

# Ontology-based Data Access: From Practice to Theory

Diego Calvanese, Martin Rezk, Guohui Xiao  
Faculty of Computer Science  
Free University of Bozen-Bolzano, Italy



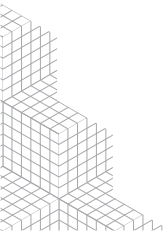
FREIE UNIVERSITÄT BOZEN  
LIBERA UNIVERSITÀ DI BOLZANO  
FREE UNIVERSITY OF BOZEN · BOLZANO

International Semantic Web Conference  
12/10/MMXV

# What are we going to learn today?

Optique

- How to organize and access your data using **ontologies**.
- How to do it with our system: **—ontop—**.
- How to use this approach for **data integration** and **consistency checking**.



## Ontology Based Data Access

- The Database:

- Ontologies

- Mappings

- Virtual Graph

- Querying

## Ontology Based Data Integration

- SQL Federation

- Checking Consistency

- Conclusions



## Ontology Based Data Access

The Database:

Ontologies

Mappings

Virtual Graph

Querying

## Ontology Based Data Integration

SQL Federation

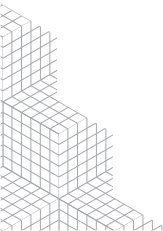
Checking Consistency

Conclusions



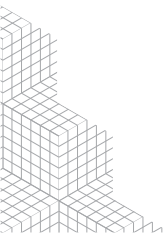
# Simple Life Science Running Example **Optique**

- **Data source(s)**: Hospital Databases with cancer patients (First 1, then 2)
- **Ontology**: A common domain vocabulary defining Patient, Cancer, LungCancer, etc.
- **Mappings**: Relating the vocabulary and the databases.



Before we start you need:

- Java
- The material online:  
`https://github.com/ontop/ontop-examples/tree/master/iswc-tutorial-2015`



## Ontology Based Data Access

### The Database:

Ontologies

Mappings

Virtual Graph

Querying

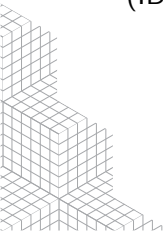
## Ontology Based Data Integration

SQL Federation

## Checking Consistency

## Conclusions

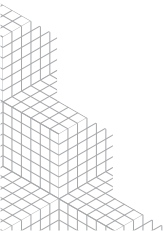
- Standard way to store **LARGE volumes** of data: Mature, Robust and FAST.
- Domain is structured as tables, data becomes rows in these tables.
- Powerful query language (**SQL**) to retrieve this data.
- Major companies developed SQL DBs for the last **30 years** (IBM, Microsoft, Oracle)..





Cancer Patient Database 1  
Table: tbl\_patient

PatientId	Name	Type	Stage
1	Mary	false	4
2	John	true	7



Cancer Patient Database 1  
Table: tbl\_patient

PatientId	Name	Type	Stage
1	Mary	false	4
2	John	true	7

Type is:

- false for Non-Small Cell Lung Cancer (NSCLC)
- true for Small Cell Lung Cancer (SCLC)

Stage is:

- 1-6 for NSCLC stages: I,II,III,IIIa,IIIb,IV
- 7-8 for SCLC stages: Limited,Extensive

- H2 is a pure java SQL database
- Just unzip the downloaded package
- Easy to run, just run the scripts:
  - Open a terminal (in mac Terminal.app, in windows run cmd.exe)
  - Move to the H2 folder (e.g., `cd h2`)
- Start H2 using the h2 scripts
  - `sh h2.sh` (in mac/linux - You might need "`chmod u+x h2.sh` ")
  - `h2w.bat` (in windows)

# How it looks:

English Options Outils Aide

**Connexion**

Configuration enregistrée: Generic H2 (Embedded)

Nom de configuration: Generic H2 (Embedded) Enregistrer Supprimer

Pilote JDBC: org.h2.Driver

URL JDBC: jdbc:h2:tcp://localhost/helloworld;DATABASE\_TO\_UPPE

Nom d'utilisateur: sa

Mot de passe:

Connecter Test de connexion

- jdbc:h2:tcp: = protocol information
- localhost = server location
- helloworld = database name

# How to access it from the web

Optique

Tables Info

Write you queries Here!

Example Queries

Exemple de script SQL	
Effacer une table si elle existe	DROP TABLE IF EXISTS TEST;
Créer une nouvelle table avec les colonnes ID et NAME	CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255));
Ajouter un nouvel enregistrement	INSERT INTO TEST VALUES(1, 'Hello');
Ajouter un autre enregistrement	INSERT INTO TEST VALUES(2, 'World');
Requêter une table	SELECT * FROM TEST ORDER BY ID;
Modifier un enregistrement	UPDATE TEST SET NAME='Hi' WHERE ID=1;
Effacer un enregistrement	DELETE FROM TEST WHERE ID=2;
Aide	HELP ...

You can use the files create.sql and insert.sql

```
CREATE TABLE "tbl_patient" (  
  patientid INT NOT NULL PRIMARY KEY,  
  name VARCHAR(40),  
  type BOOLEAN,  
  stage TINYINT  
)
```

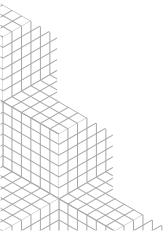
Adding Data:

```
INSERT INTO "tbl_patient"  
(patientid,name,type,stage)  
VALUES  
(1,'Mary',false,4),  
(2,'John',true,7);
```

Patients with type false and stage IIIa or above (select.sql)

```
SELECT patientid  
FROM "tbl_patient"  
WHERE  
TYPE = false AND stage >= 4
```

Give me the id and the name of the patients with a tumor at stage  
IIIa





## Ontology Based Data Access

The Database:

Ontologies

Mappings

Virtual Graph

Querying

## Ontology Based Data Integration

SQL Federation

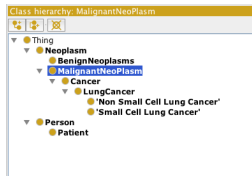
Checking Consistency

Conclusions

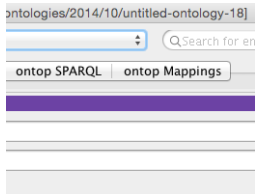
## Definition

An artifact that contains a vocabulary, relations between the terms in the vocabulary, and that is expressed in a language whose syntax and semantics (meaning of the syntax) are shared and agreed upon.

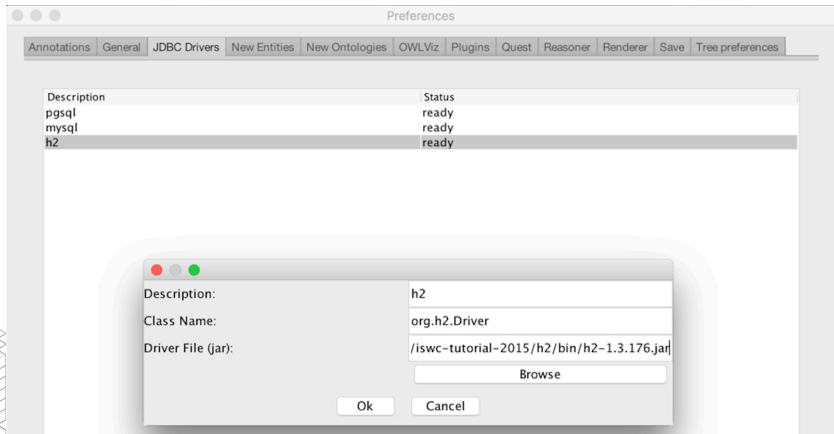
Mike **type** Patient  
NSCLC **subClassOf** LungCancer  
LungCancer **subClassOf** Cancer



- Go to the protégé-ontop folder from your material. This is a Protégé 5 package that includes the ontop (1.15) plugin
- Run Protégé from the console using the run.bat or run.sh scripts. That is, execute:
- cd Protege\_5/; run.sh



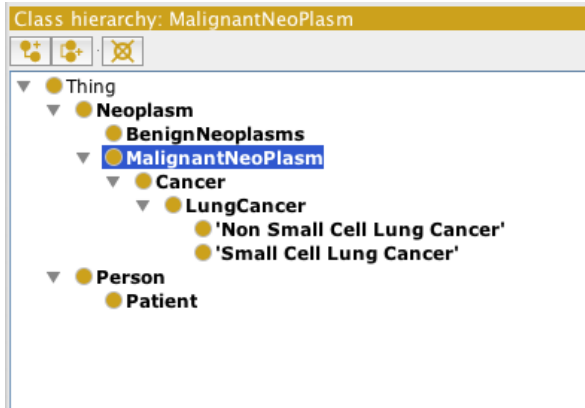
- Open “Preferences”, “JDBC Drivers” and add the configuration for H2
  - Description: h2
  - Class Name: org.h2.Driver
  - Driver File (jar): /path/to/h2/bin/h2-1.3.176.jar



# The ontology: Creating Concepts and Properties

Optique

Add the concept: Patient.

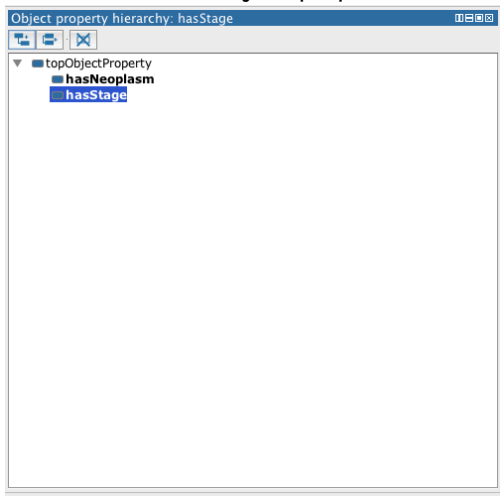


(See PatientOnto.owl)

# The ontology: Creating Concepts and Properties

Optique

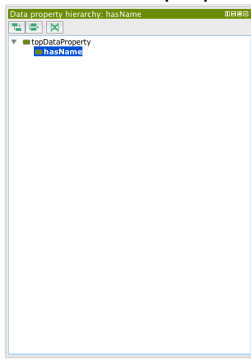
Add these object properties:



# The ontology: Creating Concepts and Properties

Optique

Add these data properties:



(See PatientOnto.owl)

## Ontology Based Data Access

The Database:

Ontologies

**Mappings**

Virtual Graph

Querying

## Ontology Based Data Integration

SQL Federation

Checking Consistency

Conclusions



- We have the vocabulary, the database, now we need to **link** those two.
- Mappings define **triples** (subject, property, object) out of SQL queries.
- These triples is accessible during query time (the **on-the-fly** approach) or can be imported into the OWL ontology (the ETL approach)

## Definition (Intuition)

A mapping have the following form:

*TripleTemplate*  $\leftarrow$  SQL Query to build the triples

represents triples constructed from each result row returned by the SQL query in the mapping.

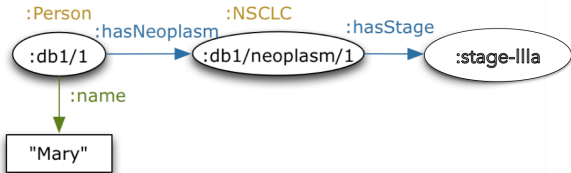
p.Id	Name	Type	Stage
1	Mary	false	2

- $(:db1/\{p.id\}, type, :Patient) \leftarrow \text{Select } p.id \text{ From } tbl\_patient$
- $(:db1/\{p.id\}, :hasName, \{name\}) \leftarrow \text{Select } p.id, name \text{ From } tbl\_patient$
- $(:db1/\{p.id\}, :hasNeoplasm, :db1/neoplasm/\{p.id\}) \leftarrow$   
Select p.id From tbl\_patient
- $(:db1/neoplasm/\{p.id\}, :hasStage, :stage-IIIa) \leftarrow$   
Select p.id From tbl\_patient where stage=4

# Optique

p.Id	Name	Type	Stage
1	Mary	false	2

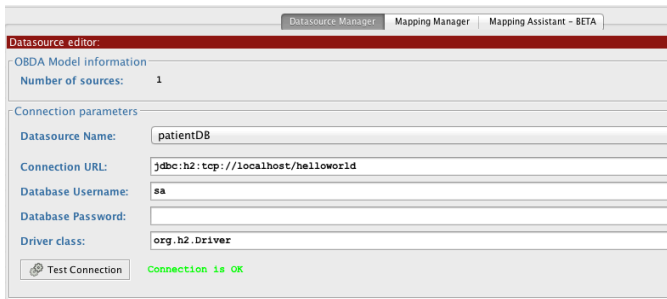
- (:db1/{p.id},type, :Patient) ← Select p.id From tbl\_patient
- (:db1/{p.id},:hasName, {name}) ← Select p.id,name From tbl\_patient
- (:db1/{p.id},:hasNeoplasm, :db1/neoplasm/{p.id}) ←  
Select p.id From tbl\_patient
- (:db1/neoplasm/{p.id},:hasStage, :stage-IIIa) ←  
Select p.id From tbl\_patient where stage=4



Using the Ontop Mapping tab, we now need to define the connection parameters to our lung cancer database

Steps:

1. Switch to the Ontop Mapping tab
2. Add a new data source (give it a name, e.g., PatientDB)
3. Define the connection parameters as follows:
  - Connection URL: jdbc:h2:tcp://localhost/helloworld
  - Username: sa
  - Password: (leave empty)
  - Driver class: org.h2.Driver (choose it from the drop down menu)
4. Test the connection using the “Test Connection” button



The screenshot shows the 'Datasource Manager' tab in the Optique application. The 'Datasource editor' section is active, displaying 'OBDA Model information' with 'Number of sources: 1'. Below this, the 'Connection parameters' section contains the following fields:

- Datasource Name:** patientDB
- Connection URL:** jdbc:h2:tcp://localhost/helloworld
- Database Username:** sa
- Database Password:** (empty field)
- Driver class:** org.h2.Driver

At the bottom, there is a 'Test Connection' button with a gear icon. To its right, the status 'Connection is OK' is displayed in green text.

- Switch to the “Mapping Manager” tab in the ontop mappings tab.
- Select your datasource
- click Create:

**target:** :db1/{patientid} a :Patient .

**source:** SELECT patientid FROM "tbl\_patient"

**target:** :db1/{patientid} :hasName {name} .

**source:** Select patientid,name FROM "tbl\_patient"

**target:** :db1/{patientid} :hasNeoplasm :db1/neoplasm/{patientid}.

**source:** SELECT patientid FROM "tbl\_patient"

**target:** :db1/neoplasm/{patientid} :hasStage :stage-IIIa .

**source:** SELECT patientid FROM "tbl\_patient" where stage=4

- Now we classify the neoplasm individual using our knowledge of the database.
- We know that “false” in the table patient indicates a “Non Small Cell Lung Cancer”, so we classify the patients as a **:NSCLC**.

**nsclc**

```
:db1/neoplasm/{PATIENTID}/ a :NSCLC .  
select * from TBL_PATIENT WHERE TYPE = false
```

**sclc**

```
:db1/neoplasm/{PATIENTID}/ a :SCLC .  
select * from TBL_PATIENT WHERE TYPE = true
```

## Ontology Based Data Access

The Database:

Ontologies

Mappings

Virtual Graph

Querying

## Ontology Based Data Integration

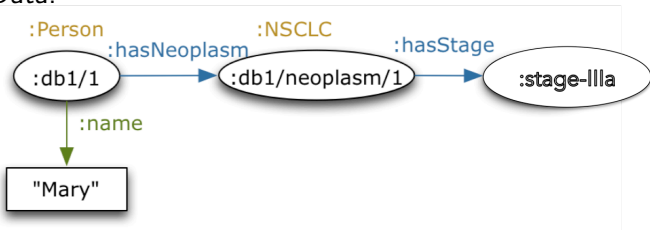
SQL Federation

Checking Consistency

Conclusions

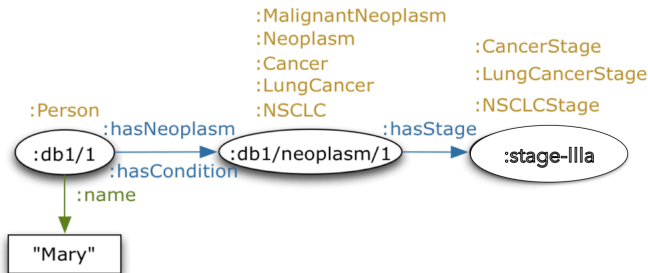


Data:



- The vocabulary is more **domain oriented** and **independent from the DB**.
- No more values to encode types or stages.
- Later, this will allow us to **easily integrate** new data or domain information (e.g., an ontology).
- Our data sources are now **documented!**.

## Data and Inference:



- There is a new individual `:db1/neoplasm/1` that stands for the cancer (tumor) of Mary. This allows the user to query specific properties of the tumor independently of the patient.
- We get extra information as shown above.

## Ontology Based Data Access

The Database:

Ontologies

Mappings

Virtual Graph

Querying

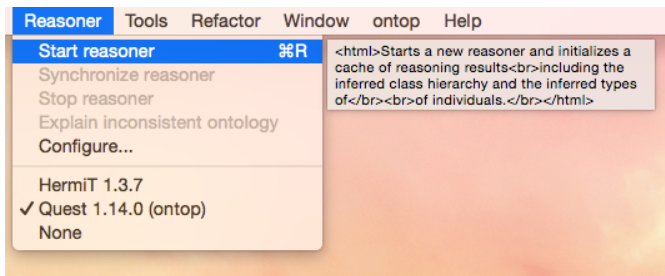
## Ontology Based Data Integration

SQL Federation

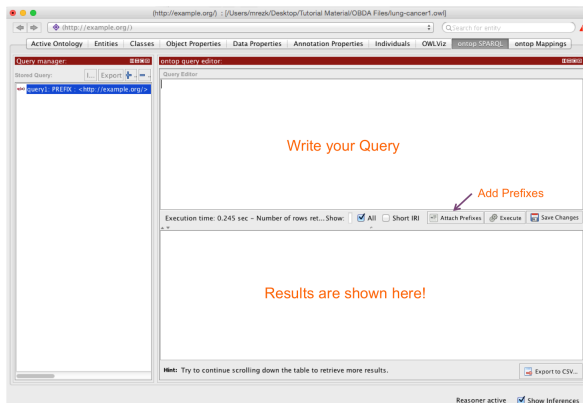
Checking Consistency

Conclusions

- Recall our information need: Give me the id and the name of the patients with a tumor at stage IIIa.
- Enable Ontop in the “Reasoner” menu



In the ontop SPARQL tab add all the prefixes

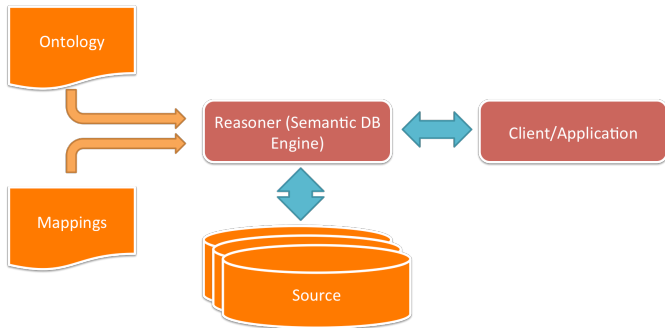


- Write the SPARQL Query

```
SELECT ?p ?name WHERE
{ ?p rdf:type :Patient .
  ?p :hasName ?name .
  ?p :hasNeoplasm ?tumor .
  ?tumor :hasStage :stage-IIIa . }
```

- Click execute
- This is the main way to access data in ontop and its done by querying ontop with SPARQL.

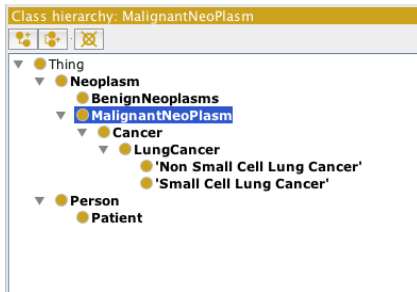
# How we do Inference...



We embed **inference** into the query  
We do **not** need to reason with the (Big) Data

- If we pose the query asking for all the instances of the class Neoplasm:

```
SELECT ?x WHERE { ?x rdf:type :Noeplasms). }
```





- If we pose the query asking for all the instances of the class Neoplasm:

```
SELECT ?x WHERE { ?x rdf:type :Neoplasms). }
```

- (Intuitively) -ontop- will translate it into:

```
SELECT ?x WHERE { { ?x rdf:type :Neoplasms. }  
  UNION  
  { ?x rdf:type :BenignNeoplasms. }  
  UNION  
  { ?x rdf:type :MalignantNeoplasm. }  
  UNION  
  :  
  { ?x rdf:type :NSCLC). }  
  UNION  
  { ?x rdf:type :SCLC). } }
```

- If we pose the query asking for all the instances of the class Neoplasm:

```
SELECT ?x WHERE { ?x rdf:type :Neoplasms). }
```

- (Intuitively) -ontop- will translate it into:

```
SELECT ?x WHERE { {?x rdf:type :Neoplasms. }  
UNION  
{ ?x rdf:type :BenignNeoplasms. }  
UNION  
{ ?x rdf:type :MalignantNeoplasm. }  
UNION  
:  
:  
{ ?x rdf:type :NSCLC). }  
UNION  
{ ?x rdf:type :SCLC). } }
```

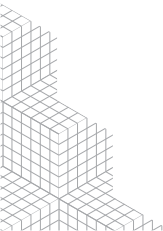
- If we pose the query asking for all the instances of the class Neoplasm:

```
SELECT ?x WHERE { ?x rdf:type :Noeplasms}. }
```

- (Intuitively) -ontop- will translate it into:

```
SELECT Concat(:db1/neoplasm/, TBL.PATIENT.id) AS ?x  
FROM TBL.PATIENT
```

- Ontology languages: RDF, RDFS, OWL (W3C recommendations)
- User Query Language: SPARQL (W3C recommendation)
- Mappings: R2RML (W3C recommendation)
- DB Query Language: SQL



## Ontology Based Data Access

The Database:

Ontologies

Mappings

Virtual Graph

Querying

## Ontology Based Data Integration SQL Federation

Checking Consistency

Conclusions

# What about Data Integration?

## Cancer Patient Database 2

T\_Name

PId	Nombre
1	Anna
2	Mike

DB information is distributed in multiple tables.  
The IDs of the two DBs overlap.

T\_NSCLC

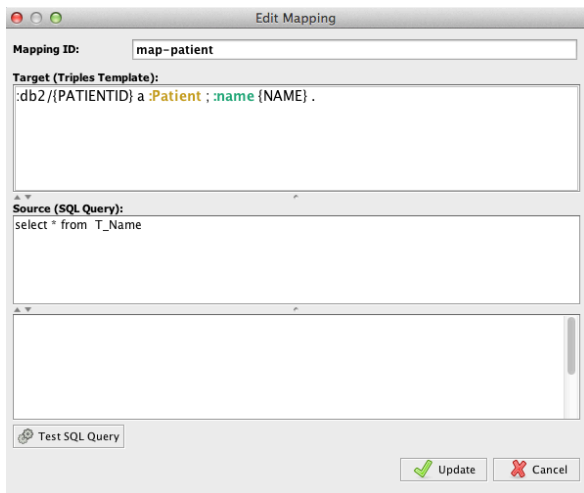
Id	hosp	Stge
1	X	two
2	Y	one

Information is encoded differently. E.g. Stage of cancer is text (one, two...)

T\_SCLC

key	hosp	St
1	XXX	
2	YYY	

# New Mappings



**Edit Mapping**

**Mapping ID:** map-patient

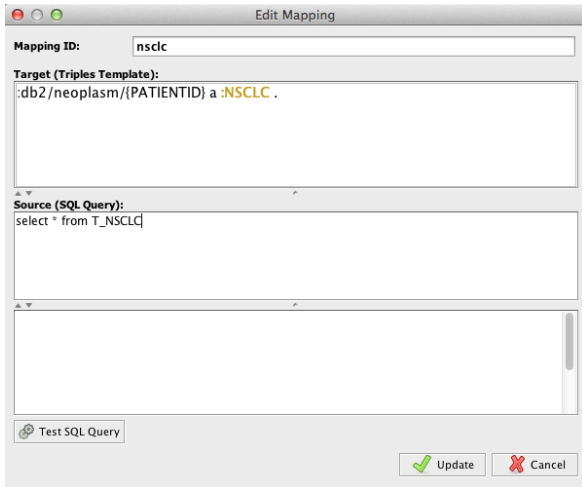
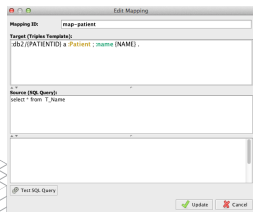
**Target (Triples Template):**  
:db2/{PATIENTID} a :Patient ; :name {NAME} .

**Source (SQL Query):**  
select \* from T\_Name

Test SQL Query

Update Cancel

# New Mappings





The screenshot shows the 'Edit Mapping' window with the title 'map-patient'. The 'Mapping ID' field contains 'map-patient'. The 'Target (Triples Template):' field contains the query `:db2/(PATIENTID) a :Patient ; :name (NAME) .`. The 'Source (SQL Query):' field contains the query `select * from T_Name`. At the bottom, there is a 'Test SQL Query' button and 'Update' and 'Cancel' buttons.

The screenshot shows the 'Edit Mapping' window with the title 'nslc'. The 'Mapping ID' field contains 'nslc'. The 'Target (Triples Template):' field contains the query `:db2/(neoplasm/(PATIENTID) a :NSCLC .`. The 'Source (SQL Query):' field contains the query `select * from T_NSCLC`. At the bottom, there is a 'Test SQL Query' button and 'Update' and 'Cancel' buttons.

- The URI's for the new individuals differentiate the data sources (db2 vs. db1)
- Being an instance of NSCLC and SCLC depends now on the table, not a column value

## Ontology Based Data Access

The Database:

Ontologies

Mappings

Virtual Graph

Querying

## Ontology Based Data Integration

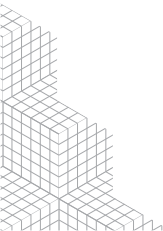
SQL Federation

Checking Consistency

Conclusions

`http://www.exareme.org`

- Developed at the University of Athens.
- Query Processing module of the Optique Platform.
- Soon it will be release as Open Source.



Steps to make it work:

- Install Exareme (Developed at TUA).
  - You will need Python 2.7 and APSW.
- Build the JDBC URL for ontop. It consists of:
  - 1 fragment for the server where Exareme is installed
  - 1 fragment for each underlying DB.

Steps to make it work:

- Install Exareme (Developed at TUA).
  - You will need Python 2.7 and APSW.
- Build the JDBC URL for ontop. It consists of:
  - 1 fragment for the server where Exareme is installed
  - 1 fragment for each underlying DB.

```
jdbc:fedadp:http://10.7.20.80:9090/tmp-  
fedDB-data1-next-jdbc:postgresql://10.7.20.80/exareme1-next-  
org.postgresql.Driver-next-postgres-next-postgres-next-public-  
fedDB-data2-next-jdbc:postgresql://10.7.20.39/exareme2-next-  
org.postgresql.Driver-next-postgres-next-postgres-next-public
```

Steps to make it work:

- Install Exareme (Developed at TUA).
  - You will need Python 2.7 and APSW.
- Build the JDBC URL for ontop. It consists of:
  - 1 fragment for the server where Exareme is installed
  - 1 fragment for each underlying DB.

Connection parameters

Datasource Name: testExareme

Connection URL: esql://10.7.20.39/exareme2-next-org.postgresql.Driver-next-postgres-next-postgres-next-public

Database Username: adp

Database Password:

Driver class: nadjik.adp.federatedjdbc.AdpDriver

Test Connection

You can build your mappings and query as usual !!

## Ontology Based Data Access

The Database:

Ontologies

Mappings

Virtual Graph

Querying

## Ontology Based Data Integration

SQL Federation

## Checking Consistency

Conclusions

- A logic based ontology language, such as OWL, allows ontologies to be specified as **logical theories**, this implies that it is possible to **constrain** the relationships between concepts, properties, and data.
- In OBDA inconsistencies arise when your mappings **violate** the constraints imposed by the ontology.
- In OBDA we have two types of constraints:
  - **Disjointness**: The intersection between classes Patient and Employee should be empty. There can also be disjoint properties.
  - **Functional Properties**: Every patient has at most one name.



# Consistency: Setting up a Constraint

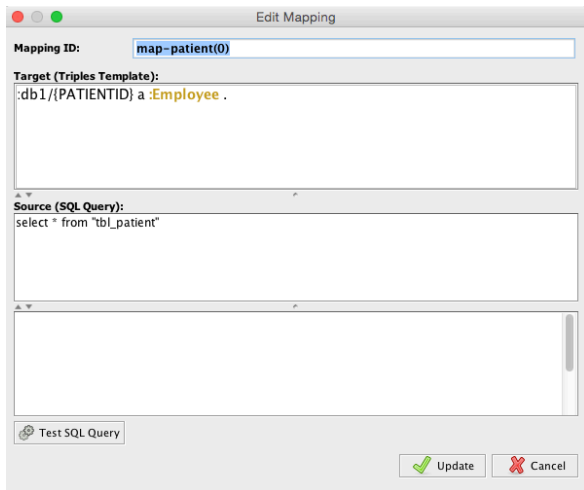
The screenshot displays the Optique web interface for an ontology. The browser address bar shows the URL `(http://example.org/)` and the file path `[/Users/mrzk/Desktop/Tutorial Material/OBDA Files/lung-cancer1.owl]`. The interface includes a top navigation bar with tabs: **Active Ontology**, **Entities**, **Classes**, **Object Properties**, **Data Properties**, **Annotation Properties**, **Individuals**, **OWL Viz**, **ontop Mappings**, and **ontop SPARQL**. A search bar labeled "Q Search for entity" is located to the right of the navigation bar.

The main content area is divided into several panels:

- Class hierarchy:** Shows a tree structure with **Thing** as the root, and **Employee** and **Patient** as subclasses.
- Object property hierarchy:** Shows a tree structure with **topObjectProperty** as the root.
- Annotations - Employee:** A panel for managing annotations for the **Employee** class.
- Description: Employee:** A panel for managing the description of the **Employee** class, including sections for **Equivalent To**, **SubClass Of**, **SubClass Of (Anonymous Ancestor)**, **Members**, **Target for Key**, **Disjoint With**, and **Disjoint Union Of**. The **Disjoint With** section shows **Patient** as a constraint.

At the bottom of the interface, there are two status indicators: **Reasoner state out of sync with active ontology** and **Show Inferences** (checked).

# Consistency: Building a wrong mapping Optique






**Edit Mapping**

**Mapping ID:** `map-patient(0)`

**Target (Triples Template):**  
`:db1/{PATIENTID} a :Employee .`

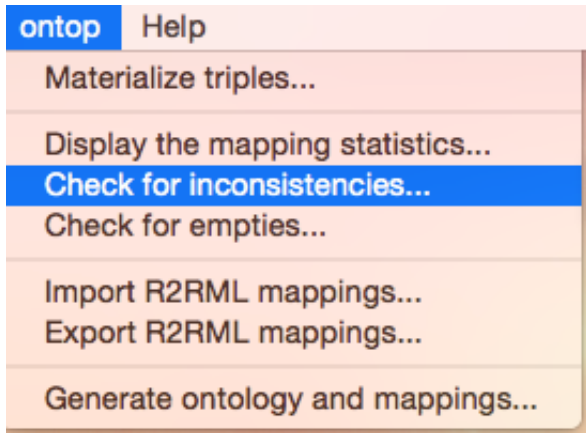
**Source (SQL Query):**  
`select * from "tbl_patient"`

 Test SQL Query

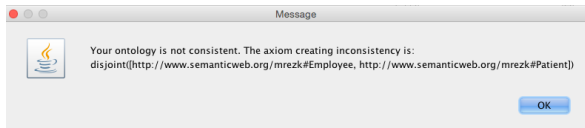
 Update  Cancel

# Consistency: Checking Inconsistency

Optique



# Consistency: Finding out the Problem Optique



## Ontology Based Data Access

The Database:

Ontologies

Mappings

Virtual Graph

Querying

## Ontology Based Data Integration

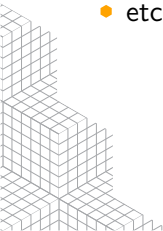
SQL Federation

Checking Consistency

**Conclusions**

- Ontologies give you a common vocabulary to formulate the **queries**, and mappings to find the **answers**.
- Ontologies and Semantic Technology can help to handle the problem of **accessing Big Data**
  - **Diversity**:
    - Using ontologies describing particular domains allows to **hide** the storage complexity.
    - Agreement on data identifiers **allows for integration** of datasets.
  - **Understanding**: Agreement on vocabulary allow to better define your data and allows for **easy information exchange**.
- There is no need of computationally **expensive ETL** processes.
- **Reasoning is scalable** because we reason at the query level.
- You do not need to have everything ready to **use it!**

- Semantic Query Optimisation
- SWRL and Recursion
- Performance Evaluation
- Aggregates and bag semantics
- Give out about Database Engines
- Tons of theory
- etc. etc. etc...



# Thanks!!!

Visit us!

<http://ontop.inf.unibz.it>  
<http://www.optique-project.eu>

