Jena: Implementing the Semantic Web Recommendations

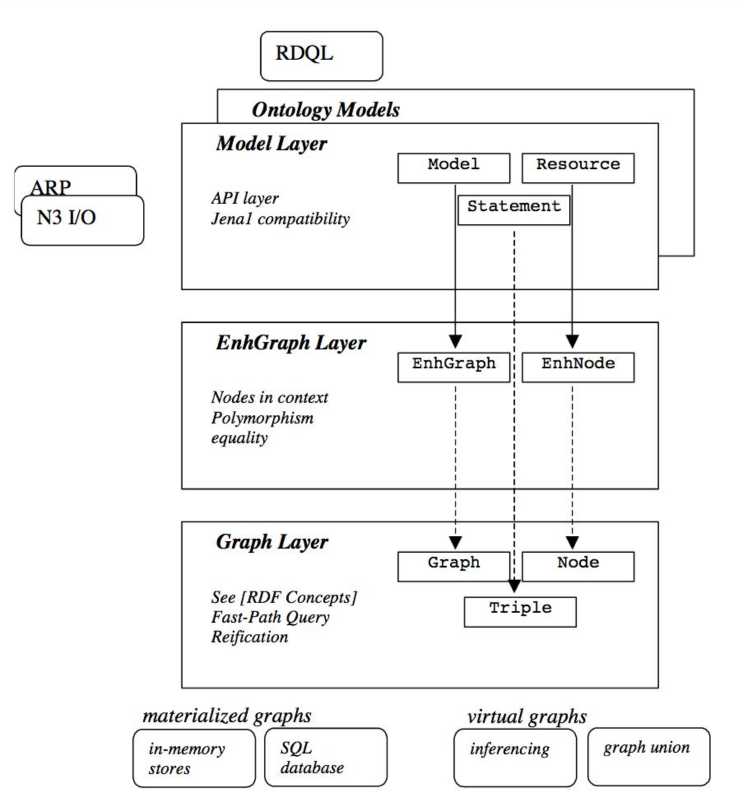
Jeremy J. Carroll, Ian Dickinson, Chris Dollin

Introduction

- RDF = Resource Description Framework
- OWL = Web Ontology Language
- Both together form a standardization for a simple triple-based representation of knowledge.
- RDF triple is <s, p, o>
- Jena is a Semantic Web Toolkit.
  - It has a graph as its core interface.
  - It provides rich API for dealing with RDF.
  - It supports RDQL (RDF Data Query Language).
- RDFS and OWL provide the vocabulary, schema and ontology.

Architecture Overview

- Jena architecture is mainly composed of three layers
- The Graph Layer:
  - It is based on RDF (set of triples of nodes).
  - It has a triple store (in-memory, persistent).
  - It has virtual triples resulting from inference on other triples.
- The Model Layer:
  - It is the abstraction of the RDF graph that is used by application programmers.
- The EnhGraph Layer:
  - It is the intermediate layer between Graph and Model layers.
  - It provides views for the graph and nodes.



Graph Layer

The Graph Layer

- Graph is composed of triples <Subject, Predicate, Object>.
- A triple's node represents RDF URI label, a blank node (bNode which is an anonymous resource), or a literal.
    - <Michael, studiesAt, UW>
- The restriction that a literal can only appear as an object and a property must be URI are forced by Model Layer not Graph Layer.
- Graph interface supports modifications (add, delete) and access (list all triples).
- find(Node s, Node P, Node O) returns an iterator on all triples matching <s,p,o>.

Fast Path Query

- Each graph has a query handler that manages complex queries.
- It implements complex query in terms of “find” primitive.
- Query consists of triples patterns to be matched against graphs.
- Ex: (?x P ?y) (?y Q ?z)
    - All possible bindings of the variables are returned.
- Jena’s memory-based Graph model implements this query by simply iterating over the graph using “find”.
- RDB-based graphs can alternatively compile queries into SQL to get the results from DB-engine.

Model Layer

APIs

- Graph layer only provides triples.
- This is not easy to work with within the application level.
- The Model layer has an API that acts as the presentation layer over graph.
- *Resource* is an abstraction corresponding to *rdfs:resource*.
  - It is represented as a URIref or bNode
  - It provides a view to a collection of facts about the node
- Ex: a URIref having a type *rdf:Bag* provides a view on the node that allows access to specific triples related to it.

Enhanced Graph Layer

Presentation Layers and Personalities

- Each presentation layer has:
  - Interfaces
  - Implementation classes
  - Mapping from interfaces to methods invoking the classes (*Personality*)
- Implementation classes extends EnhGraph or EnhNode
  - EnhGraph is a wrapper around Graph with a pointer to *personality*
  - EnhNode is a wrapper around Node with a pointer to EnhGraph

Polymorphism

- RDFS permits resources to have multiple types (*rdfs:SubClassOf*) which acts as multiple-inheritance.
- Java objects can only have one class.
- Given a Node (EnhNode) and *personality* it is possible to create a view.

Inference Support

- Inference engines consist of *reasoners* (Graph combinators):
  - Combine RDF Graphs (ontology - instances)
  - Expose entailments as another RDF Graph
  - Virtual entailments rather than materialized data
- It enables stacking reasoners after each others (flexibility).
- RDQL queries can be applied to inferred graphs.
- External reasoners are easily registered into the system.

RDQl-RDF Query

- RDQL = RDF Data Query Language
- RDQL query consists of a graph pattern (list of triple patterns)
  - pattern : URIs and named variables (?x)
  - constraints on values of variables
- RDQL can include virtual triples
- No distinction between:
  - Ground triples
  - Virtual triples

```
SELECT ?x
WHERE (?x,
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
<http://example.com/someType>)
```

Persistent Storage

- Persistency is supported by using a conventional database.
- Each triple is stored in a general-purpose triple table or property table.
- Jena uses a denormalized schema:
  - URIs, literals are stored directly in the triple table.
  - Separate literals table is used for storing large literals.
  - This enables the processing of many queries without using join.
  - It trades off time with space (more space for denormalization).
- Common namespaces prefix in URIs are stored in a separate table.
- Property tables:
  - They hold statements for a specific property.
  - They are stored as subject-value pairs.
  - Property table and triple table are disjoint (triple is stored once).
  - Property class table stores properties associated with a particular class with all of its instances.

- Queries are executed on graphs that can span multiple statement tables.
- Each statement table has a handler to convert between Jena graph view and the SQL tuple view.
- The query processor passes the triple pattern to each table handler for evaluation.
- Jena supports fast path query with a goal to use the database engine to process the entire query instead of single patterns.
    - Case one: all triple patterns access only triple table.
    - Case two: all triple patterns can be evaluated on a single property table.

Joseki

- Joseki is a webAPI for Jena.
- It provides a remote API that is simple to use.
- Access mechanism is graph-based query where the target is a remote knowledge base and the result is a graph.
- Client does not know what happens on the server. They just communicates through queries and expect results.
- A host repository can have different graphs.
- The webAPI requires each graph to have a different URL.
- HTTP is used as the protocol for querying the RDF.

Thanks