Graph Database Model supporting RDF

Renzo Angles

Department of Computer Science
Universidad de Chile

AMW’07
General objective

- Call the attention of the RDF community about related work in (graph) databases

- Use the experience of the database community to enhance research on RDF
RDF Data Model (1999)

- Domains (UBL): Resources (URIs), Blank nodes (existencial variables), Literals
- Data structures:

**RDF Triple**

Subject → Predicate → Object

Property / relation (U)

Resource (UB)

Resource (UB) or value (L)

**RDF Graph**

```
:132 author codd
```

"A Relational Model of ….."

"E.F. “Codd""
RDF(S) model features

Schema and data are mixed
RDF(S) model features

A property can occur as the subject or object of another statement
RDF(S) model features

RDF(S) presents formal semantics

INFORMATION!
### RDF from a database perspective

<table>
<thead>
<tr>
<th>Abstraction Level</th>
<th>RDF API</th>
<th>RDF REPOSITORY</th>
<th>RDF DATABASE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Functions</td>
<td>RDF QL (SPARQL) Rule-based Inference engine</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>Triples</td>
<td>Triples RDF Graphs Datasets (SPARQL)</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>File</td>
<td>Native data store Files RDBMS</td>
<td>?</td>
</tr>
</tbody>
</table>
Issues from a database perspective

- The RDF model has a very low level of abstraction
- RDF query languages need support for graph properties
- RDF does not define (formally) notions of integrity constraints
- Improve RDF visualization
- Use of graph data structures and algorithms for secondary memory
Welkin: graph-based RDF visualizer
Tabulator: semantic data browser

URL: [http://dig.csail.mit.edu/issues/tabulator](http://dig.csail.mit.edu/issues/tabulator)

This is release 0.8 of the Tabulator Project. The [live development trunk](http://dig.csail.mit.edu/) is also available.

Title: The Tabulator project


Description: The Tabulator is a generic data browser. It provides a way to browse and query RDF data in a web standard. Adding new views is a snap. The Tabulator also has features for the power user wanting source and written in Javascript. The source can be easily combined with custom web pages to requires Firefox preferences to be set -- see the tabulator help page. The Tabulator is open source.

Developer:
- Joe Prebysh
- Kenny Lu
- Adam Leer

David Li
- type: Person
- based near: 
- family name: Li
- Given name: David

Acquaintance: Tim Berners-Lee
- type: Person
- see also: [http://dbpedia.org/resource/Tim_Berners-Lee](http://dbpedia.org/resource/Tim_Berners-Lee)
- name: Tim Berners-Lee
- requested: [http://dbpedia.org/resource/Tim_Berners-Lee](http://dbpedia.org/resource/Tim_Berners-Lee)
- is acquaintance of: David Li
- James Hollenbach

Personal mailbox: david_li@mit.edu
- mailbox checksum: 41bd96421e8640d7d402c65c598e0d7d0736f8
- name: David Li
Tabulator: visualizing, browsing and querying of RDF data
Tabulator: querying and browsing

```sql
SELECT ?v0 ?v1 ?v2 ?v3
WHERE
{
    <http://dig.csail.mit.edu/2005/ajar/ajaw/data#Tabulator>
    <http://usefulinc.com/ns/doap#developer> ?v0 .
    FILTER ( ?v2 = "Tim Berners-Lee" )
    OPTIONAL {
    }
}
```
### RDF from a database perspective

<table>
<thead>
<tr>
<th>Abstraction Level</th>
<th>RDF API</th>
<th>RDF REPOSITORY</th>
<th>RDF DATABASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td>Functions</td>
<td>RDF QL (SPARQL) / Rule-based Inference engine</td>
<td>(Graph) Query Language / Rule-based inference engine / RDF Integrity Constraints / (Rule-definition language)</td>
</tr>
<tr>
<td>Logical</td>
<td>Triples</td>
<td>Triples / RDF Graphs / Datasets (SPARQL)</td>
<td>(Graph) Database Model</td>
</tr>
<tr>
<td>Physical</td>
<td>File</td>
<td>Native data store / Files / RDBMS</td>
<td>Native data store / RDBMS?</td>
</tr>
</tbody>
</table>
Graph, Hypernode, Hypergraph

$\mathbf{N} = \text{set of simple nodes} \quad \mathbf{H} = \text{set of hypernodes}$

Graph

$V \subseteq \mathbf{N} \quad E \subseteq V \times V$

$G = (V, E)$

Hypernode

$V = \mathbf{N} \cup \mathbf{H}$

Hypergraph

$E \subseteq \mathcal{P}(V)$
Hypernode Model (1990)

- **Motivation:** modeling of complex objects
- **Features:** simple and flexible data structure (hypernode) that supports complex objects and encapsulation of information
Hypergraph Model (1991)

- **Motivation:** modeling of object-oriented features, visualization, browsing
- **Features:** natural formalisation of the notions of sub-object sharing and structural inheritance
Proposal

API (?) Applications (?) Services (?)

Data Structure: Hypergraph + Hypernode

Query Language

GraphQL

Integrity Constraints

Native Data Store RDBMS
Graph Schema

A Graph Schema is a hypergraph \((V, E_d, E_u)\), where

- \(V \subseteq U\) is a set of vertices
- \(E_d \subseteq V \times V\) is a set of unlabeled directed edges (just edges)
- \(E_u\) is a set of labeled undirected hyperedges (just hyperedges) of the form \((c, P)\) where:
  - \(c \in U\) is the hyperedge label
  - \(P \subseteq E_d\) is the set of edges grouped by the hyperedge
Schema for Resources (definition)

A Resource Graph Schema is a graph schema \((V, E_d, E_u)\) with the following interpretation:

- A hyperedge \((c, P) \in E_u\) represents a Resource Class where:
  - \(c\) is the name of the class
  - Each edge \((u, v) \in P\) represents a Property as a pair \((property-name, property-value)\).

- If a resource class \(c_1\) contains a resource class \(c_2\), then \(c_1\) is a subClassOf \(c_2\) (i.e. inheritance of properties)
Schema for Resources (example)
Schema for Properties (definition)

A Property Graph Schema is a graph schema \((V, E_d, E_u)\) with the following interpretation:

- A hyperedge \((c, P) \in E_u\) represents a Property Class where:
  - \(c\) is the name of the class
  - Each edge \((u, v) \in P\) represents a valid pair \((domain, range)\) for the property class

- If a property class \(c_1\) contains a property class \(c_2\), then \(c_1\) is a \textit{subPropertyOf} \(c_2\) (i.e. inheritance of domains and ranges)
Schema for Properties (example)
Nested Graph (Hypernode)

A Nested Graph (nGraph) is defined recursively as a triple $(n, V_n, E_n)$ where:

(a) $n \in U$ is the name of the nGraph

(b) $V_n$ is a finite set of vertices, such that, each $v \in V_n$ satisfies that $v \in U \cup L$ or $v$ is a nGraph

(c) $E_n \subseteq V_n \times V_n$ is a finite set of unlabeled directed edges, such that, each triple $(u, v) \in E_n$ satisfies that:

(i) $u \in U$ or $u$ is a nGraph
(ii) $v \in U \cup L$ or $v$ is a nGraph
Resource Graph Instance (definition)

A Resource Graph Instance is a pair $(V_l, E_l)$, where:

1. $V_l$ is a finite set of nGraphs, and
2. $E_l$ is a finite set of labeled undirected hyperedges (just hyperedges) of the form $(c, R)$ satisfying that:
   - $c \in U$ is a resource class name from the schema
   - $R \subseteq V_l$ is the set of nGraphs of type $c$

Restrictions:

(C1) Given two nGraphs $(n, V_n, E_n), (n', V_{n'}, E_{n'}) \in V_l, \ n = n'$ implies that $V_n = V_{n'}$ and $E_n = E_{n'}$

(C2) $\forall (n, V_n, E_n) \in V_l$ it is not the case that $n \in V_{n'}$ for any $(n', V_{n'}, E_{n'}) \in V_l$
Resource Graph Instance

Hyperedge  Hypernode

#Artist  #Painter  #Artifact  #Painting

#goya.xml  #picasso.xml  #guernica.jpg

Iname  "Goya"  "Pablo"  "Picasso"

#Creates  #Paints  #Exhibited  #Technique

"oil on canvas"

#saturn.jpg  #woman.jpg
### Querying RDF data

<table>
<thead>
<tr>
<th>Graph notion</th>
<th>Real-life Application Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjacency</td>
<td>All relatives of degree one of Alice (adjacent nodes in a genealogy database)</td>
</tr>
<tr>
<td></td>
<td>What chemical composes does a given chemical reaction produce? (Adjacent edges in chemical information)</td>
</tr>
<tr>
<td></td>
<td>What cities are near Athens? (neighborhood in a tourism system graph)</td>
</tr>
<tr>
<td>Degree of a node</td>
<td>What is/are the most cited paper/s? (searching node/s with maximum in-degree in a database of bibliographic cites)</td>
</tr>
</tbody>
</table>
### Querying RDF data (cont)

<table>
<thead>
<tr>
<th>Graph notion</th>
<th>Real-life Application Query</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Paths</strong></td>
<td>Are suspects A and B related? (relevant paths in a police database)</td>
</tr>
<tr>
<td></td>
<td>What is the shortest route between city A and city B? (Shortest path in a database of roads)</td>
</tr>
<tr>
<td></td>
<td>What is the influence of article D? (transitive closure in database of bibliographic cites)</td>
</tr>
<tr>
<td><strong>Distance</strong></td>
<td>What is the Erdös distance between author X and author Y? (distance between nodes in a collaboration graph)</td>
</tr>
<tr>
<td><strong>Pattern Matching</strong></td>
<td>Where and how much a motif (pattern) appears? (Pattern matching in genome data)</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>Adjacent Nodes</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>RDF Query Language</td>
<td>RQL</td>
</tr>
<tr>
<td></td>
<td>SeRQL</td>
</tr>
<tr>
<td></td>
<td>RDQL</td>
</tr>
<tr>
<td></td>
<td>Triple</td>
</tr>
<tr>
<td></td>
<td>N3</td>
</tr>
<tr>
<td></td>
<td>Versa</td>
</tr>
<tr>
<td></td>
<td>RxPath</td>
</tr>
<tr>
<td></td>
<td>Sparql</td>
</tr>
<tr>
<td>Graph Query Language</td>
<td>G</td>
</tr>
<tr>
<td></td>
<td>G+</td>
</tr>
<tr>
<td></td>
<td>Graph Log</td>
</tr>
<tr>
<td></td>
<td>Gram</td>
</tr>
<tr>
<td></td>
<td>Graph DB</td>
</tr>
<tr>
<td></td>
<td>Lorel</td>
</tr>
<tr>
<td></td>
<td>F-G</td>
</tr>
</tbody>
</table>
Current and future work

- API (?)
- Applications (?)
- Services (?)

Data Structure:
- Hypergraph + Hypernode

Query Language:
- GraphQL

Integrity Constraints

Native Data Store

RDBMS
Thanks

Questions?