Semantic Data Persistence - triple/quad stores, application access to ontologies

Jana Ahmad

Czech Technical University

ahmadjan@fel.cvut.cz

January 10, 2019

Outlines

- Short review
  - Semantic web
  - RDF
  - SPARQL
- Triple Stores
  - Indexing Approaches
    - Vertical Table
    - Property Table
    - Horizontal Table
    - Mapping Dictionary
  - Example of Existing Triple store:
    - Sesame
    - Jena TDB
    - GraphDB
    - Blazegraph, and others....
- Applications to access ontology
  - OWLAPi

Review

Semantic Web

"The Semantic Web is a web of data that, provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries." [w3.org]

SW lets us:

- Represent the knowledge
- Support search queries on knowledge
- Support inference

RDF:Resource Description Framework

Figure: RDF Graph

RDF:Resource Description Framework

- A way to structure and link data
- A data model that lets us make statements about Web resources in the form of subject-predicate-object sentences, called triples
  - Subject denotes the resource
  - Object
  - Predicate (aka the property) expresses a subject-object relationship
- RDF Graph is a set of RDF triples
- RDF Term is either an IRI, a blank node, or a literal

SPARQL: Query and Update the Graph

- SPARQL Query Type
  - SELECT: returns a binding table (similarly to SQL)
  - ASK: returns a true/false indicating existence of the given pattern in the RDF graph
  - CONSTRUCT: returns an RDF graph constructed from the binding table
  - DESCRIBE: returns an RDF graph describing the given resource (semantics not fixed
- SPARQL Query Update
  - Insert
  - Delete

Indexing Approaches

Triple Store

- What is triple store
- How triple store stores RDF data
- What is the best triple store

Triple Store Overview

- Triple Stores are tools for RDF Data Management
- Framework used for storing and querying RDF data.
- Triple Stores must support SPARQL

Triple Tables Approaches

How triple store stores RDF data

- The most straightforward mapping of RDF into a relational database system
- Each triple given by (s, p, o) is added to one large table of triples with a column for the subject, predicate, and object respectively
- Indexes are then added for each of the columns.

Triple Store Tables

How triple store stores RDF data

- The most straightforward mapping of RDF into a relational database system
- Each triple given by (s, p, o) is added to one large table of triples with a column for the subject, predicate, and object respectively
- Indexes are then added for each of the columns.
    - Triple Tables
    - Property Tables
    - Vertical Portioning
    - Mapping Dictionary

Triple Table

Idea

Stores each triple in a three-column table (s, p, o)

Subject	**Predicate**	**Object**
Anna	loves	Jan
Anna	hates	John
Jan	loves	Anne

Table: Triple Table example

Property Table

Idea

Stores triples with the same subject as a n-ary table row, where predicates are modeled as table columns.
Each table includes a subject and all matched properties

Subject	**loves**	**hates**
Anna	Jan	John
John	Jana	Anna
Jan	Anna	John

Table: Property Table example

Property Tables

- Each database table includes a column for a subject and several fixed properties.
- The intent is that these properties often appear together on the same subject.
- Advantage
  - This approach eliminates many of the expensive self-joins in a triples table
- Disadvantage
  - Multi-valued properties are problematic in this approach
  - Null values

Vertical Partitioning Table

Idea

Stores triples with the same property in one table

- hates

Subject	Object
Anna	Jan
John	Jana
Jan	Anna

Table: Vertical Table example

- loves

Subject	Object
Ann	Jan
John	Jana
Jan	Ann

Table: Vertical Table example

Mapping Dictionary

- Replacing all literals by unique IDs using a mapping dictionary
- Removes redundancy and helps in saving a lot of space
- We can concentrate on a logical index structure rather than the physical storage design

Mapping Dictionary

- Replacing all literals by unique IDs using a mapping dictionary

ID	Value
Anna	1
Jan	2
John	3
Jana	4
loves	5
hates	6

Table: Mapping table

Subject	Predicate	Object
1	5	2
1	6	3
2	5	1

Table: Tribe Table example with Mapping Dictionary

Quad Stores

Quad Stores

- It extend triple table stores with one more column for representing the context (named graph) in which the triple resides, i.e. (S,P,O,C),
- The graph name corresponds normally with the name-space of the ontology.

UFO Index

Objective

- How proper Foundational-Ontology-Based Knowledge can be used to design a generic index
- Optimizes RDF query by means of an index based on the actual knowledge structure provided by Conceptual Model-Based Ontology

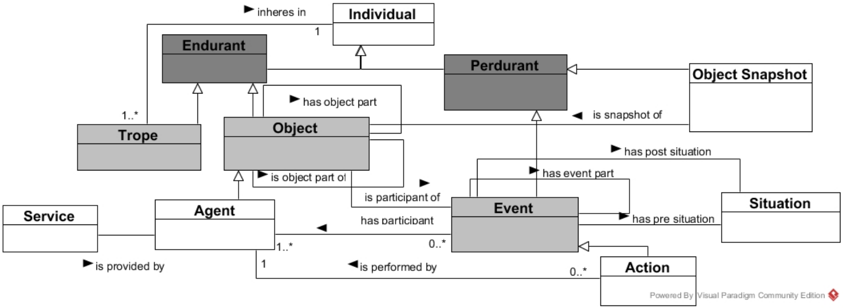


Figure: UFO Concepts

UFO Index Tables Approaches

- Triple Table
- Property Tables
- Vertical Partitioning

Triple Table

Subject	Predicate	Object
Event-i	has-participant	Agent-i
Process-i	is-event-part-of	Event-i
Person-i	is-participant-of	Event-i
Agent-i	performs	Action-i
Action-i	is-performed-by	Agent-i
En	*Pn*	*On*

Perdurant Table

Subject	Predicate	Object
Event-i	has-participant	Agent-i
Process-i	is-event-part-of	Event-i
Action-i	is-performed-by	Agent-i

Endurant Table

Subject	Predicate	Object
Person-i	is-participant-of	Event-i
Agent-i	performs	Action-i
En	*Pn*	*On*

Figure: Triple Table

Property Table

Endurant Property Table

Subject	is-object-part-of	is-participant	performs	has-trope
Object-i	Object-ii	Event-i	Event-i	Trope-i

Perdurant Property Table

Subject	is-event-part-of	has-participant	is-performed-by	is-snapshot-of
Event-i	Event-i	Person-i	Agent-i	
Object-Snapshot				Object-i

Figure: Property Table

Vertical Partitioning

has-object-part Predicate Table

Object-i	Object-ii
Object-i	Object-iii

Figure: Vertical Partitioning Table

Example of Existing Triple store

Triple Stores

Figure: Examples of Triples Stores

Example of Existing Triple store

- RDF4J
- GraphDB
- Apache Jena
- Blazegrah and others ...

Example of Existing Triple store

Triple Stores

- A triple store is normally a synonym for an RDF store.

Quad Stores

- It extend triple table stores with one more column for representing the context (named graph) in which the triple resides, i.e. (S,P,O,C),
- The graph name corresponds normally with the name-space of the ontology.

Quad Stores

Example

OSPC index means that the index table contains triples sorted according to object, then according to subject, then predicate and then context. This index is suitable for searching data given an object (i.g. matching the BGP ?x ?y :a), or object+subject (e.g. matching the BGP ?x :p :a).

Triple Stores Types

Native Triple Store

RDF stores that implement their own database engine without reusing the storage and retrieval functionalities of other database management systems (e.g., Apache Jena TDB, GraphDB,AllegroGraph, etc.).

Non-Native Triple Store

RDF Stores that use the storage and retrieval functionality provided by another database management system (e.g., Apache Jena SDB, Semantics Platform, etc.)

Hybrid Stores

RDF Stores that supports both architectural styles (native and DBMS-backed) (e.g., RDF4J (Sesame), OpenLink Virtuoso Universal Server, etc.)

RDF4J

DRF4J (former Sesame) an open source Java framework for processing RDF data. Link: `http://rdf4j.org/`

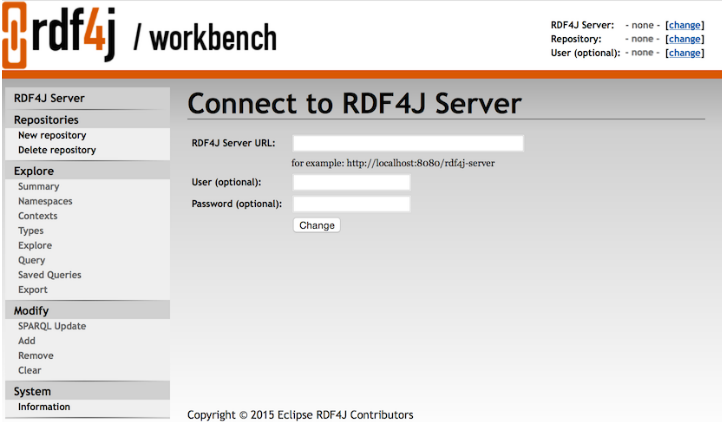


Figure: RDF4J Interface

RDF4J Architecture

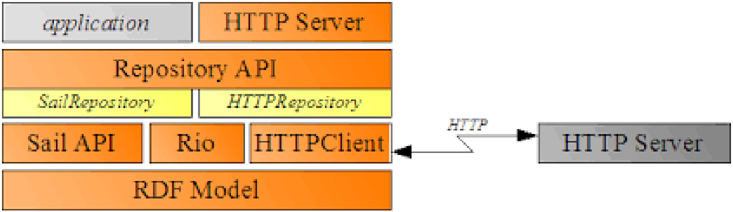


Figure: RDF4J Architecture

- Rio (RDF I/O)
    - Parsers and writers for various notations
- Sail (Storage And Inference Layer)
- Repository API
- HTTP Server
    - Accessing Sesame through HTTP

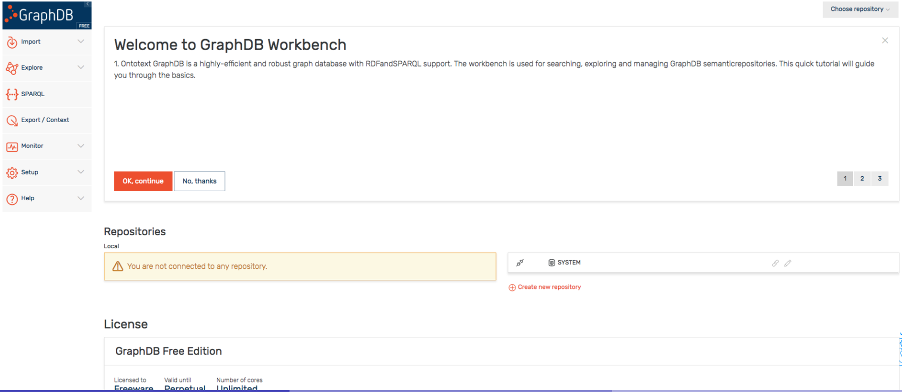
RDF4J Installation

- Simple web interface for storing and querying RDF data
- Install steps (no admin rights needed)
    - Download and unzip newest RDF4J and Tomcat
    - Copy all *.war files from RDF4Js war folder to Tomcats webapps folder
    - Start Tomcat
        - From bin folder by running startup.sh (UNIX) or startup.bat (Win)
    - Go to http://localhost:8080/rdf4j-workbench

More information: `http://docs.rdf4j.org/server-workbench-console/`

GraphDB

(formerly OWLIM) is a leading RDF Triple store built on OWL (Ontology Web Language) standards. GraphDB handles massive loads, queries and OWL inferencing in real time.
Link:http://ontotext.com/products/graphdb/,http://graphdb.ontotext.com/



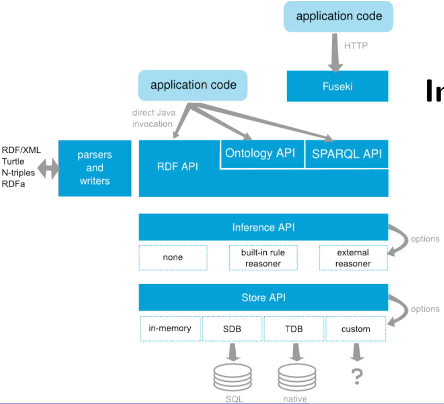
GraphDB Installation

- Download installation file and run it (https://ontotext.com/products/graphdb/)
- Access workbench via http://localhost:7200/

More information: `http://graphdb.ontotext.com/documentation/free/running-graphdb.html`

Apache Jena

Is a Java framework (collection of tools and Java libraries) to simplify the development of Semantic Web and Linked Data applications.
Link: `http://jena.apache.org/index.html`



Includes

1. RDF API for processing RDF data in various notations
2. Ontology API for OWL and RDFS
3. Rule-based inference engine and Inference API
4. TDB a native triple store
5. SPARQL query processor (called ARQ).
6. Fuseki a SPARQL end-point accessible over HTTP

Blazegraph

- Blazegraph:(former Bigdata)(open-source and commercial license) is ultra-scalable, high-performance graph database with support for the RDF/SPARQL APIs.
- Blazegraph is available in a range of versions that provide solutions to the challenge of scaling graphs. Blazegraph solutions range from millions to trillions of edges in the graph.

More information: Link: `https://www.blazegraph.com/product/`

Applications to Access Ontology

Applications to Access Ontology

Low-level APIs

1. OWLAPI
2. JENA
3. RDF4J-API
4. and others ......

High-level APIs

1. JOPA
2. JAOB
3. and others ......

OWLAPI

Standard Java API for accessing/ parsing OWL 2 ontologies. (Protege and Pellet reasoner)

Example (OntologyLoding)

```
OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
// Load an ontology from the Web
IRI iri = IRI.create("http://www.co-ode.org/ontologies/pizza/pizza.owl");
OWLOntology pizzaOntology = manager.loadOntologyFromOntologyDocument(iri);
System.out.println("Loaded ontology: " + pizzaOntology);
```

Example (ShowClasses)

```
OWLOntologyManager m = create();
OWLOntology o = m.loadOntologyFromOntologyDocument(pizza_iri);
assertNotNull(o);
// Named classes referenced by axioms in the ontology.
for (OWLClass cls : o.getClassesInSignature())
System.out.println(cls);
```

For more information. Link: http://owlapi.sourceforge.net

JENA

- Jena is a Java API which can be used to create and manipulate RDF graphs.
- The biggest disadvantage of Jena is that it does not support OWL 2, its primary focus is on RDF
- In Jena, a graph is called a model and is represented by the model interface.

Example (Read a RDF from a file and write it out)

```
// create an empty model
Model model = ModelFactory.createDefaultModel();
// use the FileManager to find the input file
InputStream in = FileManager.get().open( inputFileName );
if (in == null) {
throw new IllegalArgumentException(
"File: " + inputFileName + " not found");
}
// read the RDF/XML file
model.read(in, "");
// write it to standard out
model.write(System.out);
```

JENA

Example (Representing a RDF graph)

```
static String personURI = "http://somewhere/JohnSmith";
static String fullName = "John Smith";
// create an empty Model
Model model = ModelFactory.createDefaultModel();
// create the resource
Resource johnSmith = model.createResource(personURI);
// add the property
johnSmith.addProperty(VCARD.FN, fullName);
```

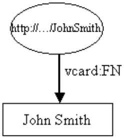


Figure: RDF Graph

For more information. Link: `http://jena.apache.org`

JOPA: Java Ontology Persistence API

- JOPA is a Java OWL persistence framework aimed at efficient programmatic access to OWL2 ontologies and RDF graphs in Java.
- Through instances of EntityManager the application can query and manipulate the object model and the requests are seamlessly transformed to the underlying ontology

Example (JOPA)

```
 EntityManager em = factory.createEntityManager();
Person person1 = em.find("http://example.org/person1");
person1.setHasName("John");
```

For more information. Link: https://sourceforge.net/projects/jopa/

JOPA and JENA

Example (JOPA)

```
 EntityManager em = factory.createEntityManager();
Person person1 = em.find("http://example.org/person1");
person1.setHasName("John");
```

Example (JENA)

```
Model m = ModelFactory.getModel("http://example.org/personal");
Resource i = m.getResource("http://example.org/person1");
i.addProperty(ResourceFactory.getProperty("http://example.org/hasName"),"John");
m.close();
```

The End