COMP718: Ontologies and Knowledge Bases

Lecture 9: Ontology/Conceptual Model based Data Access

Maria Keet

email: keet@ukzn.ac.za

home: http://www.meteck.org

School of Mathematics, Statistics, and Computer Science
University of KwaZulu-Natal, South Africa

10 April 2012

Outline

An ontology with a very large ABox

- Thus far, we've seen mostly small ontologies with lots of features, but with little data, and mostly theory
- This and next lecture:
  - Scale up to realistic size knowledge base handling large amounts of data
  - Set up an ontology-driven information system
- To realise this, we need
  - A language of relatively low computational complexity
  - A way to store large amounts of data
  - Some mechanism to link up the previous two ingredients
  - Query (and reason over) the combination of the previous three
- Use the "Ontology-Based Data Access" approach

An ontology with a very large ABox

- Thus far, we've seen mostly small ontologies with lots of features, but with little data, and mostly theory
- This and next lecture:
  - Scale up to realistic size knowledge base handling large amounts of data
  - Set up an ontology-driven information system
- To realise this, we need
  - A language of relatively low computational complexity
  - A way to store large amounts of data
  - Some mechanism to link up the previous two ingredients
  - Query (and reason over) the combination of the previous three
- Use the "Ontology-Based Data Access" approach

An ontology with a very large ABox

- Thus far, we've seen mostly small ontologies with lots of features, but with little data, and mostly theory
- This and next lecture:
  - Scale up to realistic size knowledge base handling large amounts of data
  - Set up an ontology-driven information system
- To realise this, we need
  - A language of relatively low computational complexity
  - A way to store large amounts of data
  - Some mechanism to link up the previous two ingredients
  - Query (and reason over) the combination of the previous three
- Use the "Ontology-Based Data Access" approach

Access to relational databases

- "Sysadmin interface" for developers and users:
  - SQL or any of its close variants (e.g., StruQL)
  - Need to know *how* the data is stored in the database
  - Writing large queries is still time-consuming even for experts
  - One-off queries or some manual query management for recurring queries
- "End-user interface"
  - Canned queries and pre-computed queries
  - Inflexible for data analysis
  - Burden on sysadmin for application layer updates to meet whims of the user
- Database integration

Access to relational databases

- "Sysadmin interface" for developers and users:
  - SQL or any of its close variants (e.g., StruQL)
  - Need to know *how* the data is stored in the database
  - Writing large queries is still time-consuming even for experts
  - One-off queries or some manual query management for recurring queries
- "End-user interface"
  - Canned queries and pre-computed queries
  - Inflexible for data analysis
  - Burden on sysadmin for application layer updates to meet whims of the user
- Database integration

Access to relational databases

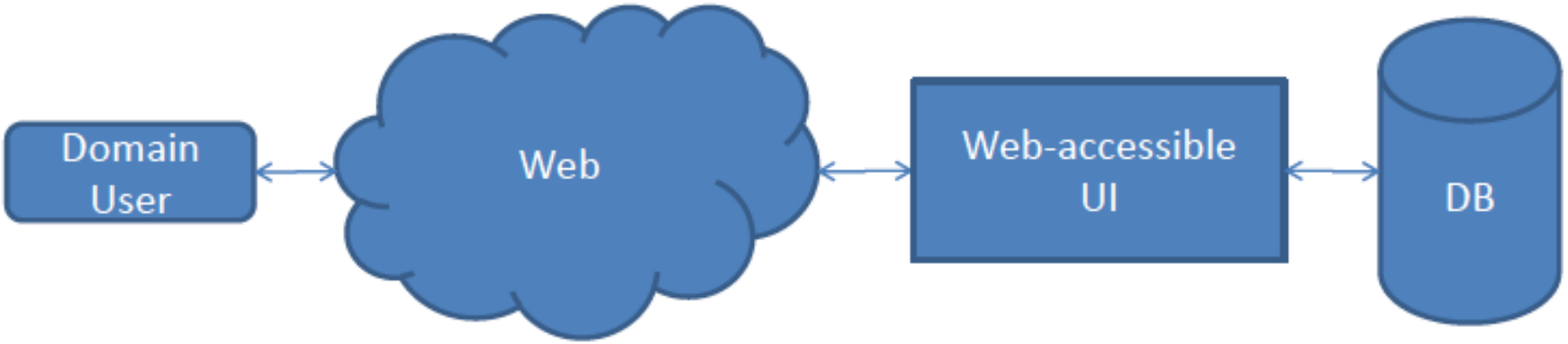
- "Sysadmin interface" for developers and users:
  - SQL or any of its close variants (e.g., StruQL)
  - Need to know *how* the data is stored in the database
  - Writing large queries is still time-consuming even for experts
  - One-off queries or some manual query management for recurring queries
- "End-user interface"
  - Canned queries and pre-computed queries
  - Inflexible for data analysis
  - Burden on sysadmin for application layer updates to meet whims of the user
- Database integration

Access to relational databases

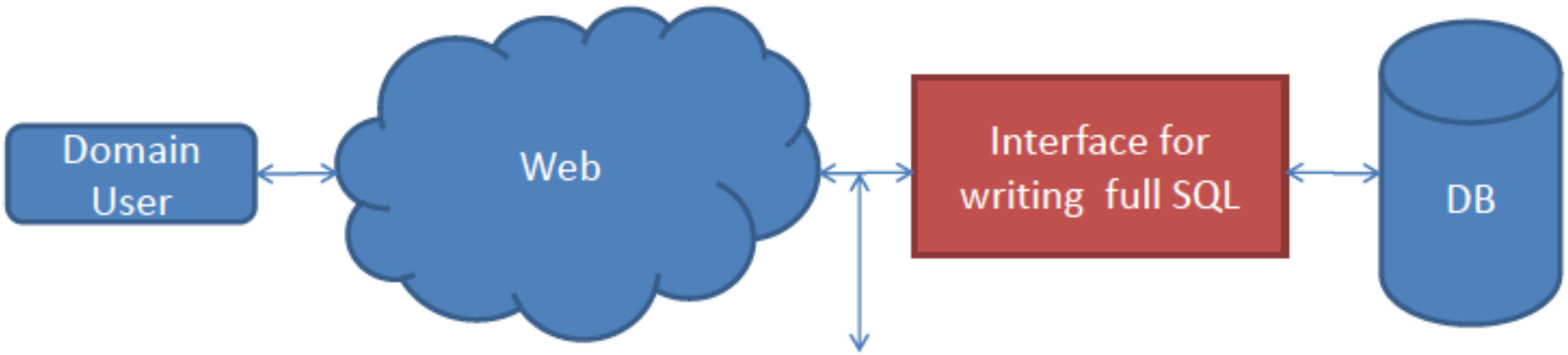
- "Sysadmin interface" for developers and users:
  - SQL or any of its close variants (e.g., StruQL)
  - Need to know *how* the data is stored in the database
  - Writing large queries is still time-consuming even for experts
  - One-off queries or some manual query management for recurring queries
- "End-user interface"
  - Canned queries and pre-computed queries
  - Inflexible for data analysis
  - Burden on sysadmin for application layer updates to meet whims of the user
- Database integration

Outline

Web-accessible databases



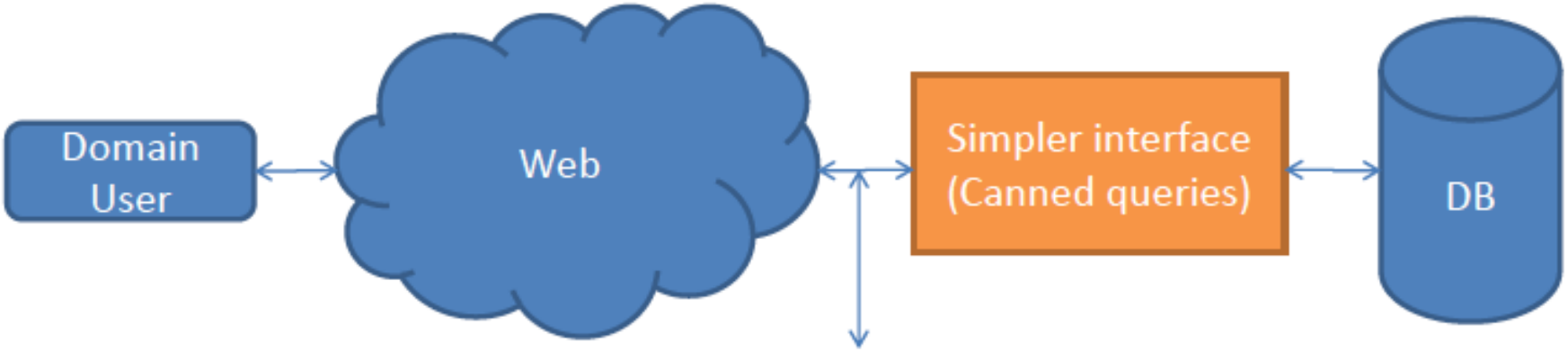
Web-accessible databases



- Fail to meet the usability requirement

+ Maximal query expressiveness

Web-accessible databases



+Meets the usability requirement

- Not all queries can be expressed

Case study and problem: the Horizontal Gene Transfer DB

- Lots of data made available on the Web by the Life Science field
- HGT-DB is a web-accessible genomics database about prokaryotic organisms
- Web interface with pre-computed queries or restricted SQL queries
- Contains 477 organisms and 1,445,840 genes
- 4GB genomics database
- Tables with 16-46 columns

Case study and problem

- Sample Information Request:
  Retrieve all genes of the organisms Neisseria *for which horizontal gene transfer is predicted or have a GC3 value* > 80
  - Simple HTML interface for posing canned queries and to retrieve text-files of pre-computed queries
  - Substantial limitations on expressiveness of queries: **Domain users cannot extract all the information contained in the database!**
- Problem: Users (geneticists) need to know *what* is in the database, *how* the data is stored, and need to know SQL or use the pre-computed queries, which is what limits their data analysis

Case study and problem

- Sample Information Request:
  Retrieve all genes of the organisms Neisseria *for which horizontal gene transfer is predicted or have a GC3 value* > 80
  - Simple HTML interface for posing canned queries and to retrieve text-files of pre-computed queries
  - Substantial limitations on expressiveness of queries: **Domain users cannot extract all the information contained in the database!**
- Problem: Users (geneticists) need to know *what* is in the database, *how* the data is stored, and need to know SQL or use the pre-computed queries, which is what limits their data analysis

Case study and problem

- Sample Information Request:
  Retrieve all genes of the organisms Neisseria *for which horizontal gene transfer is predicted or have a GC3 value* > 80
  - Simple HTML interface for posing canned queries and to retrieve text-files of pre-computed queries
  - Substantial limitations on expressiveness of queries: **Domain users cannot extract all the information contained in the database!**
- Problem: Users (geneticists) need to know *what* is in the database, *how* the data is stored, and need to know SQL or use the pre-computed queries, which is what limits their data analysis

A solution

- Solution: add a semantic layer to the database, let the users construct queries graphically, and generate the SQL/SPARQL/EQL query automatically from the graphics
- *Constraints*: any solution needs to be scalable, usable, and web-based
- **Realisation**: graphical, web-based 'Ontology'-Based Data Access (COnceptual MOdel-based Data Access – COMODA)

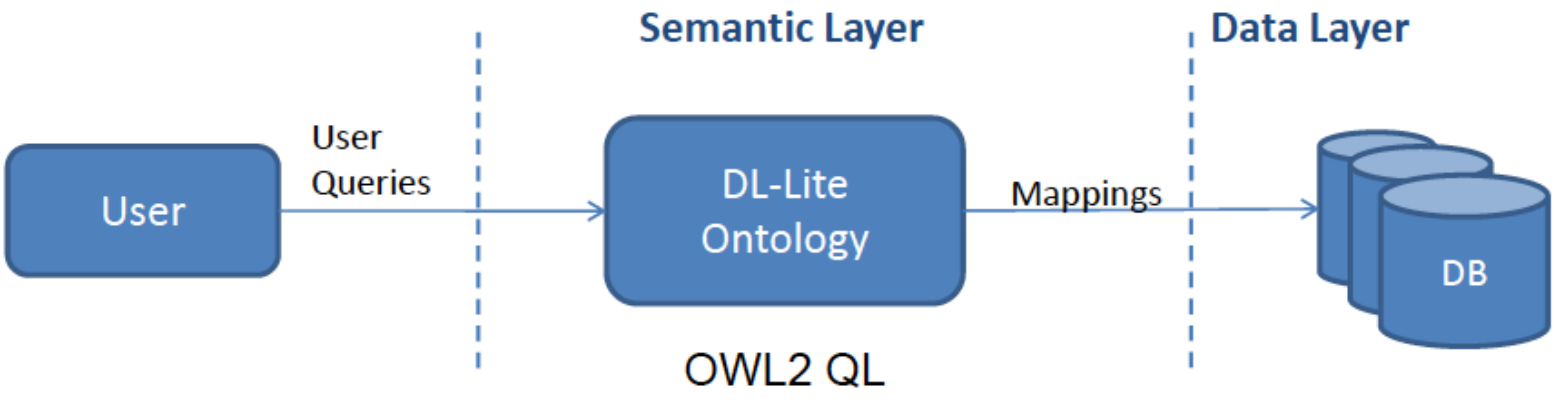
A solution

- Solution: add a semantic layer to the database, let the users construct queries graphically, and generate the SQL/SPARQL/EQL query automatically from the graphics
- *Constraints*: any solution needs to be scalable, usable, and web-based
- **Realisation**: graphical, web-based 'Ontology'-Based Data Access (COnceptual MOdel-based Data Access – COMODA)

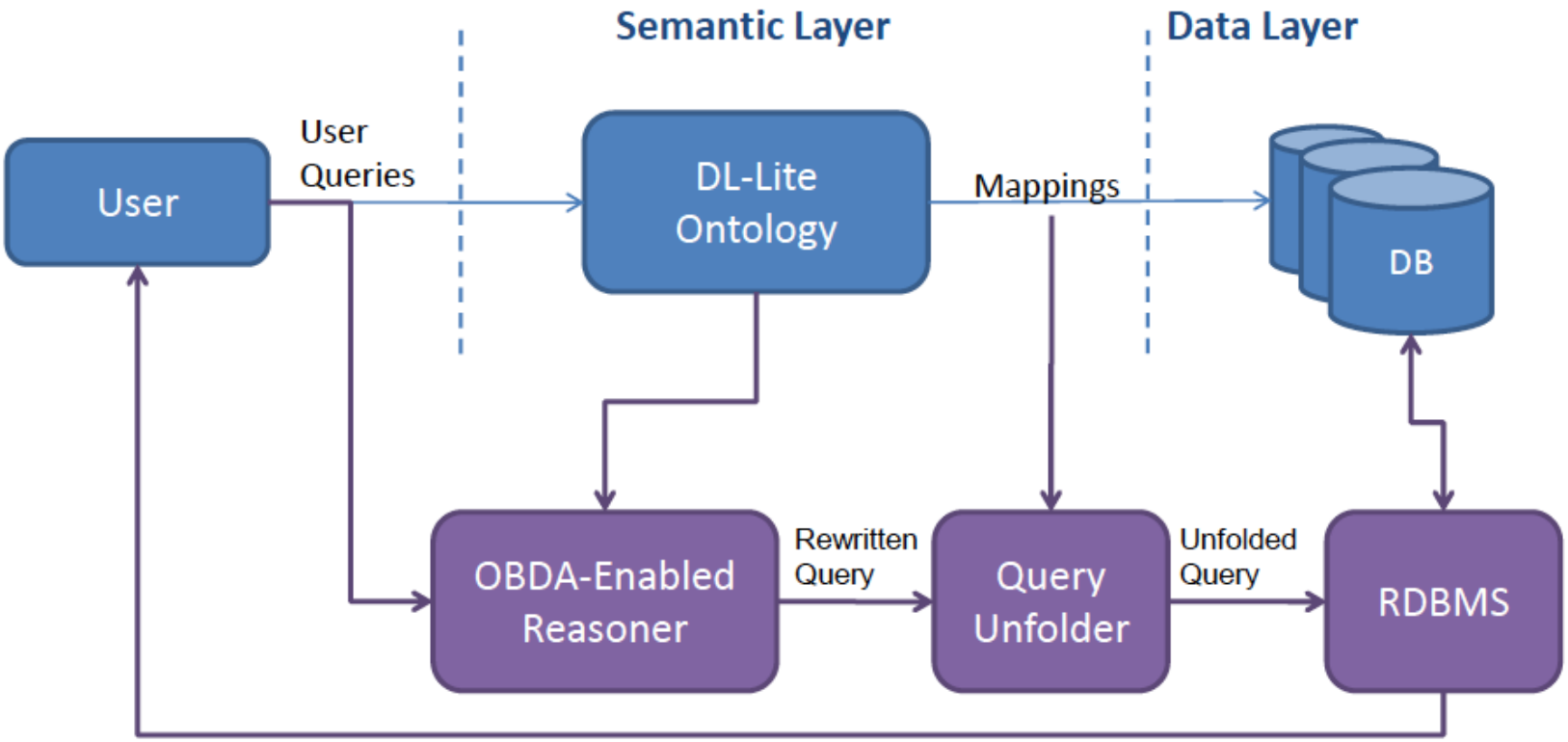
A solution

- Solution: add a semantic layer to the database, let the users construct queries graphically, and generate the SQL/SPARQL/EQL query automatically from the graphics
- *Constraints*: any solution needs to be scalable, usable, and web-based
- **Realisation**: graphical, web-based 'Ontology'-Based Data Access (COnceptual MOdel-based Data Access – COMODA)

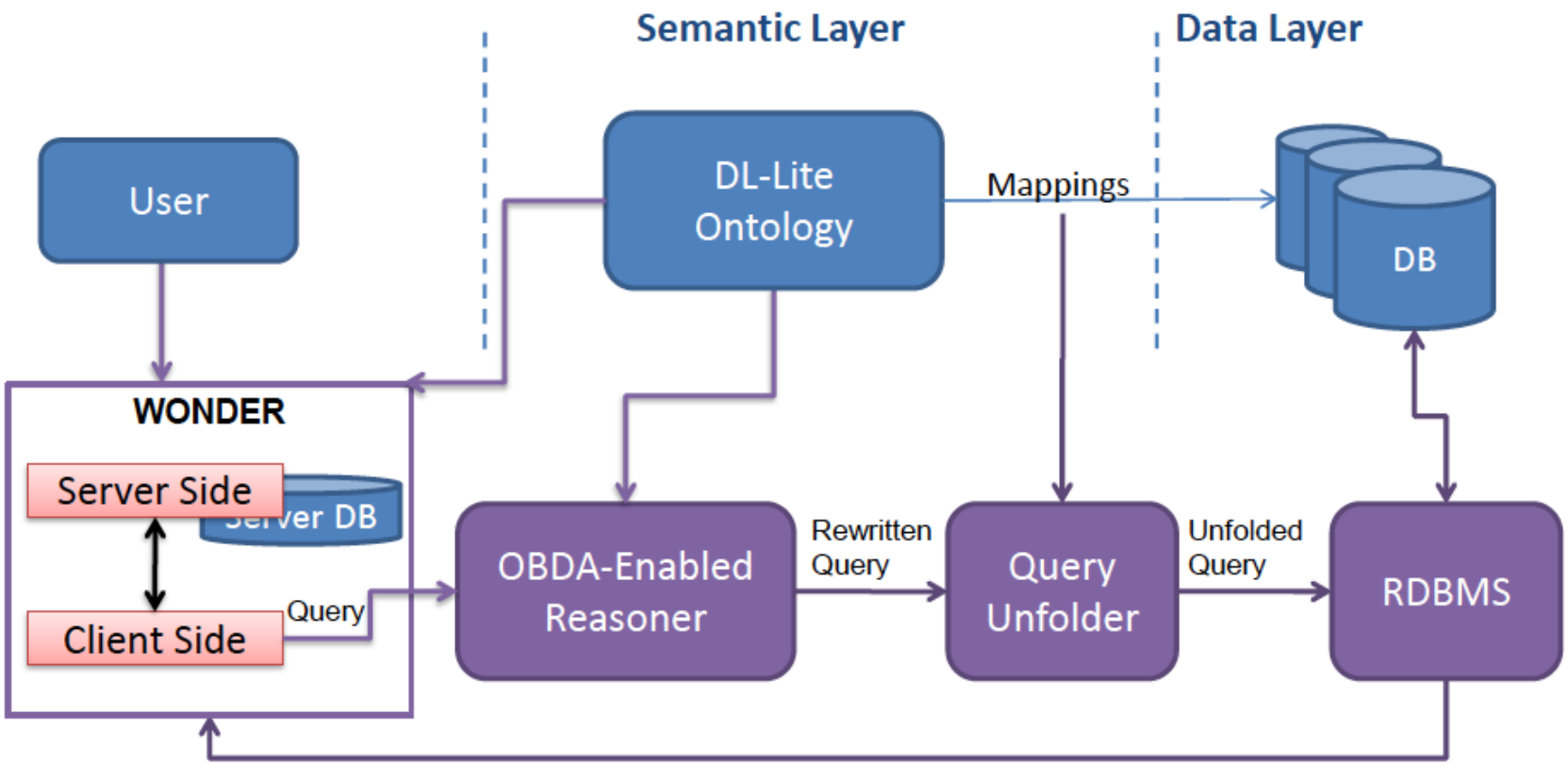
Approach


Semantic Layer
Data Layer
User
User Queries
DL-Lite Ontology
Mappings
DB
OWL2 QL

Architecture



Architecture



Web-ONtology baseD Extraction of Relational data (WONDER)

Components of this particular system

- Developed in collaboration at "La Sapienza" University in Rome and Free University of Bozen-Bolzano
- Formal languages: *DL-Lite* family, OWL 2 QL
  - This context: ontology ≡ Description Logics Knowledge Base
- OBDA-enabled reasoner: QuOnto
- RDBMS: Oracle, PostgreSQL, DB2, ...
- Developer interface: OWL ontology development tool, OBDA plugin to manage the mappings and data access
- End-user interface: OBDA plugin for Protégé for SPARQL queries and results, and the WONDER system for graphical querying

Why graphical querying?

- Querying in the basic OBDAs uses (unions of) conjunctive queries:
  - A conjunctive query is the formal counterpart of an SQL (or relational algebra) select-project-join (SPJ) query: $q(\vec{x}) \leftarrow \exists \vec{y}.conj(\vec{x}, \vec{y})$
  - $conj(\vec{x}, \vec{y})$ is a conjunction of atoms and equalities over the free variables $\vec{x}$ and the existentially quantified variables $\vec{y}$
  - The variables in $\vec{x}$ are the *distinguished variables* (i.e.: appear in the head) and in $\vec{y}$ are the *non-distinguished variables*
  - A union of CQs (UCQ) is a disjunction of CQs, corresponding to a union of SPJ queries
- (U)CQs (in SPARQL notation) embedded into ordinary SQL code is more compact, but still user-unfriendly

Why graphical querying?

- Querying in the basic OBDAs uses (unions of) conjunctive queries:
  - A conjunctive query is the formal counterpart of an SQL (or relational algebra) select-project-join (SPJ) query: $q(\vec{x}) \leftarrow \exists \vec{y}.conj(\vec{x}, \vec{y})$
  - $conj(\vec{x}, \vec{y})$ is a conjunction of atoms and equalities over the free variables $\vec{x}$ and the existentially quantified variables $\vec{y}$
  - The variables in $\vec{x}$ are the *distinguished variables* (i.e.: appear in the head) and in $\vec{y}$ are the *non-distinguished variables*
  - A union of CQs (UCQ) is a disjunction of CQs, corresponding to a union of SPJ queries
- (U)CQs (in SPARQL notation) embedded into ordinary SQL code is more compact, but still user-unfriendly

Why graphical querying?

- Querying in the basic OBDAs uses (unions of) conjunctive queries:
  - A conjunctive query is the formal counterpart of an SQL (or relational algebra) select-project-join (SPJ) query: $q(\vec{x}) \leftarrow \exists \vec{y}.conj(\vec{x}, \vec{y})$
  - $conj(\vec{x}, \vec{y})$ is a conjunction of atoms and equalities over the free variables $\vec{x}$ and the existentially quantified variables $\vec{y}$
  - The variables in $\vec{x}$ are the *distinguished variables* (i.e.: appear in the head) and in $\vec{y}$ are the *non-distinguished variables*
  - A union of CQs (UCQ) is a disjunction of CQs, corresponding to a union of SPJ queries
- (U)CQs (in SPARQL notation) embedded into ordinary SQL code is more compact, but still user-unfriendly

Sample query

```
SELECT stbl.gene
FROM sparqltable
        (SELECT $gene $orgName $gcVal $predVal
        WHERE {$gene    :GeneHasOrganism $org.
               $org     :OrganismHasOrganismInfo $info.
               $info    :OrganismName $orgName.
               $gene    :GeneHasHGTPredictionGene $pred.
               $pred    :Prediction $predVal.
               $gene    :GeneHasGCstatsGene $gcstats.
               $gcstats :GC3 $gcVal}) stbl
WHERE stbl.orgName LIKE '%Neisseria%' AND
        (stbl.predVal = 'hgt' OR stbl.gcVal > '80')
```

Retrieve all genes of the organisms Neisseria for which horizontal gene transfer is predicted or have a GC3 value > 80

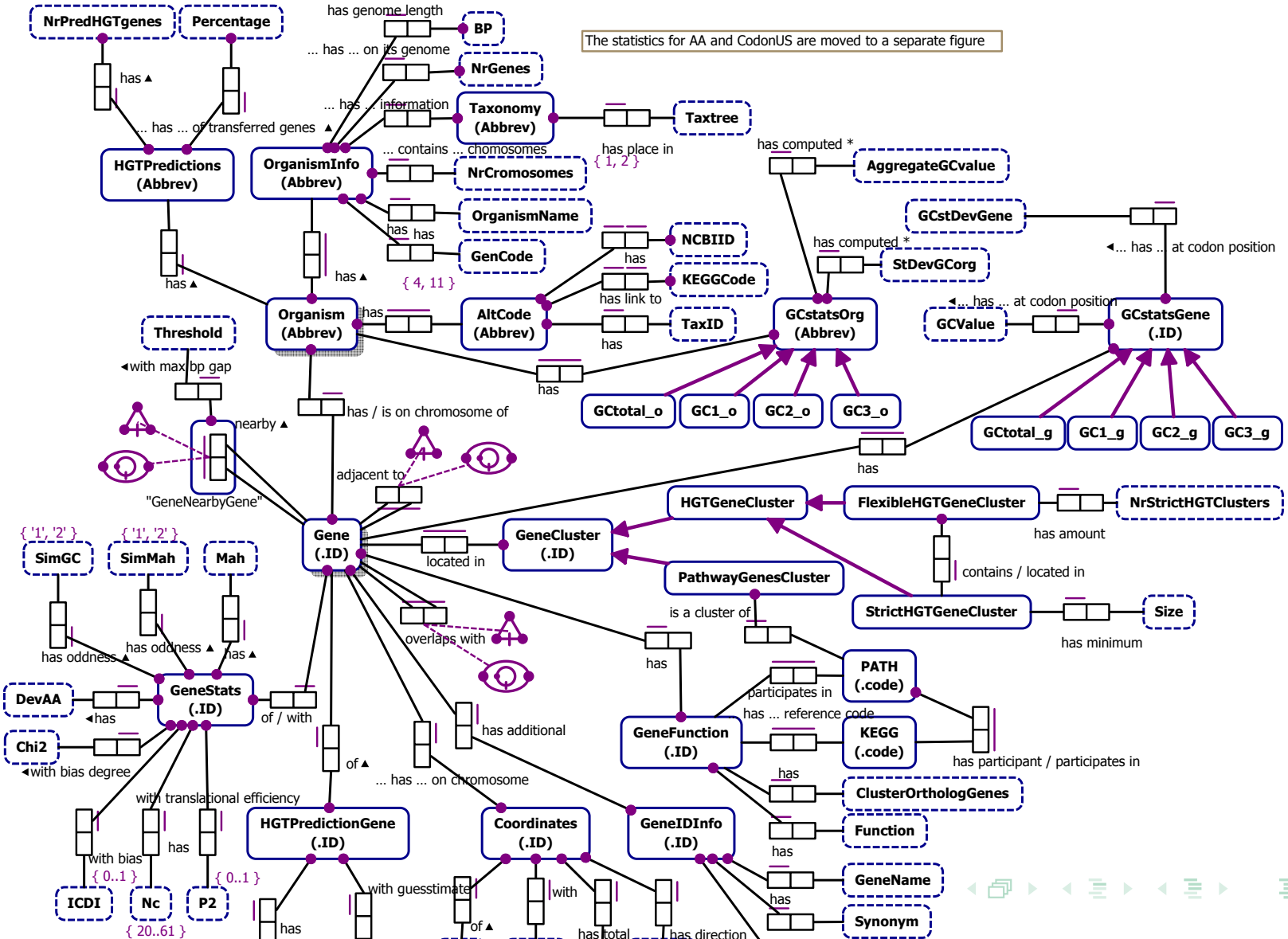
Approach to graphical querying

- Accessing information comprises three activities:
  - Browsing the ontology, to understand the structure of the information;
  - Formulating a query, to express an information request; and
  - Retrieving data that answers the query
- The WONDER Web interface consists of a separate component for each of these activities.

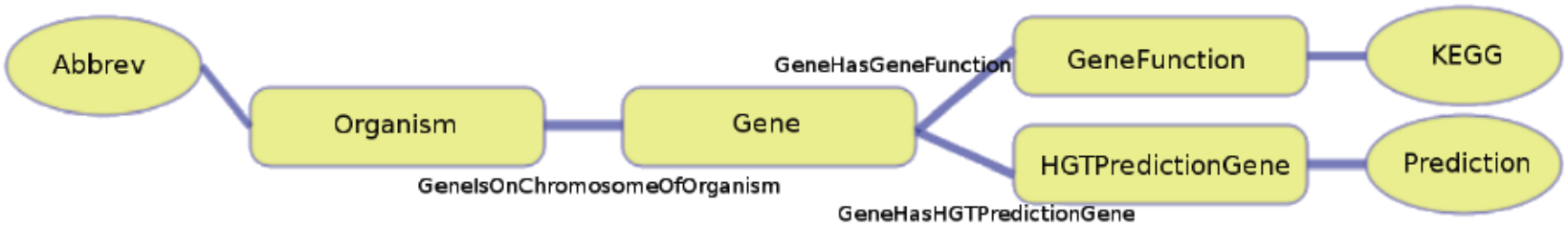
Procedure to realise the solution

- Reverse engineer the database into an ORM conceptual data model with the domain experts, cleaning and refining it
- Transform it into a *DL-lite*$_{\mathcal{A}}$ ontology, and put it in Protégé
- Declare the mappings (matching OWL classes and properties to SQL queries over the database)
- Develop the web-based front-end for browsing, query formulation, and displaying the results, using QuOnto for the automated reasoning at the back-end

Section ORM 2 diagram (in NORMA)



Example: Diagram – *DL-lite*$_{\mathcal{A}}$ correspondence



$\delta(Abbrev) \sqsubseteq Organism$	An abbreviation is for an organism
$\rho(Abbrev) \sqsubseteq \mathtt{xsd:string}$	An abbreviation is of type string
$Organism \sqsubseteq \delta(Abbrev)$	Each organism has an abbreviation
(funct $Abbrev$)	Each individual has a single abbreviation
$\exists GeneIsOnChromosomeOfOrganism \sqsubseteq Gene$	Domain of object property
$\exists GeneIsOnChromosomeOfOrganism^{-} \sqsubseteq Organism$	Range of object property
$Gene \sqsubseteq \exists GeneIsOnChromosomeOfOrganism$	Each gene belongs to some organism
$Organism \sqsubseteq \exists GeneIsOnChromosomeOfOrganism^{-}$	Each organism has some gene
(funct $GeneIsOnChrOfOrganism$)	Each gene belongs to at most one organism
...	

Figure 2: Section of the HGT application ontology.

Note: semantics of the icons

Construct	Graphical Element	Semantic
Class	C	$C \sqsubseteq \top$
Object Property	C P D	$\exists P \sqsubseteq C$ $\exists P^{-} \sqsubseteq D$
Data Property	C A	$\delta(A) \sqsubseteq C$ $\rho(A) \sqsubseteq \top_d$
SubClass Relationship	C D	$C \sqsubseteq D$

Example: mapping concepts & relations of the Ontology to SQL query over the relational database

`SELECT id, abbrev`
`FROM organism`
`JOIN genes`
`ON abbrev = idorganism`
	$\rightsquigarrow$	$OrganismHasGene($
		$\mathbf{gene}(id),$
		$\mathbf{organism}(abbrev))$

`SELECT id, kegg` `FROM genes`	$\rightsquigarrow$	$GeneHasGeneFunction($ $\mathbf{gene}(id), \mathbf{function}(id))$ $KEGG(\mathbf{function}(id), kegg)$

Figure 3: Extract of the mapping from the HGT-DB database to the *DL-Lite*$_{\mathcal{A}}$ application ontology.

Mappings between ontology and data

Metadata (Ontology1222766179.owl) | OWLClasses | Properties | Individuals | Forms | Datasource Manager | ABox Queries

DATASOURCE BROWSER

For project: hgt-app...

Datasources

HGT

For datasource: HGT
Type: RDBMS
Mapping Type: OBDAMappings
Source ID: .com/Ontology12227
JDBC URL: :thin:@obdalin.inf.unib
Database Name: obda.obdalin
Database Username:

DATASOURCE MANAGER

Mappings | SQL queries | SQL Schema Inspector

M:0

PromiscuousBacterium(getPromBact($abbrev,$ccount,$percentage))

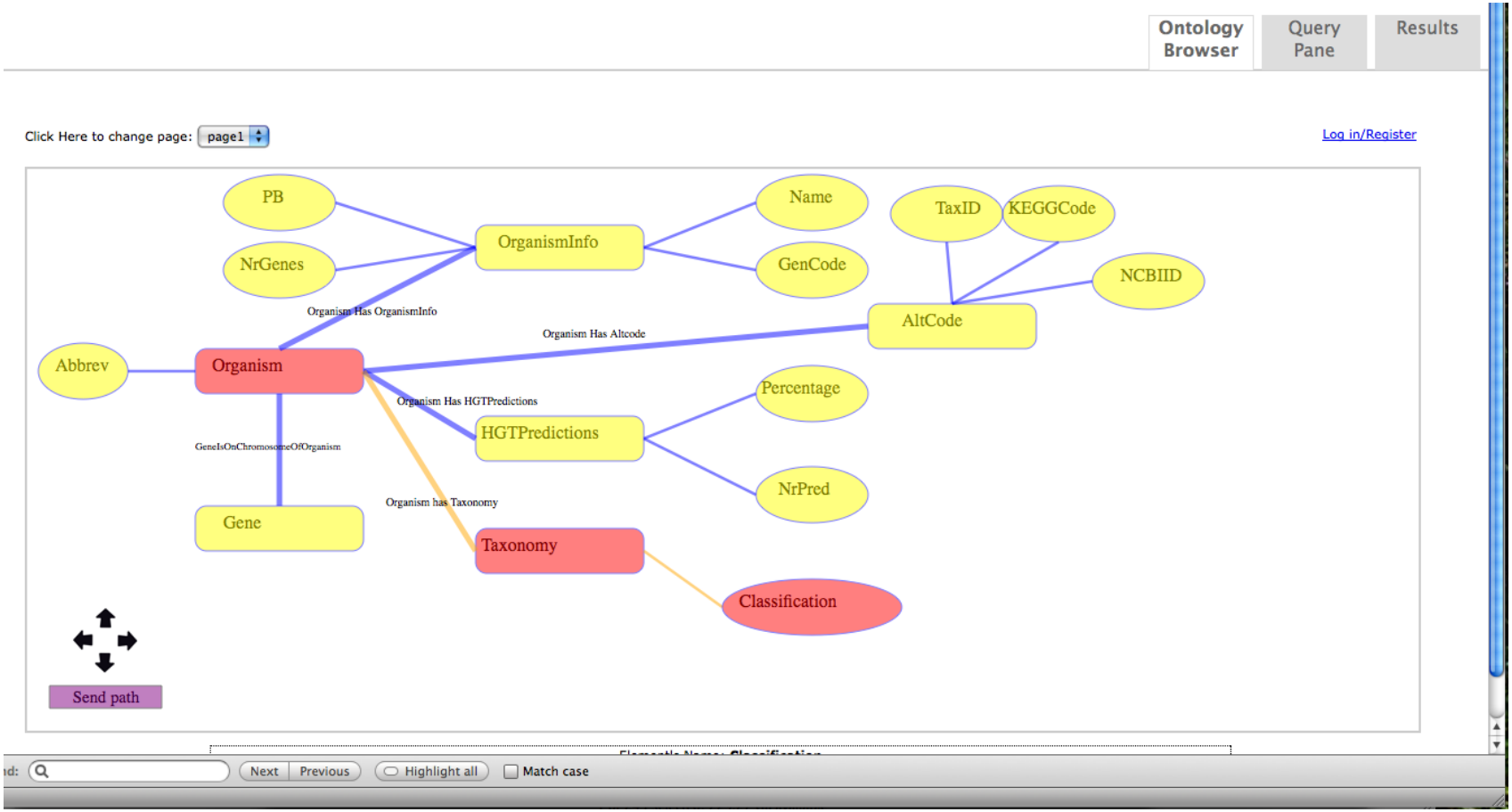
```
SELECT organisme.abbrev, ccount, organisme.percentage
FROM ( SELECT idorganisme, COUNT(distinct cstart) as ccount
       FROM COMCLUSTG2 GROUP BY idorganisme
     ) flexcount, organisme
WHERE organisme.abbrev = flexcount.idorganisme AND
      organisme.percentage > 10 AND flexcount.ccount > 5
```

M:1

PromBactPrime(getPromBactPrime($abbrev,$ccount,$percentage,$hgt))

```
SELECT organisme.abbrev, ccount, organisme.percentage,
       organisme.hgt
FROM ( SELECT idorganisme, COUNT(distinct cstart) as ccount
       FROM COMCLUSTG2 GROUP BY idorganisme
     ) flexcount, organisme
WHERE organisme.abbrev = flexcount.idorganisme AND
      organisme.percentage > 10 AND flexcount.ccount > 10 AND
      organisme.hgt > 150
```

Example: Browsing & selecting



Example: Query pane icons

Construct	Graphical Element	Semantic
Class node	C, D	$C(x), D(x)$
Object Property link	C P D	$C(x), P(x,y), D(y)$
Data Property node and link	C A	$C(x), A(x,y)$

Example: Adding a constraint

Ontology Browser | Query Pane | Results

Log in/Register

Add a Constraint for GCValue

Relational Constraint

GCValue > 60

IN Constraint

GCValue = OR

Between Constraint

<= GCValue <=

Type	Attribute	
Attribute	GCValue	
Label	GCValueFirmicutes	(edit)
Constraint	not constrained	(edit)
Selected?	true	(edit)
Delete	Delete it!	
Query	SELECT q1.bf, q1.cd, q1.cf, q1.dg FROM sparqltable (SELECT $bf $cd $cf $dg WHERE { $cc rdf:type 'GeneIDInfo' . $df rdf:type 'Taxonomy' . $ca rdf:type 'GCstatsGene' . $ca rdf:type 'GCtotal_g' . $ci rdf:type 'Gene' . $dl rdf:type 'Organism' . $ci :GeneIsOnChromosomeOfOrganism $dl . $ci :GeneHasGCstatsGene $ca . $ci :GeneHasGeneIDInfo $cc . $dl :OrganismHasTaxonomy $df . $ca :GCValue $bf . $ci :ID $cd . $cc :GeneName $cf . $df :Classification $dg }) q1 WHERE (q1.dg LIKE '%Firmicute%')	

Manage Constraints

ID

Gene

Example: Managing constraints

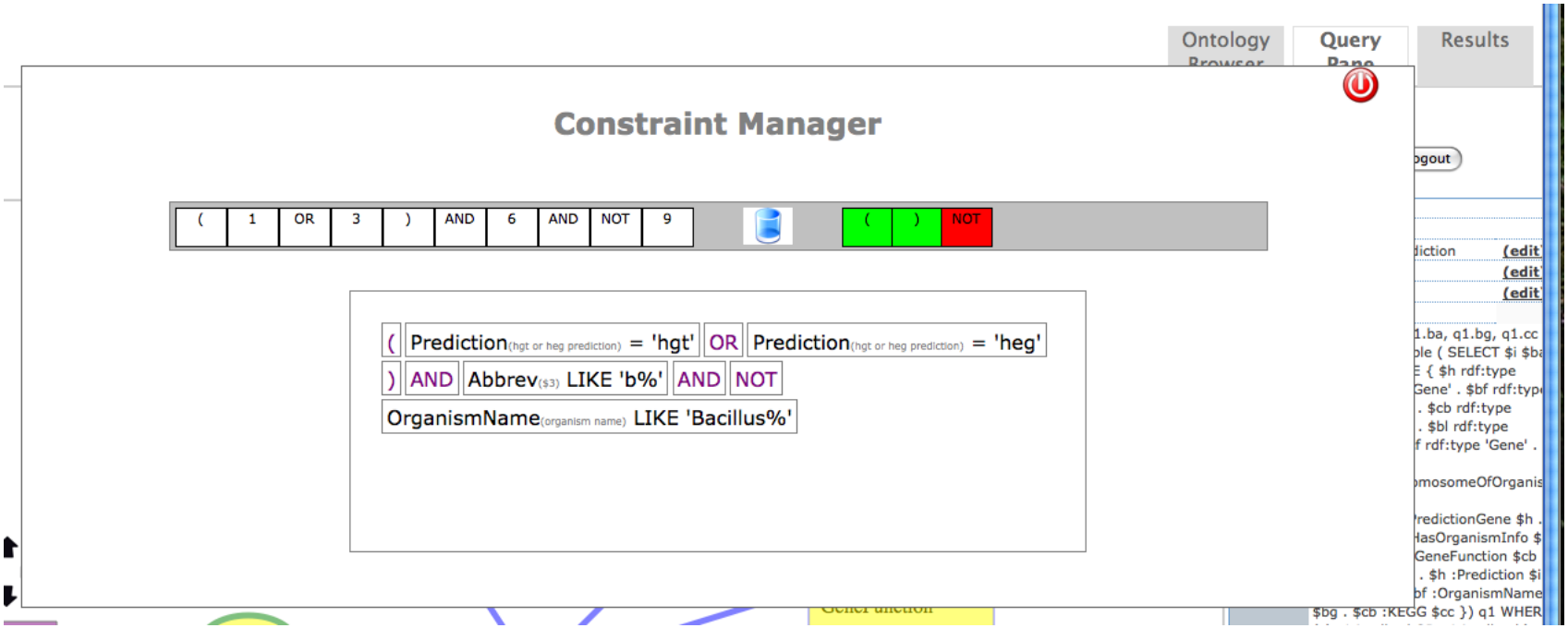


Figure: Constraint manager for getting "Give me the names of the organisms of which the abbreviation starts with a b, but not being a Bacillus, and the prediction and KEGG code of those organisms genes that are putatively either horizontally transferred or highly expressed"

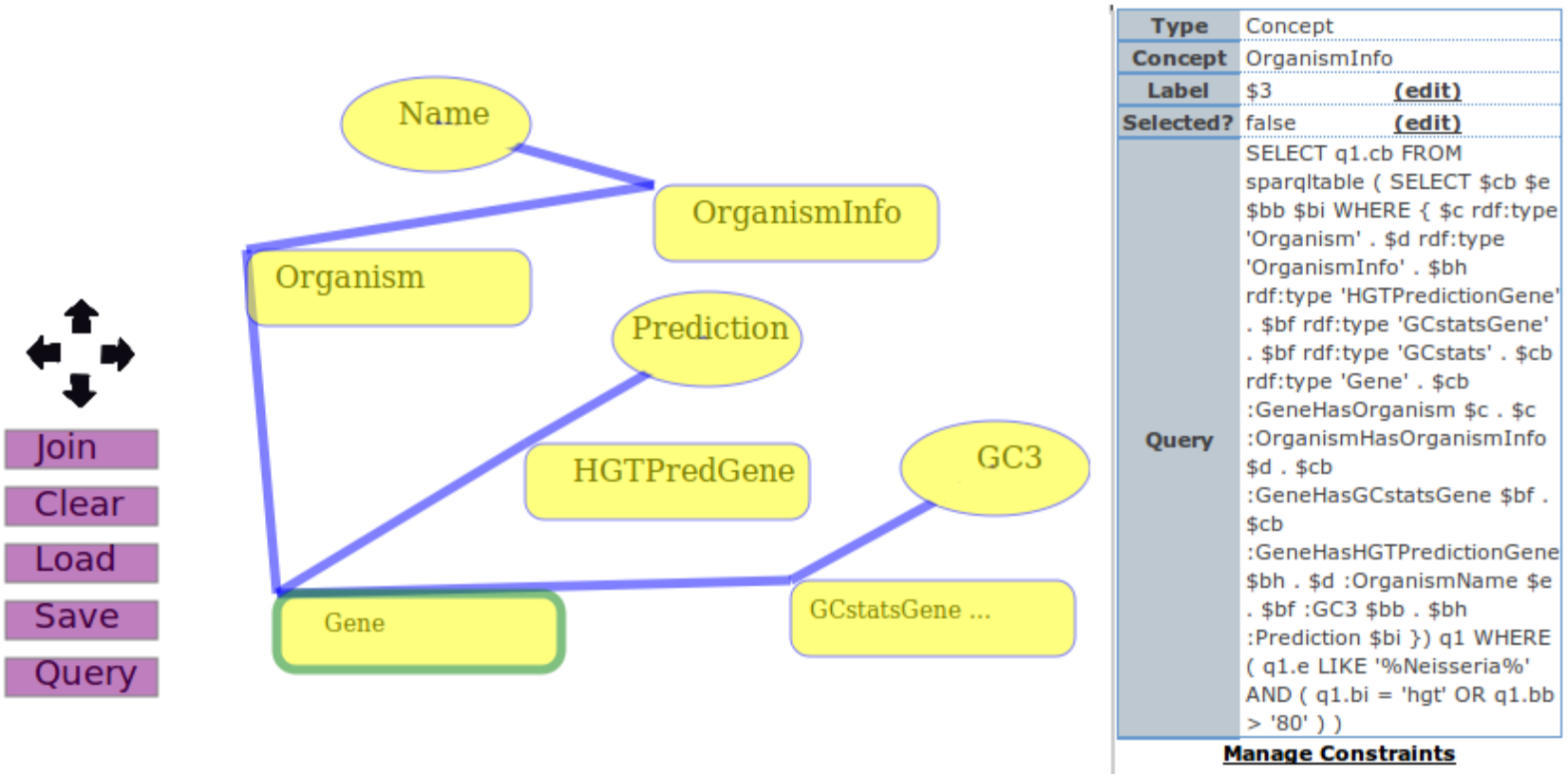


Figure: Query to retrieve the genes of *Neisseria* spp. that have a GC3 content > 80 or are predicted to be horizontally transferred. The textual version of the graphically constructed query (on the right) is generated automatically by the WONDER system.

Example: Section of the results

Ontology Browser | Query Pane | Results

Log in/Register

Download the CSV file

GCValueFirmicutes	geneID	name of the gene	family
'61.900001525878906'	'bcer1_1574'	'-'	'no rank:cellular organisms; superkingdom:Bacteria; phylum:Firmicutes; class:Bacilli; order:Bacillales; family:Bacillaceae; genus:Bacillus; species group:Bacillus cereus group; species:Bacillus cereus; '
'63.79999923706055'	'bcer1_3144'	'-'	'no rank:cellular organisms; superkingdom:Bacteria; phylum:Firmicutes; class:Bacilli; order:Bacillales; family:Bacillaceae; genus:Bacillus; species group:Bacillus cereus group; species:Bacillus cereus; '
'62.79999923706055'	'bcla_469'	'-'	'no rank:cellular organisms; superkingdom:Bacteria; phylum:Firmicutes; class:Bacilli; order:Bacillales; family:Bacillaceae; genus:Bacillus; species:Bacillus clausii; '
			'no rank:cellular

Find: Next Previous Highlight all Match case

Query and user results

- Domain users have more freedom in constructing the queries and thanks to the query loading/saving feature, the overall service is more usable
- While using the WONDER interface, domain users came up with new queries that are interesting for their studies
- The user is aided in the formulation of complex constraints over the queries (Constraint Manager)
- Syntactical correctness of the query is ensured by the formal foundation of the interface

Technological results

- The overhead caused by the graphical interface is negligible w.r.t. the standard OBDA setting
- The approach is sufficiently scalable with pretty large databases ($>$ 4GB)
- Achieved seamless integration of different (Semantic) Web Technologies: OWL 2, AJAX, JavaScript, SVG and XSLT

Summary

- Builds upon the theory, technology, and implementation developed for Ontology-Based Data Access
- Graphical ontology browsing, query formulation, and query execution in a Web browser
- Rigorous formal characterisation and uses a coupling with an OWL file, (U)CQs (in SPARQL syntax) and EQL-Lite queries managed by the DIG-QuOnto reasoner
- This WONDER system meets the scalability and usability requirements, and allows domain experts to query through a web browser the database without the need to learn SPARQL or EQL-Lite

Summary