<td>attention for his push to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>repeal 'Obamacare'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Congress Returns to a Full</td>
<td>Nancy Pelosi</td>
<td>John Boehner</td>
</tr>
<tr>
<td>Plate of Economic Issues</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Congress Returns to a Full</td>
<td>Nancy Pelosi</td>
<td>John Boehner</td>
</tr>
<tr>
<td>Plate of Economic Issues</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Explicit Nodes from Query:
- Republican
- Democrat

Explicit Predicates from Query:
- Has Faith
- Has Party
• A little demo?
How scalable is this?
Tomorrow’s Mission Critical, Today.
The latest Intel® Xeon® processor-based servers are providing powerful, scalable, and reliable solutions that let you tackle today’s toughest IT challenges.

Get the details
Franz’s AllegroGraph® Sets New Record - 1 Trillion RDF Triples”

AllegroGraph is the first NoSQL database to achieve the loading of over 1 Trillion RDF Triples – A major step forward in scalability for the Semantic Web.

OAKLAND, Calif. — August 16, 2011 — Franz Inc., a leading supplier of Graph Database technology, with critical support from Stillwater SuperComputing Inc. and Intel, today announced it has achieved its goal of being the first to load and query a NoSQL database with a trillion RDF statements. RDF (also known as triples or quads), the cornerstone of the Semantic Web, provides a more flexible way to represent data than relational database and is at the heart of the W3C push for the Semantic Web.

A trillion RDF Statements eclipses the current state of the art for the Semantic Web data management but is a primary interest for companies like Amdocs that use triples to represent real-time knowledge about telecom customers. Per-customer, Amdocs uses about 4,000 triples, so a large telecom like China Mobile would easily need 2 trillion triples to have detailed knowledge about each single customer.

Bill Guinn, CTO of Amdocs Product Enabler Group recently said, “We run the biggest Oracle installations in the world for telecom companies and we will keep using them in the foreseeable future. However, we just couldn’t make our intelligent Amdocs Intelligent Decision Automation work in a relational database. We needed the flexibility of a triple store to make this new personalized predictive CRM application work. So now we have an application that works with literally tens of other relational databases and unstructured data sources, which turns information into triples so we can raise the bar on customer experience.”
<table>
<thead>
<tr>
<th>Load Test</th>
<th># Triples</th>
<th>Time</th>
<th>Load Rate (T/Sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUBM(8000)*</td>
<td>1.106 Billion</td>
<td>36min, 49 sec</td>
<td>500,679</td>
</tr>
<tr>
<td>LUBM(160,000)*</td>
<td>22.12 Billion</td>
<td>12 hrs, 18m, 16s</td>
<td>499,188</td>
</tr>
<tr>
<td>AllegroGraph Pre-release**</td>
<td>310.269 Billion</td>
<td>78 hrs, 9m, 23s</td>
<td>1,102,737</td>
</tr>
<tr>
<td>AllegroGraph Pre-release***</td>
<td>1.009 Trillion</td>
<td>338 hrs, 5m</td>
<td>829,556</td>
</tr>
</tbody>
</table>

*32 core Intel E5520, 2.0 GHz, with 1 TB RAM, RedHat v6.1.
**64 core Intel x7560, 2.27 GHz, 2TB RAM, 22TB Disk, Redhat v6.1. LUBM-like data.
***240 core Intel x5650, 2.66GHz, 1.28TB RAM, 88TB Disk, Redhat v6.1. LUBM-like data.
Queries

- Query planner now takes 99% of SPARQL 1.0, automatically compiles it into query graph flow language...

```sparql
prefix app: <http://franz.com/phone/>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

select ?a ?b ?c
where {
  ?c app:friend app:jans .
}
```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX dcterms: <http://purl.org/dc/terms/>

SELECT DISTINCT ?title
WHERE {
  ?class rdfs:subClassOf foaf:Document .
  ?doc2 dcterms:references ?bag2
  OPTIONAL {
    ?class3 rdfs:subClassOf foaf:Document .
    ?bag3 ?member3 ?doc
    OPTIONAL {
      ?class4 rdfs:subClassOf foaf:Document .
      ?bag4 ?member4 ?doc3
    } FILTER (!bound(?doc4))
  } FILTER (!bound(?doc3))
}
This will actually work on Prolog with rules too!

(\(\text{male ~}?x\)) \(\text{(q ~}?x \text{ ~}!o:\text{sex} \text{ ~}!o:\text{male})\)

(\(\text{female ~}?x\)) \(\text{(q ~}?x \text{ ~}!o:\text{sex} \text{ ~}!o:\text{female})\)

(\(\text{father ~}?x ~?y\)) \(\text{(male ~}?x\)} \(\text{(q ~}?x \text{ ~}!o:\text{has-child} ~?y)\))

(\(\text{mother ~}?x ~?y\)) \(\text{(female ~}?x\)} \(\text{(q ~}?x \text{ ~}!o:\text{has-child} ~?y)\))

(\(\text{parent ~}?x ~?y\)) \(\text{(or ~}?x ~?y\)} \(\text{(father ~}?x ~?y\)} \(\text{(mother ~}?x ~?y)\))

(\(\text{grandparent ~}?x ~?y\)) \(\text{(parent ~}?x ~?z\)} \(\text{(parent ~}?z ~?y)\))

(\(\text{grandchild ~}?x ~?y\)) \(\text{(grandparent ~}?y ~?x)\))

(\(\text{ancestor ~}?x ~?y\))

\(\text{(or ~}?x ~?y\)}\)

\(\text{\hspace{1cm} (and ~}?x ~?z\)} \(\text{(ancestor ~}?z ~?y)\))\)

(\(\text{descendent ~}?x ~?y\)) \(\text{(ancestor ~}?y ~?x)\))

(\(\text{uncle ~}?x ~?y\))

\(\text{(father ~}?z ~?x)\)

\(\text{(male ~}?x)\)

\(\text{(father ~}?z ~?w)\)

\(\text{(not ~}?x ~?w)\)

\(\text{(parent ~}?w ~?y)\)

; normally the following is executed at run time

; impossible to optimize

\(\text{(select ~}?x\)} \(\text{(uncle ~}?x \text{ ~}o:\text{person17})\))
; unfold, apply statistics, reorder, rewrite.. and
; then executed by Prolog..

(select (?x o:person17)
 (q ?z26 !o:has-child !o:person17)
 (q ?z26 !o:sex !o:male)
 (q ?z26 !o:has-child ?w27)
 (not (= !o:person17 ?w27))
 (or (and (q ?w27 !o:sex !o:male)
         (q ?w27 !o:has-child ?x))
     (and (q ?w27 !o:sex !o:female)
         (q ?w27 !o:has-child ?x))))
The end
A proof of flexibility

- MongoDB API -> MongoGraph
Bob
Alice
Craig
Bill

monograph(707): (add-json "{"type":"phonecall","aCaller":"bob","bCaller":"alice"}" store2)
667486
monograph(708): (add-json "{"type":"phonecall","aCaller":"alice","bCaller":"bill"}" store2)
667491
monograph(709): (add-json "{"type":"phonecall","aCaller":"bill","bCaller":"craig"}" store2)
667496
monograph(710): (add-json "{"type":"phonecall","aCaller":"craig","bCaller":"bob"}" store2)
667501
monograph(711): (mgraph store2)
<6045 type phonecall store2>
<6045 aCaller bob store2>
<6045 bCaller alice store2>
<6045 id 6045 store2>
<6045 mongoblob {"type":"phonecall","aCaller":"bob","bCaller":"alice"} store2>
<6046 type phonecall store2>
<6046 aCaller alice store2>
<6046 bCaller bill store2>
<6046 id 6046 store2>
<6046 mongoblob {"type":"phonecall","aCaller":"alice","bCaller":"bill"} store2>
<6047 type phonecall store2>
<6047 aCaller bill store2>
<6047 bCaller craig store2>
<6047 id 6047 store2>
<6047 mongoblob {"type":"phonecall","aCaller":"bill","bCaller":"craig"} store2>
<6048 type phonecall store2>
<6048 aCaller craig store2>
<6048 bCaller bob store2>
<6048 id 6048 store2>
<6048 mongoblob {"type":"phonecall","aCaller":"craig","b Caller":"bob"} store2>