ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

RDF stores and data persistence

Petr Křemen

Ontologies and Semantic Web
Winter 2021

Outline

- RDF stores
  - GraphDB
  - StarDog
- Indexing Approaches
  - Vertical Table
  - Property Table
  - Horizontal Table
  - Mapping Dictionary
- Programmatic Access to Ontologies
  - low-level APIs - Jena, OWLAPI
  - high-level APIs - JOPA

RDF store

- SPARQL API
- often REST API
- indexing crucial, e.g.
  - SPOC
  - POSC
- more indexes
  - faster queries,
  - slower updates,
  - bigger disk footprint

Triple store

subject	predicate	object
:John	:loves	:Peggy
:Peggy	rdf:type	:Person
…	…	…

Quad store

subject	predicate	object	context
:John	:loves	:Peggy	:people
:Peggy	rdf:type	:Person	:people
…	…	…	…

Triple Table

subject	predicate	object
:John	:loves	:Peggy
:Peggy	rdf:type	:Person
:Mary	:loves	:George
:John	rdf:type	:Man
…	…	…

- + simple implementation
- - eliminates self-joins

Property Table

subject	:loves	rdf:type
:John	:Peggy	:Man
:Peggy		:Person
:Mary	:George	
…	…	…

- + eliminates self-joins
- - null values
- - single-valued properties

Vertical partioning table

subject	object
:John	:Peggy
:Mary	:George
…	…

:loves

subject	object
:Peggy	:Person
:John	:Man
…	…

rdf:type

- + eliminates self-joins
- - null values
- - single-valued properties

Mapping dictionary

subject	predicate	object
3	1	4
4	2	5
6	1	7
3	2	8
…	…	…

id	node
:1	:loves
:2	rdf:type
:3	:John
:4	:Peggy
…	…

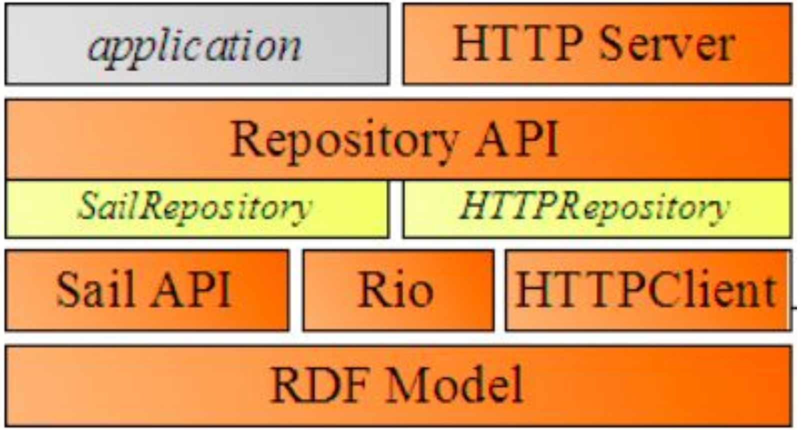
- + removes redundance
- - saving space

Triplestores

- RDF4J
- GraphDB
- Virtuoso
- Fuseki
- Stardog
- AllegroGraph
- Amazon Neptune
- BlazeGraph
- …

RDF4J-based triple store (triple table)

- **Memory Store** (speed)
  - transactional RDF database using main memory with optional persistent sync to disk.
- **Native Store** (scalability, consistency)
  - transactional RDF database using direct disk IO for persistence.
  - B-Trees
- **ElasticsearchStore** (fast for read-only scenarios)
  - experimental RDF database that uses Elasticsearch for storage.
  - Elastic indexing



taken from https://graphdb.ontotext.com/documentation/free/architecture-components.html

RDF4j Inferencing

- **Full materialization**
  - upon save data inference rules are run and new triples inferred which are then stored together with original triples
- non-complete for OWL entailment regimes

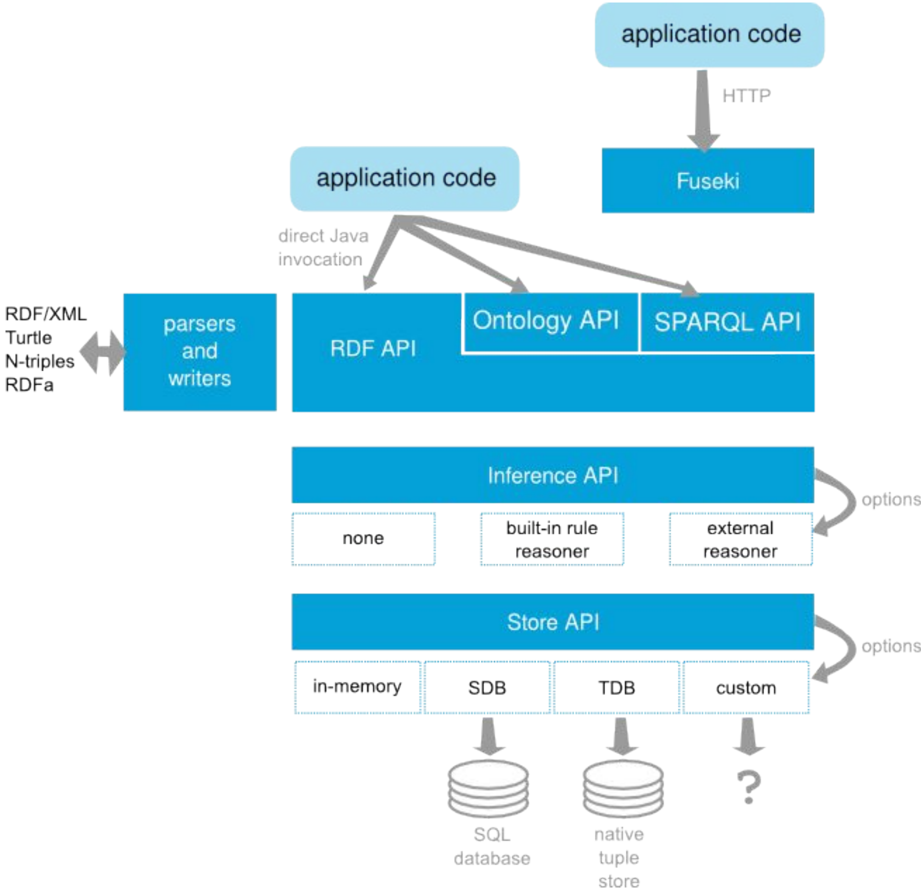
subject	predicate	object	context
:John	:loves	:Peggy	:people
:loves	rdf:type	owl:Symmetric Property	:people

```
Id: prp_symp

        a <rdf:type> <owl:SymmetricProperty>
        b a c
      -----------------------------------
        c a b [Constraint a != <blank:node>]
```

subject	predicate	object	context
:Peggy	:loves	:John	*explicit*

Jena + Fuseki

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



StarDog inferencing

- **Runtime Query Execution**
  - upon query execution new data are infered
- slower for queries
- faster for updates

subject	predicate	object	context
:John	:loves	:Peggy	:people

:loves <rdf:type> <owl:SymmetricProperty>

subject	predicate	object	context
:Peggy	:loves	:John	*explicit*

Application access to ontologies

Low-level vs. High-level APIs

- **Low-level APIs**
  - OWLAPI
  - JENA
  - RDF4J-API
  - ....
- **High-level APIs**
  - JOPA
  - JAOB
  - ....

work with individual statements

work with objects

OWLAPI

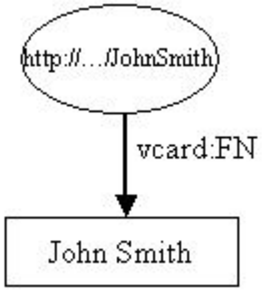
- **Reference implementation of OWL 2**
  - complete
  - pluggable architecture for reasoners

```
OWLOntologyManager m = create();

OWLOntology o = m.loadOntologyFromOntologyDocument(pizza_iri);

for (OWLClass cls : o.getClassesInSignature()) {

     System.out.println(cls);

}
```

JENA

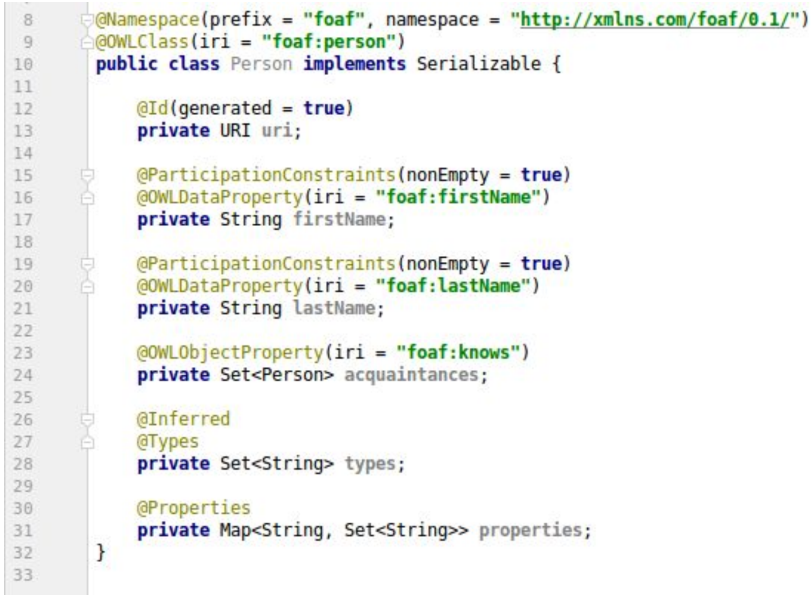
- **Long-history implementation of RDF**
  - complete
  - extended towards OWL (but incomplete support)
  - wide use

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



Java OWL persistence API (JOPA)

- Annotation-based object-ontological mapping
- Inheritance, inferred knowledge access
- Query API with automatic mapping to entities
  - JPQL-like query language also available
- Access to unmapped types and properties
- Transactions, second-level cache
- Integrity constraints
  - Mapping definition, validation of participation constraints at runtime
- Object model generator (OWL2Java)



```java
@Namespace(prefix = "foaf", namespace = "http://xmlns.com/foaf/0.1/")
@OWLClass(iri = "foaf:person")
public class Person implements Serializable {

    @Id(generated = true)
    private URI uri;

    @ParticipationConstraints(nonEmpty = true)
    @OWLDataProperty(iri = "foaf:firstName")
    private String firstName;

    @ParticipationConstraints(nonEmpty = true)
    @OWLDataProperty(iri = "foaf:lastName")
    private String lastName;

    @OWLObjectProperty(iri = "foaf:knows")
    private Set<Person> acquaintances;

    @Inferred
    @Types
    private Set<String> types;

    @Properties
    private Map<String, Set<String>> properties;
}
```

https://github.com/kbss-cvut/jopa