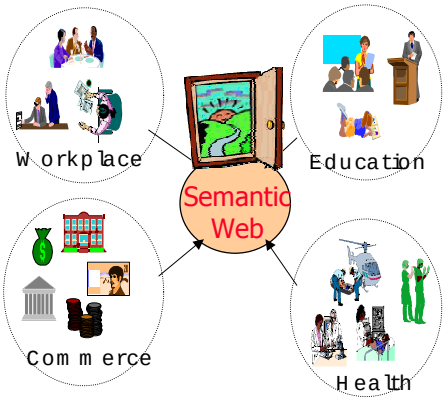
Database Technology for the Semantic Web

Vassilis Christophides
Dimitris Plexousakis

Computer Science Department, University of Crete
Institute for Computer Science - FORTH
Heraklion, Crete

On the Semantic Web

- Main infrastructure for supporting Community Webs
  - groups of people sharing a domain of discourse and a set of information resources (e.g., data, documents, services) and having some common interests/objectives
- Higher Quality Web Information Services
  - having data and programs described in a way that facilitates their reuse and integration by machines across applications

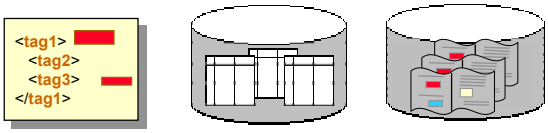


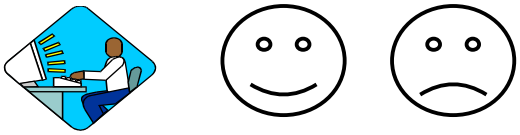
4 + 1 Webs?

1. Computers
   - XHTML
2. Voice
   - Voice XML
3. Wireless
   - WAP/WML
4. Television
   - bHTML
5. Semantic
   - RDF

Metadata exists for Almost Anything/Everywhere

- Physical Objects, Places, People,
- Devices, Networks, Infrastructure,
- Digital Documents, Data, Programs
- User Profiles, Preferences,



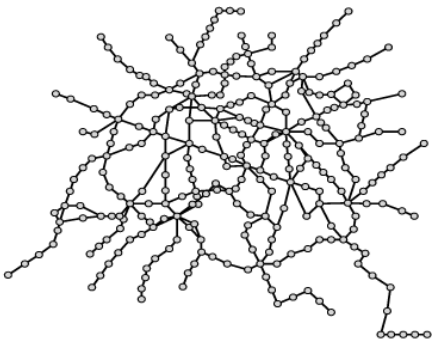


RDF Objectives

- Enables communities to define their own semantics of resource descriptions
  - we can disagree about semantics, but share the same infrastructure (syntax, editors, query languages, databases, etc.)
- Imposes structural constraints on the expression of metadata in various application contexts
  - for consistent encoding, exchange and processing of metadata on the Web
- Facilitates development of metadata vocabularies without central coordination
  - mechanisms for reusing descriptions of resources, concepts, etc.
- Focus on DBMS technology for RDF metadata
  - Related W3C efforts on XML data management

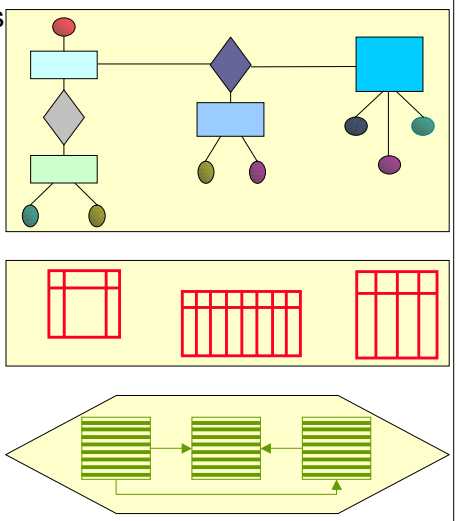
Outline

- Database issues for RDF metadata management
  - The Data Independence Issue
  - The Query Language Issue
  - The Model Issue
- RDF Query Language: RQL
  - Querying Large RDF Schemas
  - Filtering/Navigating Complex RDF descriptions
- Storing Voluminous RDF descriptions
  - Alternative DB representations
  - Performance Figures
- The ICS-FORTH RDFSuite
- Conclusions and remaining issues

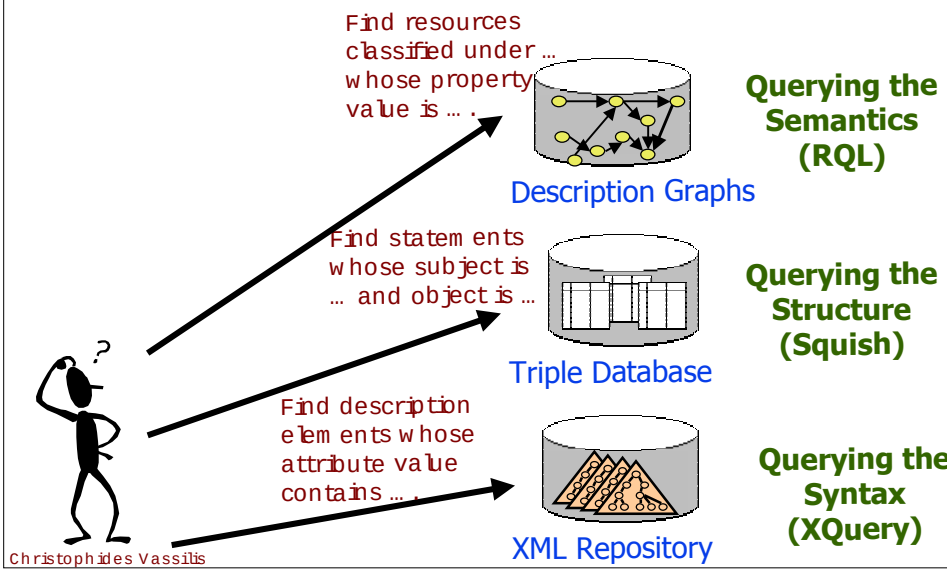


The Data Independence Issue

- Conceptual Level: Describing resources using one or several RDF schemas
- Logical Level: How RDF descriptions and schemas are physically stored
  - Logical-schema: Data organization using tables, objects, etc.
  - Physical-schema: Data organization using files, records, indices, etc.
- RDF data independence is crucial for ensuring scalability of real-scale Semantic Web applications



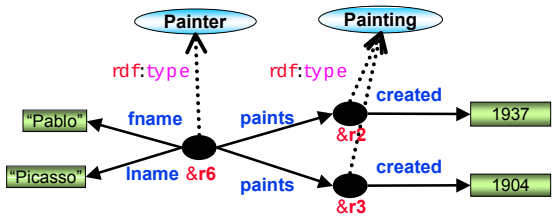
The Query Language Issue



Why a Data Model for RDF ?

- As support for physical/logical independence
  - RDF can be stored in files, a native repository, a relational database
  - RDF can be virtual, as a view of a repository, integrated sources
  - RDF can be in memory, using data structures in C, C++, Java, etc
  - RDF can be streamed between processes
- To describe information content of RDF Statements
  - to agree and reason about information content, preservation
- To define semantics of a data manipulation language:
  - A query language describes in a declarative fashion, the mapping between an input instance of the data model to an output instance of the data model

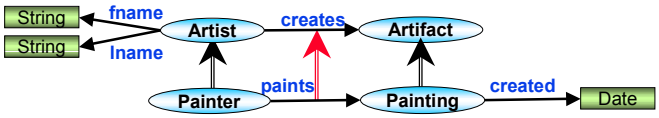
But RDF has specifics: Serialization syntax



r2: museoreinasofia.mcu.es/guernica.jpg
r3:www.artchive.com/woman.jpg
r6: picasso132

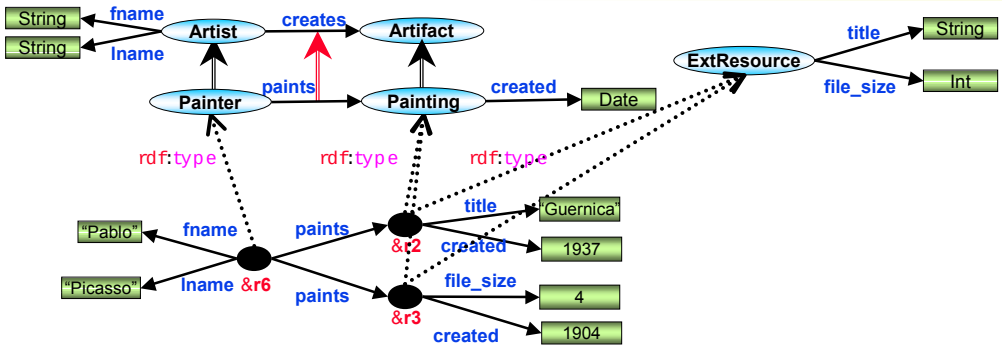
- XML attributes vs elements for RDF properties
  - fname, lname
- XML flat vs nested structures of RDF statements
  - Description vs. Painter elements
- RDF properties are unordered, optional, and multivalued
  - 2 paints and 0 creates
- One more motivation for a data model :
  - isolate the user from syntactic aspects of RDF/XML

But RDF has specifics: Schema Semantics



- Distinguish between labels of nodes and edges
  - Painter vs. paints
- Class and properties are organized in subsumption hierarchies
  - Painter <= Artist
- Properties are inherited
  - **&r6** may also have a creates property
- References are typed
  - **&r2** should be of class <= Painting
- Literal values are typed
  - 1937 is not a string but a date value !

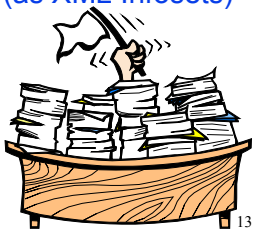
But RDF has specifics: Superimposed Descriptions



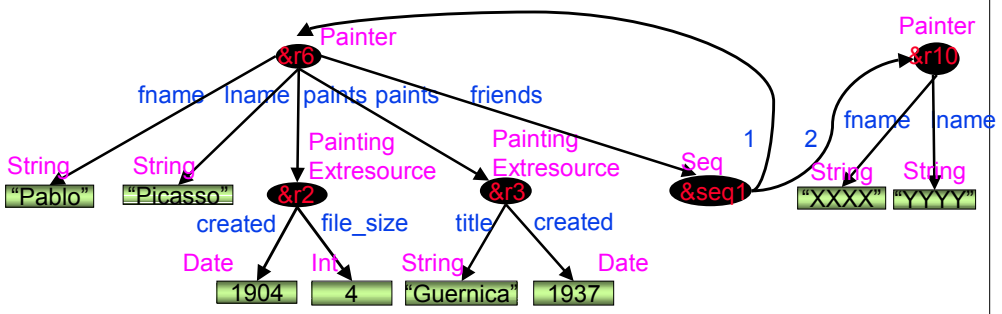
- Resources may belong to multiple (unrelated though isa) classes
  - **&r2** is both a Painting and an ExtResource
- Heterogeneous descriptions reminiscent of SGML exceptions
  - What is the structure of Painting resources?

Existing Data Models

- Graph and tree models used in research (OEM, UnQL, YAT, etc.)
- Document Object Model (DOM)
  - status: recommendation
  - programmatic interface for XML (with an object-oriented flavor)
- RDF Triple-based Model
  - describes the statements exported by RDF processors
  - can be generated after parsing or after validation (as XML Infosets)
- XML languages' Data Models:
  - Xpath: recommendation has it's own Data Model
  - XML Query Data Model: working draft



A Semistructured Data Model for RDF



- Graph based, unordered, edge/node-labeled (in the style of OEM)
  - But what about sequences (ordered)?

Towards a Formal Data Model for RDF

- An RDF schema is a 5-tuple: RS = (V_S, E_S, H, ψ, λ)
  - V_S a set of nodes
  - E_S a set of edges
  - H = (N,<) a well-formed hierarchy of names
  - ψ an incidence function: $E_s \rightarrow V_s \times V_s$
  - λ a labeling function: $V_S \cup E_S \rightarrow N \cup T$
- An RDF description base, instance of a schema RS, is a 5-tuple: RD = (V_D, E_D, ψ, ν, λ)
  - V_D a set of nodes
  - E_D a set of edges
  - ψ an incidence function: $E_D \rightarrow V_D \times V_D$
  - ν a valuation function: $V_D \rightarrow$ **V**
  - λ a labeling function: $V_D \cup E_D \rightarrow 2^{N \cup T}$:
    - $\forall$ u $\in V_D$, $\lambda \rightarrow$ n $\in$ **C**$\cup$**T**: ν(u) $\in$[[n]]
    - $\forall$ e $\in E_D$ [u,u'], $\lambda \rightarrow$ p

Why a Type System for RDF ?

- For error detection & safety:
  - to verify that statements comply to what the application expects
  - to make sure that the application accesses valid statements
  - to enforce safe operations (e.g., don't do float arithmetic on classes!)
  - to check that compositions of operations make sense
- For performance:
  - to design storage (saving space, improving clustering, etc.)
  - to process queries (algebraic laws, rewriting path expressions, etc.)
- We need a full-fledged Data Definition Language for RDF !
  - RDF Schema is viewed more as an ontology & modeling tool

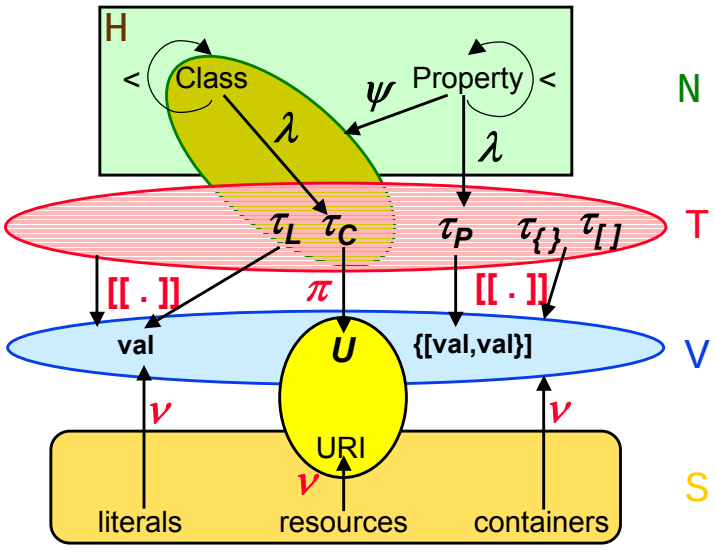
Towards a Type System for RDF

- Type System:

  $\tau = \tau_L \mid \tau_U \mid \{\tau\} \mid [\tau] \mid (1{:}\tau + 2{:}\tau + \ldots + n{:}\tau)$

- Interpretation Function:
  - Literal types, $[[\ \tau_L\ ]] = dom(\tau_L)$
  - Bag types, $[[\ \{\tau\}\ ]] = \{\nu_1, \nu_2, \ldots, \nu_n\}$, $\nu_1, \nu_2, \ldots, \nu_n \in \mathbf{V}$ are values of type τ
  - Seq types, $[[\ [\tau]\ ]] = \{\nu_1, \nu_2, \ldots, \nu_n\}$, $\nu_1, \nu_2, \ldots, \nu_n \in \mathbf{V}$ are values of type τ
  - Alt types, $[[\ (1{:}\tau_1 + 2{:}\tau_2 + \ldots + n{:}\tau_n)\ ]] = \nu_I$, $\nu_i \in \mathbf{V}$, $1<i<n$ is a value of type τ_i
  - $c \in \mathbf{C}$, $[[c]] = \{\nu \mid \nu \in \pi(c)\} \cup \{\pi(c') \mid c' < c\}$
  - $p \in \mathbf{P}$, $[[p]] = \{[\nu_1, \nu_2] \mid \nu_1 \in [[domain(p)]], \nu_2 \in [[range(p)]]\} \cup \{\pi(p') \mid p' < p\}$

A Formal Data Model for RDF/S



Schema Constraints

- Class, Property and Type names are mutually exlusive
  $C \cap P \cap T = \varnothing$
- Literal, Resources and Container values are mutually exclusive
  $L \cap U \cap V/\{U, L\} = \varnothing$
- $\forall$ c, c´ $\in$ C
  - Class is the root of class hierarchy
    $c < Class$
  - subClassOf relation is transitive
    $c < c', c' < c'' \Rightarrow c < c''$
  - subClassOf relation is antisymmetric
    $c < c' \Rightarrow c \neq c'$
- Domain and range of properties should be defined and they should be unique
  $\forall p \in P, \exists! c_1 \in C \ (c_1 = domain(p))$
  $\wedge \ \exists! c_2 \in C \cup T_L \ (c_2 = range(p))$
- $\forall$ p, p´, p´´ $\in$ P
  - Property is the root of property hierarchy
    $p < Property$
  - subPropertyOf relation is transitive
    $p < p', p' < p'' \Rightarrow p < p''$
  - subPropertyOf relation is antisymmetric
    $p < p' \Rightarrow p \neq p'$
  - If p is subPropertyOf of p´ then domain of p is subset of domain of p´ and range of p is subset of range of p´
    $p < p' \Rightarrow domain(p) \leq domain(p')$
    $\wedge \ range(p) \leq range(p')$
- A reified statement should have exactly one rdf:predicate, rdf:subject and rdf:object property

Data Constraints

- For all values: $\forall u \in V$
  - If u is a URI then it is an instance of one or more Classes
    $u \in U \Rightarrow \lambda(u) \subseteq C$
  - If u is a literal then it an instance of one and only one Literal type
    $u \in L \Rightarrow \lambda(u) \in T_L$
  - If u is a container then it an instance of one and only one Container type
    $u \in V/\{U, L\} \Rightarrow \lambda(u) \in T_{B|S|A}$
- For all properties: $\forall p \in P, \ [u_1, u_2] \in [[p]]$
  - if p belongs to the set {1, 2, 3…} then u_1 is an instance of either rdf:Bag or rdf:Seq or rdf:Alt
    if $p \in \{1, 2, 3\ldots\} \Rightarrow \lambda(u_1) \in T_{B|S|A}$
  - if p doesn't belong to {1, 2, 3, …} then u_1 belongs to the domain of p and u_2 belongs to the range of p
    if $p \in P/\{1, 2, 3\ldots\} \Rightarrow \lambda(u_1) \leq domain(p) \wedge \lambda(u_2) \leq range(p)$

Comparing RDF/S to XML DTD/Schemas

- Focus on edge-labeled, unordered graphs
  - With the exception of sequences
- Relies on global names and ids
  - With the exception of unnamed resources
- Supports a limited form of typing
  - Heterogeneous containers
- Provides subsumption relationships for classes and properties
  - With the exception of containers
- No integrity constraints
  - Skolem functions for unnamed resources

- Focus on node-labeled, ordered trees
  - With the exception of attributes
- Relies on global (elements) and local names (attributes)
  - XML Schema local elements
- Supports stronger forms of typing
  - With the exception of references
- Provides limited mechanisms for subtyping
  - Notions of extension&restriction
- Defines integrity constraints
  - Keys and foreign keys using XPath expressions

Looking at existing RDF Applications

- Publishing:
  - Biblink
  - Scholarly Link Specification (Slinks)
- Education/ Academic:
  - Common European Research Information Format (CERIF)
  - Mathematics International
  - Universal
  - IMS Global Learning Consortium
- Cultural Heritage/ Archives/ Libraries:
  - Inter. Committee for Documentation Reference Schema (CIDOC)
  - Research Support Libraries – Collection Level Description (RSLP-CLD)
  - EUropean Libraries & Electronic Resources in Mathematical Sciences (Euler)
- Audio-visual:
  - Internet Movie DataBase
  - MusicBrainz
- Mobile devices
  - Composite Capability/ Preference Profile (CC/PP)
- E-commerce
  - Basic Semantic Registry (BSR)
  - Real Estate Data Consortium
- Cross-domain:
  - MetaNet (Harmony)
  - Lexical WordNet
  - CERES/NBII Thesaurus
  - Top Level phOntology

Statistics of RDF Schemas

Schema	# Classes	# Properties	subClassOf		subPropertyOf	
			Max. Depth	Max. Breadth	Max. Depth	Max Breadth
BibLink	14	22	1	5	1	1
Slinks	20	56	2	2	1	1
CERIF	42	142	1	13	1	18
Mathematics International	211	0	11	43	-	-
Universal	5	13	-	-	-	-
IMS	17	8	2	5	1	2
CIDOC	63	85	6	9	1	4
RSLP-CLD	11	43	1	3	1	7
Euler	20	22	1	14	1	1

Statistics of RDF Schemas

Schema	# Classes	# Properties	subClassOf		subPropertyOf	
			Max. Depth	Max. Breadth	Max. Depth	Max Breadth
Internet Movie Database	65	182	2	37	-	-
CC/PP	18	3	2	4	1	1
BSR	2714	1754	4	62	-	-
Data Consortium	5073	285	5	763	3	233
MetaNet	66	11	2	17	1	2
Lexical WordNet	9	5	2	4	-	-
CERES/NBII	8	14	1	3	-	-
Top Level phOntology	189	141	11	11	6	18

Statistics of RDF Schemas

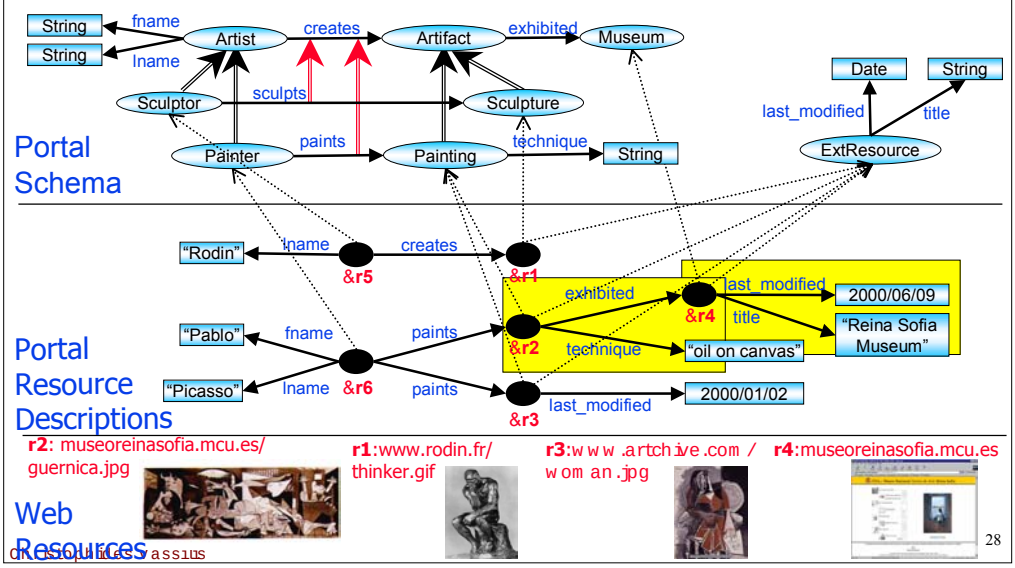
- Most of the ontologies were developed in breadth rather than in depth
  - when a small number of classes is defined, the number of properties is relatively big and vice versa
- The majority of ontologies do not use the subPropertyOf construct. In cases it is used:
  - is used mainly for relations (range classes) rather than attributes (range literals)
  - top-level properties are most of the times unconstrained (no domain/range restriction)
- Multiple inheritance for classes is far more widely used than multiple inheritance for properties
  - Multiple inheritance for properties appears only once in the set of the ontologies examined
- Multiple classification of resources was used only once in the instance files of the ontologies examined
- The only actually reused RDF Schema is Dublin Core

Querying RDF Descriptions: An Introduction to RQL

The RDF Query Language (RQL)

- Declarative query language for RDF description bases
  - relies on a typed data model (literal & container types + union types)
  - follows a functional approach (basic queries and filters)
  - adapts the functionality of XML query languages to RDF, but also:
    - treats properties as self-existent individuals
    - exploits taxonomies of node and edge labels
    - allows querying of schemas as semistructured data
- Relational interpretation of schemas & resource descriptions
  - Classes (unary relations)
  - Properties (binary relations)
  - Containers (n-ary relations)

A Cultural Community Resource Description Example



Querying Large RDF Schemas with RQL

- Basic Class Queries
  - topclass
  - subclassof(Artist)
  - subclassof^ (Artist)
  - superclassof(Painter)
  - superclassof^ (Painter)
- Basic Property Queries
  - topproperty
  - subpropertyof(creates)
  - subpropertyof^ (creates)
  - superpropertyof(paints)
  - superpropertyof^ (paints)
- Querying the RDF/S meta-schema
  - Class
  - Property
  - Literal

Class & Property Querying

- Find the domain and range of the property creates

```
seq(domain(creates), range(creates))
```

  while thanks to functional composition we can express

```
subclassof(seq(domain(creates), range(creates))[0]) or
select X from
subclassof(seq(domain(creates), range(creates))[0]){X}
```

- Which classes can appear as domain and range of property creates

```
select $X, $Y from {:$X}creates{:$Y} or
select X, Y from Class{X}, Class{Y}, {:X}creates{:Y}
```

- Find all properties defined on class Painting and its superclasses

```
select @P, range(@P) from {:Painting}@P or
select P, range(P) from Property{P} where domain(P)>=Painting
```

Schema Navigation using RQL

- Iterate over the subclasses of class Artist

```
select $X from Artist{:$X} or
select X from subclassof(Artist){X}
```

- Find the ranges of the property exhibited which can be reached from a class in the range of property creates

```
select $Y, $Z from creates{:$Y}.exhibited{:$Z}
```

- Find the properties that can be reached from a range class of property creates, as well as, their respective ranges

```
select * from creates{:$Y}.@P{:$$Z} or
from Class{Y}, (Class union Literal){Z}, creates{:Y}.@P{:Z}
```


Exporting Schemas using RQL Queries

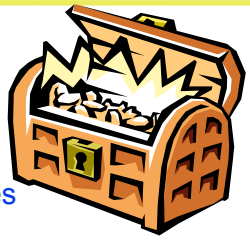
- Find Leaf Classes (i.e., classes without subclasses)

```
select C1
from Class{C1}
where not (C1 in (select C1
                  from Class{C2}
                  where C2 < C1))
```

- Find all schema information (i.e., group related superclasses and properties for each class)

```
select C, superclassof^(C), (select P, range(P)
                             from Property{P}
                             where domain(P) = C)
from Class{C}
```

Querying Complex RDF Descriptions with RQL



- Find all resources

```
Resource
```

- Find the resources in the extent of the property creates

```
creates    or
select * from {X}creates{Y}
```

- Find the resources of type ExtResource and Sculpture

```
ExtResource intersect Sculpture
ExtResource minus Sculpture
ExtResource union Sculpture
```


Navigating in Description Graphs using RQL

- Find the Museum resources that have been modified after year 2000 (i.e., data path with node and edge labels)

```
select X
from   Museum {X}.last_modified{Y}
where Y >= 2000-01-01T12:12:34+5
```

- Find the resources that have been created and their respective titles (i.e., data path using only edge labels)

```
select X,Z from creates{Y}.title{Z}
```

- Find the titles of exhibited resources that have been created by a Sculptor (i.e., multiple data paths)

```
select Z,W
from Sculptor.creates{Y}.exhibited{Z},{Z}title{W }
```

Using Schema to Filter Resource Descriptions

- Find the Painting resources that have been exhibited as well as the related target resources of type ExtResource (i.e., restrict multiply classified property target values using node labels)

```
select X, Y from {X:Painting}exhibited{Y}.ExtResource
```

Note the difference with the following path exression

```
select X, Y from {X:Painting}exhibited{Y:ExtResource}
```

- Find modified resources which can be reached by a property applied to the class Painting and its subclasses (i.e., restrict property source values using edge labels)

```
select @P, Y, Z
from {:$X}@P.{Y}last_modified{Z}
where $X <= Painting
```

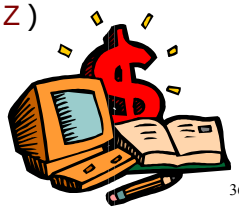
Discover the Schema of RDF Descriptions

- Find the description of a resource with URI "http://www.museum.es"

```
select $X, (select @P, Y
            from {Z : $Z} @P {Y}
            where X = Z and $X = $Z)
from $X {X}
where X = &http://www.museum.es
```

- Find the descriptions of resources whose URI match "www.museum.es"

```
select X, (select $W , (select @P, Y
                        from {Z : $Z} @P {Y}
                        where W = Z and $W = $Z)
           from $W {W}
           where W = X)
from Resource {X}
where X like "*www.museum.es*"
```



And if you still like triples …

- Find the description of resources which are not of type ExtResource

```
(
 (select X,@P,Y from {X}@P{Y})
  union
 (select X,type,$X from $X{X})
)
minus
(
 (select X,@P,Y from {X:ExtResource}@P{Y})
  union
 (select X,type,ExtResource from ExtResource{X})
)
```

Comparing RQL to W3C XQuery

- Find the names of those who have created artifacts which are exhibited in Museums, along with the Museum titles
  - RQL

```
select Y,Z,V,R
from {X}creates.exhibited{Y}.title{Z},
     {X}first_name{V},{X}last_name{R}
```

Comparing RQL to W3C XQuery

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  - <rdf:Bag ID="bag2287967">
    - <rdf:li>
      - <rdf:Seq>
          <rdf:li rdf:type="resource" rdf:resource="http://www.rodin.fr" />
          <rdf:li rdf:type="string">Rodin Museum</rdf:li>
          <rdf:li rdf:type="string">August</rdf:li>
          <rdf:li rdf:type="string">Rodin</rdf:li>
        </rdf:Seq>
      </rdf:li>
    - <rdf:li>
      - <rdf:Seq>
          <rdf:li rdf:type="resource" rdf:resource="http://www.museum.es" />
          <rdf:li rdf:type="string">Reina Sofia Museum</rdf:li>
          <rdf:li rdf:type="string">Pablo</rdf:li>
          <rdf:li rdf:type="string">Picasso</rdf:li>
        </rdf:Seq>
      </rdf:li>
    - <rdf:li>
      - <rdf:Seq>
          <rdf:li rdf:type="resource" rdf:resource="http://www.louvre.fr/" />
          <rdf:li rdf:type="string">Louvre Museum</rdf:li>
          <rdf:li rdf:type="string">Michelangelo</rdf:li>
          <rdf:li rdf:type="string">Buonarroti</rdf:li>
        </rdf:Seq>
      </rdf:li>
    </rdf:Bag>
</RDF>
```


Comparing RQL to W3C XQuery

- XQuery

```
LET $t := document("sirpac-culture-merged.rdf")//description
FOR $artist IN rdf:instance-of-class($t, rdf:predicate-domain($t,
  "creates"))
LET $artifact := rdf:join-on-property($t, $artist,"creates"),
    $museum := rdf:join-on-property($t, $artifact, "exhibited")
RETURN
   <result>
     {filter($artist | $artist/last_name | $artist/first_name),
      filter($museum | $museum/title)}
   </result>
```

Comparing RQL to W3C XQuery

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  - <result>
    - <description rdf:about="rodin424">
        <first_name>August</first_name>
        <last_name>Rodin</last_name>
      </description>
    - <description rdf:about="http://www.rodin.fr">
        <title>Rodin Museum</title>
      </description>
    </result>
  - <result>
    - <description rdf:about="michelangelo">
        <first_name>Michelangelo</first_name>
        <last_name>Buonarroti</last_name>
      </description>
    - <description rdf:about="http://www.louvre.fr/">
        <title>Louvre Museum</title>
      </description>
    </result>
  - <result>
    - <description rdf:about="picasso132">
        <first_name>Pablo</first_name>
        <last_name>Picasso</last_name>
      </description>
    - <description rdf:about="http://www.museum.es">
        <title>Reina Sofia Museum</title>
      </description>
    </result>
</RDF>
```

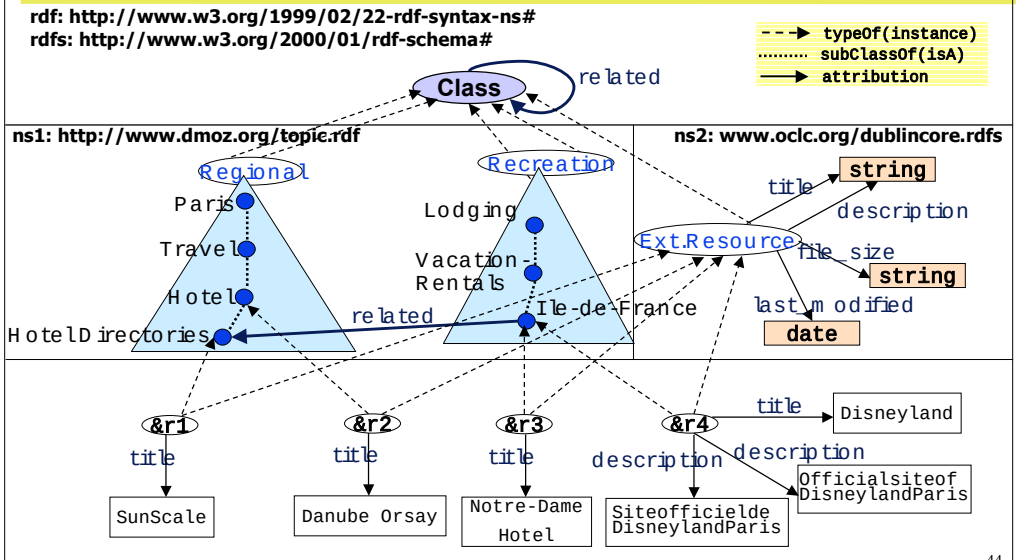

ICS-FORTH
EU-NSF Semantic Web Workshop 3-5 Oct

Comparing RQL to W3C XQuery

- XML syntactic and schematic discrepancies of semantically equivalent RDF statements
  - normalized representation under the form of merged descriptions
- XQuery has no built-in knowledge of the RDF schema information
  - function library that exploits the RDF schema if the assertions of the schema are also present in the normalized representation
- Data model mismatches between XML and RDF impact type safety of functions and queries
  - `bag( range(Artist) ) union subclassof(Artifact)`
    - In RQL Type Error
    - In XQuery All the subclasses of Artifact !

Storing RDF Descriptions: RSSDB Preliminary Performance Results

Modeling the ODP Catalog with RDF/S



ODP Statistics

- ODP Version: 16-01-2001
  - 170 Mbytes of class hierarchies
  - 700 Mbytes of resource descriptions
  - 337,085 topics
  - 16 hierarchies with
    - max depth: 13 (6.86 on average)
    - max # subclasses: 314 (4.02 on average)
  - 2,342,978 URIs

Generic Representation

Resources

id: int	uri: text
1	http://www.dmoz.org/topics.rdfs#Hotel
2	http://www.dmoz.org/topics.rdfs#Hotel Directories
3	http://www.oclc.org/dublincore.rdfs#title
4	http://www.dmoz.org/schema.rdf#Ext.Resource
5	http://www.w3.org/1999/02/22-rdf-syntax-ns#type
6	http://www.w3.org/2000/01/rdf-schema#subClassOf
7	http://www.w3.org/1999/02/22-rdf-syntax-ns#Property
8	http://www.w3.org/2000/01/rdf-schema#Class
9	r1

Triples

predid: int	subid: int	objid: int	objvalue: text
6	2	1	
5	3	7	
5	1	8	
5	9	2	
3	9		SunScale

Specific Representation

Namespace

id:int	uri: text
1	http://www.w3.org/2000/01/rdf-schema#
2	http://www.w3.org/1999/02/22-rdf-syntax-ns#
3	http://www.oclc.org/dublincore.rdfs#
4	http://www.dmoz.org/topics.rdfs#

Type

id: int	nsid: int	lpart: text
1	1	Resource
2	2	Bag
3	2	Seq
4		String

Class

id: int	nsid: int	lpart: text
11	5	Ext.Resource
12	4	Hotel
13	4	Hotel Directories

Property

id: int	nsid: int	lpart: text	domainid: int	rangeid: int
14	3	title	1	4
15	3	description	1	4
16	5	title	11	4

SubClass

subid: int	superid: int
11	1
12	1
13	12

SubProperty

subid: int	superid: int
16	14

Instances

uri: text	classid: int
r1	11
r2	11
r1	13
r2	12

t14

source: text	target: text
r1	SunScale
r2	Pulitzer Opera

t15

source: text	target: text

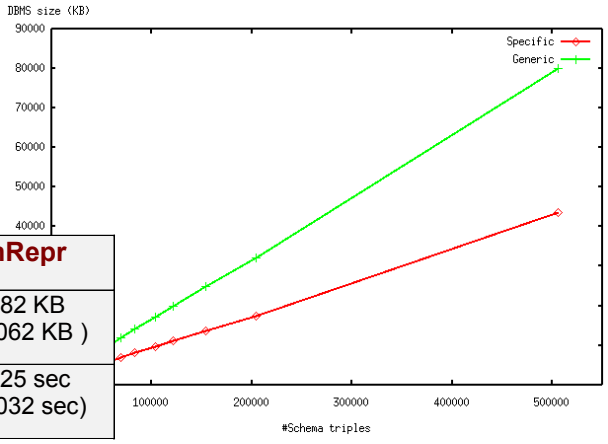
t16

source: text	target: text

—— subtable

DBMS Size vs. Schema Triples

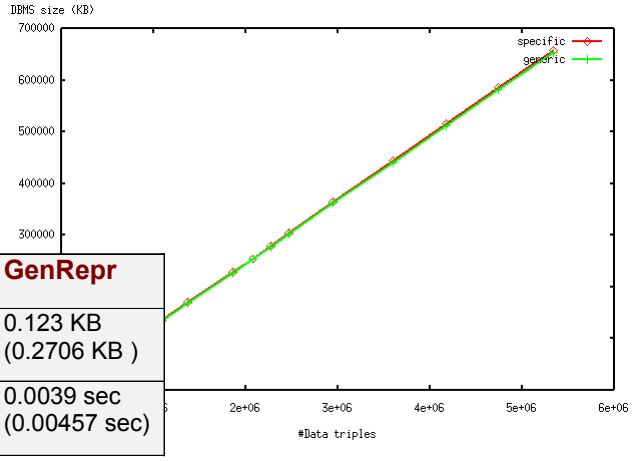
➔DBMS size scales linearly with the number of schema triples



	SpecRepr	GenRepr
Aver. triple size (with indexes)	0.086 KB (0.1734 KB)	0.1582 KB (0.3062 KB)
Aver. triple storage time (with indexes)	0.0021 sec (0.0025) sec	0.0025 sec (0.0032 sec)

DBMS Size vs. Data Triples

➔DBMS size scales linearly with the number of data triples



	SpecRepr	GenRepr
Aver. triple size (with indexes)	0.123 KB (0.2566 KB)	0.123 KB (0.2706 KB)
Aver. triple storage time (with indexes)	0.0033 sec (0.0043) sec	0.0039 sec (0.00457 sec)

Query Templates for RDF description bases

Pure schema queries	
Q1	*Find the range (or domain) of a property*
Q2	*Find the direct subclasses of a class*
Q3	*Find the transitive subclasses of a class*
Q4	*Check if a class is a subclass of another class*
Queries on resource descriptions using available schema knowledge	
Q5	*Find the direct extent of a class (or property)*
Q6	*Find the transitive extent of a class (or property)*
Q7	*Find if a resource is an instance of a class*
Q8	*Find the resources having a property with a specific (or range of) value(s)*
Q9	*Find the instances of a class having a given property*
Schema queries for specific resource descriptions	
Q10	*Find the properties of a resource and their values*
Q11	*Find the classes under which a resource is classified*

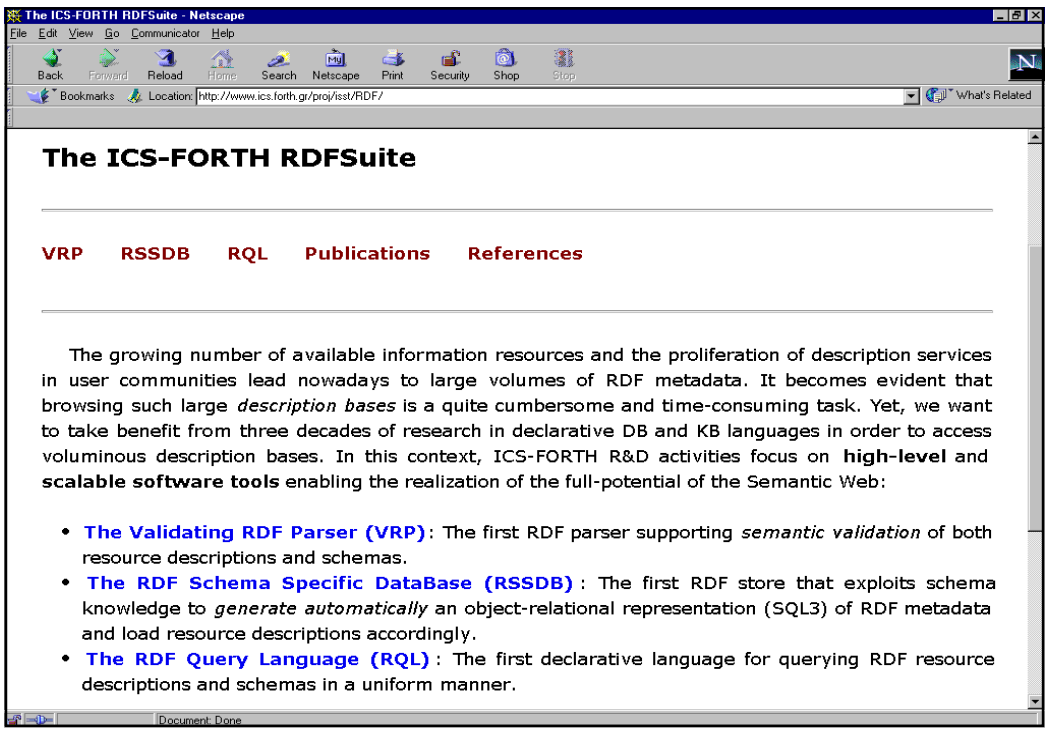
Execution Time of RDF Benchmark Queries

Query	Generic			Specific		
	Case 1	Case 2	Case 3	Case 1	Case 2	Case 3
Q1	0.0015			0.0012		
Q2	0.0017	0.0028	0.02	0.0012	0.0022	0.0124
Q3	0.0460	0.082	344.91	0.0463	0.0612	341.98
Q4	0.033	0.0415	0.0662	0.0333	0.0415	0.0662
Q5	0.0043	0.008	0.04	0.0015	0.0028	0.027
Q6	0.0573	0.315	627.43	0.0508	0.1118	482.45
Q7	0.0034	0.0034	0.0034	0.0016	0.0016	0.0017
Q8	**124.20**	**365.73**	**675.42**	**0.0013**	**0.0069**	**0.0466**
Q9	**110.58**	**117.68**	**185.7**	**0.031**	**0.0338**	**0.1059**
Q10	0.0072	0.0072	0.0072	0.0071	0.0071	0.0076
Q11	0.0035	0.0043	0.0056	0.0013	0.0015	0.0015

Comparison

- Specific Representation permits the customization of the database representation of RDF metadata
- Specific Representation outperforms the Generic Representation for all types of queries
  - Q1, Q2, Q5, Q7, Q10, Q11: by a factor up to 3.73
  - Q3, Q4, Q6: by a factor up to 2.8
  - Q8, Q9: by a factor up to **95,538**
- Generic representation pays severe penalty for maintaining large tables (**Triples**, **Resources**)
  - e.g., queries Q8, Q9 require (self-) joins of **Triples**, **Resources**

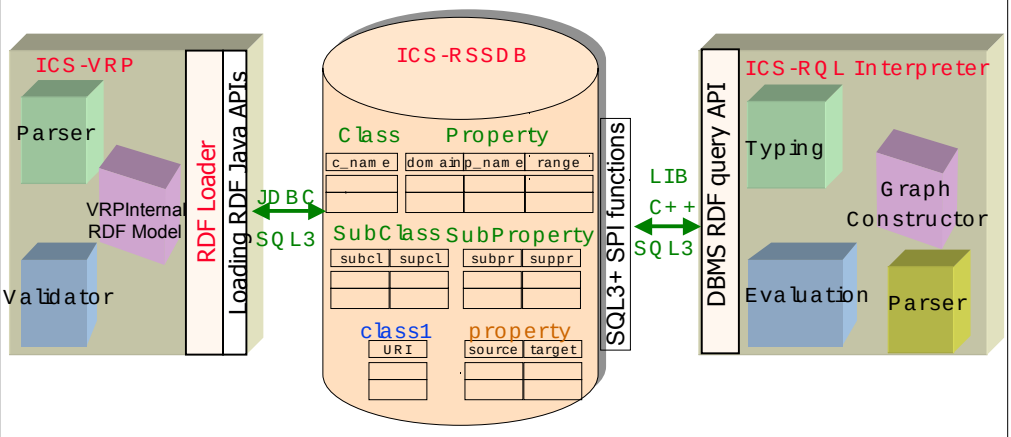
The ICS-FORTH RDFSuite: High-level and Scalable Tools for the Semantic Web
http://139.91.183.30:9090/RDF/



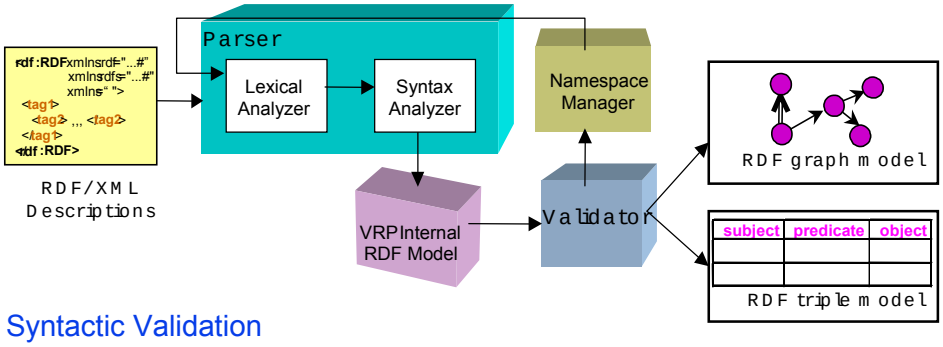
The RDFSuite Main Components

- The Validating RDF Parser (VRP): Karsten Tolle Diploma Thesis
  - The First RDF Parser supporting semantic validation of both resource descriptions and schemas
- The RDF Schema Specific DataBase (RSSDB): Sophia Alexaki MSc. Thesis
  - The First RDF Store using schema knowledge to automatically generate an Object-Relational (SQL3) representation of RDF metadata and load resource descriptions
- The RDF Query Language (RQL): Greg Karvournarakis MSc. Thesis
  - The First Declarative Language for uniformly querying RDF schemas and resource descriptions

The RDFSuite Architecture

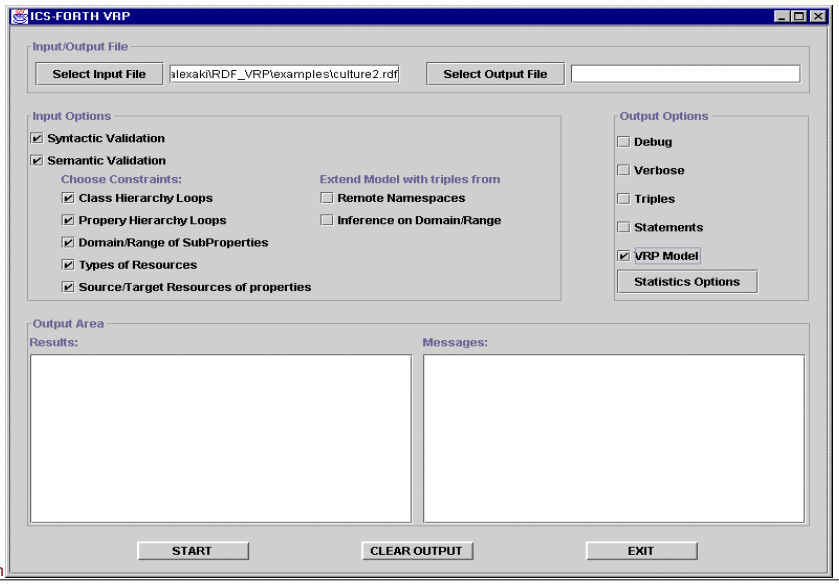


Validating RDF Parser (VRP)



- Syntactic Validation
  - RDF/XML syntax described in the RDF M&S Specification
- Semantic Validation
  - Semantic constraints derived from the RDF Schema Specification
- Implementation
  - Standard compiler generator tools for Java CUP (0.1) JFLEX (1.3.2)
  - 100% Java(TM) development (Java 1.2.2)

VRP Interface



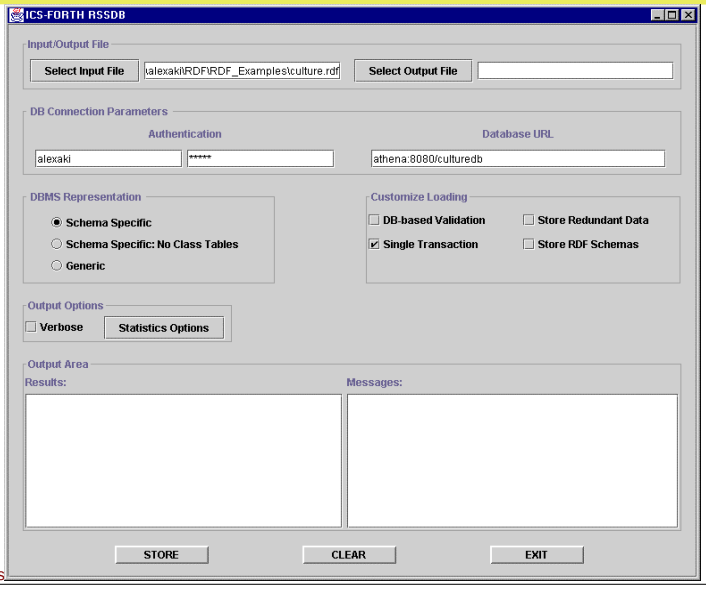
VRP Features

- Understands embedded RDF in HTML or XML
  - Full XML Schema Data Types support
  - Full Unicode support
- Statement validation across several RDF/XML namespaces
  - Persistent namespaces (for consistency, optimization)
- Various Output Options
  - Debugging
  - Serialization in files under the form of triples and graphs
  - Statistics for schema characteristics (class/property hierarchies) and resource distribution (class population)
- Easy to use as a standalone application
  - No other software needs to be installed (e.g., XML Parsers)
- Easy to integrate with other applications e.g., visualization tools
  - RDF Model Construction and Validation Java APIs

RDF Schema Specific DataBase (RSSDB)

- Persistent RDF Store using standard database technology
  - Separates schema form data information
  - Distinguishes between classes and properties
- Preserves the flexibility of RDF in
  - Refining schemas
  - Enriching descriptions
  - Using multiple schemas
- Implementation
  - On top of an object-relational DBMS (SQL3) like PostgreSql
  - Using JDBC Interface (2.0)

RSSDB Interface



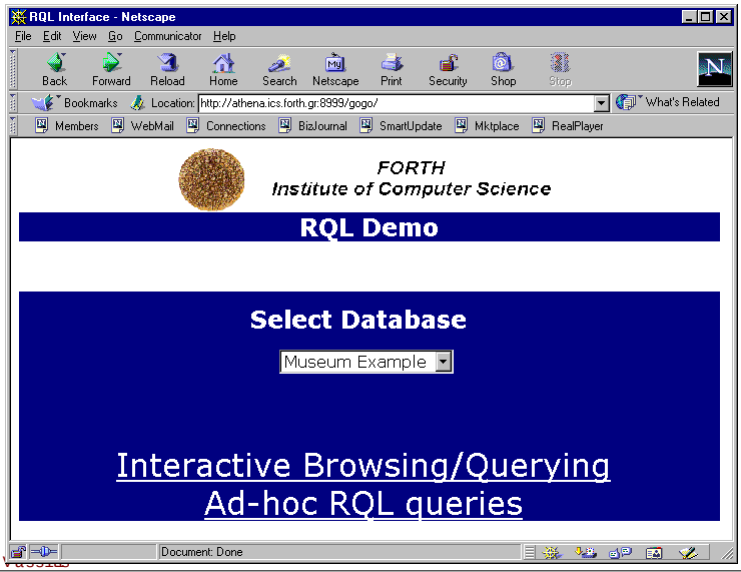
RSSDB Features

- Customization of the database representation according to
  - Employed meta-schemas (RDF/S, DAML-OIL)
  - RDF schemas and description bases peculiarities (number of classes vs. properties, resource distribution per classes)
  - Query functionality of applications
- Scalability
  - size of DBMS scales linearly with the number of loaded triples (tested with the Open Directory Portal comprising about 6 million triples)
  - incremental loading of voluminous description bases
- Easy to use as a standalone application
  - Requires only JDBC-compliant ORDBMS
- Easy to integrate with other applications e.g., metadata servers
  - RDF Model Loading & Update Java APIs

RDF Query Language (RQL)

- Declarative language (like ODMG OQL) for conceptual browsing & querying of voluminous RDF Description Bases
  - Easy navigation and resource discovery (using few query terms)
  - Task-specific personalization of RDF description bases (views)
  - Seamless querying of RDF schemas and resource descriptions
  - Flexible export facilities of RDF metadata (restructuring)
- RQL fully supports:
  - XML Schema data types (for filtering literal values)
  - grouping primitives (for constructing complex XML results)
  - aggregate functions (for extracting statistics)
  - recursive traversal of class and property hierarchies (for matchmaking)
- Implementation:
  - C++ development (GCC 2.95.1) on top of an ORDBMS (Unix, Linux)
  - Client/Server architecture (XDR-based)

RQL Web Interface



RQL Features

- Pushes as much as possible query evaluation to the underlying DBMS
  - Benefit from robust SQL3 query engines
  - Extensive use of DB indices
- Generic RDF/XML result form (Containers)
  - Standard XSL/XSL processing for customized rendering
- Easy to couple with commercial ORDBMSs (Oracle, DB2)
  - RDF querying APIs (SQL3/C++ functions)
- Easy to integrate with different Application Servers (Zope, JetSpeed)
  - C++ or Java drivers to RQL servers
- Easy to learn and use
  - One day training

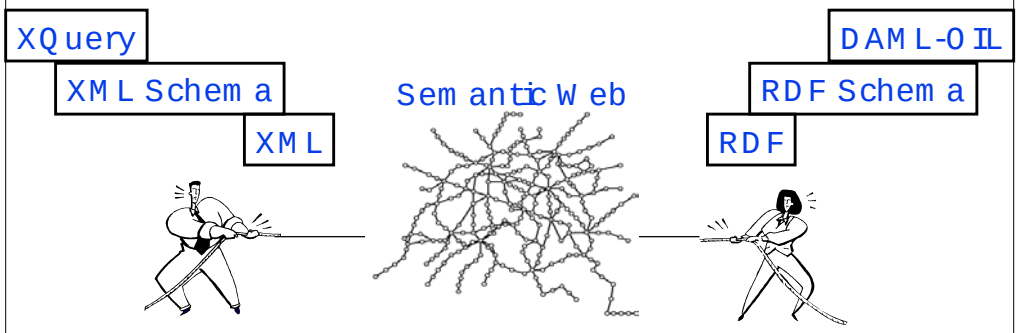
RDFSuite Summary

- RDFSuite addresses the needs of effective and efficient RDF metadata management by providing tools for validation, storage and querying
  - validation follows a formal data model and constraints enforcing consistency of RDF schemas
  - scalability
  - declarative query language for schema and data querying
- Ongoing efforts:
  - RQL query optimization
  - RQL update and transactional aspects

Other Issues

- RDF Metadata Generation from Legacy Repositories:
  - need to capture schemas from heterogeneous resources
- RDF Schema Evolution and Metadata Revision:
  - to support the dynamics of resource descriptions
- RDF Repositories Distribution:
  - for integration with WebDAV or LDAP-like architectures
- RDF Query Languages Optimization:
  - for real-scale Semantic Web applications

RDF and XML: Convergence or Divergence ?



- Will the SW be inside, above or beside the normal Web ?


Thank you
Hvala
Danke
Merci
Gracias
Grazie