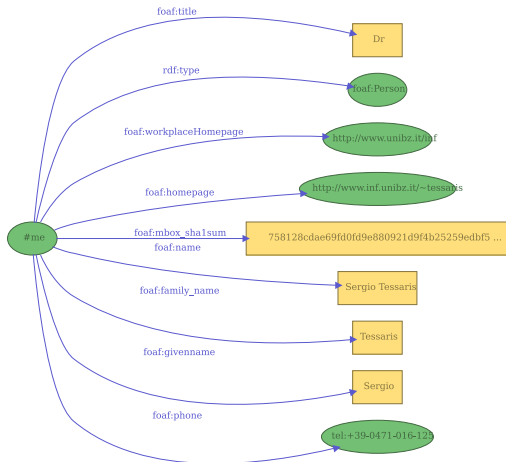


RDF Data Model and Query Languages

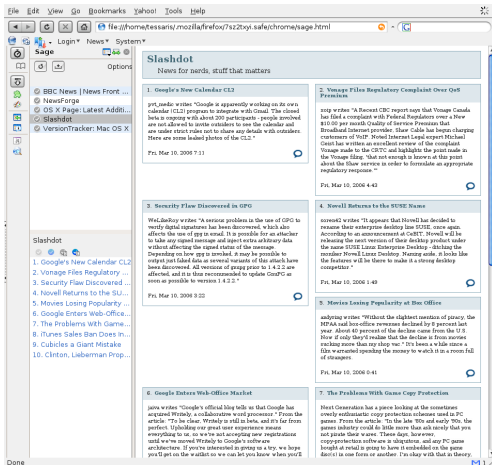
Sergio Tessaris

10 March 2006

FOAF example



RDF Site Summary (RSS) 1.0



- 1 Introduction
 - Building Blocks
 - RDF Abstract Syntax
 - RDF Vocabulary
- 2 RDF Semantics
 - RDF model theory
 - Entailment
 - Casting RDF into FOL
- 3 Querying RDF
 - Introduction
 - Graph Patterns
 - Query languages

Basic Concepts

- **RDF**: language for representing information about resources (e.g. Metadata)

Basic Concepts

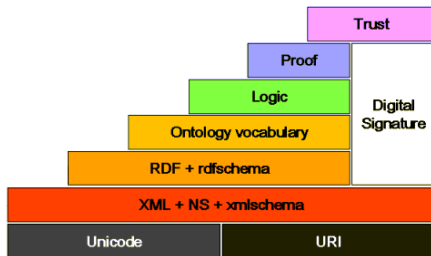
- **RDF**: language for representing information about resources (e.g. Metadata)
- information about things *identified* on the Web
 - identifiable doesn't mean *retrievable*
 - e.g. goods from an eShop, prices, availability, etc.

Basic Concepts

- **RDF**: language for representing information about resources (e.g. Metadata)
- information about things *identified* on the Web
 - identifiable doesn't mean *retrievable*
 - e.g. goods from an eShop, prices, availability, etc.
- information processed by applications, not human beings
 - Semantic Web

Digression: Semantic Web

The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.
Tim Berners-Lee, James Hendler, Ora Lassila, The Semantic Web, Scientific American, May 2001



Design Goals

- having a simple data model
- having formal semantics and provable inference
- using an extensible URI-based vocabulary
- using an XML-based syntax
- supporting use of XML schema datatypes
- allowing anyone to make statements about any resource

RDF Statements

RDF is about making statements about resources

- E.g. Sergio Tessaris is the author of the web page
`http://www.inf.unibz.it/~tessaris/index.html`
- This can be stated as a property of the web page
`http://www.inf.unibz.it/~tessaris/index.html` has
an author whose value is "Sergio Tessaris"
- RDF statements
 - **subject**: e.g. URL
`http://www.inf.unibz.it/~tessaris/index.html`
 - **predicate**: e.g. property `author`
 - **object**: e.g. string `"Sergio Tessaris"`

Identifying Resources

- RDF identifiers: **Uniform Resource Identifiers** (URI)
- URIs, URLs, and URNs
 - **URL** identifies resources via a representation of their primary access mechanism
 - **URN** URIs that are required to remain globally unique and persistent

Example

```
ftp://ftp.is.co.za/rfc/rfc1808.txt  
http://www.math.uio.no/faq/compression-faq/part1.html  
news:comp.infosystems.www.servers.unix  
telnet://melvyl.ucop.edu/  
mailto:someone@example.com
```

Literals

- RDF allows the use of values
 - strings
 - numbers
 - booleans
- **Literals** are basically UNICODE strings
 - plain just strings (w optional language tag)
 - typed have associated datatype URI
- RDF literals and typing
- literals are **not** URIs
 - e.g. "<http://www.unicode.org>" and <http://www.unicode.org> are different

Triples and Graphs

- **RDF triples** basic element of RDF model
 - subject
 - predicate
 - object

Triples and Graphs

- **RDF triples** basic element of RDF model
 - subject
 - predicate
 - object
- triple: two nodes (subject, object) connected by a labelled edge (predicate)

Triples and Graphs

- **RDF triples** basic element of RDF model
 - subject
 - predicate
 - object
- triple: two nodes (subject, object) connected by a labelled edge (predicate)
- set of triples: a labelled directed graph

Blank Nodes

- RDF graphs may contain additional nodes
- arbitrary set of **blank nodes** (bnodes)
 - infinite
 - disjoint from URIs and literals
- given two blank nodes it is possible to determine whether or not they are the same

Blank Nodes

- RDF graphs may contain additional nodes
- arbitrary set of **blank nodes** (bnodes)
 - infinite
 - disjoint from URIs and literals
- given two blank nodes it is possible to determine whether or not they are the same
- **intuition**: a bnode represents the existence of something

Scope of RDF Terms

- URIs and Literals have a global scope
- two equal URIs (Literals) always represent the same object
 - equal means that the two unicode strings are the same
 - there are not contextual to the graph
- bnodes are contextual to the graph in which they appear

Scope of RDF Terms

- URIs and Literals have a global scope
- two equal URIs (Literals) always represent the same object
 - equal means that the two unicode strings are the same
 - there are not contextual to the graph
- bnodes are contextual to the graph in which they appear
- ... more to come on the role of bnodes

RDF Graphs

- RDF triple
 - subject: RDF URI reference or a **bnode**
 - predicate: RDF URI reference
 - object: **literal**, RDF URI reference or a **bnode**

RDF Graphs

- RDF triple
 - **subject**: **RDF URI** reference or a **bnode**
 - **predicate**: **RDF URI** reference
 - **object**: **literal**, **RDF URI** reference or a **bnode**
- RDF graph: set of RDF triples
 - **nodes** of an RDF graph are subjects and objects in the triples

RDF Graphs (cont.)

- no literals as subject
- predicates are just URIs
- URIs are used for both resources (nodes) and predicates (edges)
- literals can be non well formed datatypes
- no complete information about any resource

On Bnodes

- bnodes are different from other RDF terms
- starting from the syntax

On Bnodes

- bnodes are different from other RDF terms
- starting from the syntax
- Graph equivalence

Definition

G , G' are equivalent if there is a bijection M between the nodes of the two graphs, s.t.:

- 1 M maps bnodes to bnodes.
- 2 $M(lit) = lit$ for literals lit in G .
- 3 $M(uri) = uri$ for URI in G .
- 4 $\langle s, p, o \rangle$ in G iff the triple $\langle M(s), p, M(o) \rangle$ in G' .

On Bnodes

- bnodes are different from other RDF terms
- starting from the syntax
- Graph equivalence

Definition

G , G' are equivalent if there is a bijection M between the nodes of the two graphs, s.t.:

- 1 M maps bnodes to bnodes.
- 2 $M(lit) = lit$ for literals lit in G .
- 3 $M(uri) = uri$ for URI in G .
- 4 $\langle s, p, o \rangle$ in G iff the triple $\langle M(s), p, M(o) \rangle$ in G' .

- I.e. a sort of graph isomorphism

Syntax vs. Semantics

- 'still talking about (abstract) syntax
- nothing has been said on the actual semantics of RDF
 - introduced with an RDF *vocabulary*
 - given by means of a *Model Theory*
- how do you write/exchange RDF?
- in particular bnodes and literals
- several possibilities
 - N-Triples
 - RDF/XML (normative)
 - Turtle notation (subset of N3)

N-Triples

- documents contain a set of assertions
subject predicate object .
- URI references written out completely:
<http://example.org/resource30>

- Literals as strings:

plain "chat"@fr

typed "<a>"^^<http://www.w3.org/2000/01/
rdf-schema#XMLLiteral>

- Bnodes:
_:anon

N-Triples Example

```
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
  <http://xmlns.com/foaf/0.1/title> "Dr" .
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Person> .
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
  <http://xmlns.com/foaf/0.1/workplaceHomepage> <http://www.unibz.it/inf> .
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
  <http://xmlns.com/foaf/0.1/homepage> <http://www.inf.unibz.it/~tessaris> .
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
  <http://xmlns.com/foaf/0.1/mbox_sha1sum>
    "758128cdae69fd0fd9e880921d9f4b25259edbf5" .
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
  <http://xmlns.com/foaf/0.1/name> "Sergio Tessaris" .
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
  <http://xmlns.com/foaf/0.1/family_name> "Tessaris" .
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
  <http://xmlns.com/foaf/0.1/givenname> "Sergio" .
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
  <http://xmlns.com/foaf/0.1/phone> <tel:+39-0471-016-125> .
```

RDF/XML

- Normative XML serialisation for RDF documents
- Use namespace abbreviations: e.g.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:myfoaf="http://www.inf.unibz.it/~tessaris/myfoaf.xml">
```

- Encode paths of the RDF graph
- There are several ways of encoding the same graph!

RDF/XML

- document root is `rdf:RDF`
- `rdf:Description` to represent nodes
- predicates use the corresponding URI

```
<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">  
  <dc:title>RDF/XML Syntax Specification (Revised)</dc:title>  
</rdf:Description>
```

- bnodes can be omitted

RDF/XML Example

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <foaf:Person rdf:ID="me">
    <foaf:name>Sergio Tessaris</foaf:name>
    <foaf:title>Dr</foaf:title>
    <foaf:givenname>Sergio</foaf:givenname>
    <foaf:family_name>Tessaris</foaf:family_name>
    <foaf:mbox_sha1sum>
      758128cdae69fd0fd9e880921d9f4b25259edbf5</foaf:mbox_sha1sum>
    <foaf:homepage
      rdf:resource="http://www.inf.unibz.it/~tessaris"/>
    <foaf:phone rdf:resource="tel:+39-0471-016-125"/>
    <foaf:workplaceHomepage
      rdf:resource="http://www.unibz.it/inf"/>
  </foaf:Person>
</rdf:RDF>
```

Turtle Notation

- Extension of N-Triples
- Compact representation of graphs

Turtle Notation

- Extension of N-Triples
- Compact representation of graphs
- I will use this notation, explaining it on the way

Turtle Example

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix : <http://www.inf.unibz.it/~tessaris/myfoaf.xml#> .

:me rdf:type foaf:Person ;
    foaf:family_name "Tessaris" ;
    foaf:givenname "Sergio" ;
    foaf:homepage
        <http://www.inf.unibz.it/~tessaris> ;
    foaf:mbox_sha1sum
        "758128cdae69fd0fd9e880921d9f4b25259edbf5" ;
    foaf:name "Sergio Tessaris" ;
    foaf:phone
        <tel:+39-0471-016-125> ;
    foaf:title "Dr" ;
    foaf:workplaceHomepage
        <http://www.unibz.it/inf> .
```

RDF Data Model

- up till now we have
 - abstract syntax for graphs
 - a way to exchange these graphs
- where's the semantics?
- how to express “rich” semantic constructs?

RDF Data Model

- up till now we have
 - abstract syntax for graphs
 - a way to exchange these graphs
- where's the semantics?
- how to express “rich” semantic constructs?
- everything is going to be in RDF itself

RDF Data Model

- up till now we have
 - abstract syntax for graphs
 - a way to exchange these graphs
- where's the semantics?
- how to express “rich” semantic constructs?
- everything is going to be in RDF itself
- URIs are *global*: RDF prescribes the **meaning** of same URIs

RDF, RDF Schema, and more

- RDF: basic vocabulary, e.g.
 - class membership
- RDFS: extends the basic vocabulary, e.g.
 - subclass relation
 - domain and range for predicates
- OWL family: “extends” to a full fledged ontology language

RDF, RDF Schema, and more

- RDF: basic vocabulary, e.g.
 - class membership
- RDFS: extends the basic vocabulary, e.g.
 - subclass relation
 - domain and range for predicates
- OWL family: “extends” to a full fledged ontology language
 - we wont discuss OWL

RDF Vocabulary

- Classes
 - `rdf:Property`
 - `rdf:XMLLiteral`
- Properties
 - `rdf:type`
- Reification
 - `rdf:Statement`
 - `rdf:subject`, `rdf:predicate`, `rdf:object`
- Collections and Containers
 - `rdf:Bag`, `rdf:Seq`, `rdf:Alt`
 - `rdf:List`, `rdf:first`, `rdf:rest`, `rdf:nil`
 - `rdf:_1`, `rdf:_2`, `rdf:_3`,... etc.

RDFS Vocabulary

- Classes
 - `rdfs:Resource`
 - `rdfs:Class`
 - `rdfs:Literal`
 - `rdfs:Datatype`
- Properties
 - `rdfs:range`, `rdfs:domain`
 - `rdfs:subClassOf`
 - `rdfs:subPropertyOf`
 - `rdfs:label`, `rdfs:comment`

Meaning and Semantics

- the *meaning* of an RDF document is context dependent
 - mostly not machine accessible
- we are interested in the **semantics** of RDF(S) as a *formal language*
 - implications of the asserted statements
 - **entailment** as the basic tool (query answering)
 - inference rules to decide entailment between two graphs
- semantics provided by means of a **model theory** (normative)
 - over the abstract syntax already described
- monotonic
 - no closed-world assumptions
 - no defaults

RDF Interpretations

Consider a set of terms \mathcal{T} (URIs and Literals in a graph)

Definition (Simple Interpretation)

a simple interpretation \mathcal{I} over \mathcal{T} is composed by

- non-empty set Δ of resources (domain of \mathcal{I})
- set \mathcal{P} (properties of \mathcal{I})
- mapping \mathcal{E} from \mathcal{P} into the powerset of $\Delta \times \Delta$
- mapping \mathcal{V} from URI references in \mathcal{T} into $\Delta \cup \mathcal{P}$
- mapping \mathcal{L} from typed literals in \mathcal{T} into Δ
- $\mathcal{PL} \subseteq \Delta$, which contains all the plain literals in \mathcal{T}

RDF Interpretations

- “double level” interpretation
- the key is in \mathcal{E}
- literals are in the domain
- no bnodes in \mathcal{T}

Triples Satisfiability

- interpretation $\mathcal{I}(t)$ of a term $t \in \mathcal{T}$
 - $\mathcal{I}(t) = t$ if t is a plain literal
 - $\mathcal{I}(t) = \mathcal{L}(t)$ if t is a typed literal
 - $\mathcal{I}(t) = \mathcal{V}(t)$ if t is a URI reference

Triples Satisfiability

- interpretation $\mathcal{I}(t)$ of a term $t \in \mathcal{T}$
 - $\mathcal{I}(t) = t$ if t is a plain literal
 - $\mathcal{I}(t) = \mathcal{L}(t)$ if t is a typed literal
 - $\mathcal{I}(t) = \mathcal{V}(t)$ if t is a URI reference
- a triple $s p o .$ is **satisfied** by \mathcal{I} iff
 - $\langle \mathcal{I}(s), \mathcal{I}(o) \rangle \in \mathcal{E}(\mathcal{I}(p))$

Triples Satisfiability

- interpretation $\mathcal{I}(t)$ of a term $t \in \mathcal{T}$
 - $\mathcal{I}(t) = t$ if t is a plain literal
 - $\mathcal{I}(t) = \mathcal{L}(t)$ if t is a typed literal
 - $\mathcal{I}(t) = \mathcal{V}(t)$ if t is a URI reference
- a triple $s p o$ is **satisfied** by \mathcal{I} iff
 - $\langle \mathcal{I}(s), \mathcal{I}(o) \rangle \in \mathcal{E}(\mathcal{I}(p))$
- **Bnodes**
 - $\mathcal{B}(G)$ is the set of bnodes in G
 - \mathcal{A} a mapping from $\mathcal{B}(G)$ to Δ
 - $\mathcal{I}_{\mathcal{A}}$ is the extension of \mathcal{I} with \mathcal{A} (i.e. $\mathcal{I}_{\mathcal{A}}(b) = \mathcal{A}(b)$)

Models of RDF Graphs

- Ground graphs (without bnodes)

Definition (Ground Satisfiability)

\mathcal{I} is a **model** of a ground RDF graph G iff
satisfies all the triples in G

Models of RDF Graphs

- Ground graphs (without bnodes)

Definition (Ground Satisfiability)

\mathcal{I} is a **model** of a ground RDF graph G iff
satisfies all the triples in G

- Graphs with bnodes

Definition (Satisfiability)

\mathcal{I} is a **model** of an RDF graph G iff
there is a mapping \mathcal{A} from $\mathcal{B}(G)$ to Δ s.t.
 $\mathcal{I}_{\mathcal{A}}$ satisfies all the triples in G

Simple Entailment

- Entailment is defined in terms of models
- G **entails** G' iff every model of G is a model of G'

Simple Entailment

- Entailment is defined in terms of models
- G **entails** G' iff every model of G is a model of G'
- bnodes are “existential variables”
 - \mathcal{I} model of G if exists \mathcal{A} s.t. $\mathcal{I}_{\mathcal{A}}$ satisfies all the triples in G
 - \mathcal{I} model of G' if exists \mathcal{A}' s.t. $\mathcal{I}_{\mathcal{A}'}$ satisfies all the triples in G'

Simple Entailment

- Entailment is defined in terms of models
- G **entails** G' iff every model of G is a model of G'
- bnodes are “existential variables”
 - \mathcal{I} model of G if exists \mathcal{A} s.t. $\mathcal{I}_{\mathcal{A}}$ satisfies all the triples in G
 - \mathcal{I} model of G' if exists \mathcal{A}' s.t. $\mathcal{I}_{\mathcal{A}'}$ satisfies all the triples in G'
- What about RDF and RDFS?

RDF(S) Vocabulary

- additional semantic conditions over the vocabulary
e.g. `rdf:type`
 - **restrict** the set of models
 - derived from the intended meaning (RDF data model)
e.g. `rdf:type` is a `rdf:Property`
- **Semantic conditions**
e.g. $x \in \mathcal{P}$ iff $\langle x, \mathcal{V}(\text{rdf:Property}) \rangle \in \mathcal{E}(\mathcal{V}(\text{rdf:type}))$
- **Axiomatic triples**
e.g. `rdf:type rdf:type rdf:Property .`
- RDF-MT doesn't cover reification, containers and collections

Entailment Rules

- a set of **inference rules** to capture RDF(S)-entailment
- rules *complete* an RDF graph
add a triple if there is some pattern
 - rules application **terminates**
 - in a **polynomial** number of steps
- set of rules for RDF and RDFS (informative)

Entailment Rules

Lemma (Entailment Lemma)

G rdf(s)-entails E iff exists G' derived from G with axiomatic triples using the entailment rules s.t. G' simply entails E .

- simple entailment is enough
- effective procedure (needs simple entailment)

Grounding RDF Graphs

Grounding a graph G

Def *completed*: added axiomatic triples and applied entailment rules

Def **Herbrand** model: each bnode replaced by an URIs or Literal

Def **Canonical** model (\widehat{G}): bnodes replaced with fresh URIs

Graph entailment

Theorem

RDF graphs entailment: G entails E iff some herbrand model of E is a subgraph of the canonical model of G

- Connects our definition to W3C normative semantics
- Complexity of entailment
 - 1 NP-complete in the size of the RDF graphs
 - 2 PTIME in the size of the entailing graph G
 - 3 PTIME if E is acyclic or ground

RDF and FOL

- RDF has an high order flavour
 - URIs can play different roles
 - $\langle \text{ex:o}, \text{rdf:type}, \text{ex:o} \rangle$
- We introduce a different model theoretic semantics for RDF
 - Compatible with FOL
 - Use standard technologies (e.g. databases or theorem provers)

FO Compatible Model Theory

- key idea: polymorphic interpretation of URIs (contextual PC)
 - abstract domain Δ
 - $u^{I_O} \in \Delta$: individual + function mapping valid literals to datavalues
 - $u^{I_C} \subseteq \Delta$: class
 - $u^{I_R} \subseteq \Delta \times \Delta$: binary relation
- E.g.

$\langle \text{ex:o}, \text{rdf:type}, \text{ex:o} \rangle$

leads to

$$\text{ex:o}^{I_O} \in \text{ex:o}^{I_C}$$

Classical Logic Interoperability

Def *non-high order graph*: no blank nodes as objects of `rdf:type`

Def The *classical logic translation* $FO(G)$ of a non-high order graph G

- URIs and literals are constants
- blank nodes are existentially quantified variables
- binary atomic formulas in correspondence with triples in G
- $\langle u_1, \text{rdf:type}, u_2 \rangle$ triples introduce $u_2(u_1)$ atomic formulae

Theorem

Given an RDF graph G and a non-high order graph E , G entails E iff $FO(\hat{G}) \models_c FO(E)$

Querying RDF documents

- How do we get informations out of an RDF document?
- RDF graphs can be stored in a variety of means
 - RDF/XML documents
 - databases
 - dedicated RDF service providers (e.g. RSS 1.0)
 - generated on the fly; e.g. from web pages (Gleaning Resource Descriptions from Dialects of Languages [GRDDL])
- Often retrieving the whole graph is not feasible/desirable
 - too big
 - lot of uninteresting parts
 - dynamic
- Solution: a (standard) protocol for querying RDF graphs

Entailment and Querying

- entailment and query answering are strictly related
- an answer to a query is a set of entailed facts
 - tuples representing variable bindings
 - complex structures like RDF graphs or XML documents
 - it depends on the query language
- RDF aims at big volumes of data
 - look out for efficiency (i.e. low data complexity)
- two main factors
 - representational language (RDF)
 - query language

Querying RDF

- RDF(S) is a *simple* language
 - entailment can be checked on a single canonical model
 - see entailment rules and lemma
- more freedom on the query language
 - truly graph based query language (e.g. a la XQuery)
 - simple language based on graph entailment

Querying RDF

- RDF(S) is a *simple* language
 - entailment can be checked on a single canonical model
 - see entailment rules and lemma
- more freedom on the query language
 - truly graph based query language (e.g. a la XQuery)
 - simple language based on graph entailment

Graph Patterns

- **Dataset**: graph to be queried
- define a *new* kind of graphs: **RDF graph patterns**
 - same as RDF graphs, but with an additional type of nodes: variables

```
?x foaf:nick "Alice" .
```

Problem

Find all the assignments for the variables that make the pattern a logical consequence of the Dataset

- nothing more than conjunctive queries using a single **ternary** predicate (e.g. *triple*(*x*, foaf:nick, "Alice"))

Subgraph Matching

- can we query in an efficient way?

Subgraph Matching

- can we query in an efficient way?
- Dataset can be stored in a relational database

Subgraph Matching

- can we query in an efficient way?
- Dataset can be stored in a relational database
- entailment lemma: simple entailment is subgraph matching

Subgraph Matching

- can we query in an efficient way?
- Dataset can be stored in a relational database
- entailment lemma: simple entailment is subgraph matching
- subgraph matching is conjunctive query answering

Subgraph Matching

- can we query in an efficient way?
- Dataset can be stored in a relational database
- entailment lemma: simple entailment is subgraph matching
- subgraph matching is conjunctive query answering
- the answer is yes!
 - e.g. Oracle supports RDF and an extension to SQL
 - most SPARQL implementations rely on a database back-end

ORACLE SQL Extension

```
SELECT t.r reviewer, e.emailid emailid
FROM TABLE(RDF_MATCH(
    '(?r ReviewerOf ?c)
    (?r rdf:type Faculty)',
    RDFModels('reviewers'),
    NULL, NULL)) t, employees e
WHERE t.r = e.name;
```

SPARQL

- query language for RDF becoming a W3C recommendation
- based on Graph Patterns
 - patterns “extract” a set of variable bindings (tuples)
- results from different graph patterns can be combined using an algebra
 - union of answersets
 - left outer join
 - filtering based on XQuery operators
 - optionally, an RDF graph can be returned (using a template)

SPARQL Example

```
BASE <http://example.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX ex: <properties/1.0#>
SELECT DISTINCT ?person ?name ?age
FROM <http://rdf.example.org/people.rdf>
WHERE {
  ?person a foaf:Person ;
          foaf:name ?name.
  OPTIONAL { ?person ex:age ?age } .
  FILTER ! REGEX(?name, "Bob")
}
LIMIT 3 ORDER BY ASC[?name]
```