Generating On the Fly Queries for the Semantic Web: The ICS-FORTH Graphical RQL Interface (GRQL)

Nikos Athanasis
Vassilis Christophides
Dimitris Kotzinos

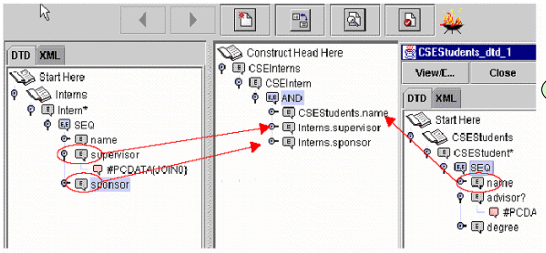
Computer Science Department, University of Crete
and Institute for Computer Science - FORTH
Heraklion, Crete, Greece

Agenda

- Motivation
- Navigation scenarios in GRQL
- The GRQL Translation Algorithm of Browsing Actions
- GRQL Prototype and overall architecture
- summary & future work



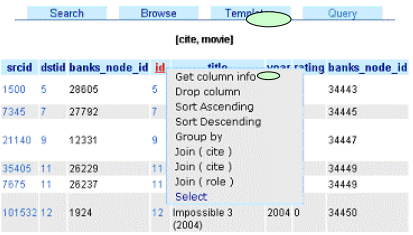
Visual Abstractions for Accessing Data Collections



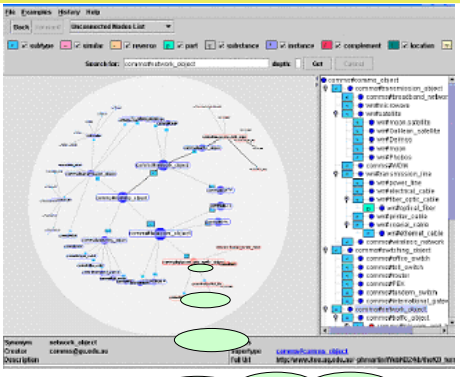
BANKS:
Navigation and querying in RDBMS

Banks

BBQ:
Browsing XML data and DTD's



Ontology Browsers

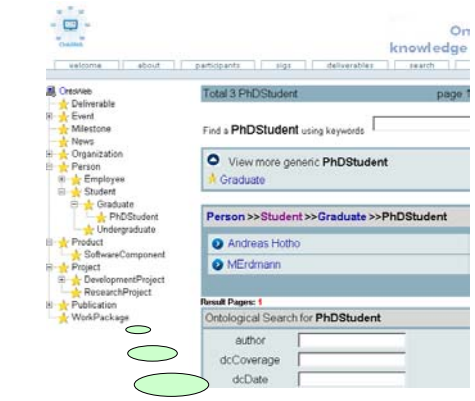


Ontopia Omnigator



RDF Distillery

Semantic Web Portals



OntoWeb Portal

ODESeW Portal

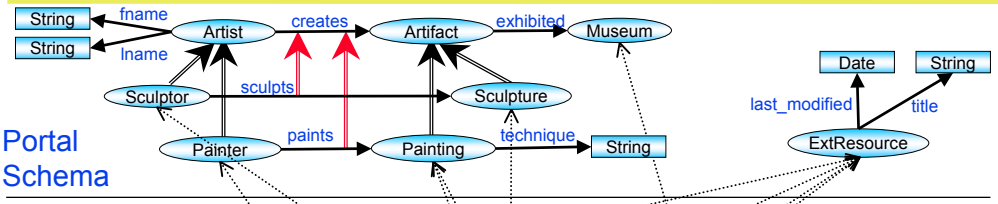


Our Solution: GRQL

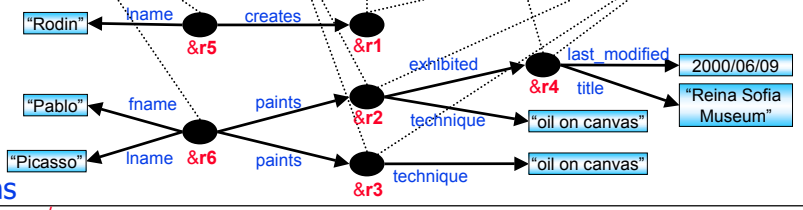
- Integrates smoothly conceptual navigation and querying for end-user Semantic Web applications
- Relies on the full power of the RDF/S data model for constructing, in a transparent to the end-user way, queries expressed in a declarative language such as RQL
- Creates dynamic views based on the incremental computation of adequate RQL queries capturing the cumulative effect of browsing actions performed by a user during its navigation session
- Combines
  - the simplicity of SW browsers with
  - the expressiveness of DB query generators

Motivating Example: Cultural Portal

Portal Schema



Portal Resource Descriptions



Web Resources

r2: www.museum.es/guernica.jpg



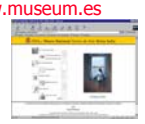
r1:www.rodin.fr/thinker.gif



r3:www.museum.es/woman.qti



r4:www.museum.es



GRQL Translation of Schema Browsing Actions

- Translation of the user navigation navigation into minimal schema triplets, used to:
  - generate the corresponding RQL query
  - visualize the traversed navigation paths

- Algorithm based on triplets:

<subject, predicate, object>

Capturing: < (class), (property), (class or literal type) >

Example: <Artist, creates, Artifact>

- For each browsing action a new triplet will be created, an existing one could be modified or everything will be left unchanged (in case of redundancy)

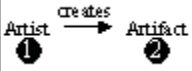
A Complete GRQL Navigation Scenario

- Entry point of the navigation session
  - user chooses a class (or property) provided from the tree-shaped graphical representation of the class (or property) subsumption relationships

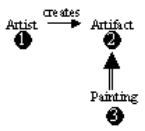
Navigation	triplets	Path	RQL query
	<Artist,empty,empty>	Artist	select X1 from Artist{X1}

A Complete GRQL Navigation Scenario (Cont.)

Browsing class associations:

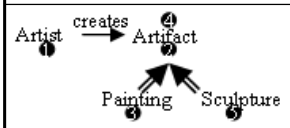
Navigation	triplets	Path	RQL query
	<Artist, creates, Artifact>	Artist→creates→Artifact	select X2 from {X1}creates{X2}

Browsing class subsumption relationships:

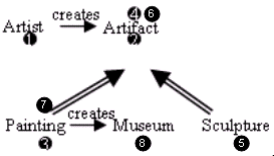
Navigation	triplets	Path	RQL query
	<Artist, creates, Painting>	Artist→creates→Painting	select X2 from {X1}creates{X2;Painting}

A Complete GRQL Navigation Scenario (Cont.)

Navigating to sibling subclasses in a path:

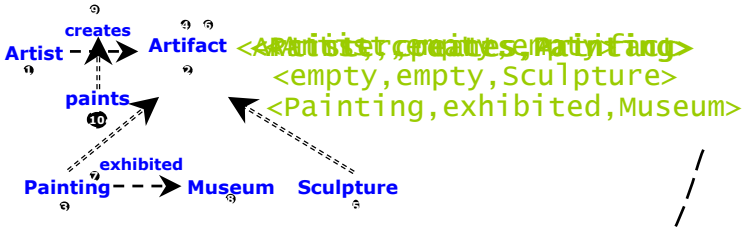
Navigation	triplets	RQL query
	<Artist,creates,Painting> <empty,empty,Sculpture>	select X2 from {X1}creates{X2;Painting}, Sculpture{X2}

Navigating to associated classes belonging to different paths:

Navigation	triplets	RQL query
	<Artist,creates,Painting> <Painting,exhibited,Museum> <empty,empty,Sculpture>	select X3 from {X1}creates{X2;Painting}.{;Painting}exhibited{X3},Sculpture{X2}

A Complete GRQL Navigation Scenario (Cont.)

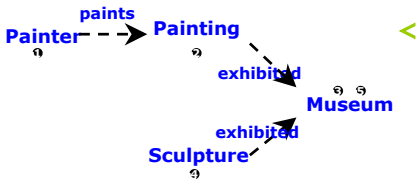
Navigating to specialized properties:



select X1,X2 from
{X1}paints{X2}.
{;Painting}exhibited{X3},
Sculpture{X2}

A Complete GRQL Navigation Scenario (Cont.)

Navigating to associated classes from different paths:

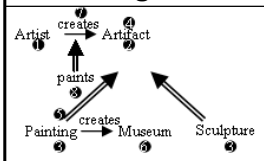


<Painter,paints,Painting>
<Painting,exhibited,Museum>
<empty,empty,Sculpture>

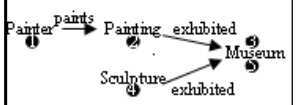
select X3 from
{X1}paints{X2}.
{;Painting}exhibited{X3},
Sculpture{X2}

A Complete GRQL Navigation Scenario (Cont.)

Navigating to specialized properties:

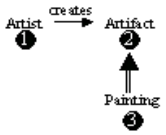
Navigation	triplets	RQL query
	<Artist,paints,Painting> <Painting,exhibited,Museum> <empty,empty,Sculpture>	select X1,X2 from {X1}paints{X2}. {;Painting}exhibited{X3}, Sculpture{X2}

Navigating to associated classes from different paths:

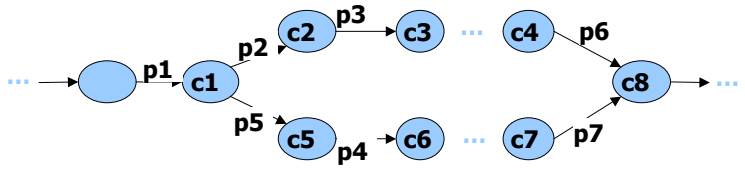
Navigation	triplets	RQL query
	<Artist,creates,Painting> <Painting,exhibited,Museum> <empty,empty,Sculpture>	select X3 from {X1}paints{X2}. {;Painting}exhibited{X3}, Sculpture{X2}

Enriching Path Expressions with Filtering Conditions

- users can formulate filtering conditions on the attributes of the class visited at each navigation step

Navigation	triplets	RQL query
	<Artist,paints,Painting> <last_name, like, "*P*>	select X1 from {X1}:creates{X2;Painting}, {X1}last_name{X4} where X4 like "*P*"

The GRQL Translation Algorithm



- Algorithm captures successfully a sequence of complex browsing actions performed by a user at the schema level
  - Forward and backward navigation
  - Branching paths with common destination or origin
- Multiple navigation paths are translated into a conjunctive RQL query
  - Conjunctive predicates on node/edge labels
- Algorithm computes at each browsing step the minimal RQL query corresponding to the conjunction of the current and the previous navigation paths

GRQL Minimal Query Translation: Example

From the property *creates*, refining its range to *Painter* and finally to the subproperty *paints*, GRQL generates the following three RQL queries:

Q3 ⊆	**Q2** ⊆	**Q1**
`select * from {X1}paints{X2}`	`select * from {X1;Painter}creates{X2}`	`select * from {X1}creates{X2}`

The result of **Q2** is contained in the result of **Q1**, the conjunction of the queries (or more precisely of their path expressions) generated at the current (2) and previous steps (1) results to a redundant RQL query, which can be simplified as follows:

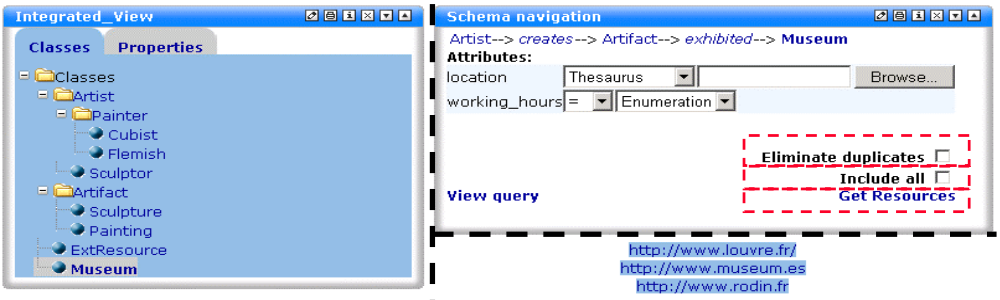
Q2 ≡	**Q2'**
`select * from{X1;Painter}creates{X2}`	`select * from{X1;Painter} creates{X2},{X1}creates{X2}`

In the same way, the minimal RQL query produced at the navigation step 3 is **Q3**

The GRQL Interface

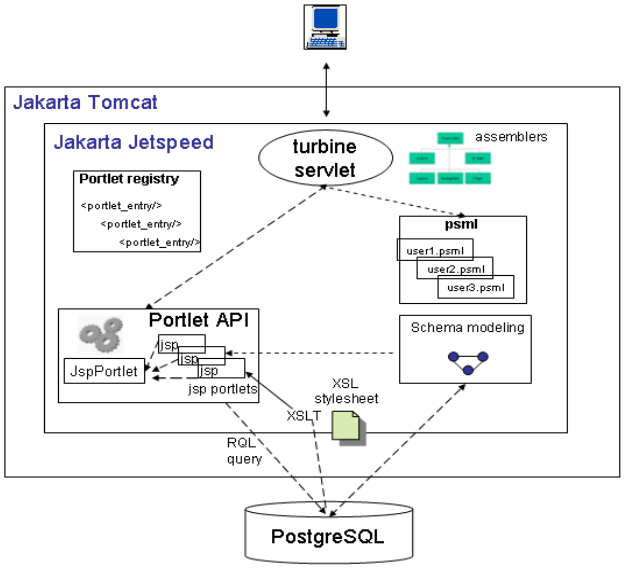
Tree-shaped display of the subsumption hierarchies of both the classes and properties defined in an RDF/S schema

Explore progressively the individual RDF/S class and property definitions



Visualization of the constructed query/view results

The GRQL High-Level Architecture



GRQL Summary

- Visual metaphor for assisting non-expert users to browse/filter RDF/S resource descriptions and schemas in a personalized way
  - No need to learn the syntax and semantics of RDF/S QLs
  - Capture navigational context for accessing resource descriptions
- Translates into a minimal set of schema triples a sequence of browsing actions during an entire navigation session performed by a user
  - Exploit semantics for reasoning over queries and not data
  - Implemented for RQL but can be deployed with other declarative RDF/S QLs
- Reduced development and maintenance costs for GRQL applications
  - No need to program new applications from scratch with low-level APIs
  - Presentation customization using a variety of menu-driven options
- Visit our online demo: http://139.91.183.30:8999/RQLdemo/

Future Work

- Extend the GRQL translation algorithm to:
  - support disjunctive queries,
  - support updates of RDF/S descriptions and schemas
- Thus equipping users with a uniform graphical interface for
  - conceptual navigation/filtering as well as
  - updating resource description bases and schemas

  users are released from learning the RDF/S syntax and semantics!!!
- Store generated user views in intelligent bookmarks
  - Implement a semantics-based navigation history mechanism

Thank you



Back-up Slides

GRQL Translation Algorithm (pseudo code)

```
if (user navigates through subclasses)
 {
   if (the navigation path is empty)
   {//create a new triplet where its domain is the selected class
     create triplet (selClass, empty, empty);
   }
   else
    for each triplet (c1,p,c2) in list of navigation paths
     {//if the selected class is a subclass of triplet's domain
       if (selClass.isSubClassOf(c1) )
       {//modify the triplet's domain to the selected class
         modify triplet (selClass, p, c2);
       }
       //if the selected class is a subclass of triplet's range
       else if (selClass.isSubClassOf(c2))
            {//modify the triplet's range to the selected class
              modify triplet (c1, p, selClass);
            }
            //if the selected class is a sibling of triplet's domain/range
            else if (selClass.isSubClassOf(p.getDomain()) or
                     selClass.isSubClassOf(p.getRange()))
                 {//create a new branching triplet for path nodes
                create triplet (selClass, empty, empty); }
     }
```

GRQL Translation Algorithm (pseudo code)

```
if (user navigates through subproperties)
{//get the domain of the selected property
   domain = selProp.getDomain();
   //get the range of the selected property
   range =  selProp.getRange();
 for each triplet (c1,p,c2) in list of navigation paths
    { //if the selected property selProp is a subproperty of the current
   triplet's property
       if (selProp.isSubPropertyOf(p)){
           modify triplet (domain, selProp, range);
      if !((domain==c1) or (domain.isSubClassOf(c1))
   if (candidate triplet is not in list of navigation paths)
                  create triplet (c1, empty, empty);
           if !((range==c2) or (range.isSubClassOf(c2))
        create triplet (empty, empty, c2);
       }
       else if (selProp != p)
            {//adjust the domain of the immediate following triples in path
              if (range.isSubClassOf(c1))
        modify triplet (range, p, c2);
              //if current triplet differs from the selected property
              else if ((c1 != domain) and (c2 != range))
              { //create a new branching triplet for path edges
        create triplet (domain, selProp, range);
              }
```