Ontology Based Data Access

Elena Botoeva

Free University of Bozen-Bolzano, Italy

January 9
CSWR 2015
Santiago, Chile

Acknowledgements: Diego Calvanese, Roman Kontchakov,
Martin Rezk, Davide Lanti

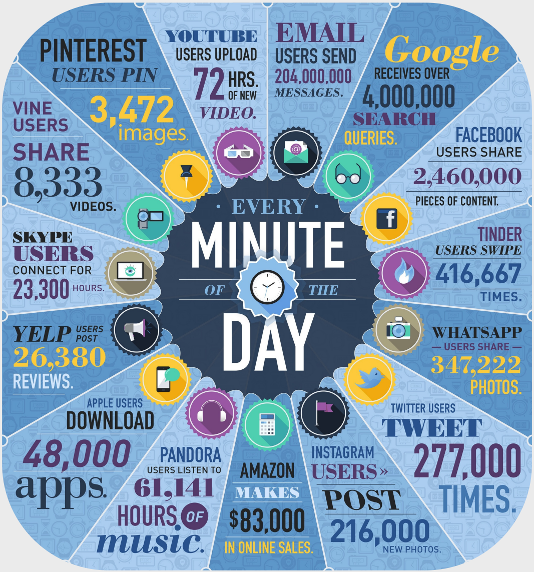
Outline

Outline

The Problem: Information Need

how to formulate the right question
to find the right answer
in the ocean of Big Data

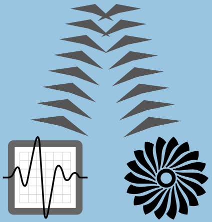
We are Living in the Era of Big Data



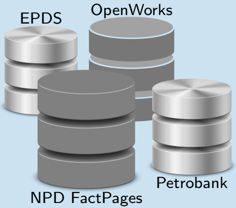
Data Never Sleeps 2.0

Big Data in Industry

30GB/ 24h



3000 Tables
>20000 Columns
TBs and PBs

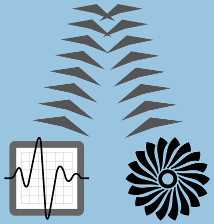


Big Data in Industry

VELOCITY

SIEMENS

30GB/ 24h



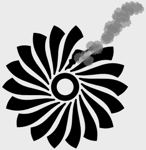
Statoil

3000 Tables
>20000 Columns
TBs and PBs

COMPLEXITY

SIEMENS: Information Need

1. Reactive diagnostics of turbines.



2. Predictive analysis of turbine condition.

3. Product engineering and maintenance support.

How Much Time is Spent Searching for Data?

Engineers in industry spend a significant amount of their time searching for data that they require for their core tasks. E.g., in the oil&gas industry, 30-70% of engineers time is spent looking for and assessing the quality of data (Crompton, 2008).

How Much Time/Money is Spent Searching for Data?

A user query in Statoil:

Show all norwegian wellbores with some aditional attributes (wellbore id, completion date, oldest penetrated age,result). Limit to all wellbores with a core and show attributes like (wellbore id, core number, top core depth, base core depth, intersecting stratigraphy). Limit to all wellbores with core in Brentgruppen and show key atributes in a table. After connecting to EPDS (slegge) we could for instance limit futher to cores in Brent with measured permeability and where it is larger than a given value, for instance 1 mD. We could also find out whether there are cores in Brent which are not stored in EPDS (based on NPD info) and where there could be permeability values. Some of the missing data we possibly own, other not.

How Much Time/Money is Spent Searching for Data?

A use

Show
comp core
and s re
depth
Brent S
(slegg
perm Ve
could n
EPDS
Some

```
SELECT [...]
FROM
db_name.table1 table1,
db_name.table2 table2a,
db_name.table2 table2b,
db_name.table3 table3a,
db_name.table3 table3b,
db_name.table3 table3c,
db_name.table3 table3d,
db_name.table4 table4a,
db_name.table4 table4b,
db_name.table4 table4c,
db_name.table4 table4d,
db_name.table4 table4e,
db_name.table4 table4f,
db_name.table5 table5a,
db_name.table5 table5b,
db_name.table6 table6a,
db_name.table6 table6b,
db_name.table7 table7a,
db_name.table7 table7b,
db_name.table8 table8,
db_name.table9 table9,
db_name.table10 table10a,
db_name.table10 table10b,
db_name.table10 table10c,
db_name.table11 table11,
db_name.table12 table12,
db_name.table13 table13,
db_name.table14 table14,
db_name.table15 table15,
db_name.table16 table16
WHERE [...]
table2a.attr1=‘keyword’ AND
table3a.attr2=table10c.attr1 AND
table3a.attr6=table6a.attr3 AND
table3a.attr9=‘keyword’ AND
table4a.attr10 IN (‘keyword’) AND
table4a.attr1 IN (‘keyword’) AND
table5a.kinds=table4a.attr13 AND
table5b.kinds=table4c.attr74 AND
table5b.name=‘keyword’ AND
(table6a.attr19=table10c.attr17 OR
(table6a.attr2 IS NULL AND
table10c.attr4 IS NULL)) AND
table6a.attr14=table5b.attr14 AND
table6a.attr2=‘keyword’ AND
(table6b.attr14=table10c.attr8 OR
(table6b.attr4 IS NULL AND
table10c.attr7 IS NULL)) AND
table6b.attr19=table5a.attr55 AND
table6b.attr2=‘keyword’ AND
table7a.attr19=table2b.attr19 AND
table7a.attr17=table15.attr19 AND
table4b.attr11=‘keyword’ AND
table8.attr19=table7a.attr80 AND
table8.attr19=table13.attr20 AND
table8.attr4=‘keyword’ AND
table9.attr10=table16.attr11 AND
table3b.attr19=table10c.attr18 AND
table3b.attr22=table12.attr63 AND
table3b.attr66=‘keyword’ AND
table10a.attr54=table7a.attr8 AND
table10a.attr70=table10c.attr10 AND
table10a.attr16=table4d.attr11 AND
table4c.attr99=‘keyword’ AND
table4c.attr1=‘keyword’ AND
table11.attr10=table5a.attr10 AND
table11.attr40=‘keyword’ AND
table11.attr50=‘keyword’ AND
table2b.attr1=table1.attr8 AND
table2b.attr9 IN (‘keyword’) AND
table2b.attr2 LIKE ‘keyword’% AND
table12.attr9 IN (‘keyword’) AND
table7b.attr1=table2a.attr10 AND
table3c.attr13=table10c.attr1 AND
table3c.attr10=table6b.attr20 AND
table3c.attr13=‘keyword’ AND
table10b.attr16=table10a.attr7 AND
table10b.attr11=table7b.attr8 AND
table10b.attr13=table4b.attr89 AND
table13.attr1=table2b.attr10 AND
table13.attr20=’‘keyword’’ AND
table13.attr15=‘keyword’ AND
table3d.attr49=table12.attr18 AND
table3d.attr18=table10c.attr11 AND
table3d.attr14=‘keyword’ AND
table4d.attr17 IN (‘keyword’) AND
table4d.attr19 IN (‘keyword’) AND
table16.attr28=table11.attr56 AND
table16.attr16=table10b.attr78 AND
table16.attr5=table14.attr56 AND
table4e.attr34 IN (‘keyword’) AND
table4e.attr48 IN (‘keyword’) AND
table4f.attr89=table5b.attr7 AND
table4f.attr45 IN (‘keyword’) AND
table4f.attr1=‘keyword’ AND
table10c.attr2=table4e.attr19 AND
(table10c.attr78=table12.attr56 OR
(table10c.attr55 IS NULL AND
table12.attr17 IS NULL))
```

How Much Time/Money is Spent Searching for Data?

A use

Show

```
SELECT [...]
FROM
db_name.table1 table1,
db_name.table2 table2a,
db_name.table2 table2b,
db_name.table3 table3a,
db_name.table3 table3b,
db_name.table3 table3c,
db_name.table3 table3d,
db_name.table4 table4a,
db_name.table4 table4b,
db_name.table4 table4c,

table2a.attr1='keyword' AND
table3a.attr2=table10c.attr1 AND
table3a.attr6=table6a.attr3 AND
table3a.attr9='keyword' AND
table4a.attr10 IN ('keyword') AND
table4a.attr1 IN ('keyword') AND
table5a.kinds=table4a.attr13 AND
table5b.kinds=table4c.attr74 AND
table5b.name='keyword' AND
(table6a.attr19=table10c.attr17 OR
(table6a.attr2 IS NULL AND
table10c.attr4 IS NULL)) AND

table11.attr10=table5a.attr10 AND
table11.attr40='keyword' AND
table11.attr50='keyword' AND
table2b.attr1=table1.attr8 AND
table2b.attr9 IN ('keyword') AND
table2b.attr2 LIKE 'keyword'% AND
table12.attr9 IN ('keyword') AND
table7b.attr1=table2a.attr10 AND
table3c.attr13=table10c.attr1 AND
table3c.attr10=table6b.attr20 AND
table3c.attr13='keyword' AND
table10b.attr16=table10a.attr7 AND
```

In SIEMENS, it usually takes 4 to 6 weeks to formulate a query in SQL.

SIEMENS loses up to **50.000.000€** per year because of this!!

perm

could

EPDS

Some

```
db_name.table9 table9,
db_name.table10 table10a,
db_name.table10 table10b,
db_name.table10 table10c,
db_name.table11 table11,
db_name.table12 table12,
db_name.table13 table13,
db_name.table14 table14,
db_name.table15 table15,
db_name.table16 table16
WHERE [...]

table8.attr19=table7a.attr80 AND
table8.attr19=table13.attr20 AND
table8.attr4='keyword' AND
table9.attr10=table16.attr11 AND
table3b.attr19=table10c.attr18 AND
table3b.attr22=table12.attr63 AND
table3b.attr66='keyword' AND
table10a.attr54=table7a.attr8 AND
table10a.attr70=table10c.attr10 AND
table10a.attr16=table4d.attr11 AND
table4c.attr99='keyword' AND
table4c.attr1='keyword' AND

table16.attr28=table11.attr56 AND
table16.attr16=table10b.attr78 AND
table16.attr5=table14.attr56 AND
table4e.attr34 IN ('keyword') AND
table4e.attr48 IN ('keyword') AND
table4f.attr89=table5b.attr7 AND
table4f.attr45 IN ('keyword') AND
table4f.attr1='keyword' AND
table10c.attr2=table4e.attr19 AND
(table10c.attr78=table12.attr56 OR
(table10c.attr55 IS NULL AND
table12.attr17 IS NULL))
```

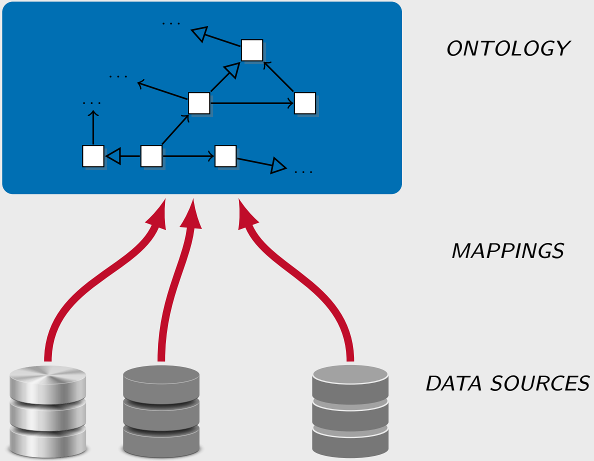
Need for Abstraction

We need to facilitate access to Data

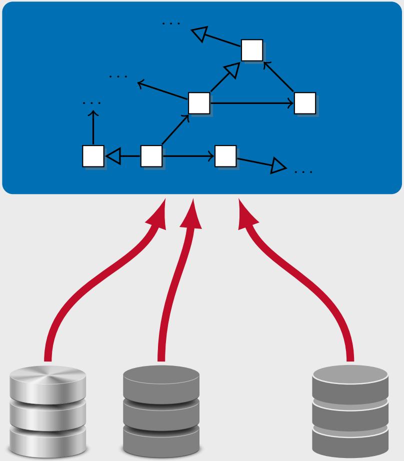
- by abstracting away from how the data is stored, and
- by making use of high level views on the data, so called **ontologies**.

Outline

Ontology Based Data Access Framework



Ontology Based Data Access Framework



ONTOLOGY
=
global vocabulary
+
conceptual view

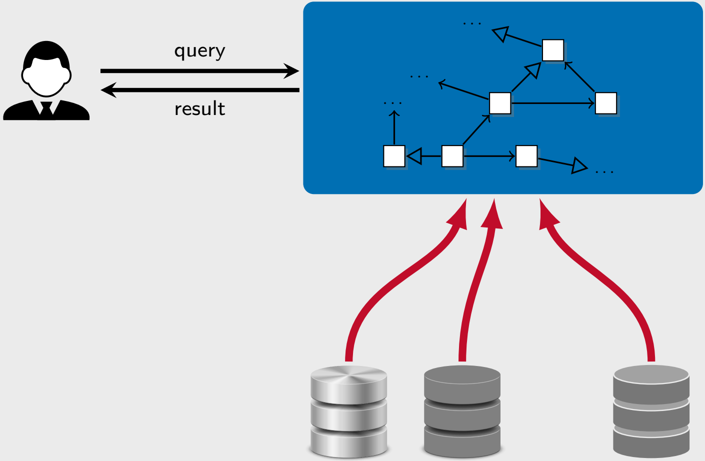
MAPPINGS

how to populate the ontology

DATA SOURCES

external and heterogeneous

Ontology Based Data Access Framework



ONTOLOGY
=
global vocabulary
+
conceptual view

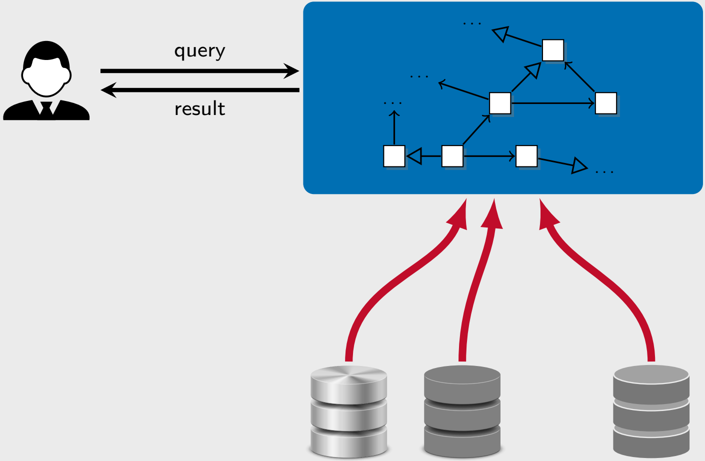
MAPPINGS

how to populate the ontology

DATA SOURCES

external and heterogeneous

Ontology Based Data Access Framework



ONTOLOGY
=
global vocabulary
+
conceptual view

MAPPINGS

how to populate the ontology

DATA SOURCES

external and heterogeneous

Logical transparency in accessing data:

- does not know where and how **data** is stored;
- can only see a **conceptual view** of **data**.

Questions to be Addressed in OBDA

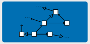
- What is the right ontology language?
- What is the right query language?
- What is the right mapping language?
- What are the available tools?

Outline

Databases, Knowledge Bases and Ontologies

A **data**base is an organized collection of **data** 
that can be accessed by a computer program to quickly **select** pieces of data.

A **knowledge** base is an organized collection of

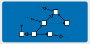
- **axioms**  (formalization of how things are in the world), and of
- **data**  (facts about the world),

that can be accessed by a computer program to (quickly) **reason** about data.

Databases, Knowledge Bases and Ontologies

A **data**base is an organized collection of **data**
that can be accessed by a computer program to quickly **select** pieces of data.

A **knowledge** base is an organized collection of

- **axioms**  (formalization of how things are in the world), and of
- **data** (facts about the world),

that can be accessed by a computer program to (quickly) **reason** about data.

Axioms:
Lung Cancer is Cancer.
Every patient must have a name, age and SSN.

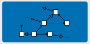
Data:
Mary is a patient with
Non Small Cell Lung Cancer on stage 2.

Databases, Knowledge Bases and Ontologies

A **data**base is an organized collection of **data** 
that can be accessed by a computer program to quickly **select** pieces of data.

- Database schema also includes **integrity constraints** (≈ **axioms**)
- The **data** is assumed to be **complete** w.r.t. them: **Closed World Assumption**

A **knowledge** base is an organized collection of

- **axioms**  (formalization of how things are in the world), and of
- **data**  (facts about the world),

that can be accessed by a computer program to (quickly) **reason** about data.

- The **data** may be **incomplete** w.r.t. the **axioms**: **Open World Assumption**

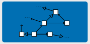
Axioms:	**Data**:
Lung Cancer is Cancer.	Mary is a patient with
Every patient must have a name, age and SSN.	Non Small Cell Lung Cancer on stage 2.

Databases, Knowledge Bases and Ontologies

A **data**base is an organized collection of **data** 
that can be accessed by a computer program to quickly **select** pieces of data.

- Database schema also includes **integrity constraints** (≈ **axioms**)
- The **data** is assumed to be **complete** w.r.t. them: **Closed World Assumption**

A **knowledge** base is an organized collection of

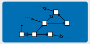
- **axioms**  (formalization of how things are in the world), and of
- **data**  (facts about the world),

that can be accessed by a computer program to (quickly) **reason** about data.

- The **data** may be **incomplete** w.r.t. the **axioms**: **Open World Assumption**

Axioms:
Lung Cancer is Cancer.
Every patient must have a name, age and SSN.

Data:
Mary is a patient with
Non Small Cell Lung Cancer on stage 2.

In this tutorial, by **ontology** we mean the **axioms**  part of knowledge bases.
It is also referred to as intentional level, conceptual schema, domain structure.

Knowledge Representation Formalisms

What is a **formalization** of the world?

Knowledge Representation Formalisms

What is a formalization of the world?

A representation of the information about the world in a **formal language**.

Knowledge Representation Formalisms

What is a formalization of the world?

A representation of the information about the world in a **formal language**.

Some formal logical languages:

- First-Order Logic
- Predicate Logic
- Modal Logics
- Description Logics
- Datalog
- DDL (in Databases)

Knowledge Representation Formalisms

What is a formalization of the world?

A representation of the information about the world in a **formal language**.

Some formal logical languages:

- First-Order Logic
- Predicate Logic
- Modal Logics
- Description Logics
- Datalog
- DDL (in Databases)

What are the requirements to a formal language?

1. should be **decidable**
2. should be **easy** to decide
3. should be **expressive** enough
4. should be **readable** not only by machines, but also by humans

Knowledge Representation Formalisms

What is a formalization of the world?

A representation of the information about the world in a **formal language**.

Some formal logical languages:

- First-Order Logic $\Leftarrow$ **UNDECIDABLE**
- Predicate Logic
- Modal Logics
- Description Logics
- Datalog
- DDL (in Databases)

What are the requirements to a formal language?

1. should be **decidable**
2. should be **easy** to decide
3. should be **expressive** enough
4. should be **readable** not only by machines, but also by humans

Knowledge Representation Formalisms

What is a formalization of the world?

A representation of the information about the world in a **formal language**.

Some formal logical languages:

- First-Order Logic $\Leftarrow$ **UNDECIDABLE**
- Predicate Logic
- Modal Logics
- **Description Logics**
- Datalog
- DDL (in Databases)

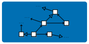
What are the requirements to a formal language?

1. should be **decidable**
2. should be **easy** to decide
3. should be **expressive** enough
4. should be **readable** not only by machines, but also by humans

Description Logics (DLs)

is a family of Knowledge Representation formalisms, fragments of First-Order Logic.

In a description logic knowledge base,

- the **TBox** $\mathcal{T}$ encodes **axioms** ,
- the **ABox** $\mathcal{A}$ encodes **data** .

Thus, a knowledge base is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$.

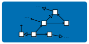
A DL KB talks about

- **individuals** (objects): mary, neoplasm1;
- **concepts** (classes): *Patient*, *Cancer*;
- **roles** (properties): *hasNeoplasm*, *hasSSN*.

Description Logics (DLs)

is a family of Knowledge Representation formalisms, fragments of First-Order Logic.

In a description logic knowledge base,

- the **TBox** $\mathcal{T}$ encodes **axioms** ,
- the **ABox** $\mathcal{A}$ encodes **data**.

Thus, a knowledge base is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$.

A DL KB talks about

- **individuals** (objects): mary, neoplasm1;
- **concepts** (classes): *Patient*, *Cancer*;
- **roles** (properties): *hasNeoplasm*, *hasSSN*.

$$\begin{aligned} \mathit{LungCancer} &\sqsubseteq \mathit{Cancer} \\ \mathit{NSCLCancer} &\sqsubseteq \mathit{LungCancer} \\ \mathit{Patient} &\sqsubseteq \exists \mathit{hasSSN} \\ &\dots \end{aligned}$$

$\mathit{Patient}(\mathrm{mary})$

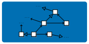
$\mathit{hasNeoplasm}(\mathit{mary}, \mathit{neoplasm}1)$

$\mathit{NSCLCancer}(\mathit{neoplasm}1)$

Description Logics (DLs)

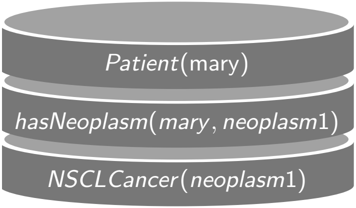
is a family of Knowledge Representation formalisms, fragments of First-Order Logic.

In a description logic knowledge base,

- the **TBox** $\mathcal{T}$ encodes **axioms** ,
- the **ABox** $\mathcal{A}$ encodes **data** .

Thus, a knowledge base is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$.

$$
\begin{aligned}
\mathit{LungCancer} &\sqsubseteq \mathit{Cancer}\\
\mathit{NSCLCancer} &\sqsubseteq \mathit{LungCancer}\\
\mathit{Patient} &\sqsubseteq \exists \mathit{hasSSN}\\
&\dots
\end{aligned}
$$



A DL KB talks about

- **individuals** (objects): mary, neoplasm1;
- **concepts** (classes): *Patient*, *Cancer*;
- **roles** (properties): *hasNeoplasm*, *hasSSN*.

Why to use description logics?

Description Logics (DLs)

is a family of Knowledge Representation formalisms, fragments of First-Order Logic.

In a description logic knowledge base,

- the **TBox** $\mathcal{T}$ encodes **axioms**,
- the **ABox** $\mathcal{A}$ encodes **data**.

Thus, a knowledge base is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$.

A DL KB talks about

- **individuals** (objects): mary, neoplasm1;
- **concepts** (classes): *Patient*, *Cancer*;
- **roles** (properties): *hasNeoplasm*, *hasSSN*.

$$\begin{aligned} LungCancer &\sqsubseteq Cancer \\ NSCLCancer &\sqsubseteq LungCancer \\ Patient &\sqsubseteq \exists hasSSN \\ &\cdots \end{aligned}$$

$Patient(\mathrm{mary})$

$hasNeoplasm(mary, neoplasm1)$

$NSCLCancer(neoplasm1)$

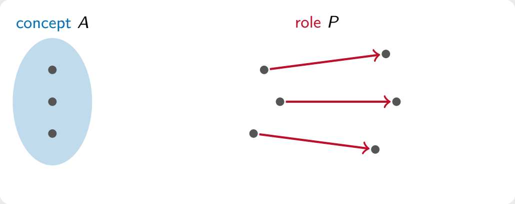
Why to use description logics?

- good trade-off between expressivity and computational complexity
- efficient reasoning algorithms
- concise syntax

DL-Lite$_{\mathcal{R}}$ at the Basis of OWL 2 QL

DL-Lite$_{\mathcal{R}}$ provides the basis for the **OWL 2 QL** profile of **OWL 2**.
It is specifically tailored to efficient **query answering**.

In *DL-Lite*$_{\mathcal{R}}$ we deal with the following **concepts** and **roles**:

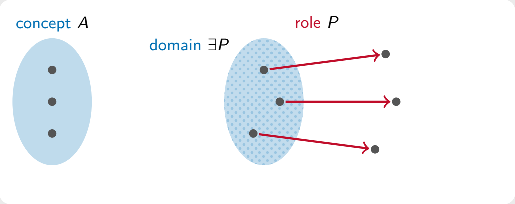


DL-Lite$_{\mathcal{R}}$ at the Basis of OWL 2 QL

DL-Lite$_{\mathcal{R}}$ provides the basis for the **OWL 2 QL** profile of **OWL 2**.

It is specifically tailored to efficient **query answering**.

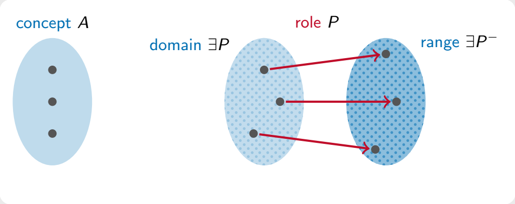
In *DL-Lite*$_{\mathcal{R}}$ we deal with the following **concepts** and **roles**:



$DL\text{-}Lite_{\mathcal{R}}$ at the Basis of OWL 2 QL

$DL\text{-}Lite_{\mathcal{R}}$ provides the basis for the **OWL 2 QL** profile of **OWL 2**.
It is specifically tailored to efficient **query answering**.

In $DL\text{-}Lite_{\mathcal{R}}$ we deal with the following **concepts** and **roles**:

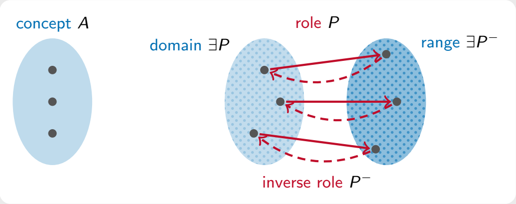


DL-Lite$_\mathcal{R}$ at the Basis of OWL 2 QL

DL-Lite$_\mathcal{R}$ provides the basis for the **OWL 2 QL** profile of **OWL 2**.

It is specifically tailored to efficient **query answering**.

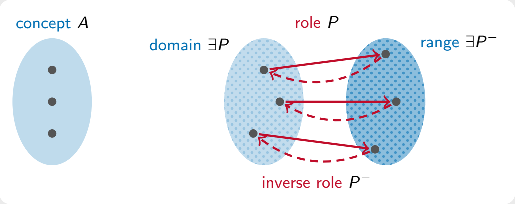
In *DL-Lite*$_\mathcal{R}$ we deal with the following **concepts** and **roles**:



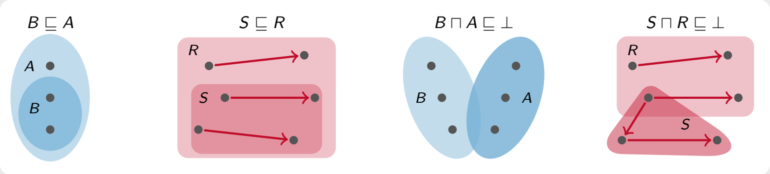
DL-Lite$_\mathcal{R}$ at the Basis of OWL 2 QL

DL-Lite$_\mathcal{R}$ provides the basis for the **OWL 2 QL** profile of **OWL 2**.
It is specifically tailored to efficient **query answering**.

In *DL-Lite*$_\mathcal{R}$ we deal with the following **concepts** and **roles**:



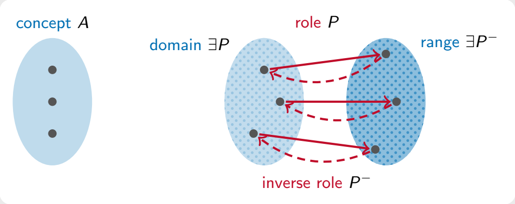
A *DL-Lite*$_\mathcal{R}$ TBox is a finite set of inclusion and disjointness axioms.



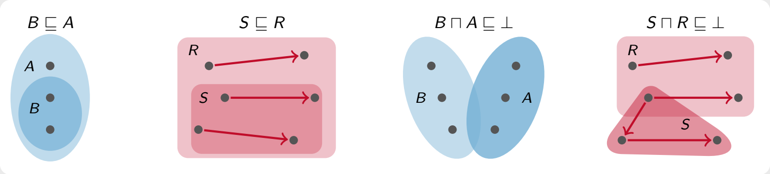
DL-Lite$_\mathcal{R}$ at the Basis of OWL 2 QL

DL-Lite$_\mathcal{R}$ provides the basis for the **OWL 2 QL** profile of **OWL 2**.
It is specifically tailored to efficient **query answering**.

In *DL-Lite*$_\mathcal{R}$ we deal with the following **concepts** and **roles**:



A *DL-Lite*$_\mathcal{R}$ TBox is a finite set of inclusion and disjointness axioms.



A *DL-Lite*$_\mathcal{R}$ ABox is a finite set of facts of the form $A(c)$, $P(c, d)$.

$\mathit{DL\text{-}Lite}_{\mathcal{R}}$ in the OWL 2 and RDF Syntaxes

$\mathit{DL\text{-}Lite}_{\mathcal{R}}$	OWL 2 Functional Style Syntax	RDF/Turtle
$\mathit{LungCancer} \sqsubseteq \mathit{Cancer}$	**SubClassOf**(LungCancer, Cancer)	LungCancer **rdfs:subClassOf** Cancer .
$\mathit{Patient} \sqsubseteq \exists \mathit{hasSSN}$	**SubClassOf**(Patient, **ObjectSomeValuesFrom**(hasSSN, **owl:Thing**))	Patient **rdfs:subClassOf** $\_b_1$. $\_b_1$ **rdf:type owl:Restriction** ; **owl:onProperty** hasSSN ; **owl:someValuesFrom owl:Thing** .
$\exists \mathit{hasNeoplasm}^{-} \sqsubseteq \mathit{Neoplasm}$	**SubClassOf**(**ObjectSomeValuesFrom**(**ObjectInverseOf**(hasNeoplasm), **owl:Thing**), Neoplasm)	$\_b_2$ **rdfs:subClassOf**, Neoplasm ; **rdf:type owl:Restriction** ; **owl:onProperty** $\_b_{22}$; **owl:someValuesFrom owl:Thing** . $\_b_{22}$ **owl:inverseOf** hasNeoplasm .
$\mathit{hasNeoplasm} \sqsubseteq \mathit{hasCondition}$	**SubObjectPropertyOf**(hasNeoplasm, hasCondition)	hasNeoplasm **rdfs:subPropertyOf** hasCondition .
$\mathit{Patient}(\mathrm{mary})$	**ClassAssertion**(Patient, mary)	mary **rdf:type** Patient .
$\mathit{hasNeoplasm}(\mathrm{mary}, \mathrm{neoplasm1})$	**ObjectPropertyAssertion**(hasNeoplasm, mary, neoplasm1)	mary hasNeoplasm neoplasm1 .

Outline

Query Languages

Intuitively, a **query** is a question:

- have all patients diagnosed with cancer received chemotherapy?
- give me all patients diagnosed with cancer who have not received chemotherapy.
- give me all patients diagnosed with cancer.
- give me all patients diagnosed with lung or skin cancer.

Standard query languages:

- **First-Order Queries** ($\approx$ SQL)

$$q() = \forall x, d \Big((Patient(x) \land hasCondition(x, d) \land Cancer(d)) \rightarrow (\exists t.hadTreat(x, t) \land Chemo(t)) \Big)$$

$$q(x) = Patient(x) \land (\exists d.hasCondition(x, d) \land Cancer(d)) \land (\neg \exists t.hadTreat(x, t) \land Chemo(t))$$

- **Conjunctive Queries** (Select-Project-Join)

$$q(x) = \exists d.Patient(x) \land hasCondition(x, d) \land Cancer(d)$$

- **Unions of Conjunctive Queries** (Union-of-Select-Project-Join)

$$q(x) = \begin{array}{c} \exists d.Patient(x) \land hasCondition(x, d) \land LungCancer(d) \\ \lor \\ \exists d.Patient(x) \land hasCondition(x, d) \land SkinCancer(d) \end{array}$$

Query Answering in Databases and in *DL-Lite*$_{\mathcal{R}}$

The Query Answering Problem: Is it true that **c** is an answer to a query q?

Query Answering in Databases and in *DL-Lite*$_{\mathcal{R}}$

The Query Answering Problem: Is it true that **c** is an answer to a query q?

- is **Mary** a patient?
- is **Mary** a cancer patient?
- does **Mary** have lung or skin cancer?

Query Answering in Databases and in *DL-Lite*$_{\mathcal{R}}$

The Query Answering Problem: Is it true that **c** is an answer to a query q?

- is **Mary** a patient?
- is **Mary** a cancer patient?
- does **Mary** have lung or skin cancer?

Query Answering in Databases

Data is complete.

- Query answering amounts to **query evaluation**, which is computationally **easy**.

Query Answering in Knowledge Bases

Data is incomplete, so the **axioms** must be taken into account.

- Query answering amounts to **logical inference**, which is computationally **costly**.

Query Evaluation Vs. Query Answering

Axioms:

$LungCancer \sqsubseteq Cancer$ $\qquad$ $Patient \sqsubseteq \exists hasSSN$

$NSCLCancer \sqsubseteq LungCancer$ $\qquad$ $hasNeoplasm \sqsubseteq hasCondition$

Queries:

q_1: Is Mary a patient?

q_2: Does Mary have neoplasm that is NSCLCancer?

q_3: Does Mary have neoplasm that is Cancer?

q_4: Does Mary have neoplasm that is SkinCancer?

q_5: Does Mary have condition that is NSCLCancer?

q_6: Does Mary have SSN?

Query Evaluation Vs. Query Answering

Axioms:

$LungCancer \sqsubseteq Cancer$ $\quad$ $Patient \sqsubseteq \exists hasSSN$

$NSCLCancer \sqsubseteq LungCancer$ $\quad$ $hasNeoplasm \sqsubseteq hasCondition$

Queries:

q_1: Is Mary a patient?

q_2: Does Mary have neoplasm that is NSCLCancer?

q_3: Does Mary have neoplasm that is Cancer?

q_4: Does Mary have neoplasm that is SkinCancer?

q_5: Does Mary have condition that is NSCLCancer?

q_6: Does Mary have SSN?

Data: (CWA)

q_1	q_2	q_3	q_4	q_5	q_6
Yes	Yes	No	No	No	No

Query Evaluation Vs. Query Answering

Axioms:

$$LungCancer \sqsubseteq Cancer \qquad Patient \sqsubseteq \exists hasSSN$$
$$NSCLCancer \sqsubseteq LungCancer \qquad hasNeoplasm \sqsubseteq hasCondition$$

Queries:

q_1: Is Mary a patient?
q_2: Does Mary have neoplasm that is NSCLCancer?
q_3: Does Mary have neoplasm that is Cancer?
q_4: Does Mary have neoplasm that is SkinCancer?
q_5: Does Mary have condition that is NSCLCancer?
q_6: Does Mary have SSN?

Data: (CWA)

q_1	q_2	q_3	q_4	q_5	q_6
Yes	Yes	No	No	No	No

Data

q_1	q_2	q_3	q_4	q_5	q_6
Yes	Yes				

Query Evaluation Vs. Query Answering

Axioms:

$$LungCancer \sqsubseteq Cancer \qquad Patient \sqsubseteq \exists hasSSN$$
$$NSCLCancer \sqsubseteq LungCancer \qquad hasNeoplasm \sqsubseteq hasCondition$$

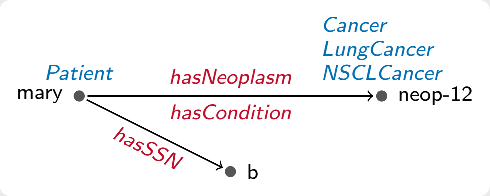
Queries:

- q_1: Is Mary a patient?
- q_2: Does Mary have neoplasm that is NSCLCancer?
- q_3: Does Mary have neoplasm that is Cancer?
- q_4: Does Mary have neoplasm that is SkinCancer?
- q_5: Does Mary have condition that is NSCLCancer?
- q_6: Does Mary have SSN?

Data: (CWA)

q_1	q_2	q_3	q_4	q_5	q_6
Yes	Yes	No	No	No	No

Data + **Axioms** = **inferred data**



q_1	q_2	q_3	q_4	q_5	q_6
Yes	Yes				

Query Evaluation Vs. Query Answering

Axioms:

$$\mathit{LungCancer} \sqsubseteq \mathit{Cancer} \qquad \mathit{Patient} \sqsubseteq \exists \mathit{hasSSN}$$
$$\mathit{NSCLCancer} \sqsubseteq \mathit{LungCancer} \qquad \mathit{hasNeoplasm} \sqsubseteq \mathit{hasCondition}$$

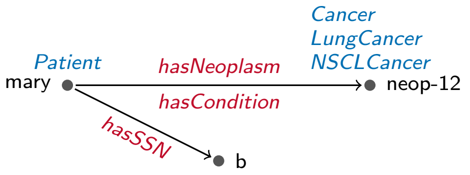
Queries:

q_1: Is Mary a patient?
q_2: Does Mary have neoplasm that is NSCLCancer?
q_3: Does Mary have neoplasm that is Cancer?
q_4: Does Mary have neoplasm that is SkinCancer?
q_5: Does Mary have condition that is NSCLCancer?
q_6: Does Mary have SSN?

Data: (CWA)

q_1	q_2	q_3	q_4	q_5	q_6
Yes	Yes	No	No	No	No

Data + **Axioms** = **inferred data**



q_1	q_2	q_3	q_4	q_5	q_6
Yes	Yes	Yes	No	Yes	Yes

Complexity of Query Answering in Databases and in *DL-Lite*$_\mathcal{R}$

	Databases	*DL-Lite*$_\mathcal{R}$ Knowledge Bases
FO	**PSpace-complete** in **LogSpace**	**undecidable**
CQs & UCQs	**NP-complete** in **LogSpace**	**NP-complete** in **LogSpace** in **PTime**

LogSpace $\subseteq$ **PTime** $\subseteq$ **NP** $\subseteq$ **PSpace**

Combined complexity: the **query**, the **data** (and the **TBox**) make the input.

Data complexity: the **data** is the only input (the **query** and the **TBox** are fixed).

TBox complexity: the **TBox** is the only input.

Complexity of Query Answering in Databases and in *DL-Lite*$_\mathcal{R}$

	Databases	*DL-Lite*$_\mathcal{R}$ Knowledge Bases
FO	**PSpace-complete** in **LogSpace**	**undecidable**
CQs & UCQs	**NP-complete** in **LogSpace**	**NP-complete** in **LogSpace** in **PTime**

LogSpace $\subseteq$ **PTime** $\subseteq$ **NP** $\subseteq$ **PSpace**

Combined complexity: the **query**, the **data** (and the **TBox**) make the input.

Data complexity: the **data** is the only input (the **query** and the **TBox** are fixed).

TBox complexity: the **TBox** is the only input.

What this table tells us

- Databases are able to perform well for queries of small size.
- Databases are able to perform well even if the data is huge.
- Answering UCQs over *DL-Lite*$_\mathcal{R}$ KBs can be reduced to QE over Databases.

Outline

Connecting Inherently Different Domains

Problem

- **Data sources** store **values** from concrete domains, such as strings, integers, etc.
- Instead, instances of concepts and roles in an **ontology** are **(abstract) objects**.

Connecting Inherently Different Domains

Problem

- **Data sources** store **values** from concrete domains, such as strings, integers, etc.
- Instead, instances of concepts and roles in an **ontology** are **(abstract) objects**.

Hence, **mappings** should specify how to construct

- from the concrete values in the **data sources**
- the (abstract) objects that populate the ABox of the **ontology**.

Connecting Inherently Different Domains

Problem

- **Data sources** store **values** from concrete domains, such as strings, integers, etc.
- Instead, instances of concepts and roles in an **ontology** are **(abstract) objects**.

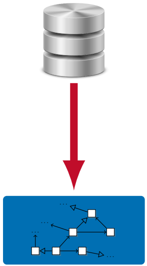
Hence, **mappings** should specify how to construct

- from the concrete values in the **data sources**
- the (abstract) objects that populate the ABox of the **ontology**.

This is done by using values (normally, keys) to create object identifiers
(think of **Uniform Resource Identifiers, URIs**).

Mapping, Intuitively

A **mapping** has two parts:



Source Query: An SQL query over the data source

Target Query: An ABox template of the ontology
$\approx$ a CQ without existentially quantified variables

Mapping, Intuitively

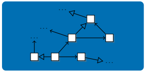
We assume a cancer patient database containing the table PATIENT:

PatientID	Name	Type	Stage
12	Mary	true	2
34	John	false	7

- Type is: true for Non-Small Cell Lung Cancer, and false for Small Cell Lung Cancer
- Stage is: 1-6 for NSCLC stages I,II,III,IIIa,IIIb,IV, and 7-8 for SCLC stages Limited,Extensive

A **mapping** has two parts:

Source Query: An SQL query over the data source



Target Query: An ABox template of the ontology

$\approx$ a CQ without existentially quantified variables

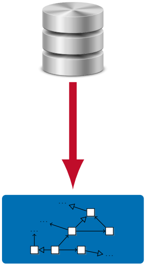
Mapping, Intuitively

We assume a cancer patient database containing the table PATIENT:

PatientID	Name	Type	Stage
12	Mary	true	2
34	John	false	7

- Type is: true for Non-Small Cell Lung Cancer, and false for Small Cell Lung Cancer
- Stage is: 1-6 for NSCLC stages I,II,III,IIIa,IIIb,IV, and 7-8 for SCLC stages Limited,Extensive

A **mapping** has two parts:



Source Query: An SQL query over the data source

select $\star$ from PATIENT where Type=true

Patient(pat-{PatientID})
hasNeoplasm(pat-{PatientID}, neop-{PatientID})
NSCLCancer(neop-{PatientID})

Target Query: An ABox template of the ontology

$\approx$ a CQ without existentially quantified variables

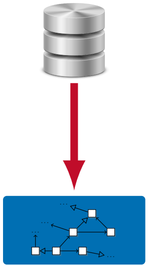
Mapping, Intuitively

We assume a cancer patient database containing the table PATIENT:

PatientID	Name	Type	Stage
12	Mary	true	2
34	John	false	7

- Type is: true for Non-Small Cell Lung Cancer, and false for Small Cell Lung Cancer
- Stage is: 1-6 for NSCLC stages I,II,III,IIIa,IIIb,IV, and 7-8 for SCLC stages Limited,Extensive

A **mapping** has two parts:



Source Query: An SQL query over the data source

select $\star$ from PATIENT where Type=true

Patient(pat-{PatientID})
hasNeoplasm(pat-{PatientID}, neop-{PatientID})
NSCLCancer(neop-{PatientID})

Target Query: An ABox template of the ontology
$\approx$ a CQ without existentially quantified variables

If we were to materialize, we would obtain the following ABox from our database:

Patient pat-12 —*hasNeoplasm*→ neop-12 *NSCLCancer*

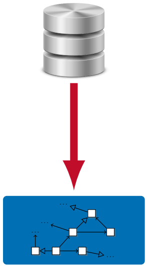
Mapping, Intuitively

We assume a cancer patient database containing the table PATIENT:

PatientID	Name	Type	Stage
12	Mary	true	2
34	John	false	7

- Type is: true for Non-Small Cell Lung Cancer, and false for Small Cell Lung Cancer
- Stage is: 1-6 for NSCLC stages I,II,III,IIIa,IIIb,IV, and 7-8 for SCLC stages Limited,Extensive

A **mapping** has two parts:



Source Query: An SQL query over the data source

select $\star$ from PATIENT where Type=true

Patient(pat-{PatientID})
hasNeoplasm(pat-{PatientID}, neop-{PatientID})
NSCLCancer(neop-{PatientID})

Target Query: An ABox template of the ontology
$\approx$ a CQ without existentially quantified variables

If we were to materialize, we would obtain the following ABox from our database:

Patient pat-12 $\xrightarrow{\textit{hasNeoplasm}}$ neop-12 *NSCLCancer*

However, the ABox is only **virtual**, and the objects are not materialized.

The Standard Mapping Languages

R2RML is an RDB to RDF mapping language, a W3C recommendation
`http://www.w3.org/TR/r2rml/`

Direct mapping is a default RDB to RDF mapping language, a W3C recommendation
`http://www.w3.org/TR/rdb-direct-mapping/`

Outline

Approaches

How does OBDA actually work?

- We want to keep the Data in the data sources.
- We embed reasoning either into the query or into the Data.

Query Rewriting	Combined Approach	Materialization

Approaches

How does OBDA actually work?

- We want to keep the Data in the data sources.
- We embed reasoning either into the query or into the Data.

Query Rewriting — Combined Approach — Materialization

In practice:

Query Rewriting

1. the data is not manipulated
2. the axioms are embedded into the query
3. the rewritten query is translated to SQL
4. the translated query is answered by the DBMS

Combined Approach

1. the data is partially completed (materialized) w.r.t. the axioms
2. the query is rewritten w.r.t. the axioms
3. the rewritten query is translated to SQL
4. the translated query is answered by the DBMS

Approaches

How does OBDA actually work?

- We want to keep the Data in the data sources.
- We embed reasoning either into the query or into the Data.

Query Rewriting — Combined Approach — Materialization

In practice:

Query Rewriting

1. the data is not manipulated
2. the axioms are embedded into the query
3. the rewritten query is translated to SQL **of strange shape and quite big size**
4. the translated query is answered by the DBMS

Combined Approach

1. the data is partially completed (materialized) w.r.t. the axioms
2. the query is rewritten w.r.t. the axioms
3. the rewritten query is translated to SQL
4. the translated query is answered by the DBMS

Approaches

How does OBDA actually work?

- We want to keep the Data in the data sources.
- We embed reasoning either into the query or into the Data.

Query Rewriting — Combined Approach — Materialization

In practice:

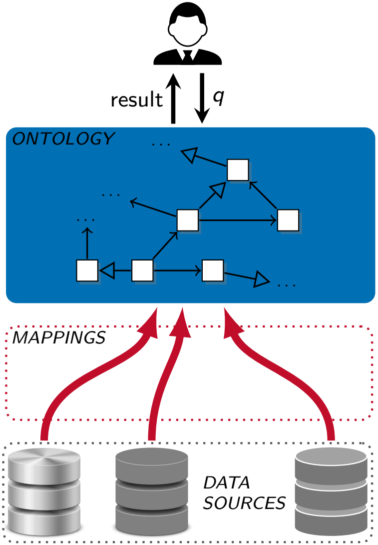
Query Rewriting

1. the data is not manipulated
2. the axioms are embedded into the query
3. the rewritten query is translated to SQL **of strange shape and quite big size**
4. the translated query is answered by the DBMS

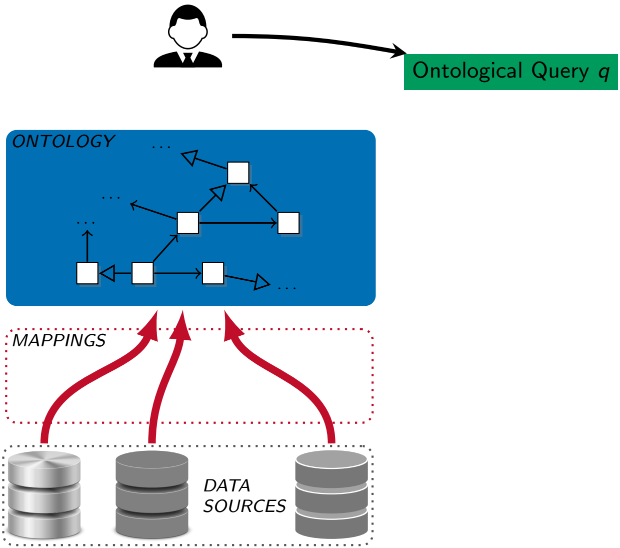
Combined Approach

1. the data is partially completed (materialized) w.r.t. the axioms
2. the query is rewritten w.r.t. the axioms
3. the rewritten query is translated to SQL **hopefully of small size**
4. the translated query is answered by the DBMS

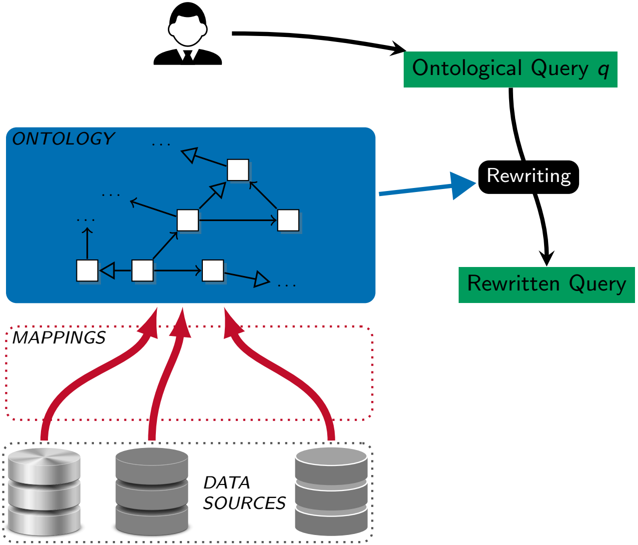
Query Answering by Query Rewriting



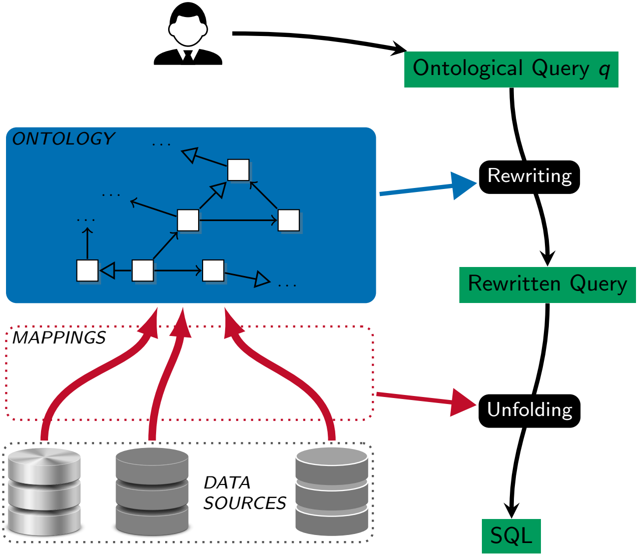
Query Answering by Query Rewriting



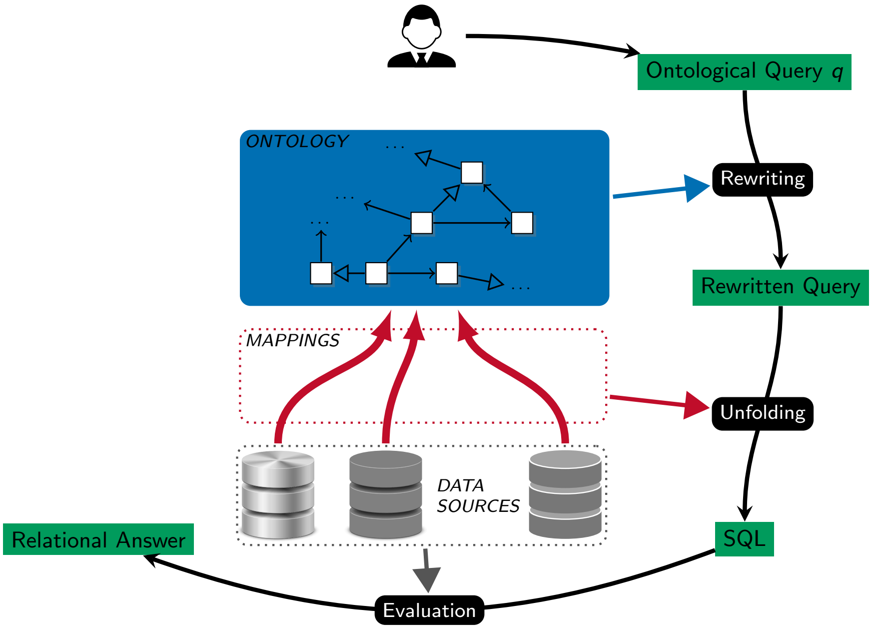
Query Answering by Query Rewriting



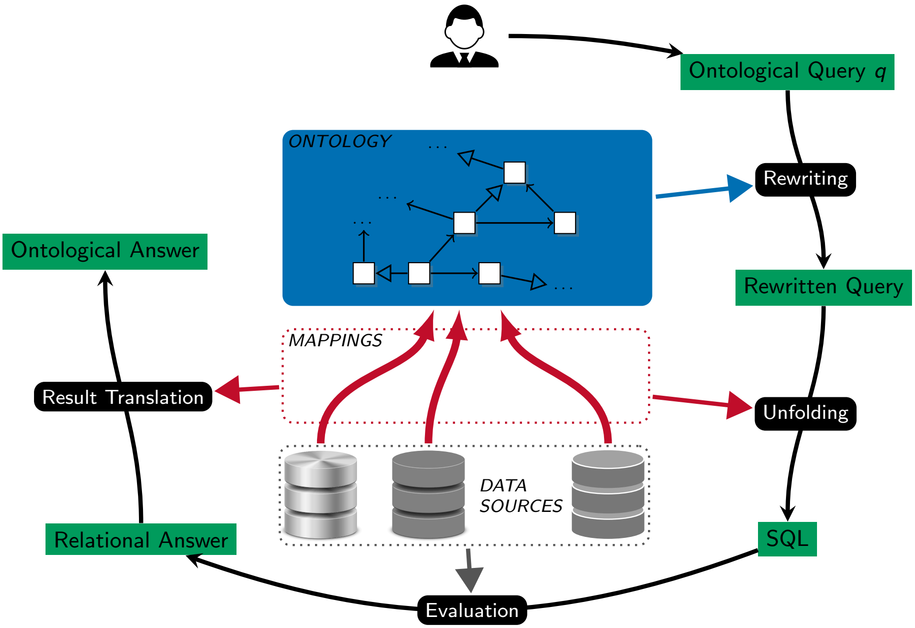
Query Answering by Query Rewriting



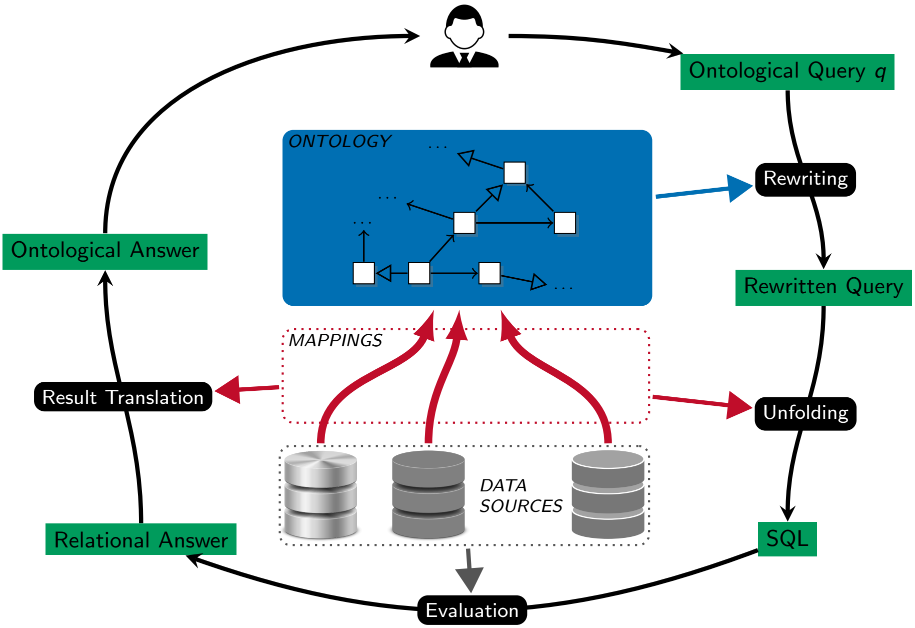
Query Answering by Query Rewriting



Query Answering by Query Rewriting



Query Answering by Query Rewriting



How Query Rewriting Works

C is an answer to a **query** q over a knowledge base $\langle \mathcal{T}, \mathcal{A} \rangle$

if and only if

C is an answer to the **rewritten query** $q_{\mathcal{T}}$ over the database $\mathcal{A}$

Query Rewriting Example

Axioms:

$$LungCancer \sqsubseteq Cancer \qquad Patient \sqsubseteq \exists hasSSN$$

$$NSCLCancer \sqsubseteq LungCancer \qquad hasNeoplasm \sqsubseteq hasCondition$$

Data:

$$\underset{\text{mary}}{Patient} \bullet \xrightarrow{\quad hasNeoplasm \quad} \bullet \underset{\text{neop-12}}{NSCLCancer}$$

Query: Does Mary have neoplasm that is Cancer?

$$q(x) = \exists y.hasNeoplasm(x, y) \land Cancer(y)$$

Query Rewriting Example

Axioms:
$$LungCancer \sqsubseteq Cancer \qquad Patient \sqsubseteq \exists hasSSN$$
$$NSCLCancer \sqsubseteq LungCancer \qquad hasNeoplasm \sqsubseteq hasCondition$$

Data:

$$\underset{mary}{Patient} \xrightarrow{hasNeoplasm} \underset{neop\text{-}12}{NSCLCancer}$$

Query: Does Mary have neoplasm that is Cancer?

$$q(x) = \exists y.hasNeoplasm(x, y) \wedge Cancer(y)$$

Rewritten Query:

$$\exists y.hasNeoplasm(x, y) \wedge Cancer(y)$$

$$q'(x) =$$

Query Rewriting Example

Axioms:

$$LungCancer \sqsubseteq Cancer \qquad Patient \sqsubseteq \exists hasSSN$$
$$NSCLCancer \sqsubseteq LungCancer \qquad hasNeoplasm \sqsubseteq hasCondition$$

Data:

$$\underset{mary}{Patient} \xrightarrow{hasNeoplasm} \underset{neop\text{-}12}{NSCLCancer}$$

Query: Does Mary have neoplasm that is Cancer?

$$q(x) = \exists y. hasNeoplasm(x, y) \wedge Cancer(y)$$

Rewritten Query:

$$\begin{aligned} q'(x) = {} & \exists y. hasNeoplasm(x, y) \wedge Cancer(y) \\ & \vee \\ & \exists y. hasNeoplasm(x, y) \wedge LungCancer(y) \end{aligned}$$

Query Rewriting Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad hasNeoplasm \sqsubseteq hasCondition$

Data:

$$\underset{\text{mary}}{Patient} \xrightarrow{hasNeoplasm} \underset{\text{neop-12}}{NSCLCancer}$$

Query: Does Mary have neoplasm that is Cancer?

$$q(x) = \exists y.hasNeoplasm(x,y) \wedge Cancer(y)$$

Rewritten Query:

$$q'(x) = \begin{array}{c} \exists y.hasNeoplasm(x,y) \wedge Cancer(y) \\ \vee \\ \exists y.hasNeoplasm(x,y) \wedge LungCancer(y) \\ \vee \\ \exists y.hasNeoplasm(x,y) \wedge NSCLCancer(y) \end{array}$$

Query Rewriting Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad hasNeoplasm \sqsubseteq hasCondition$

Data:

$$\underset{\text{mary}}{Patient} \xrightarrow{hasNeoplasm} \underset{\text{neop-12}}{NSCLCancer}$$

Query: Does Mary have neoplasm that is Cancer?

$$q(x) = \exists y.hasNeoplasm(x, y) \wedge Cancer(y)$$

Rewritten Query:

$$\begin{aligned} & \exists y.hasNeoplasm(x, y) \wedge Cancer(y) \\ & \qquad \vee \\ q'(x) = & \exists y.hasNeoplasm(x, y) \wedge LungCancer(y) \\ & \qquad \vee \\ & \exists y.hasNeoplasm(x, y) \wedge NSCLCancer(y) \end{aligned}$$

The answer is **Yes**: Mary is an answer to the query q' over **data** alone.

Query Rewriting Example

Axioms:
$$LungCancer \sqsubseteq Cancer \qquad Patient \sqsubseteq \exists hasSSN$$
$$NSCLCancer \sqsubseteq LungCancer \qquad hasNeoplasm \sqsubseteq hasCondition$$

Data:

$$\underset{mary}{\overset{Patient}{\bullet}} \xrightarrow{hasNeoplasm} \underset{neop\text{-}12}{\overset{NSCLCancer}{\bullet}}$$

Query: Does Mary have condition that is NSCLCancer?

$$q(x) = \exists y.hasCondition(x, y) \land NSCLCancer(y)$$

Query Rewriting Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad$ $Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad$ $hasNeoplasm \sqsubseteq hasCondition$

Data:

$$\begin{tikzcd} \underset{\text{mary}}{\mathit{Patient}} \arrow[r, "hasNeoplasm"] & \underset{\text{neop-12}}{\mathit{NSCLCancer}} \end{tikzcd}$$

Query: Does Mary have condition that is NSCLCancer?

$$q(x) = \exists y.hasCondition(x, y) \wedge NSCLCancer(y)$$

Rewritten Query:

$$q'(x) = \exists y.hasCondition(x, y) \wedge NSCLCancer(y)$$

Query Rewriting Example

Axioms:

$$LungCancer \sqsubseteq Cancer \qquad Patient \sqsubseteq \exists hasSSN$$
$$NSCLCancer \sqsubseteq LungCancer \qquad hasNeoplasm \sqsubseteq hasCondition$$

Data:

$$\underset{mary}{Patient} \xrightarrow{hasNeoplasm} \underset{neop\text{-}12}{NSCLCancer}$$

Query: Does Mary have condition that is NSCLCancer?

$$q(x) = \exists y.hasCondition(x, y) \wedge NSCLCancer(y)$$

Rewritten Query:

$$q'(x) = \begin{array}{c} \exists y.hasCondition(x, y) \wedge NSCLCancer(y) \\ \vee \\ \exists y.hasNeoplasm(x, y) \wedge NSCLCancer(y) \end{array}$$

Query Rewriting Example

Axioms:

$$LungCancer \sqsubseteq Cancer \qquad Patient \sqsubseteq \exists hasSSN$$
$$NSCLCancer \sqsubseteq LungCancer \qquad hasNeoplasm \sqsubseteq hasCondition$$

Data:

$$\underset{mary}{Patient} \bullet \xrightarrow{hasNeoplasm} \bullet \underset{neop\text{-}12}{NSCLCancer}$$

Query: Does Mary have condition that is NSCLCancer?

$$q(x) = \exists y.hasCondition(x,y) \wedge NSCLCancer(y)$$

Rewritten Query:

$$q'(x) = \begin{array}{c} \exists y.hasCondition(x,y) \wedge NSCLCancer(y) \\ \vee \\ \exists y.hasNeoplasm(x,y) \wedge NSCLCancer(y) \end{array}$$

The answer is **Yes**: Mary is an answer to the query q' over **data** alone.

Query Rewriting Example

Axioms:

$$LungCancer \sqsubseteq Cancer \qquad Patient \sqsubseteq \exists hasSSN$$
$$NSCLCancer \sqsubseteq LungCancer \qquad hasNeoplasm \sqsubseteq hasCondition$$

Data:

$$\underset{mary}{\mathit{Patient}} \xrightarrow{\mathit{hasNeoplasm}} \underset{neop\text{-}12}{\mathit{NSCLCancer}}$$

Query: Does Mary have SSN?

$$q(x) = \exists y.hasSSN(x, y)$$

Query Rewriting Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad hasNeoplasm \sqsubseteq hasCondition$

Data:

$$\underset{\text{mary}}{\overset{\mathit{Patient}}{\bullet}} \xrightarrow{\mathit{hasNeoplasm}} \underset{\text{neop-12}}{\overset{\mathit{NSCLCancer}}{\bullet}}$$

Query: Does Mary have SSN?

$$q(x) = \exists y.hasSSN(x, y)$$

Rewritten Query:

$$q'(x) = \exists y.hasSSN(x, y)$$

Query Rewriting Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad hasNeoplasm \sqsubseteq hasCondition$

Data:

$$\underset{\text{mary}}{Patient} \xrightarrow{hasNeoplasm} \underset{\text{neop-12}}{NSCLCancer}$$

Query: Does Mary have SSN?

$$q(x) = \exists y.hasSSN(x, y)$$

Rewritten Query:

$$q'(x) = \begin{array}{c} \exists y.hasSSN(x, y) \\ \vee \\ Patient(x) \end{array}$$

Query Rewriting Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad$ $Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad$ $hasNeoplasm \sqsubseteq hasCondition$

Data:

$$\underset{\text{mary}}{\overset{Patient}{\bullet}} \xrightarrow{\quad hasNeoplasm \quad} \underset{\text{neop-12}}{\overset{NSCLCancer}{\bullet}}$$

Query: Does Mary have SSN?

$$q(x) = \exists y.hasSSN(x, y)$$

Rewritten Query:

$$q'(x) = \begin{array}{c} \exists y.hasSSN(x, y) \\ \vee \\ Patient(x) \end{array}$$

The answer is **Yes**: Mary is an answer to the query q' over **data** alone.

Query Rewriting Practical Issues

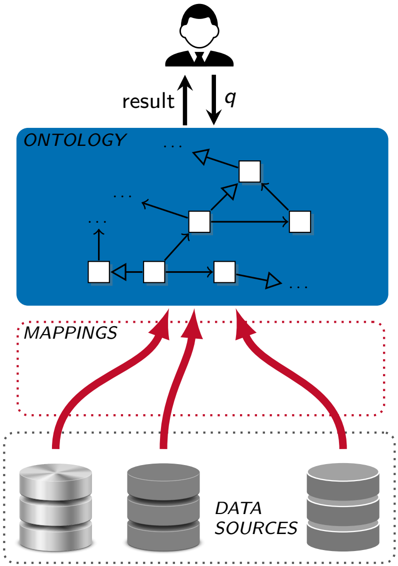
- The rewritten query can be huge: **exponential** in the size of the original query.
- Databases are **very bad** at evaluating big queries.

How to deal with that?

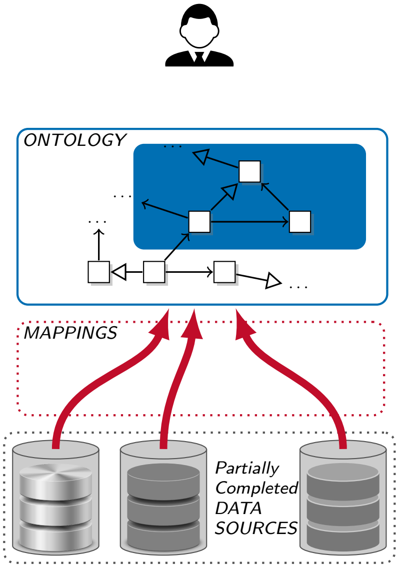
Employ various optimization techniques

- reduce the number of joins
- reduce the number of CQs in the big Union of CQs
- massage the query to a shape optimal for the database engine

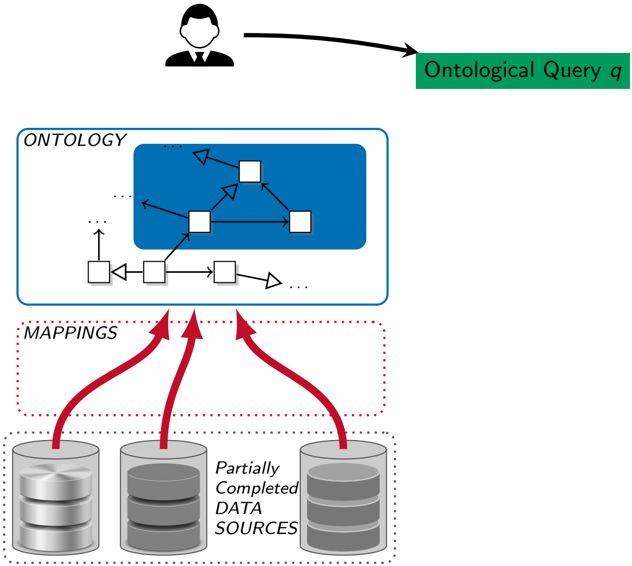
Query Answering by Combined Approach



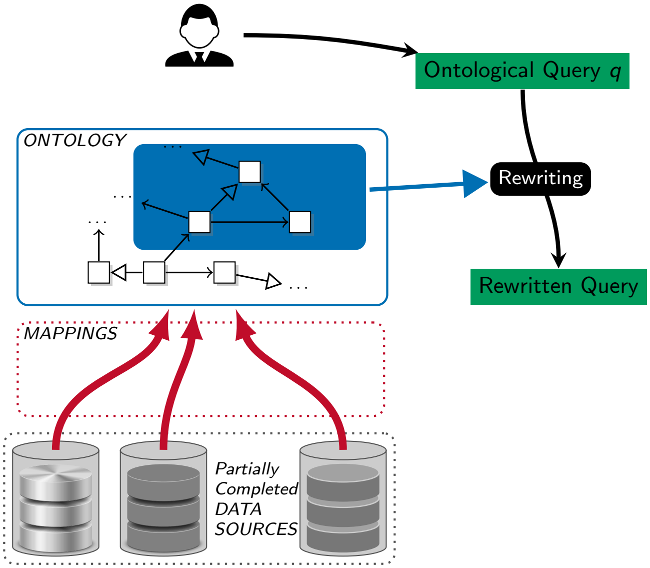
Query Answering by Combined Approach



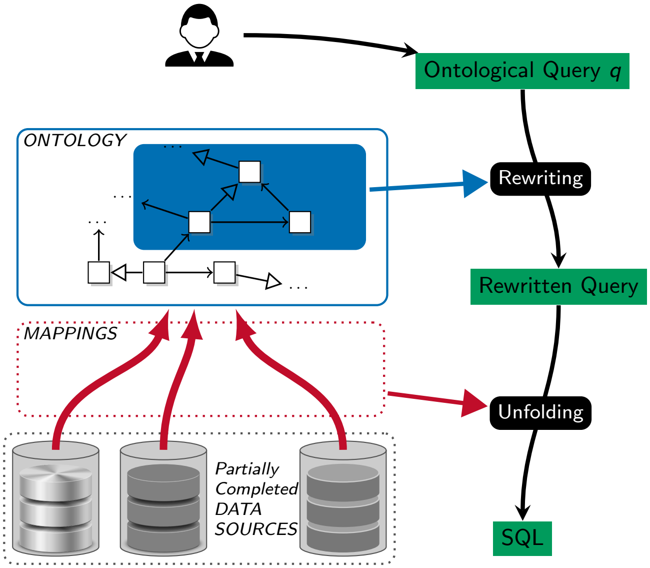
Query Answering by Combined Approach



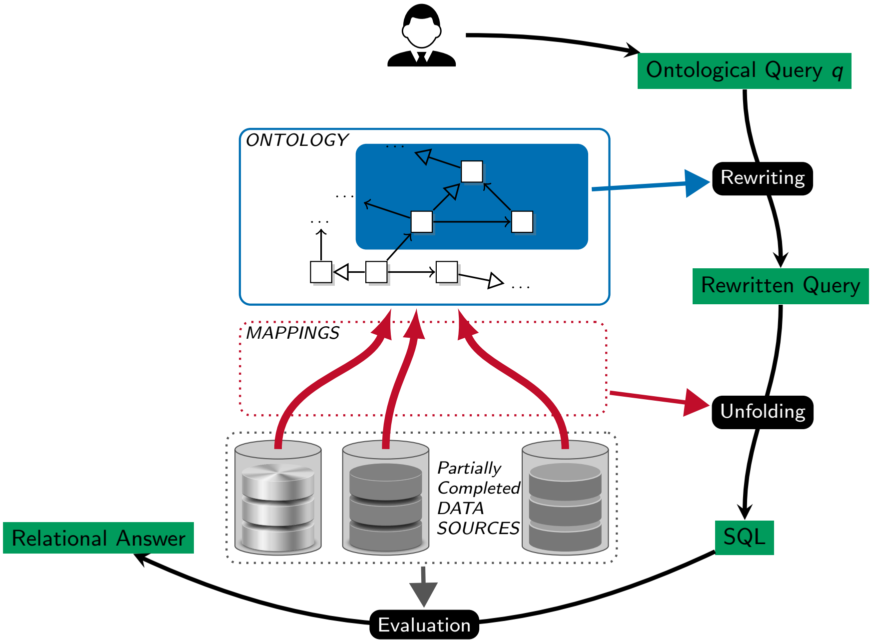
Query Answering by Combined Approach



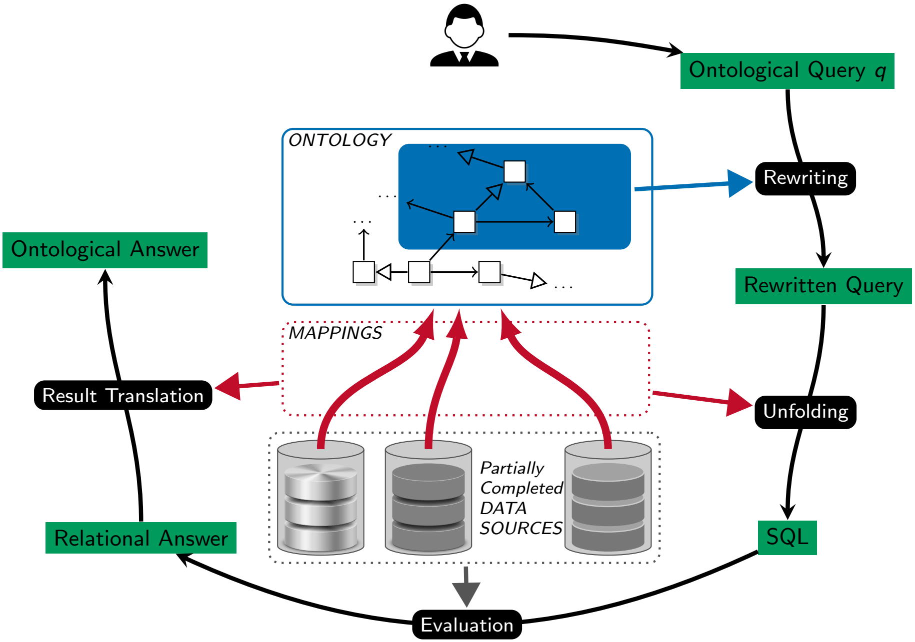
Query Answering by Combined Approach



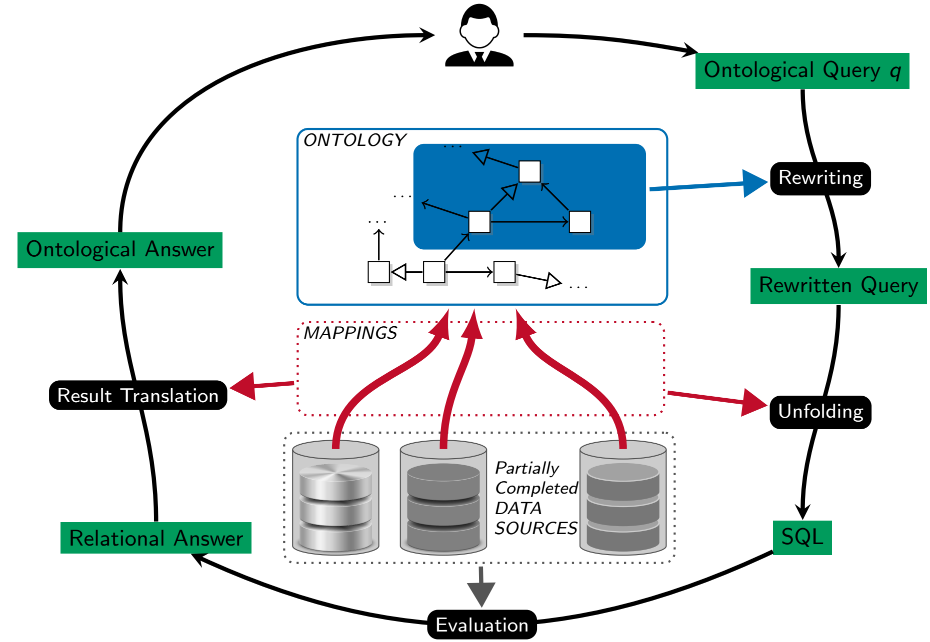
Query Answering by Combined Approach



Query Answering by Combined Approach



Query Answering by Combined Approach



Combined Approach Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad$ $Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad$ $hasNeoplasm \sqsubseteq hasCondition$

Data:

Combined Approach Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad$ $Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad$ $hasNeoplasm \sqsubseteq hasCondition$

Data:

Partially Completed Data:

Combined Approach Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad$ $Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad$ $hasNeoplasm \sqsubseteq hasCondition$

Data:

Partially Completed Data:

Query: Does Mary have neoplasm that is Cancer?

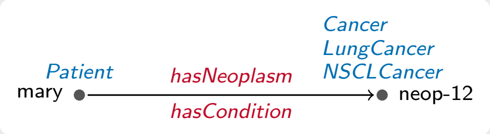
$$q(x) = \exists y.hasNeoplasm(x, y) \wedge Cancer(y)$$

Combined Approach Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad$ $Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad$ $hasNeoplasm \sqsubseteq hasCondition$

Data:

Partially Completed Data:



Query: Does Mary have neoplasm that is Cancer?

$$q(x) = \exists y.hasNeoplasm(x, y) \wedge Cancer(y)$$

Rewritten Query:

$$q'(x) = q(x)$$

Combined Approach Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad hasNeoplasm \sqsubseteq hasCondition$

Data:

$$\underset{mary}{\overset{Patient}{\bullet}} \xrightarrow{hasNeoplasm} \underset{neop\text{-}12}{\overset{NSCLCancer}{\bullet}}$$

Partially Completed Data:

$$\underset{mary}{\overset{Patient}{\bullet}} \xrightarrow[hasCondition]{hasNeoplasm} \underset{neop\text{-}12}{\overset{Cancer,\ LungCancer,\ NSCLCancer}{\bullet}}$$

Query: Does Mary have neoplasm that is Cancer?

$$q(x) = \exists y.hasNeoplasm(x, y) \wedge Cancer(y)$$

Rewritten Query:

$$q'(x) = q(x)$$

The answer is **Yes**: Mary is an answer to the query q' over **partially completed data**.

Combined Approach Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad$ $Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad$ $hasNeoplasm \sqsubseteq hasCondition$

Data:

Partially Completed Data:

Query: Does Mary have condition that is NSCLCancer?

$$q(x) = \exists y.hasCondition(x, y) \wedge NSCLCancer(y)$$

Combined Approach Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad$ $Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad$ $hasNeoplasm \sqsubseteq hasCondition$

Data:

Partially Completed Data:

Query: Does Mary have condition that is NSCLCancer?

$$q(x) = \exists y.hasCondition(x, y) \wedge NSCLCancer(y)$$

Rewritten Query:

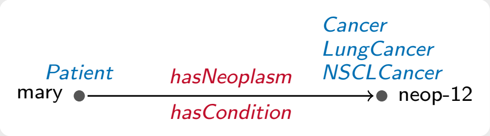
$$q'(x) = q(x)$$

Combined Approach Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad hasNeoplasm \sqsubseteq hasCondition$

Data:

Partially Completed Data:



Query: Does Mary have condition that is NSCLCancer?

$$q(x) = \exists y.hasCondition(x, y) \wedge NSCLCancer(y)$$

Rewritten Query:

$$q'(x) = q(x)$$

The answer is **Yes**: Mary is an answer to the query q' over **partially completed data**.

Combined Approach Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad$ $Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad$ $hasNeoplasm \sqsubseteq hasCondition$

Data:

$Patient$ $\quad$ $NSCLCancer$

$\text{mary} \xrightarrow{hasNeoplasm} \text{neop-12}$

Partially Completed Data:

$Patient$ $\quad$ $Cancer$, $LungCancer$, $NSCLCancer$

$\text{mary} \xrightarrow[hasCondition]{hasNeoplasm} \text{neop-12}$

Query: Does Mary have SSN?

$$q(x) = \exists y.hasSSN(x, y)$$

Combined Approach Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad hasNeoplasm \sqsubseteq hasCondition$

Data:

$$\underset{\text{mary}}{\overset{Patient}{\bullet}} \xrightarrow{hasNeoplasm} \underset{\text{neop-12}}{\overset{NSCLCancer}{\bullet}}$$

Partially Completed Data:

$$\underset{\text{mary}}{\overset{Patient}{\bullet}} \xrightarrow[hasCondition]{hasNeoplasm} \underset{\text{neop-12}}{\overset{\substack{Cancer\\ LungCancer\\ NSCLCancer}}{\bullet}}$$

Query: Does Mary have SSN?

$$q(x) = \exists y.hasSSN(x, y)$$

Rewritten Query:

$$q'(x) = \begin{array}{c} \exists y.hasSSN(x, y) \\ \vee \\ Patient(x) \end{array}$$

Combined Approach Example

Axioms: $LungCancer \sqsubseteq Cancer$ $\quad$ $Patient \sqsubseteq \exists hasSSN$
$NSCLCancer \sqsubseteq LungCancer$ $\quad$ $hasNeoplasm \sqsubseteq hasCondition$

Data:

Partially Completed Data:

Query: Does Mary have SSN?

$$q(x) = \exists y.hasSSN(x, y)$$

Rewritten Query:

$$q'(x) = \begin{array}{c} \exists y.hasSSN(x, y) \\ \vee \\ Patient(x) \end{array}$$

The answer is **Yes**: Mary is an answer to the query q' over **partially completed data**.

Combined Approach Issues

- in general does not work well for *DL-Lite*$_{\mathcal{R}}$
- however, it works well for $\mathcal{EL} \approx$ OWL 2 EL
- requires updating data sources, which might not be always possible

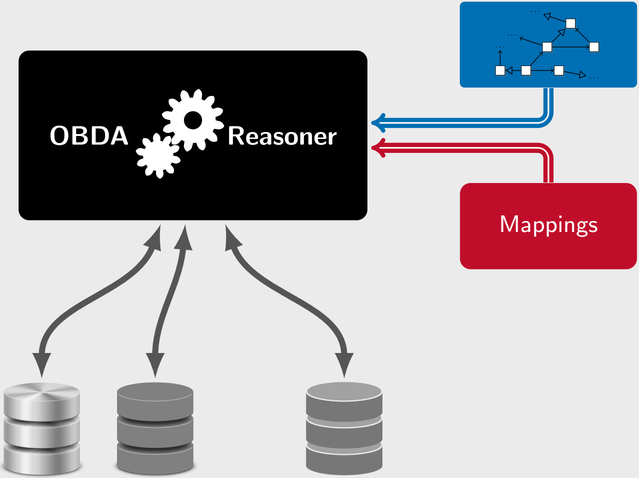
What I Did Not Mention

is how **Query Unfolding** is done:

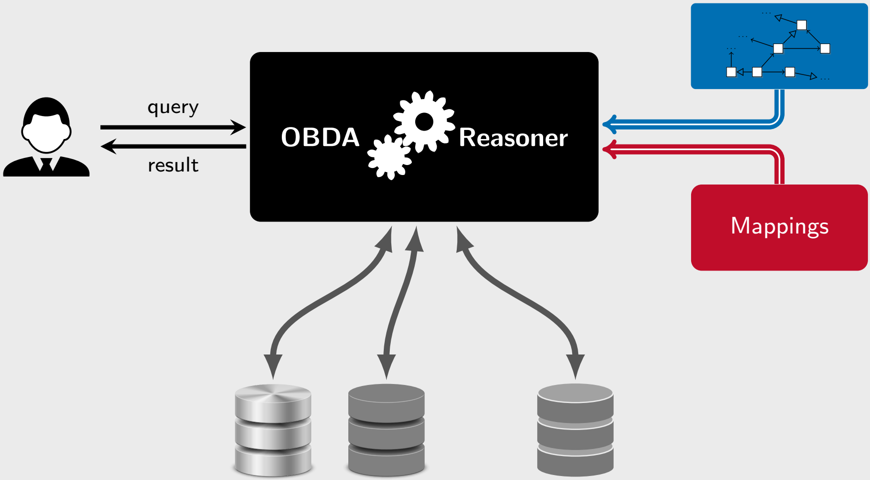
how to obtain from an ontological query an SQL query

Outline

The Architecture of an OBDA System



The Architecture of an OBDA System



The Existing OBDA Systems

- MASTRO
- Ultrawrap
- ontop / Quest
- ...

-ontop-

-ontop- is an OBDA framework developed at the Free University of Bozen-Bolzano.
`http://ontop.inf.unibz.it/`

-ontop- is freely available for download and comes as

- a plugin for Protege
- OWLAPI
- a SPARQL end-point

Let us see it at work![1]

[1] We follow the tutorial that can be found here:
`https://github.com/ontop/ontop/wiki/Easy-Tutorial%3A-Using-Ontop-from-Protege`

Thank you
for your attention!

QUESTIONS?

Recommended Reading

- D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini and R. Rosati. DL-Lite: Tractable Description Logics for Ontologies. In Proc. AAAI 2005: 602–607
- A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini and R. Rosati. Linking Data to Ontologies. J. Data Semantics 10: 133–173 (2008)
- A. Chortaras, D. Trivela and G. Stamou. Optimized query rewriting for OWL 2 QL. In Proc. CADE 2011
- A. Artale, D. Calvanese, R. Kontchakov and M. Zakharyaschev: The DL-Lite Family and Relations. J. Artif. Intell. Res. (JAIR) (2009)
- M. Rodriguez-Muro and D. Calvanese. Semantic Index: Scalable Query Answering without Forward Chaining or Exponential Rewritings. Posters of ISWC 2011
- M. Rodriguez-Muro and D. Calvanese. Dependencies: Making Ontology Based Data Access Work. In Proc. AMW 2011
- M. Rodríguez-Muro, R. Kontchakov and M. Zakharyaschev. Ontology-Based Data Access: Ontop of Databases. In Proc. ISWC 2013
- R. Kontchakov, M. Rezk, M. Rodríguez-Muro, G. Xiao and M. Zakharyaschev. Answering SPARQL Queries under the OWL 2 QL Entailment Regime with Databases. In Proc. ISWC 2014
- J. Sequeda, M. Arenas, D. P. Miranker. OBDA: Query Rewriting or Materialization? In Practice, Both! In Proc. ISWC 2014
- S. Kikot, R. Kontchakov, V. Podolskii and M. Zakharyaschev: Exponential Lower Bounds and Separation for Query Rewriting. In Proc. ICALP 2012
- R. Kontchakov, C. Lutz, D. Toman, F. Wolter and M. Zakharyaschev. The Combined Approach to Ontology-Based Data Access. In Proc. IJCAI 2011
- F. Baader, S. Brandt and C. Lutz. Pushing the EL envelope. In Proc. IJCAI 2005
- C. Lutz, D. Toman and F. Wolter. Conjunctive Query Answering in EL using a Database System. In Proc. OWLED 2008
- C. Lutz, I. Seylan, D. Toman, and F Wolter. The Combined Approach to OBDA: Taming Role Hierarchies using Filters. In Proc. ISWC 2013
- A. Calì, G. Gottlob and A. Pieris. Query Rewriting under Non-Guarded Rules. In Proc. AMW 2010
- G. Gottlob and T. Schwentick. Rewriting Ontological Queries into Small Nonrecursive Datalog Programs. In Proc. KR 2012