Viewing the Semantic Web through RVL Lenses

Aimilia Maganaraki, Val Tannen*,
Vassilis Christophides, Dimitris Plexousakis

University of Crete and Institute of Computer Science - FORTH
Heraklion, Crete, Greece

*University of Pennsylvania, Philadelphia, PA, USA

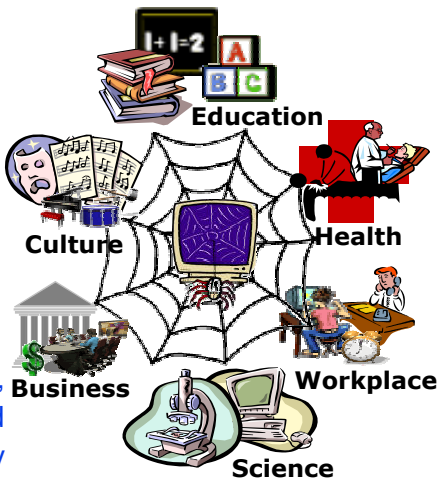
Ack: This work has been supported in part by the IST Project "SeLeNe"

The Semantic Web…

- Community Webs
  - Groups sharing a domain of discourse and a set of information resources (documents, data, services, etc.)
  - Use of the Web as a medium to exchange information and knowledge

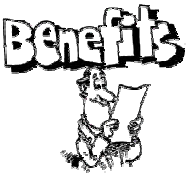
The objective:

- to provide a single point of useful, ubiquitous, comprehensive, and integrated access to community information resources



…and why bother with views on the SW?

- For the good old reasons
  - Data Independence
  - Personalization
  - Data Protection Mechanism - Access Control
  - Integration of Heterogeneous Databases
  - Integrity Constraint Verification
  - Versioning / Schema Evolution
  - Structuring schema-less data
  - Publishing Relational Databases on the Web

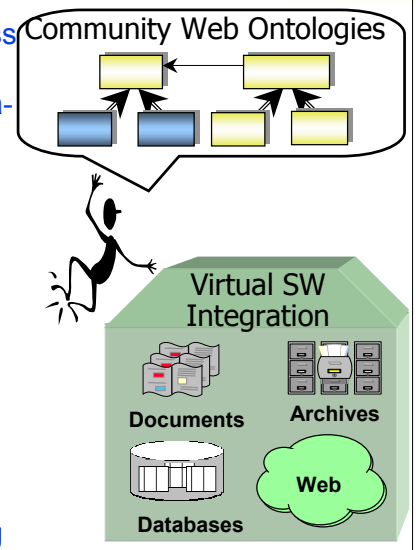


Still, why bother with views on the SW?

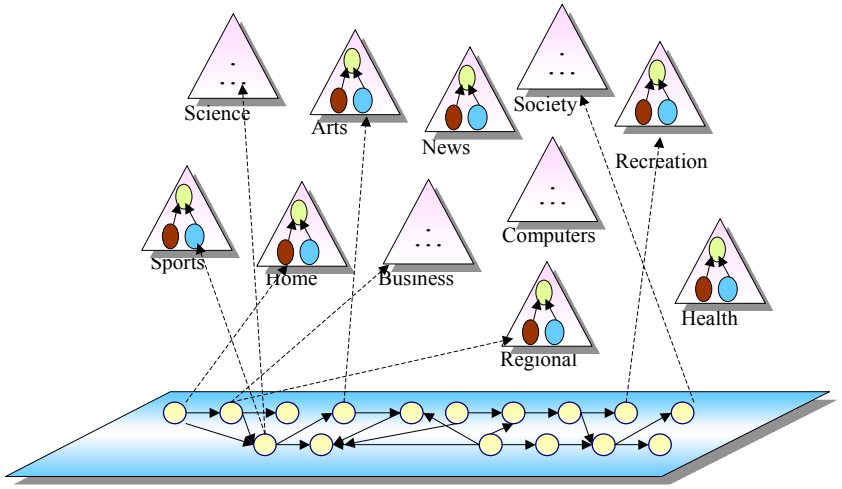
- …and for a bunch of new ones!
  - Web Resource Personalization
    - Subjective ontologies
    - Personalized schema navigation maps
    - Smart bookmarks
  - Mediation of heterogeneous web resources
    - Translation of structures according to different schemas
    - Ontology Integration / Interoperation
  - Ontology management
    - Modularity
    - Versioning
    - Evolution

Our Vision: Combine DB and KR Approaches

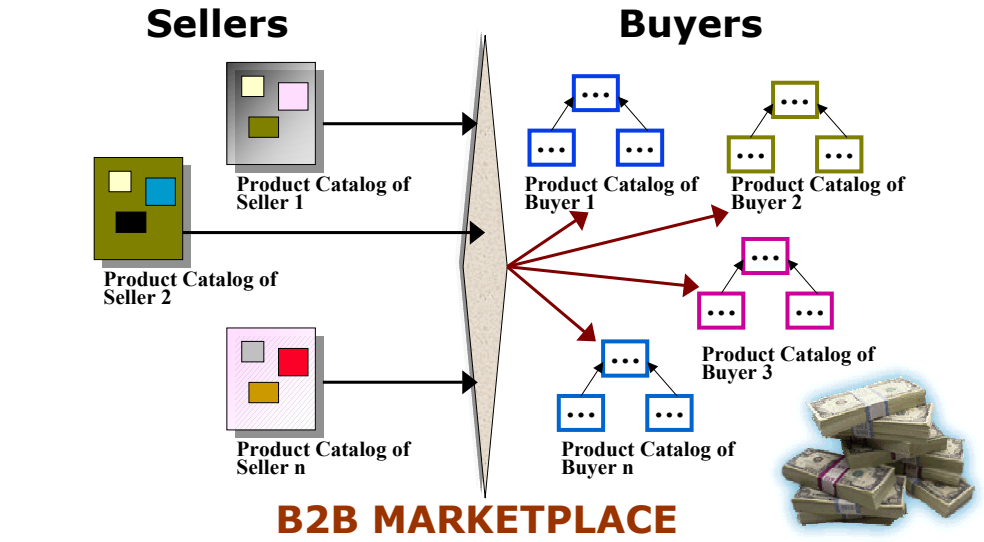
- Provide a comprehensive, high-level access to community resources
  - Ontologies as shared, formal conceptualizations of domains of interest
- Build scalable technologies for managing semantically rich data and metadata
  - Declarative Query / View Languages
  - Efficient Storage for Voluminous Descriptive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



Example Application: Web Personalization



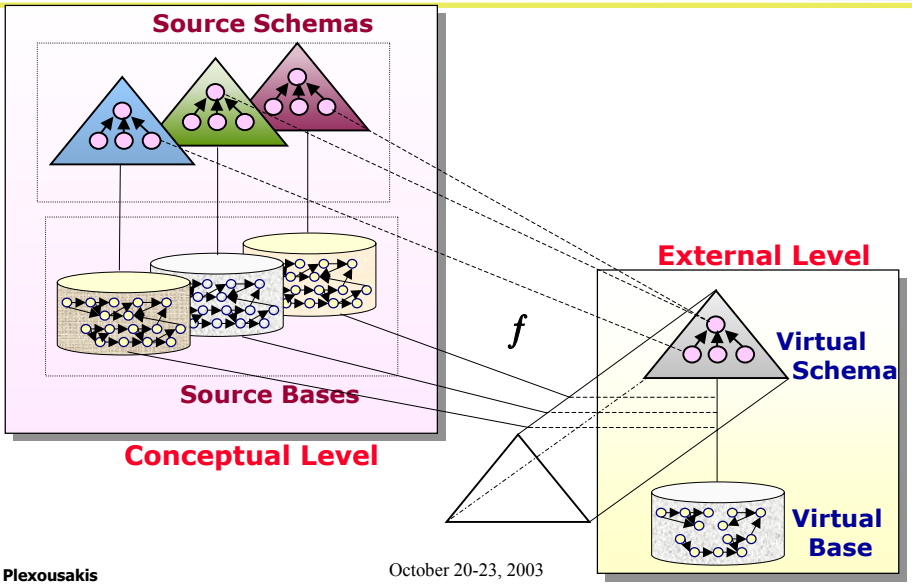
Example Application: Ontology Integration



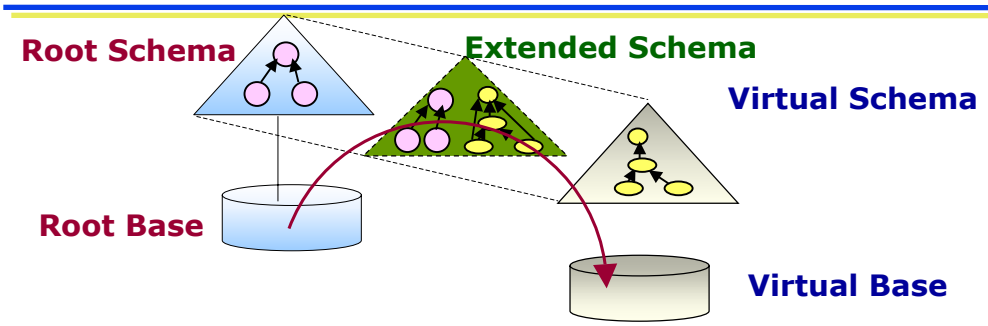
The RDF/S View Language: RVL

- Declarative view definition language for virtual RDF description bases and schemas
  - relies on the RQL typed data model
  - also follows 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 altered by the definition of new ones
  - supports object-preserving and object-generating views
  - provides heavy-duty data restructuring facilities
  - allows users to query and create views using both source and virtual schemas

RVL Approach

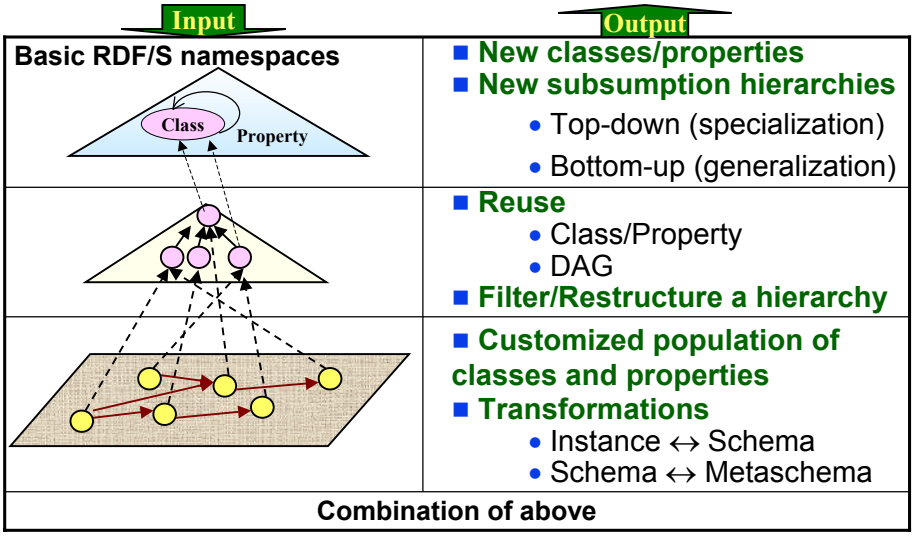


The View Definition Process



- A view defines a set of virtual names and their hierarchies. Everything defined in the view is identified by the view namespace. Everything reused from a root schema is identified by the root schema namespace.
  - Extended schema: the root schema is extended with new constructs
  - Virtual schema: a projection over the extended schema for selecting the constructs of the view

RVL Functionality



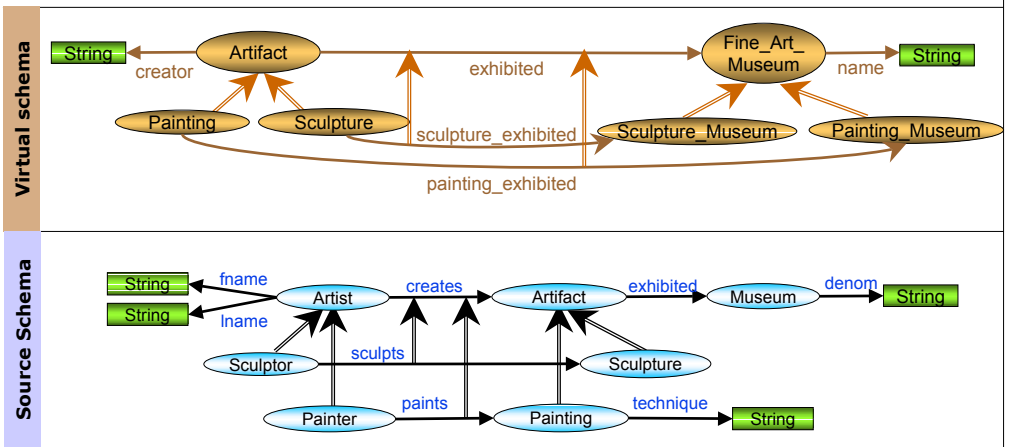
The RVL Syntax

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

RVL Operators

- RVL integrates in a uniform way the functionality needed, whilst taking into account the peculiarities of the RDF/S data model
- 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#

RVL Design Issues

- What is a good specification of a view language for the RDF/S data model?

❶ How are the virtual schema (meta-) classes and properties of a view related to the source description schema(s)?

❷ How are the virtual base resources and property values of a view related to source description base(s)?

❸ What is the expressiveness of the input/output transformations supported by the view specification language?

❹ How can the output of view specifications be used in queries and other views?

RVL Design Choices

❶ Logical Data Independence: the view specifications should be independent from those of the source schemas and bases, while the semantics of existing virtual schemas should not be altered by the definition of new ones

- the scope of virtual (meta-) class and property definitions is determined by the namespace of the view
- virtual subsumption hierarchies instead of global hierarchies

❷ View Instantiation Capabilities: population of virtual (meta-) classes and properties

- object-preserving views vs object-generating views

RVL Design Choices

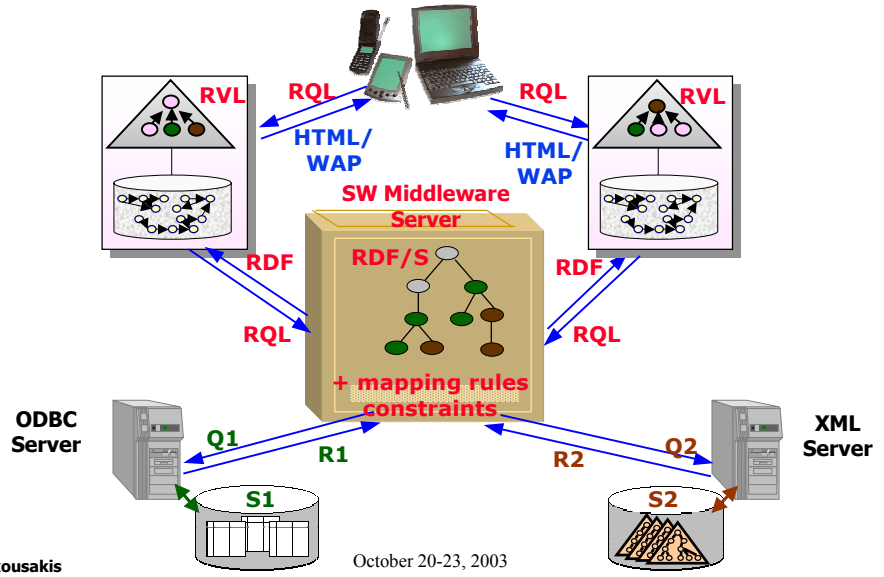
❸ Transformation Expressiveness: provide the ability to both create and reconcile different conceptual representations
- heavy-duty data restructuring facilities enabling users to change the abstraction level in which a particular view construct is defined

❹ Closure of View Language: ability to query and create views using both source and virtual schemas
- the namespace of a view can be used to formulate RQL queries and define views

RVL vs other View Languages

- ODMG-compliant view definition languages: O_2Views, MultiView, Chimera, K2
  - Differences in data models and underlying design choices
  - RVL is capable of creating virtual classes and properties using RQL queries on (meta-) schema and data information
- RDF view definition languages:
  - KAON Views: violates the logical data independence of views (one global hierarchy), while restructuring constructs for subsumption hierarchies are not supported
  - Triple Views: relies on F-Logic rules to define only virtual description bases
  - SeRQL: proposes a variation of RQL in order to produce resource description graphs
  - RVL is the only full-fledged RDF/S view definition language

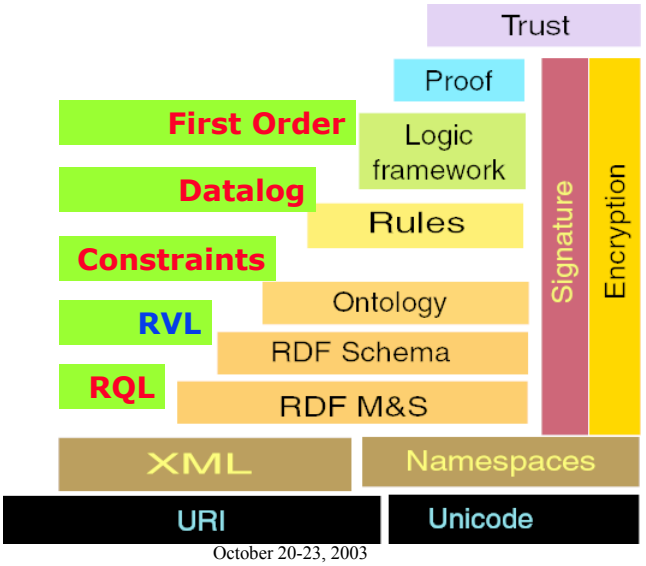
Advanced Semantic Web Infrastructure



Advanced Semantic Web Services

- Semantic Integration of Heterogeneous Resources
  - Consistency Checking of Mappings
- Semantic Query Optimization
  - Minimization of RQL Queries
- Semantic Query Mediation
  - Reformulation of RQL to SQL/XQuery
- Peer-to-Peer Personalization
  - Unconstrained RVL/RQL Composition

The SW Layer Cake



Conclusions

- Re-use of existing data/schemas/ontologies is crucial in describing the semantics of various information sources in the Semantic Web
- Defining views is a way of reusing the same information set by different users
- RVL is a declarative, functional view definition language integrating into a uniform way all the functionality needed with the minimal set of operators.
  - It takes advantage of the underlying data model, the underlying type system and the expressiveness of the RQL query language to constitute a simple, yet very expressive view definition language.