The Semantic Web: Myths and Reality

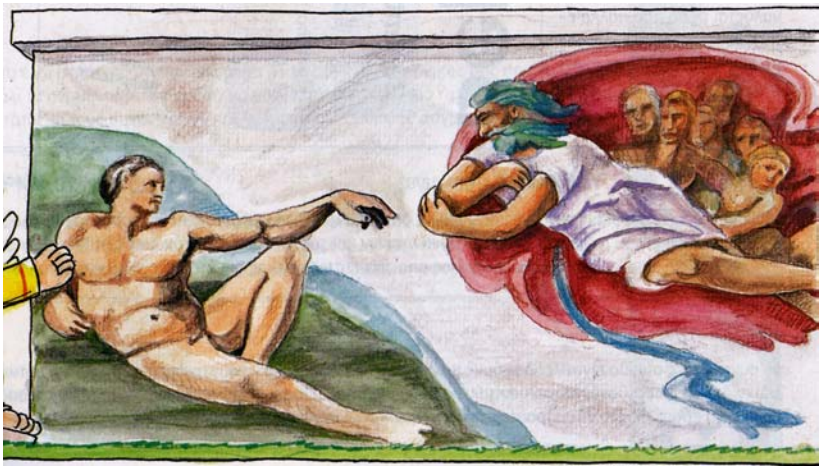
V. Christophides D. Plexousakis
Computer Science Department, University of Crete
and Institute for Computer Science - FORTH
Heraklion, Crete, Greece
in collaboration with

M. Scholl
CEDRIC/CNAM
292 Rue St Martin
75141 Paris, Cedex 03, France

V. Tannen
Computer and Information Science Department, UPenn
200 South 33rd Street,
Philadelphia, Pennsylvania, USA

This Talk is not about Myths !

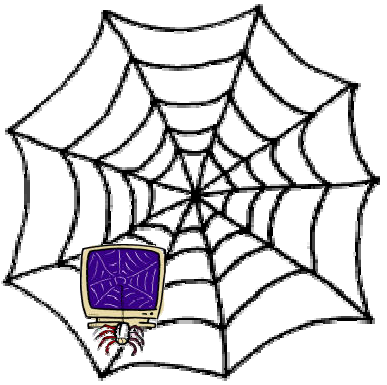
Is about Reality !!!



www.arkas.gr

How the Web is Today ?

- Information and its presentations are mixed up in the form of HTML documents
  - all intended for human consumption
  - many generated automatically by applications
- Easy to fetch any Web page, from any server, any platform
  - access through a uniform interface



The Secrets of HTML Success

- Everybody can write it:
  - HTML is simple
  - HTML is textual: it is human readable, you can use any editor, ...
- Everybody can read it:
  - HTML is portable on any platform
  - The browser is the universal application
- Everybody can search it:
  - Keyword-based Search Engines: high recall, low precision
- It connects pieces of information together
  - Through hypertext links

What's Wrong with HTML

- If written properly, normal HTML markup may reflect document presentation, but it cannot adequately represent the semantics & structure of data

Artist Name

Artifact Title

Date

Dimensions

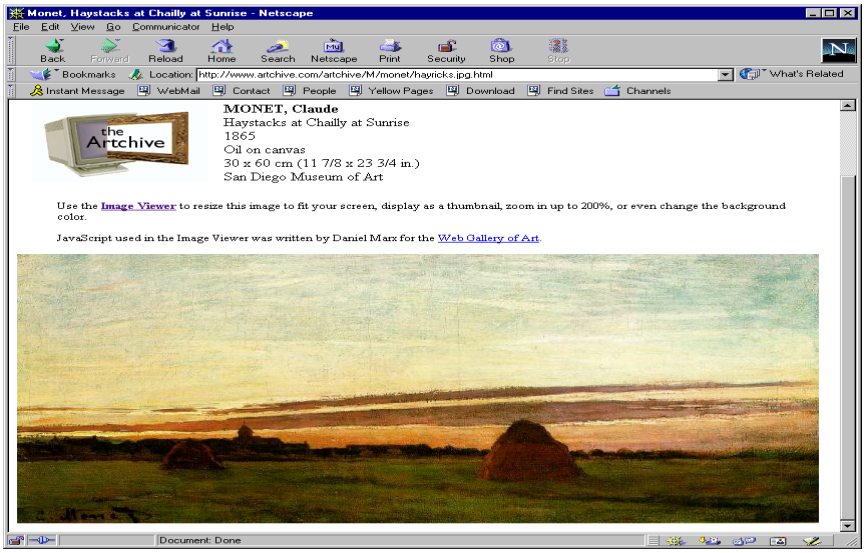
Material

Image Reference

Museum

```
<B>MONET, Claude<B><BR>
Haystacks at Chailly at Sunrise<BR>
1865<BR>
Oil on canvas<BR>
30 x 60 cm (11 7/8 x 23 3/4 in.)<BR>
San Diego Museum of Art <BR>
<P>
<IMG SRC="http://192.41.13.240/artchive/
m/monet/hayricks.jpg">
```

HTML Document Presentation



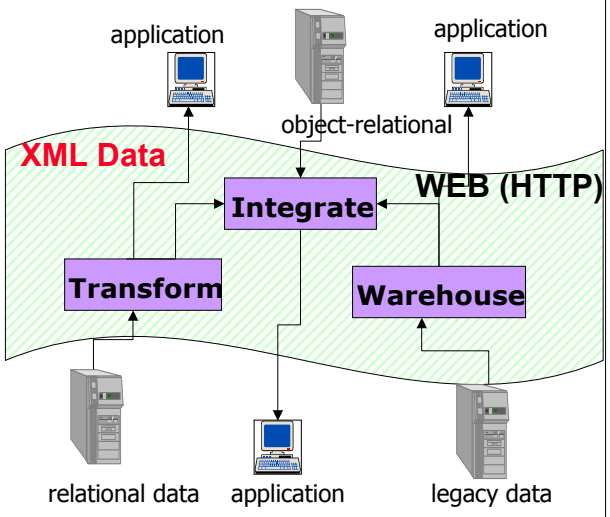
But Modern Web Applications Need More!

- Infomediaries:
  - Community Web Portals
  - Digital Museums & Libraries
- Electronic commerce:
  - On-line Catalogs & Procurement
  - Comparison Shoppers
  - Market Places
  - Virtual Enterprises
- Scientific applications:
  - E-learning
  - Data & Knowledge Grids
- Advanced Information Management
  - finding,
  - extracting,
  - representing,
  - interpreting,
  - maintaining
- Flexible, Quick Interoperation: the ability to uniformly share, interpret and manipulate heterogeneous information
  - applications cannot consume HTML

More than HTML documents: Data on the Web
More than Web browsers: Web-enabled Applications

Paradigm Shift on the Web

- New Web standard XML:
  - XML generated by applications
  - XML consumed by applications
- Data exchange:
  - across platforms
  - across organizations
- Web: from collection of documents to Web data published as documents

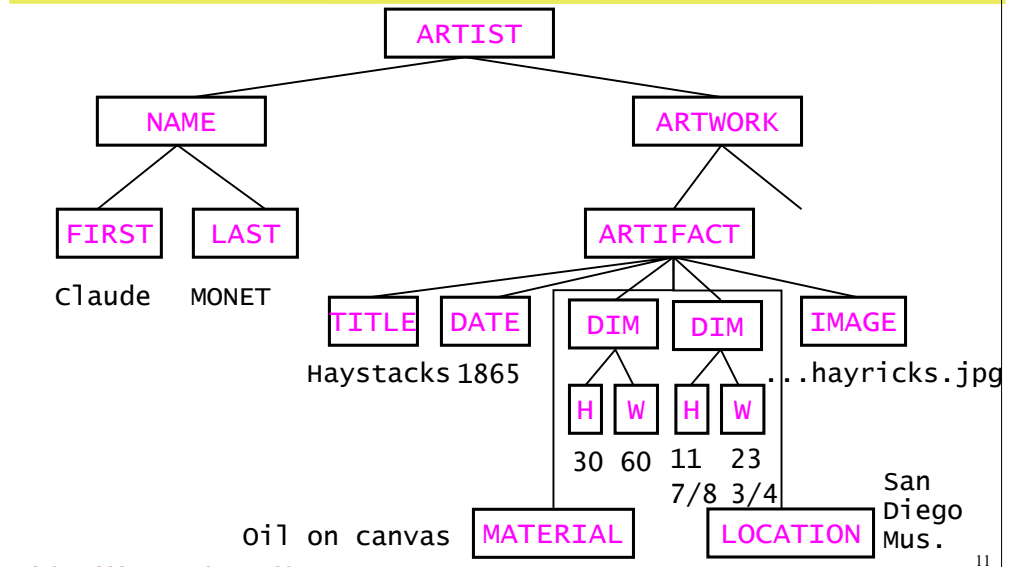


XML Data Representation: The Document View

```
<ARTIST>
  <NAME>
    <FIRST>Claude</FIRST> <LAST>Monet</LAST>
  </NAME>
  <ARTWORK>
    <ARTIFACT>
      <TITLE>Haystacks at Chailly at Sunrise</TITLE>
      <DATE>1865</DATE>
      <MATERIAL>Oil on canvas</MATERIAL>
      <DIM Metric='cm'>
       <HEIGHT>30</HEIGHT><WIDTH>60</WIDTH></DIM>
      <DIM Metric='in'>
       <HEIGHT>11 7/8</HEIGHT><WIDTH>23 3/4</WIDTH></DIM>
      <LOCATION>San Diego Museum of Art</LOCATION>
      <IMAGE File='http://192.41.13.240/artchive/m/monet/hayricks.jpg'/>
    </ARTIFACT>
  </ARTWORK>
</ARTIST>
```

Element Name
Element Content
Attribute Name
Attribute Value
Empty Element

XML Data Representation: The Database View



The Secrets of XML Popularity

- It looks like HTML...
  - Simple, familiar, easy to learn, human-readable
  - Universal and portable
  - Supported by the W3C: trusted and quickly adopted by the industry
- …but it's more than HTML!
  - flexible: you can represent any information
  - extensible: you can represent it the way you want!
- Increasing precision in XML specifications
  - Well-Formed: already better than plain text
  - Valid: Structure conforms to a DTD or an XML Schema

Well-Formed XML

- An object is said to be a well-formed XML document if it meets all the well-formedness constraints (WFCs) of the XML syntax:
  - tags (etc.) are syntactically correct
  - every tag has an end-tag
  - tags are properly nested
  - there exists a root
- By definition if a document is not well-formed, it is not XML
  - This means that there is no an XML document which is not well-formed, and XML processors are not required to do anything with such documents

Valid XML

- A well-formed document is valid only if it contains a proper DTD (or Schema) and if the document obeys the constraints of that DTD (or Schema) and therefore the XML Validity Constraints (VCs)
  - only declared tags (element or attribute names) are used
  - all tag occurrences conform to specified content models
- Examples:
  - The following XML Document is well-formed but not valid

```
<ARTIST>Claude Monet</ARTIST>
```

  - The following XML Document is not even well-formed

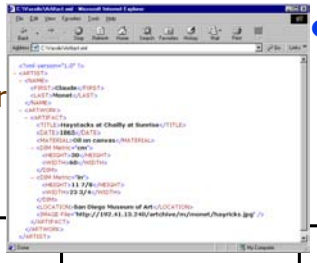
```
<FIRST>Claude</FIRST><LAST>Monet</LAST>
```

XML Document Type Definition (DTD)

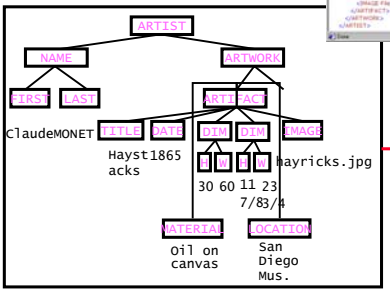
```
<!DOCTYPE artist [
<!ELEMENT artist (name, born, death, artwork, nationality?,
                  influences)>
<!ATTLIST artist oid ID #REQUIRED>
<!ELEMENT name      (first, last)>
<!ELEMENT first     (#PCDATA)>
<!ELEMENT last      (#PCDATA)> ...
<!ELEMENT artwork   (artifact+)>
<!ELEMENT artifact (title, date, material, dim*, location, image)>
<!ELEMENT title     (#PCDATA)> ...
<!ELEMENT dim       (height, width)>
<!ATTLIST dim metric (cm | in) 'cm'>
<!ELEMENT location (#PCDATA)>
<!ELEMENT image EMPTY>
<!ATTLIST image file ENTITY #REQUIRED>
<!ELEMENT influences (PCDATA | aref)*>
<!ELEMENT aref EMPTY>
<!ATTLIST aref oref IDREF #IMPLIED>
]>
```


Is XML the Solution to Interoperability?

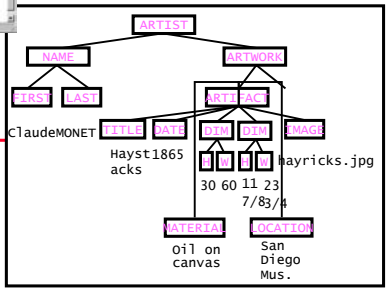
Document = medium for exchanging information



- Still need to agree on:
  - DTDs or Schemas
  - Meaning of tags
  - "Operations" on data
  - Meaning of operations



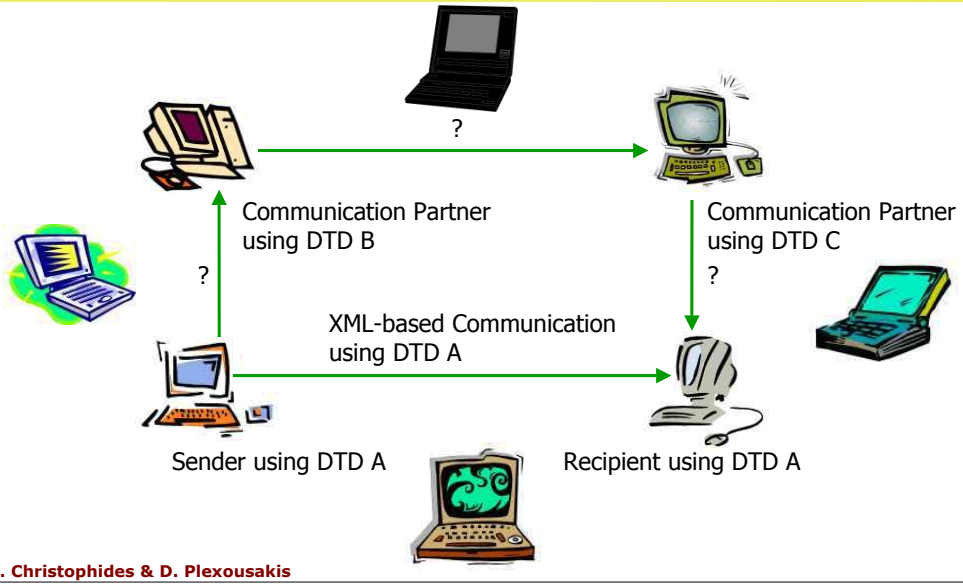
Communication



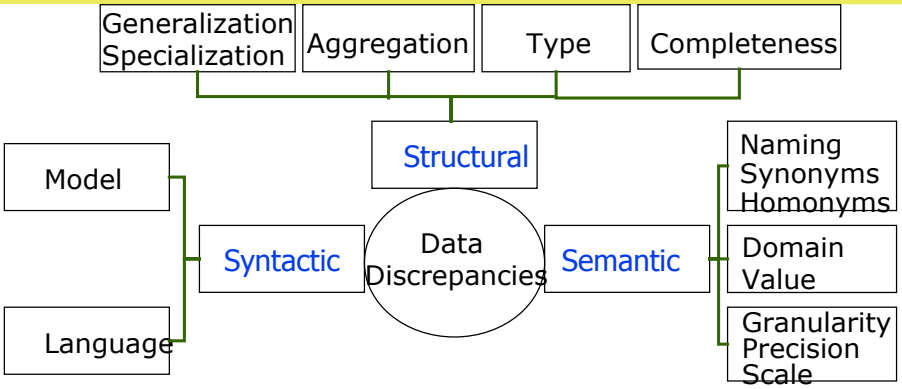
Application 1

Application 2

Large Scale Interoperation on the Web



Recall Data Heterogeneity



- XML is a Universal Format capturing data from different Models
  - Relational or Object DBMS
  - Document and File Repositories
- Semantic (and structural) heterogeneity occurs when there is a disagreement about the meaning, interpretation, or intended use of the same or related data

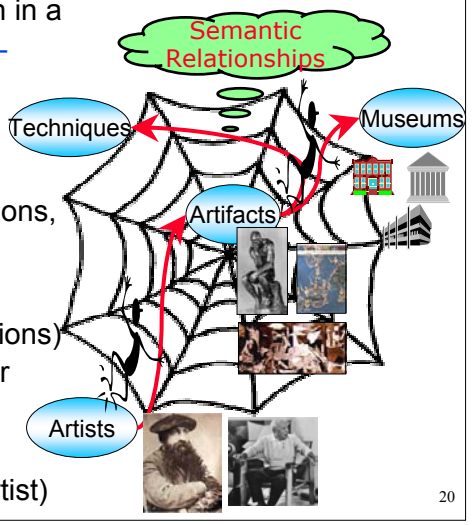
Interoperability is still an Open Issue !

- Semantic discrepancies :
  - Synonymy & Polysemy & Taxonomy
    - <ARTIFACT> vs. <ARTEFACT>
    - is <ARTWORK> paintings or songs ?
    - how < … Style='Impressionism'> is related to < … Style='Pointillism'> ?
- Structural discrepancies :
  - Aggregation
    - <NAME><FIRST>Claude</FIRST><LAST>Monet</LAST></NAME> vs <NAME>Claude Monet</NAME>
  - Type
    - <ARTIFACT Kind='Painting'> ... </ARTIFACT> vs <PAINTING>Claude Monet</PAINTING>
- Syntactic discrepancies :
  - <ARTIST Name='Claude Monet'> ... </ARTIST> vs <ARTIST> <NAME>Claude Monet</NAME> ... </ARTIST>

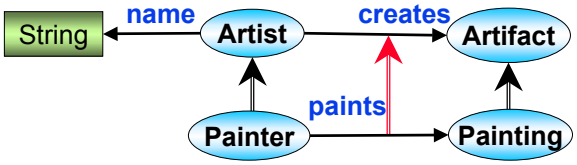
More than Web Data: Semantics on the Web
More than Web Applications: Web Services

The Semantic Web Vision: A Web of Meaning

- The "Next Generation Web" aims to provide infrastructure for expressing information in a precise, human-readable, and machine-interpretable form
- Enable both syntactic and semantic/ structural interoperability among independently-developed Web applications, allowing them to efficiently perform sophisticated tasks for humans
- Enable Web resources (data & applications) to be accessible by their meaning rather than by keywords and syntactic forms
  - Conceptual Navigation & Querying
  - Inference Services (Picasso is an Artist)



A First Step Towards the SW: RDF and RDFS

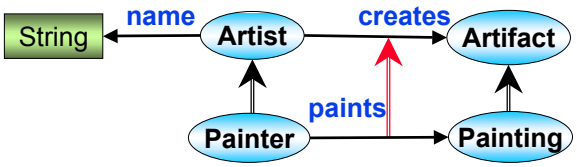



```
<Artist rdf:about="picasso132">
  <name>Pablo Picasso</name>
  <creates>
    <Artifact rdf:about=
    http://www.artchive.com/woman.jpg/>
  </creates>
</Artifact>
```

```
<Artist rdf:about="picasso132"
        name ="Pablo Picasso">
  <creates Artifact =
    http://www.artchive.com/woman.jpg/>
</Artifact>
```

```
<Painter rdf:about="picasso132">
  <name>Pablo Picasso </name>
  <paints>
   <Painting rdf:about=
   "http://www.artchive.com/woman.jpg"/>
 </paints>
 <paints>
   <Painting rdf:about="http://
    museoreinasofia.mcu.es/guernica.gif">
    </Painting>
 </paints>
</Painter>
```


A First Step Towards the SW: RDF and RDFS




```
<rdfs:Class rdf:ID="Artist"/>

<rdfs:Class rdf:ID="Artifact"/>

<rdfs:Class rdf:ID="Painter">
 <rdfs:subClassOf rdf:resource="#Artist"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Painting">
 <rdfs:subClassOf rdf:resource="#Artifact"/>
</rdfs:Class>

<rdf:Property rdf:ID="name">
 <rdfs:domain rdf:resource="#Artist"/>
 <rdfs:range rdf:resource="http://www.w3.org/
                  rdf-datatypes.xsd#String"/>
</rdf:Property>
```

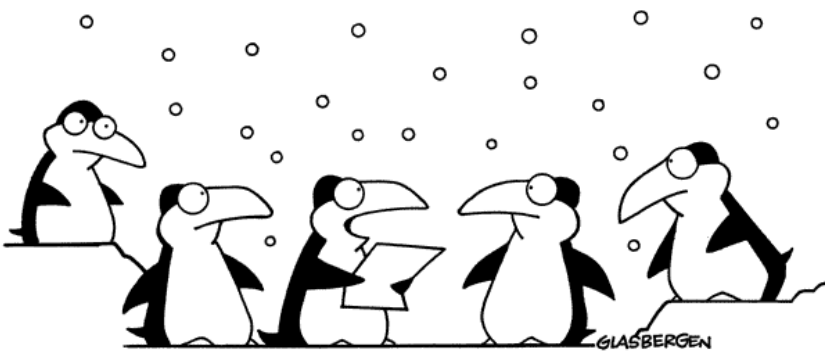
```
<rdf:Property rdf:ID="creates">
 <rdfs:domain rdf:resource="#Artist"/>
 <rdfs:range rdf:resource="#Artifact"/>
</rdf:Property>

<rdf:Property rdf:ID="paints">
 <rdfs:domain rdf:resource="#Painter"/>
 <rdfs:range rdf:resource="#Painting"/>
 <rdfs:subPropertyOf
              rdf:resource="#creates"/>
</rdf:Property>

<rdf:Property rdf:ID="created">
 <rdfs:domain rdf:resource="#Painting"/>
 <rdfs:rangerdf:resource="http://www.w3.org/
                  rdf-datatypes.xsd#Date"/>
</rdf:Property>
```

Is RDF/S the Solution to Interoperability?

- RDF/S abstracts from the syntactic discrepancies of XML data (elements vs attributes)
  - but it introduce new ones, related to its own model & syntax (classes vs properties, unique identifiers of resources)
  - ➔we can't read arbitrary XML data and interpret them as RDF!
- RDF/S provides core primitives for modeling the semantics of data in a domain of discourse (extended ER models)
  - however application data reside in autonomous sources, structured according to different schemas
  - ➔we can't expect that all existing data will be published on the SW as RDF/S data committing to one commonly agreed ontology (schema)!
- We still need expressive languages for mapping ontologies as well as translate accordingly the data from one application to another
  - finding semantic mappings is now the bottleneck!
  - ➔largely done by hand, labor intensive & error prone !

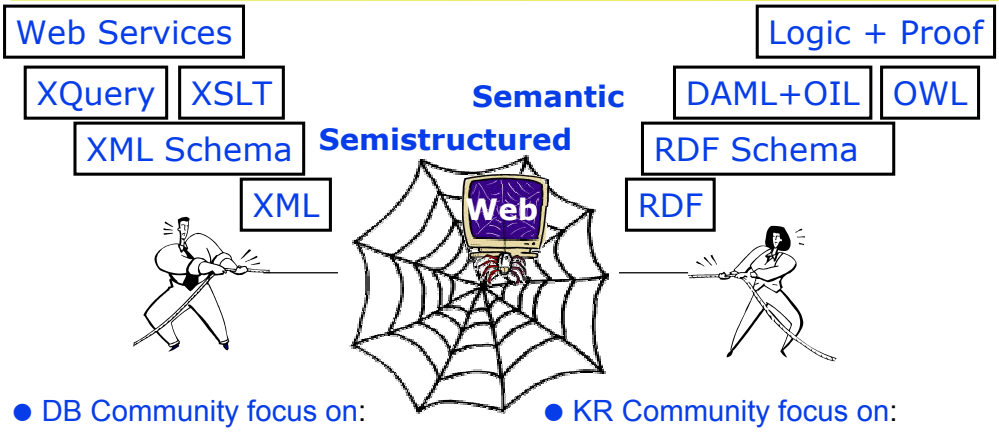
Diversity is a Feature!

- Semantic/Structural heterogeneity is not a drawback, but a feature of large scale distributed systems in a dynamic and open information universe



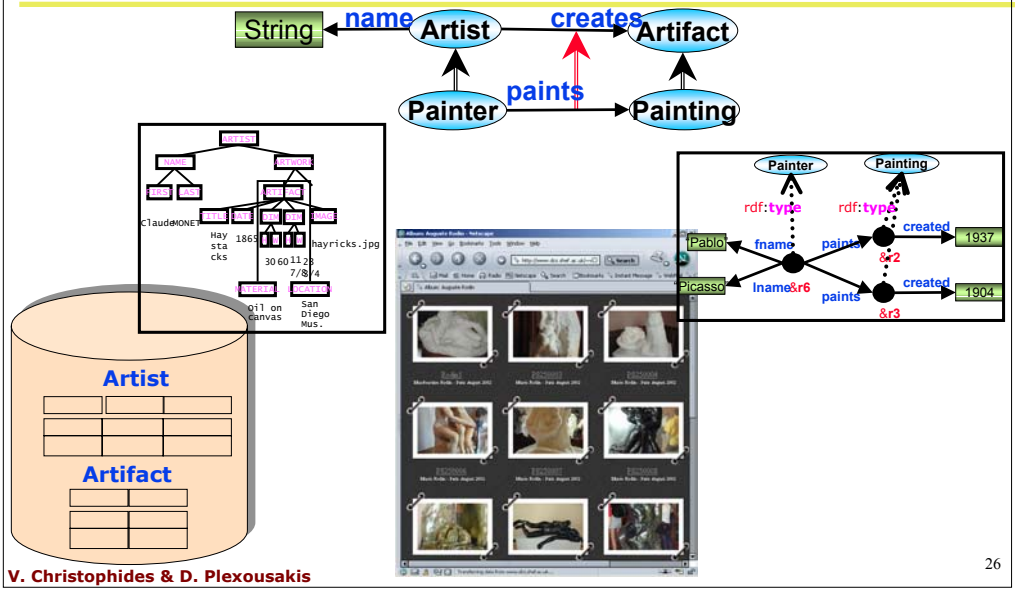
"They say we're not placing enough emphasis on diversity."

Two Cultures on the Future Web: DB vs KR

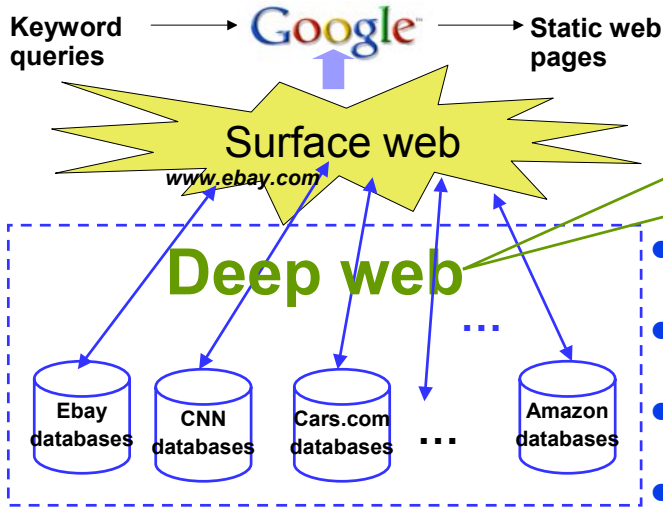


- DB Community focus on:
  - XML Data Semantics (Typing, Constraints)
  - XML Data Manipulation Languages (Querying, Views, Programming)
- KR Community focus on:
  - Ontology Languages (Frame / Description Logics)
  - Reasoners and Theorem Provers

Similar Motivations but different Application Contexts!



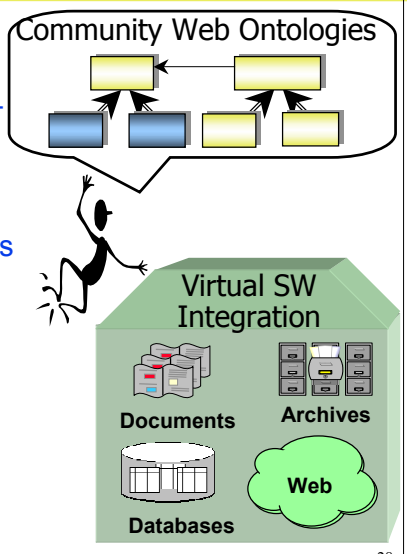
Visible (Surface) vs Invisible (Deep) Web



- Variety of Data formats & search mechanisms
- Accessible from specific HTML pages
- Higher Quality Information
- Not indexed by Google or other major search engines

Our Vision: Combine DB and KR Approaches

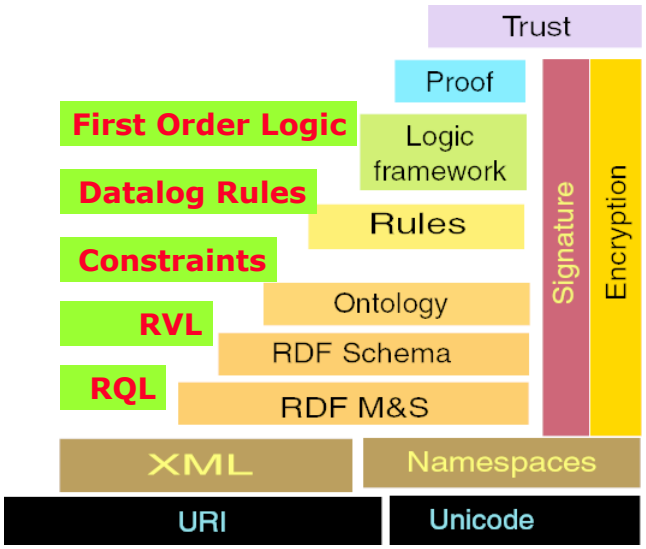
- Provide a useful, comprehensive, and high-level access to community resources
  - Ontologies as shared, formal conceptualizations of particular domains
- Build scalable technologies for managing semantically rich data and metadata
  - Declarative Querying/Viewing Languages
  - Efficient Storage for Voluminous Descriptive Information
- Support an expressive SW Integration Middleware
  - Establish Mapping/Translation Rules
  - Reformulate Conceptual Queries
  - Exploit data semantics for Query Optimization and Consistency Checking



W3C Semantic Web Activity

- Semantic Web Activity (http://www.w3.org/2001/sw/)
  - "Established to serve a leadership role, in both the design of enabling specifications and the open, collaborative development of technologies that support the automation, integration and reuse of data across various applications"
  - Successor to the W3C Metadata Activity
- RDF Core Working Group (http://www.w3.org/2001/sw/RDFCore/)
  - Responsible for the Resource Description Framework (RDF)
- Web Ontology Working Group (http://www.w3.org/2001/sw/WebOnt/)
  - Charter: Build upon the RDF Core work a language for defining structured web based ontologies which will provide richer integration and interoperability of data among descriptive communities
  - Developing Ontology Web Language (OWL)
    - Based on DAML+OIL, developed in DARPA's Agent Markup Language program

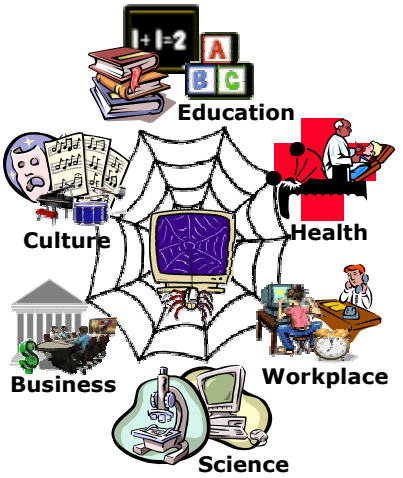
SW Layer Cake and ICS-FORTH Vision



Resource Description Framework (RDF)

RDF Objectives

- Enables communities to define their own descriptive semantics of Web resources
  - we can disagree about semantics, but share the same infrastructure (editors, query languages, databases, etc.)
- Imposes some structural constraints on the encoding of resource descriptions
  - for consistent exchange and processing of metadata on the Web
- Facilitates the development of descriptive vocabularies without central coordination
  - mechanisms for reusing and refining concepts, properties, etc.
  - mechanisms for extending resource descriptions in a peer-to-peer fashion



What is a Resource Description ?

Resource Description

author

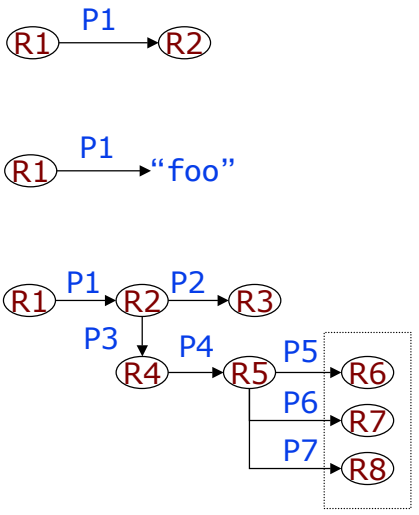
title

publisher

Resource

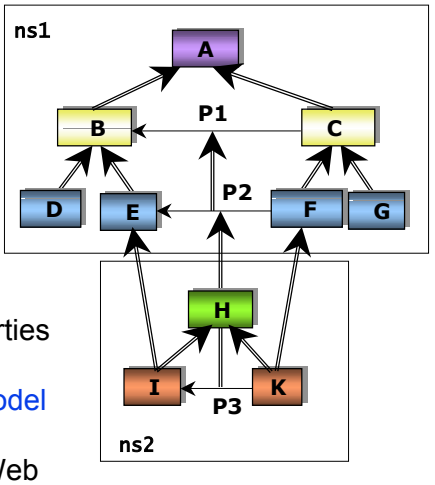
The Core RDF Data Model

- RDF: enables communities to describe their resources in a quite natural and flexible way
  - Data Model: Directed Labeled Graphs
    - Nodes: Resources (URIs) or Literals
    - Edges: Properties – Attributes or Relationships
    - Statement: assertion of the form *resource, property, value*
    - Description: set of statements concerning a resource
  - XML syntax



The Core RDFS Data Model

- RDFS: enables communities to share machine readable tokens and define human readable labels
  - Node labels (types) are defined as classes
    - XML Schema Literal data types
  - Edge labels (predicates) are defined as properties of these classes
    - domain and range constraint
  - Subsumption of both classes & properties (simple & multiple *is_A*)
- RDFS is expressible in the basic RDF model and syntax
  - vocabularies can be also viewed as Web resources identified by a namespace URI

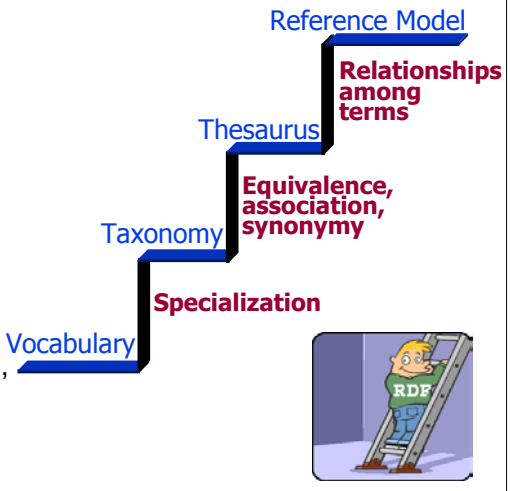


Looking at Existing RDF Applications

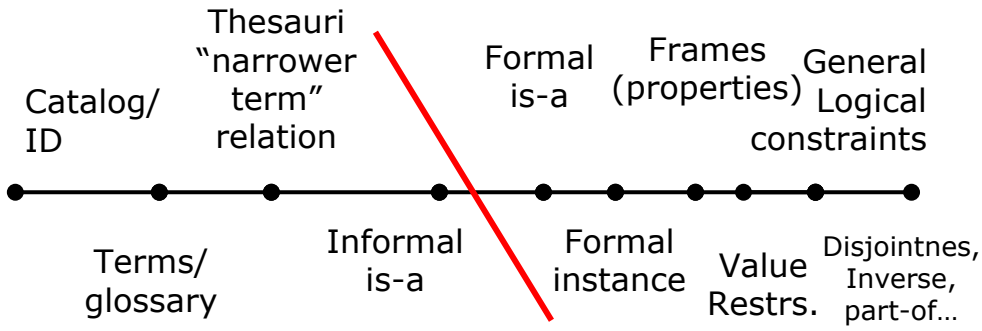
- Cultural Heritage/ Archives/ Libraries
- Educational/ Academic /Learning
- Publishing/ News
- Audio-Visual
- Geospatial/ Environmental
- Biology/ Medicine
- E-Commerce
- Ubiquitous/ Mobile/ Grid Computing
- Cross-Domain

What Descriptive Semantics RDF/S can capture?

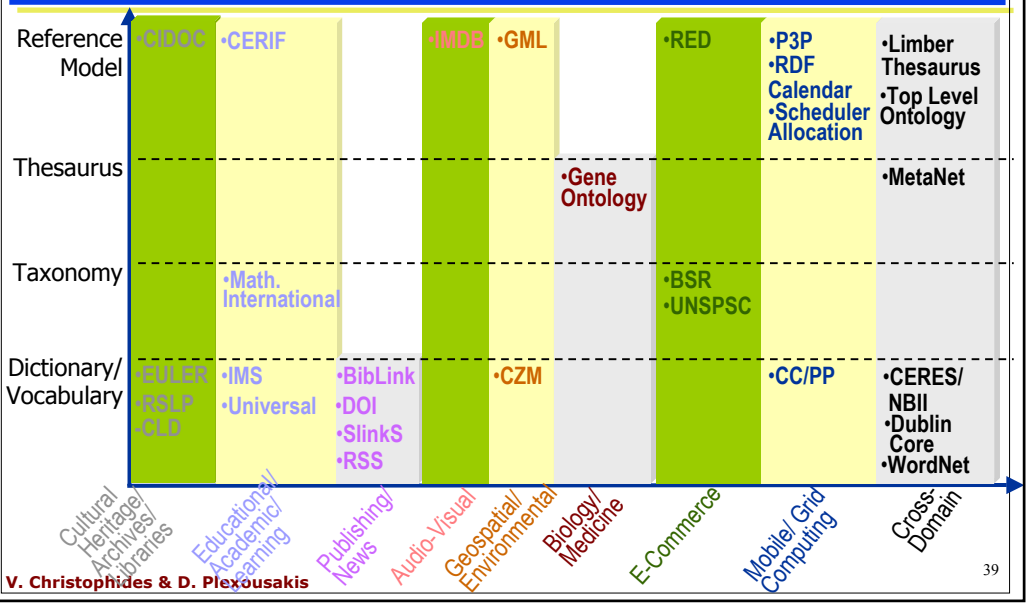
- Dictionaries/ Vocabularies
  - simple lists of terms and their definitions
- Taxonomies
  - Specialization between terms
- Thesauri
  - Broader/narrower terms, equivalence, association and synonymy relations
- Reference Models
  - A representation vocabulary of the concepts in the subject area, the relations among the terms and the way the terms can or cannot be related to each other



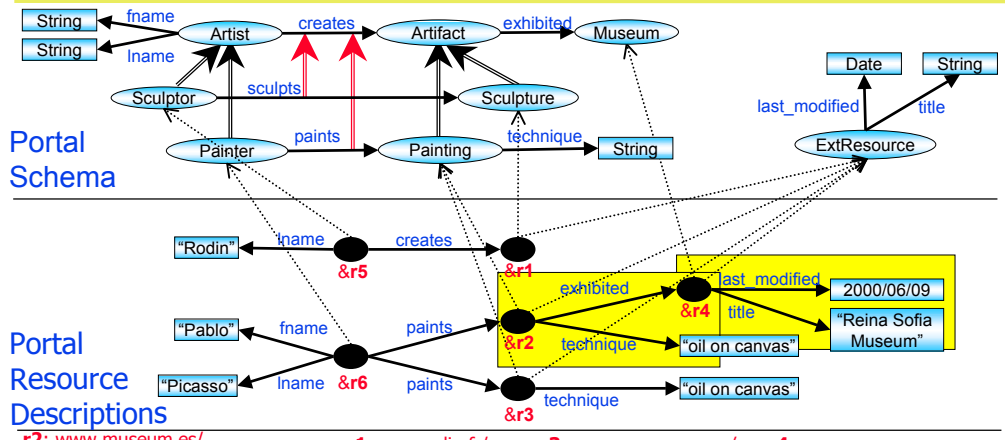
Ontologies - What Are They?



A First Categorization of existing RDF Schemas



A Cultural Community Web Portal in RDF



Web Resources

r2: www.museum.es/ guernica.jpg

r1:www.rodin.fr/ thinker.gif

r3:www.museum.es/ woman.qti

r4:www.museum.es

Advantages of RDF/S vs. Well-Known Formalisms

- Relational or Object Database Models (ODMG, SQL)
  - Instances may be associated with different properties
  - Heterogeneous Collections
- Semistructured or XML Data Models (OEM, UnQL, YAT, XML Schema)
  - Labels on both nodes or edges
  - Both class and property subsumption
- Knowledge Representation Languages (Telos, DL, F-Logic)
  - Supports complex values (bags, sequences)

Why a Formal 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

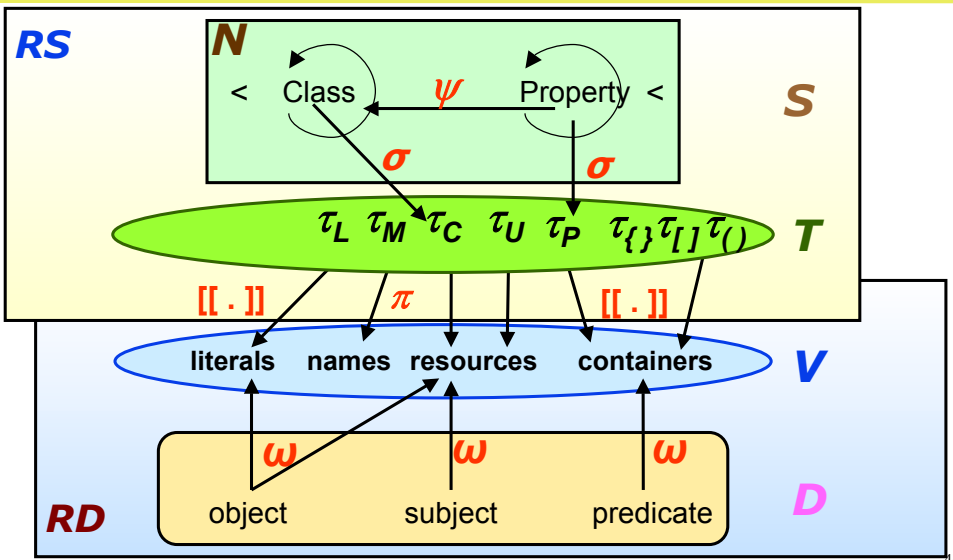
Why a Type System for RDF ?

- For error detection & safety:
  - to correctly understand statements of interest
    - e.g., don't confuse resource URIs with class/property names!
  - to enforce safety of operations
    - e.g., don't do float arithmetic on classes!
  - to check valid compositions of operations
    - e.g., don't ask the subproperties of the range of a class!

- For performance:
  - to design better storage (improving clustering, etc.)
  - to efficiently process queries (rewriting path expressions, etc.)

- We need a full-fledged Data Definition Language for RDF !
  - RDF Schema is viewed more as an ontology & modeling tool

A Formal Data Model for RDF/S



A Formal Data Model for RDF/S

- Type System:

  $\tau = \tau_L \mid \tau_U \mid \tau_M \mid \tau_C \mid \tau_P[\tau_1, \tau_2] \mid \{\tau\} \mid [1{:}\tau + 2{:}\tau + \ldots + n{:}\tau] \mid (1{:}\tau + 2{:}\tau + \ldots + n{:}\tau)$

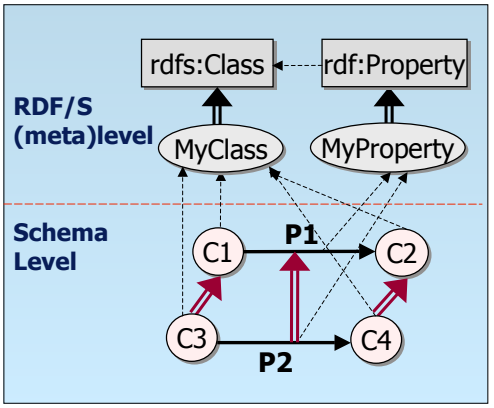
- Interpretation Function:
  - Literal types: $[[\ \tau_L\ ]] = dom(\tau_L)$
  - Resource types: $[[\ \tau_u\ ]] = u \in U$
  - MetaClass types: $[[\ \tau_M\ ]] = \{v \mid v \in \pi^*(m)\}$
  - Class types: $[[\ \tau_C\ ]] = \{v \mid v \in \pi^*(c)\}$
  - Property types: $[[\ \tau_p[\tau_1, \tau_2]\ ]] = \{ [v_1, v_2] \mid v_1 \in [[\tau_1]],\ v_2 \in [[\tau_2]] \} \cup \{[[\ \tau_p[\tau'_1, \tau'_2]\ ]] = \{ [v'_1, v_2] \mid v'_1 \in [[\tau'_1]],\ v'_2 \in [[\tau'_2]] \}\ p' < p\}$
  - Bag types: $[[\ \{\tau\}\ ]] = \{ \{v_1, \ldots, v_j\} \mid j > 0,\ \forall\ i \in [1..j]\ v_i \in [[\ \tau\ ]] \}$
  - Seq types: $[[\ [\tau]\ ]] = \{ [1{:}v_1, 2{:}v_2, \ldots, n{:}v_n] \mid n > 0,\ \forall\ i \in [1..n]\ v_i \in [[\ \tau_i\ ]] \}$
  - Alt types: $[[\ (1{:}\tau_1 + 2{:}\tau_2 + \ldots + n{:}\tau_n)\ ]] = \{i{:}v_i \mid,\ \forall\ i \in [1..n]\ v_i \in [[\ \tau_i\ ]] \}$

A Formal Data Model for RDF/S

- An RDF schema is a tuple: S = (RS, σ)
  - RS = (V_S, E_S, H, ψ, λ, N, <) is a valid RDF Schema
  - σ is a type function: N $\rightarrow$ **T**
- An RDF description base, instance of a schema S, is a tuple: D = (RD,ω)
  - RD=(RS, V_D, E_D, ψ, λ) is a set of valid resource descriptions
  - ω is a valuation function: $V_D \cup E_D \rightarrow$ **V** such that:
    - $\forall$ n $\in V_D$, ω (n) $\in$ [[σ (λ (n))]]
    - $\forall$ p $\in E_D$ from node n to n', [ω(n), ω(n')] $\in$ [[p]]

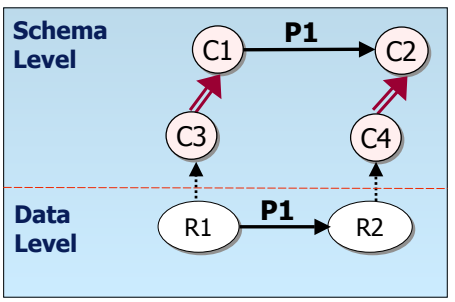
Imposed Constraints (1)

- For a valid RDF/S schema:
  - The domain and range of a property must be unique and always defined
  - The domain (range) of a sub-property must be subsumed by the domain (range) of the super-property
  - A subsumption hierarchy can be defined only among names of the same type (metaclasses, classes and properties)
  - No cycles in the subsumption hierarchies



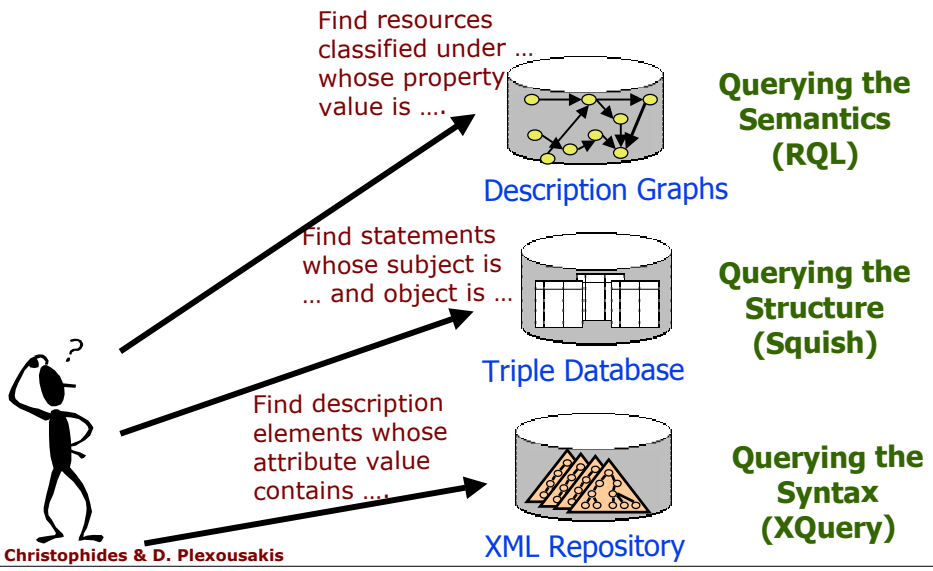
Imposed Constraints (2)

- For a valid RDF/S description base:
  - A literal value is instance of one and only one literal type
  - A resource is always instance of the most "specialized" class w.r.t the subsumption hierarchy
  - The resources connected by a property at data level must be instances of classes equal or subsumed by the property domain and range



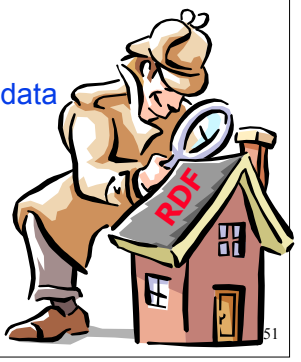
Querying and Viewing RDF/S

The RDF Query Language



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 semistructured or XML query languages to RDF, but also:
    - treats properties as first-class citizens
    - exploits taxonomies of node and edge labels
    - allows querying of schemas as semistructured data



Using Names to Access RDF Schema/Data Graphs

- Querying the RDF/S (or user-defined) meta-schema names
  - **Class**
  - **Property**
  - **Literal**

Includes Painter & Sculptor

- Querying the RDF/S user-defined schema names
  - Artist
  - creates

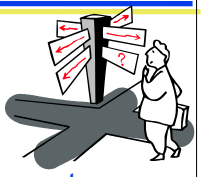
Includes paints & sculpts

- The Namespace Clause
  - ns1:ExtResource
    using namespace ns1= &ns2:www.oclc.org/schema.rdf

Querying Large RDF Schemas with RQL

- Basic Class Queries
  - subclassof(Artist)
  - subclassof^(Artist)
  - superclassof(Painter)
  - superclassof^(Painter)
  - topclass
  - leafclass
  - nca(Sculptor,Painting)
- Basic Property Queries
  - subpropertyof(creates)
  - subpropertyof^(creates)
  - superpropertyof(paints)
  - superpropertyof^(paints)
  - topproperty
  - leafclass
  - nca(paints,sculpts)
- Basic Class and Property Queries
  - domain(creates)
  - range(creates)

Class & Property Querying



- Find the domain and range of the property creates

  seq (**domain**(creates), **range**(creates))

- 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

RQL Query Result

property exhibited	class Museum
property technique	literal string

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  - <rdf:Bag ID="bag177976867">
    - <rdf:li>
      - <rdf:Seq>
          <rdf:li rdf:type="property" rdf:resource="exhibited" />
        - <rdf:li>
          - <rdf:Alt>
              <rdf:li rdf:type="class" rdf:resource="Museum" />
            </rdf:Alt>
          </rdf:li>
        </rdf:Seq>
      </rdf:li>
    - <rdf:li>
      - <rdf:Seq>
          <rdf:li rdf:type="property" rdf:resource="technique" />
        - <rdf:li>
          - <rdf:Alt>
              <rdf:li rdf:type="literal" rdf:resource="string" />
            </rdf:Alt>
          </rdf:li>
        </rdf:Seq>
      </rdf:li>
    </rdf:Bag>
</RDF>
```


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**} or
  select **$Y**, **$Z** from creates{**$Y**}, exhibited{**$Z**}
  where **$Y** <= domain(exhibited)

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

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

- Find schema properties having as domain or range a meta-class

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


Querying Complex Portal Descriptions with RQL

- Find all resources

Resource

Multiply classified resources

- Find the resources of type ExtResource and Sculpture

ExtResource **intersect** Sculpture
ExtResource **minus** Sculpture
ExtResource **union** Sculpture

- Count the total number of Painter resources

count(Painter)

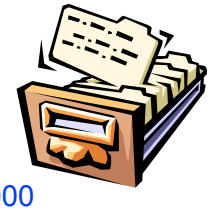
Aggregate functions

Filtering RDF Descriptions with RQL

- Find the file size of the resource with URI "www.artchive.com/rembrandt/abraham.jpg"

```
select X
from {X}file_size{Y}
where X = &www.artchive.com/rembrandt/abraham.jpg
```

Conditions on URIs



- Find the resources that have been modified after year 2000

```
select X
from {X}last_modified{Y}
where Y >= 2000-01-01
```

Conditions on Dates

Navigating in Description Graphs using RQL

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

```
select X
from Museum{X}.last_modified{Y}
```

- 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 schema properties and their values of resources classified under the class Artist or its subclasses (i.e., restrict property source values using node labels)

```
select X, @P , Y
from {X}@P{Y}
where domain(@P) <= Artist
```

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

```
select @P, Y, Z
from {;Painting}@P.{Y}last_modified{Z}
```


Using Schema to Filter Resource Descriptions

- Find the properties emanating from ExtResources and their source and target values

```
select  x , @P , y
from    {x;ExtResource}@P{y}
```

Data paths foreseen in the schema

- Find the properties applied on instances of the class ExtResource and their source and target values

```
select  x, @P, y
from    ExtResource{x}.@P{y}
```

Data paths not foreseen in the schema

Notice the difference

resource	property	resource
http://www.museum.es/guernica.jpg	exhibited	http://www.museum.es

resource	property	string
http://www.museum.es/guernica.jpg	technique	oil on canvas

resource	property	string
http://www.museum.es/woman.qti	technique	oil on canvas

resource	property	string
http://www.museum.es	title	Reina Sofia Museum

resource	property	date
http://www.museum.es	last_modified	2000-06-09T12:30:34+00:00

Discover the Schema of RDF Descriptions

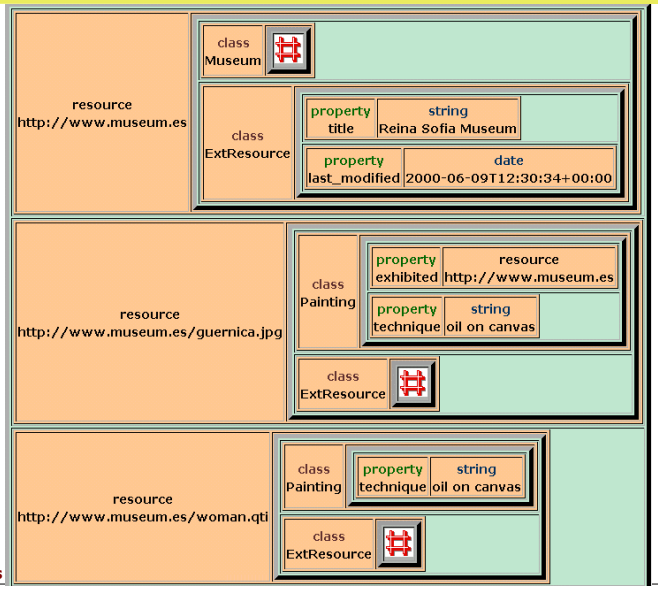
- Find the classes under which is classified the resource with URL "www.museum.es"

typeof (&www.museum.es)

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

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

RQL Query Result



And if you still like triples …

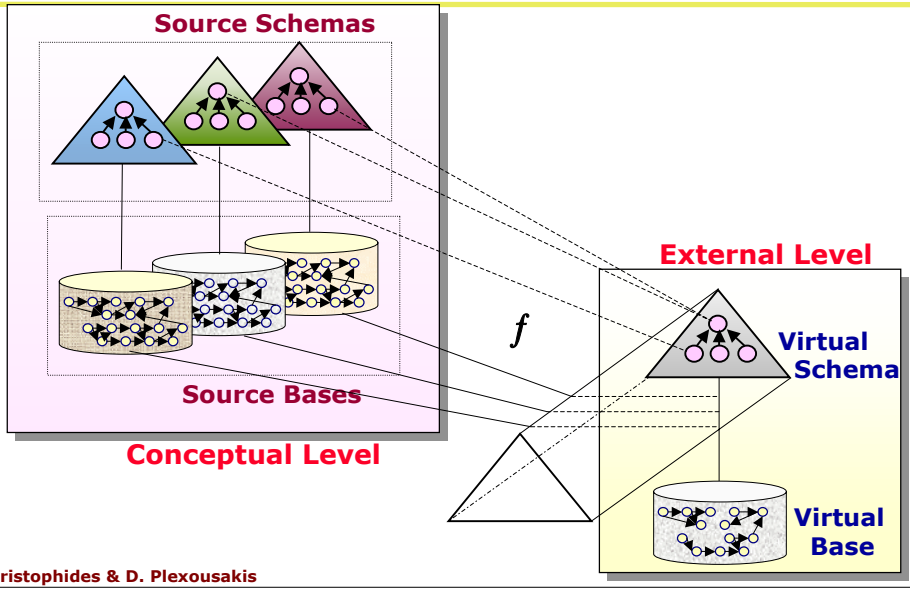
- Find the description of resources which are not of type ExtResource

```
(
  (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})
)
```

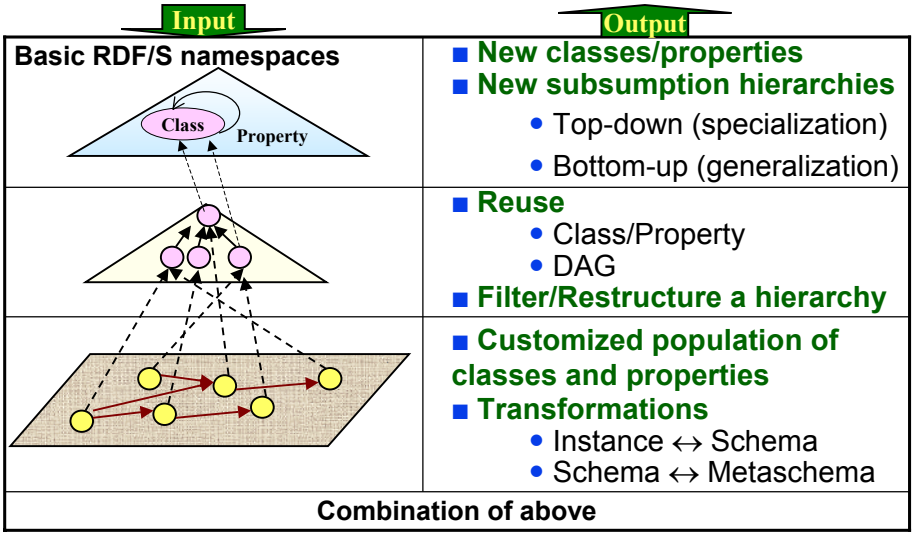
The RDF View Language: RVL

- Declarative view definition language for virtual RDF description bases and schemas
  - relies on the RQL typed data model
  - follows also a functional approach (object construction operators)
  - ensures logical data independence
    - view specifications are independent from those of the source schemas and bases,
    - the semantics of existing virtual schemas is not be altered by the definition of new ones
  - supports object-preserving and object-generating views
  - provides heavy data restructuring facilities
  - allows users to query and create views using both source and virtual schemas

The RVL Approach



The RVL Functionality



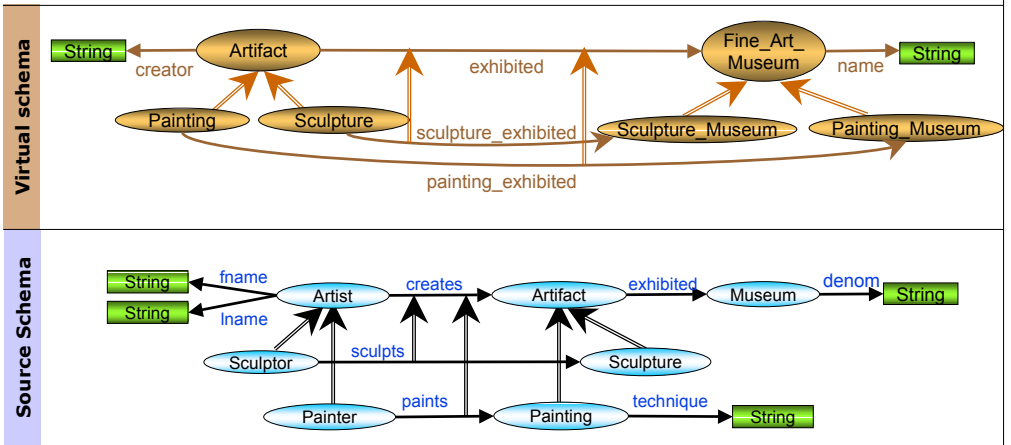
The RVL Syntax

```
[ VIEW operator
  FROM RQL_path_expression
  WHERE filtering_conditions
  USING NAMESPACE source_schema_namespace]
.....
.....
.....
CREATE NAMESPACE RVL_view_namespace
```

The RVL Operators

- Instantiation Operator
  - Creates virtual (meta)classes and properties
  - Populates virtual (meta)classes and properties
  - Up(Down)grades the abstraction level of a source entity
- Subsumption Operator
  - Creates new subsumption hierarchies of virtual (meta)classes and properties
  - Reorganizes source subsumption hierarchies of (meta)classes and properties

An RVL virtual RDF/S schema and base



An RVL virtual RDF/S schema and base

- CREATE NAMESPACE myview=&http://www.ics.forth.gr/mycult.rdf#
- VIEW Class("Fine_Art_Museum"), Class("Painting_Museum"),
  Class("Sculpture_Museum"), Class("Artifact"),
  Class("Painting"), Class("Sculpture")
- VIEW Property("name", Fine_Art_Museum, xsd:string),
  Property("title", Artifact, xsd:string),
  Property("creator", Artifact, xsd:string),
  Property("exhibited", Artifact, Fine_Art_Museum),
  Property("sculpture_exhibited",Sculpture, Sculpture_Museum),
  Property("painting_exhibited", Painting, Painting_Museum)
- VIEW Fine_Art_Museum<Sculpture_Museum>,
  Fine_Art_Museum<Painting_Museum>,
  Artifact<Painting>, Artifact<Sculpture>
  exhibited<sculpture_exhibited>,
  exhibited<painting_exhibited>

An RVL virtual RDF/S schema and base

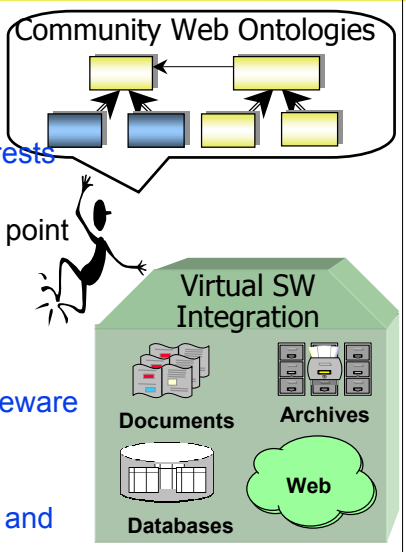
- VIEW Painting(X), painting_exhibited(X,Y), Painting_Museum(Y),
  name(Y,W), title(X,K), creator(X,Z)
  FROM {Z}n1:creates{X; n1:Painting}.n1:exhibited{Y}.n1:denom{W},
  {X}n1:title{K}
  USING NAMESPACE n1=&http://www.culture.mus/cult.rdf#

- VIEW Sculpture(X), sculpture_exhibited(X,Y), Sculpture_Museum(Y),
  name(Y,W), title(X,K), creator(X,Z)
  FROM {Z}n1:creates{X; n1:Sculpture}.n1:exhibited{Y}.n1:denom{W},
  {X}n1:title{K}
  USING NAMESPACE n1=&http://www.culture.mus/cult.rdf#

Semantic Interoperability: the role of Semantic Web Middleware

Our Vision for the SW: Community Webs

- What is a Community Web?
  - A group of people sharing a domain of discourse and a set of information resources (e.g., data, documents, services) and having some common interests
    - Commerce, Education, Health
- The main requirement is to provide a single point of useful, ubiquitous, comprehensive, and integrated access to community information resources
  - Web Portals
- Support an expressive SW Integration Middleware
  - Establish Mapping/Translation Rules
  - Reformulate Conceptual Queries
  - Exploit semantics for Query Optimization and Consistency Checking

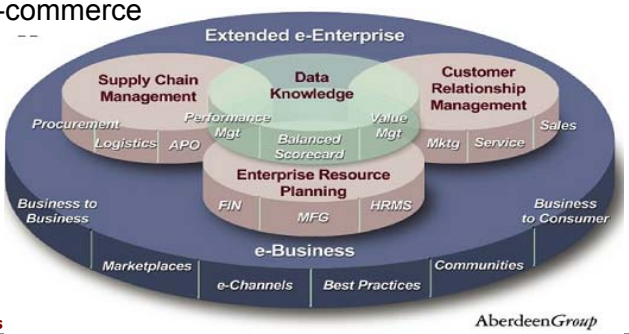


Impact

- The Enterprise Portal Software Market Size (source: Plumtree)

Analyst Firm	Report Date	Market Size	Growth Rate
Gartner	06 - 2002	2001: $709M	24% - 2006
IDC	06 - 2002	2001: $550M	41% - 2006
Delphi	12 - 2002	2002: $787M	20% - 2004

- The case of B2B E-commerce



Old Wine in New Bottles?

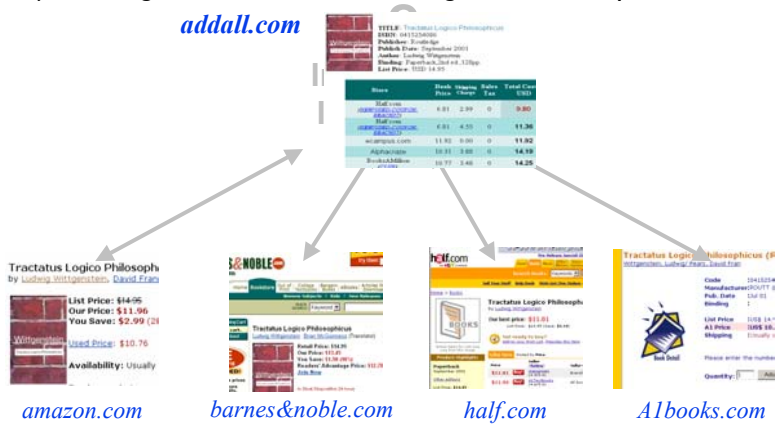
- The Information Integration Challenge:
  - *Given*: data sources S_1, ..., S_k (DBMS, web sites, ...) and user questions Q_1,...,Q_n that can be answered using the S_i
  - *Find*: the answers to Q_1, ..., Q_n
- The Database Perspective: source = "database"
  - S_i has a schema (relational, XML, OO, ...)
  - S_i can be queried
  - define virtual (or materialized) integrated views V over S_1,...,S_k using database query languages
  - questions become queries Q_i against V(S_1,...,S_k)
- Why a Database Perspective?
  - For all the good reasons: scalability, efficiency, reusability (declarative queries), physical and logical data independence
- … complemented by salient KR abstractions / languages / mechanisms

Technical Issues

- Integration Method and Architecture
  - federated DBs, wrapper-mediator approach, GAV/LAV, warehouse/on-demand, ...
- Suitable KRDB Formalisms and Frameworks
  - XML, DTDs/XML Schema, XPath, XQuery, ...
  - RDF(S), Ontologies, Description Logics, DAML+OIL, OWL
  - querying, deduction, subsumption, classification, ...
- Algorithms and Implementation
  - query answering using views, query reformulation, query / view composition, reasoning, source capabilities, ...
- Information Integration Scenario and Scope
  - simple/complex, single/multiple worlds, ...

Scenario #1: a "simple" world

- On-line shopping
  - Scroodge: "*Where can I get the cheapest copy (including shipping cost) of Wittgenstein's Tractatus Logicus-Philosophicus within a week?*"



Scenario #2: multiple "simple" worlds

- Buying a house: *What houses for sale under 300kE have at least 2 bathrooms, 2 bedrooms, a nearby school ranking in the upper third, in a neighborhood with below-average crime rate and diverse population?*

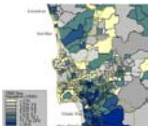
?

Information Integration


Realtor


Crime Stats


School Rankings

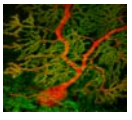

Demographics

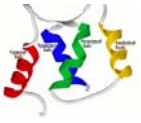
Scenario #3: multiple complex worlds

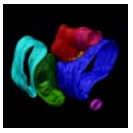
- E-neuroscience: *What is the distribution of rat proteins with more than 70% homology with human NCS-1? Any structure specificity? How about other rodents?*

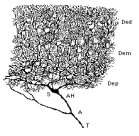
?

Information Integration

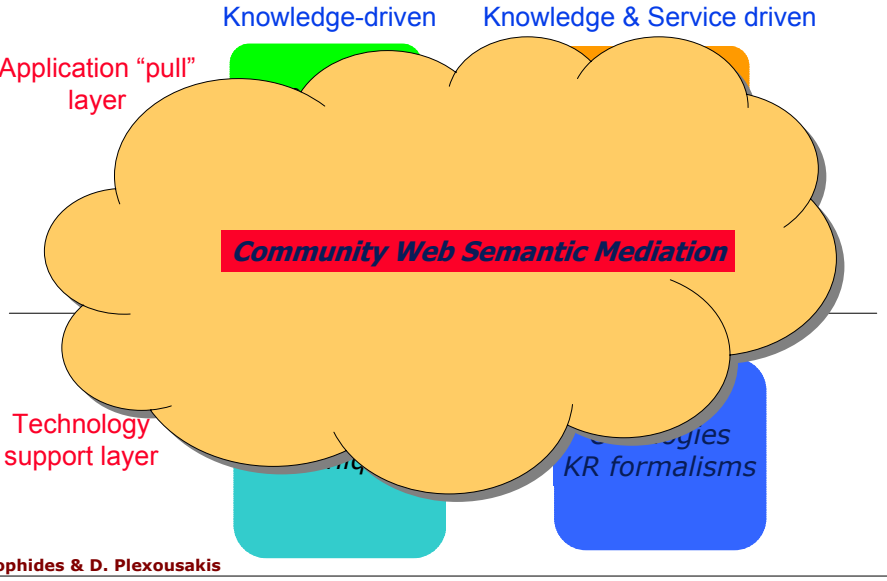

protein localization (NCMIR)


sequence info (CaPROT)


morphometry (SYNAPSE)


neurotransmission (SENSELAB)

The Integration Landscape: Contributing Forces



Semantic Web Middleware

- Design Principles:
  - Philosophical:
    1. K.I.S.S. (keep it simple stupid)
    2. Think globally, work locally
    3. Learn from history (internet and web evolution)
  - Technical:
    1. Formal basis
    2. Makes semantics explicit
    3. Accounts for expressive data models and KR schemes
    4. Serves as a "glue" for information integration and service interoperability
    5. Abstains from low-level commitments

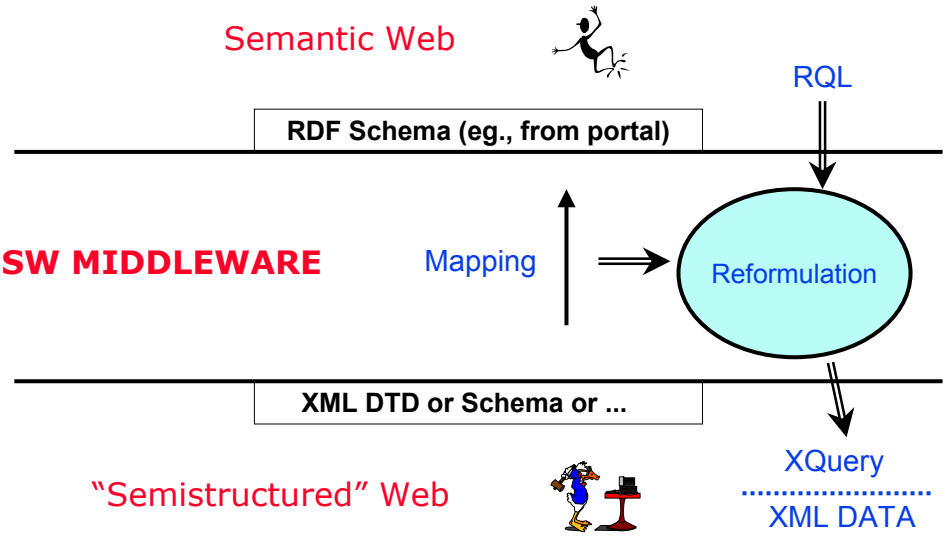
Semantic Web Middleware

- The bulk of existing data is not yet in RDF/S (or any other form suitable for the SW)
  - Data physically stored in relational DBs and/or published as virtual XML
- SW applications require viewing data as virtual RDF
  - valid instances of domain or application-specific RDF/S schemas
- Need the ability to manipulate data with high-level query or view languages (RQL, RVL)
- How to do it?
  - republish XML as RDF
  - publish relational data as RDF
  - do both

Semantic Web Middleware

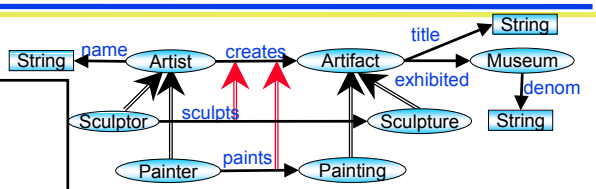
- Practical concerns:
  - XML publishing systems often provide an XML query interface.
    - SW middleware can function as an alternative to the XML publishing systems; SW middleware provides direct access to underlying DBMSs
  - SW middleware may also be required to integrate DBMS data with data in native XML storage
- SW middleware tasks:
  - Specify mappings: XML→ RDF, RDB → RDF
  - Verify conformance to the semantics of employed schemas
  - Reformulate queries (i.e., compose RQL queries with mappings to produce XML or RDB queries)
  - Provide abstractions of RDF data/schemas (views)
  - Compose queries with views

Republish XML as RDF



Motivating Example

```
<ArtDB>
<Sculptor name="Rodin">
   <sculpts>
      <Sculpture title="thinker"/>
   </sculpts>
</Sculptor>
<Painter name="Rodin">
   <paints>
      <Painting title="crucifixion"/>
   </paints>
</Painter>
<Painter name="Picasso">
   <paints>
      <Painting title="guernica">
         <exhibited>
           Reina Sofia
         </exhibited>
      </Painting>
   </paints>
</Painter>
</ArtDB>
```



Artifacts

title(key)	Artist	exhibited	kind
guernica	Picasso	ReinaSofia	Painting
crucifixion	Rodin	NULL	Painting
thinker	Rodin	NULL	Sculpture

Introducing a SW Middleware Server

- By designing (or importing) a (virtual) RDF/S cultural schema, we can answer queries using RQL
  - E.g., Q1: *"List the last names of all artists that have created artifacts exhibited at the Reina Sofia Museum"*

```
SELECT   Z
FROM     {X} creates.exhibited.title {V}, {X} name {Z}
WHERE    V = "Reina Sofia Museum"
```

- Actual data can only be queried using an XML language (e.g., XQuery) or SQL
- The RQL query needs to be reformulated into an XML query
- Reformulation cannot be ad hoc; needs to be driven by a formal description of the relationship between XML and RDF data
- Need a formal basis for expressing such mappings

Mappings: Background

- From relational database theory
  - query containment, query + view composition, query rewriting using views are solvable for a fairly large class of queries in the presence of certain classes of constraints (embedded implicational dependencies)
- A robust formalism to rely on: conjunctive queries and views (non-recursive Datalog)
- A formal data model for RDF/S
  - Validity constraints
- High-level query and view languages for RDF/S adhering to the formal model

XML to RDF Mapping

- Datalog rules with RVL atoms (head) and Xpath atoms (body)

```
...
Painter(X) :--  //Painter (X)                    populates class  Painter
...
Sculpture(X) :--  //Sculpture (X)                "          Sculpture
...
paints(X, Y) :--  //Painter (X),  .//Painting (X, Y)   populates relationship  paints
...
name(X, Y) :--  //Painter (X),  ./@name (X, Y)
                                                  populates   attribute   name
...
```

direct instances

abs-xpath (x) root — xpath — x

rel-xpath (x,y) x — xpath — y

RDB to RDF Mapping

- Datalog rules with RVL atoms (head) and Datalog atoms (body)

```
...
Painter(X) :--  Artifacts(_,X,_,"Painting")          populates class  Painter
...
Sculptor(X) :-- Artifacts(_,X,_,"Sculpture")          "          Sculpture
...
paints(X, Y) :-- Artifacts(Y,X,_,"Painting")   populates relationship   paints
...
name(X, Y) :-- Artifacts(_,X,_,"Painting"), Y=X
name(X, Y) :-- Artifacts(_,X,_,"Sculpture"), Y=X
                                                  populates   attribute   name
...
```

direct instances

N.B.: need to work around schematic and semantic discrepancies

Middleware Internal Model (1)

C_EXT : Class x Resource

P_EXT : Resource x Property x Resource

For reformulation, we translate into the internal model:

Sculpture(X) :-- //Sculpture (X) → C_EXT(Sculpture,X) :-- //Sculpture (X)

paints(X, Y) :-- //Painter (X), .//Painting (X, Y)

→ P_EXT(X, paints, Y) :-- //Painter (X), .//Painting (X, Y)

Middleware Internal Model (2)

CLASS : Class

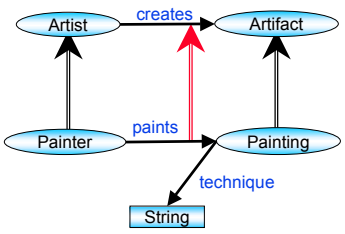
C_SUB : Class x Class

PROP : Class x Property x Class

P_SUB : Property x Property

+ a bunch of constraints

RDF Schema also gets translated into the internal model:



PROP(Painter, paints, Painting) :--

PROP(Painting, technique, String) :--

P_SUB(paints, creates) :--

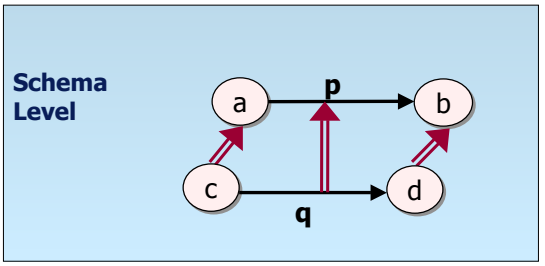
C_SUB(Painting, Artifact) :--

…

RDF/S Compatibility Constraints (1)

For a valid RDF Schema:

The domain (range) of a subproperty must be subsumed by the domain (range) of the super-property

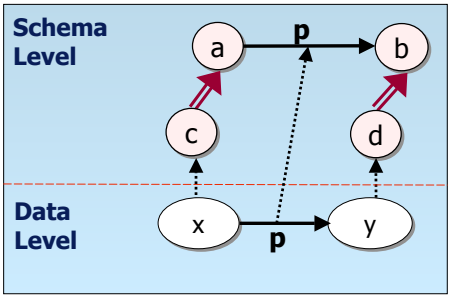


$$\forall a,p,b,c,q,d \;\; PROP(a,p,b) \wedge PROP(c,q.d) \wedge P\_SUB(q,p) \rightarrow C\_SUB(c,a) \wedge C\_SUB(d,b)$$

RDF/S Compatibility Constraints (2)

For a valid RDF description base:

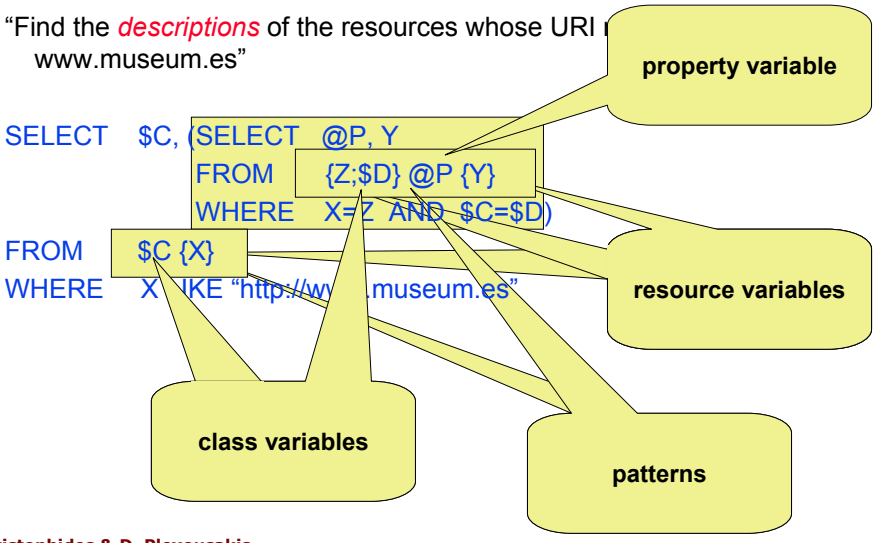
The resources connected by a property at the data level must be instances (i.e., direct instances of some subclasses) of the classes that are the property's domain and range



$$\forall a,p,b,x,y \; PROP(a,p,b) \wedge P\_EXT(x,p,y) \rightarrow \exists c,d \;\; C\_SUB(c,a) \wedge C\_SUB(d,b) \wedge C\_EXT(c,x) \wedge C\_EXT(d,y)$$

More Complex RQL Queries

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



Internal Translation of RQL Patterns

Conjunctive queries: $ans(X_1, X_2, \ldots, X_k) :- C_1, \ldots, C_n$,

where the C_i's are RQL class or property patterns

ans($C,X) :-- $C {X}

ans(x,c) :-- C_SUB(d,c), C_EXT(d,x)

---

ans(X,$C,@P,Y,$D) :-- {X;$C} @P {Y;$D}

ans(x,c,p,y,d) :-- PROP(a,p,b), P_SUB(q,p), P_EXT(x,q,y),
C_SUB(c,a), C_EXT(c,x), C_SUB(d,b), C_EXT(d,y)

simplifies only under the compatibility constraints

---

ans(X,@P,Y) :-- {X} @P {Y}

ans(x,p,y) :-- P_SUB(q,p), P_EXT(x,q,y),

Translation of Query Q1

```
SELECT   Z
FROM     {X} creates.exhibited.title {V}, {X} name {Z}
WHERE    V = "Reina Sofia Museum"
```

"Paths" provide shorthand notation for sequences of patterns:

```
SELECT   Z
FROM     {X} creates {Y}, {Y} exhibited {U}, {U} title {V}, {X} name {Z}
WHERE    V = "Reina Sofia Museum"
```

In the internal model:

```
ans(Z) :-- P_SUB(P1, name), P_EXT(X, P1, Z),
           P_SUB(P2, creates), P_EXT(X, P2, Y),
           P_SUB(P3, exhibited), P_EXT(Y, P3, U),
           P_SUB(P4, title), P_EXT(U, P4, "Reina Sofia Museum")
```

A conjunctive query!

All Together: An XPath/Datalog Program

```
ans(Z)   :--  P_SUB(P1, name), P_EXT(X, P1, Z),
              P_SUB(P2, creates), P_EXT(X, P2, Y),  …
…
P_SUB(paints, creates)   :--
P_SUB(sculpts, creates)   :--
…
P_EXT(X, paints, Y)  :--   //Painter (X),  .//Painting (X, Y)
…
P_EXT(X, name, X)  :--   //Sculptor (X),  ./@name(X, Y)
P_EXT(X, name, Y)  :--   //Painter (X),  ./@name(X, Y)
…
```

from query

from schema

from mapping

A reformulation, of sorts, but unacceptably inefficient!

Improving the Reformulation (1)

After "partial evaluation" using the schema facts:

ans(Z) :-- P_EXT(X, name, Z), P_EXT(X, paints, Y), …

ans(z) :-- P_EXT(X, name, Z), P_EXT(X, sculpts, Y), …
…
P_EXT(X, paints, Y) :-- //Painter (X), .//Painting (X, Y)
P_EXT(X, sculpts, Y) :-- //Sculptor (X), .//Sculpture (X, Y)
…
P_EXT(X, name, Y) :-- //Sculptor (X), ./@name(X, Y)
P_EXT(X, name, Y) :-- //Painter (X), ./@name(X, Y)
…

Improving the Reformulation (2)

After eliminating the intermediate predicates:

ans(Z) :-- //Painter (X), ./@name(X, Z) ,
//Painter (X), .//Painting (X, Y), …

ans(z) :-- //Sculptor (X), ./@name(X, Z),
//Painter (X), .//Painting (X, Y), …

unsatisfiable!

…

ans(z) :-- //Painter (X), ./@name (X, Z) ,
//Sculptor (X), .//Sculpture (X, Y), …

unsatisfiable!

ans(z) :-- //Sculptor (X), ./@name(X, Z),
//Sculptor (X), .//Sculpture (X, Y), …

…

Requires some reasoning about XPath that can be done with FO tools.

Reformulation, Finally

```
ans(Z)  :-  //Painter (X), .//Painting (X, Y),
            ./exhibited/text() (Y,"Reina Sofia Museum"),
            ./@name (X, Z)

ans(Z)  :-  //Sculptor(X), .//Sculpture (X, Y),
            ./exhibited/text() (y,"Reina Sofia Museum"),
            ./@name (x, z)
```

- More minimization techniques were used to get to this:

  A. Deutsch, V. Tannen,
  "Reformulation of XML Queries…", in ICDT'03,
  "MARS: a System for Publishing XML…", in VLDB'03

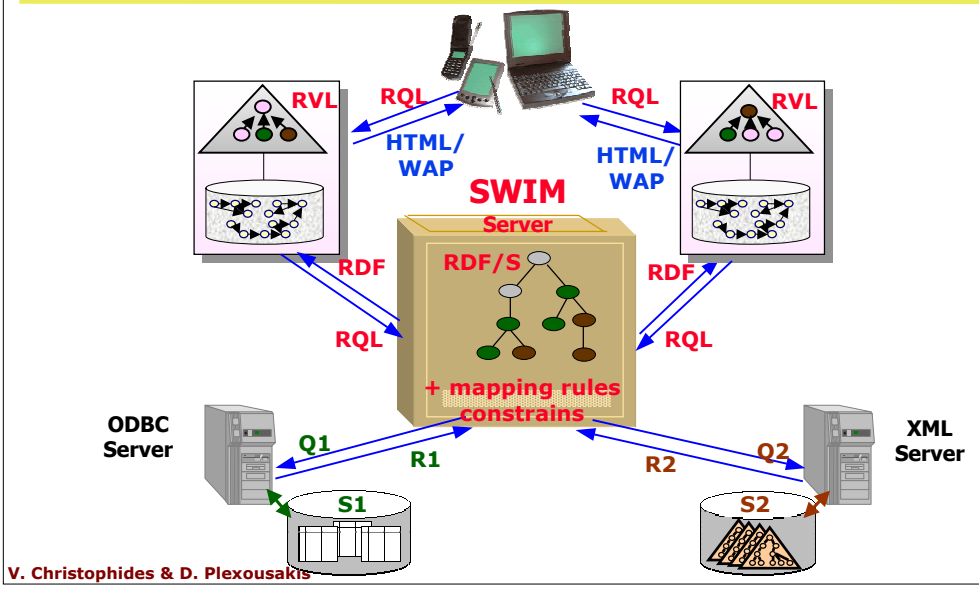
- This can be easily translated into, eg., XQuery

Flexibility

- Same framework can be used for publishing relational data directly as RDF.
- Same framework can be used for composing RQL with RVL views.
- Same framework can be used for heterogeneous integration (mediation).
- Minimization (eliminating redundancies) is essential.
- Many desirable minimizations only hold under constraints.
- For minimization under constraints, use the Chase&Backchase algorithm:

  A. Deutsch, L. Popa, V. Tannen, "… Constraints and Optimization...", in VLDB'03

Let's go SWIM-ming
(Semantic Web Integration Middleware)



Acknowledgements to our Students

- Sophia Alexaki (Master thesis 1998-2000)
- Nikos Athanasis (Master thesis 2001-2003)
- Grigoris Karvounarakis (Master thesis 1998-2000)
- Ioanna Koffina (Master thesis 2002-)
- Giorgos Kokkinidis (Master thesis 2002-)
- Aimilia Maganaraki (Master thesis 2000-2002)
- Stavros Saxtouris (Master thesis 2003-)
- Lefteris Sidirourgos (Master thesis 2003-)
- Giorgos Serfiotis (Master thesis 2002-)
- Karsten Tolle (Diploma Thesis 1999-2000)
- Sotiris Tourtounis (Master thesis 2001-2002)

Bibliography

1. Viewing the Semantic Web through RVL Lenses, Aimilia Magkanaraki, Val Tannen, Vassilis Christophides, Dimitris Plexousakis. Second International Semantic Web Conference (ISWC'03), Sanibel Island, Forida, USA, 2003.
2. RQL: A Functional Query Language for RDF, G. Karvounarakis, A. Magkanaraki, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl, K. Tolle. Functional Approaches to Computing With Data, P.M.D.Gray, L.Kerschberg, P.J.H.King, A.Poulovassilis (eds.), LNCS Series, Springer-Verlag 2003.
3. On Labeling Schemes for the Semantic Web, V. Christophides, D. Plexousakis, M. Scholl, S. Tourtounis. 12th International World Wide Web Conference (WWW'03), Budapest, Hungary, May 20-24, 2003.
4. Benchmarking RDF Schemas for the Semantic Web, A. Maganaraki, S. Alexaki, V. Christophides, and Dimitris Plexousakis. First International Semantic Web Conference (ISWC'02), Sardinia, Italy, June 9-12, 2002.

Bibliography

5. RQL: A Declarative Query Language for RDF, G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, Michel Scholl. The Eleventh International World Wide Web Conference (WWW'02), Honolulu, Hawaii, USA, May 7-11, 2002.
6. On Storing Voluminous RDF Descriptions: The case of Web Portal Catalogs, S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis. Fourth International Workshop on the the Web and Databases (WebDB'01) - in conjunction with ACM SIGMOD/PODS, Santa Barbara, CA, May 24-25, 2001.
7. The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases, S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, K. Tolle. Second International Workshop on the Semantic Web (SemWeb'01), in conjunction with Tenth International World Wide Web Conference (WWW10), pp. 1-13, Hongkong, May 1, 2001.

Thanks!
Questions?

How to Cope with Heterogeneity in the SW?