Mapping Management and Expressive Ontologies in Ontology-Based Data Access

1. Introduction to OBDA

Diego Calvanese, Benjamin Cogrel, Guohui Xiao

KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy

unibz
Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen-Bolzano

20th Int. Conf. on Knowledge Engineering and Knowledge Management (EKAW)
Bologna, 19 November 2016

Outline

1. Motivation
2. Semantic Web standards
3. OBDA framework
4. Ontop system and its ecosystem
5. Use Cases
6. Conclusion

Outline

1. Motivation
   - Semantic gap
   - Solutions

2. Semantic Web standards

3. OBDA framework

4. Ontop system and its ecosystem

5. Use Cases

6. Conclusion

Typical view of Big Data

But: data has a lot of structure

Challenge: how to use the data – Statoil Exploration

Challenge: how to use the data – Statoil Exploration

900 geologists and geophysicists in Statoil Exploration develop stratigraphic models of unexplored areas on the basis of data acquired from previous operations at nearby locations.

Challenge: how to use the data – Statoil Exploration

900 geologists and geophysicists in Statoil Exploration develop stratigraphic models of unexplored areas on the basis of data acquired from previous operations at nearby locations.

Statoil

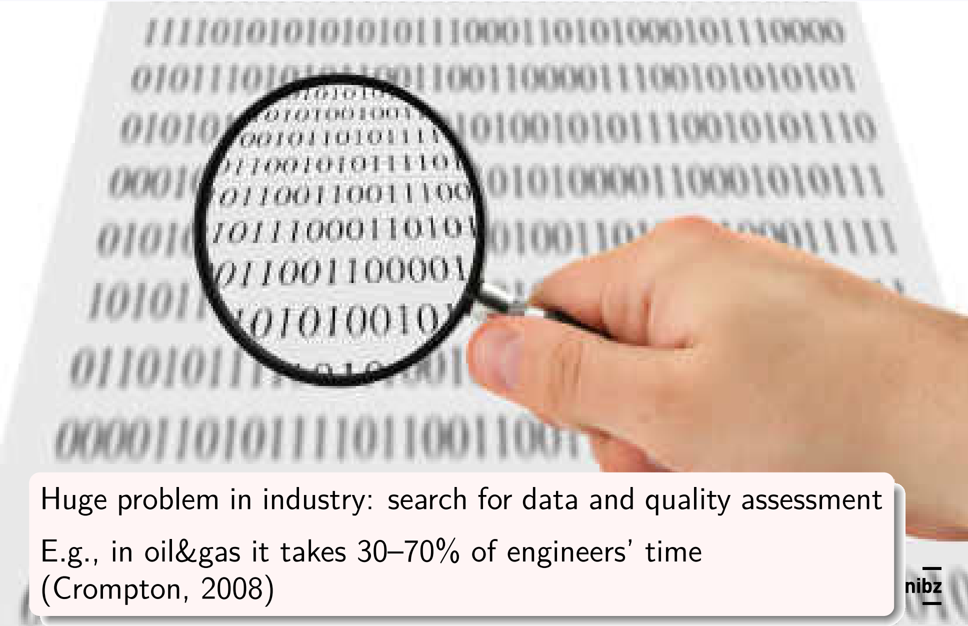
Data stores:

- Exploration and Production Data Store (EPDS): ∼1500 tables (100s GBs)
- OpenWorks
- Norwegian Petroleum Directorate FactPages

Optique

unibz

How much time is spent searching for data?

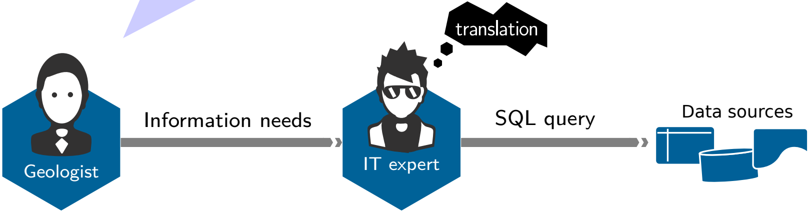


Huge problem in industry: search for data and quality assessment

E.g., in oil&gas it takes 30–70% of engineers' time (Crompton, 2008)

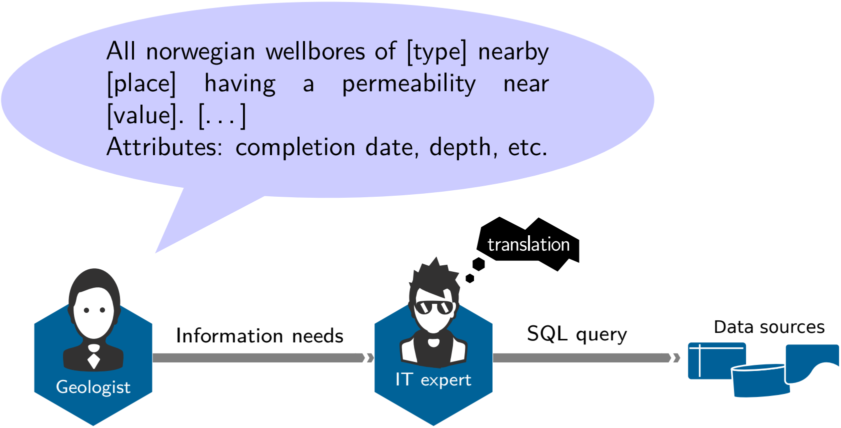
Designing a new (ad-hoc) query

All norwegian wellbores of [type] nearby [place] having a permeability near [value]. [. . .]
Attributes: completion date, depth, etc.



NB: Simplified information needs

Designing a new (ad-hoc) query



Takes 4 days in average (with EPDS only)

NB: Simplified information needs

A typical query at Statoil

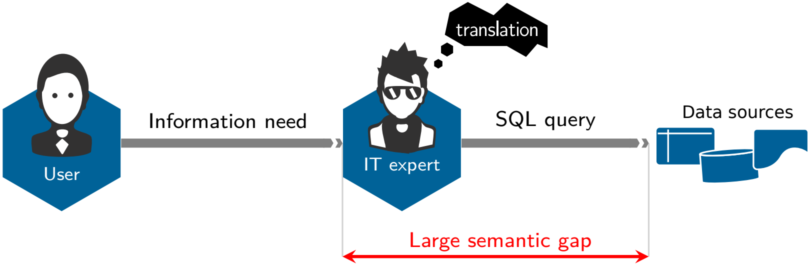
Anonymized extract

```
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))
```

Outline

1. Motivation
   - Semantic gap
   - Solutions
2. Semantic Web standards
3. OBDA framework
4. Ontop system and its ecosystem
5. Use Cases
6. Conclusion

Semantic gap



Querying over tables

Requires a lot of knowledge about:

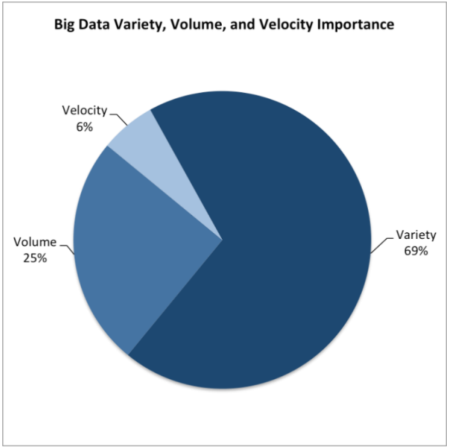
- Magic numbers
  (e.g., 1 → full professor)
- Cardinalities and normal forms
- Closely-related information spread over many tables

Data integration

- Exacerbates these issues
- Variety: **challenge #1** for most Big Data initiatives

unibz

Challenges in “Big Data” era



“Variety, Not Volume, Is Driving Big Data Initiatives”
MIT Sloan Management Review (28 March 2016)

http://sloanreview.mit.edu/article/variety-not-volume-is-driving-big-data-initiatives/

Outline

unibz

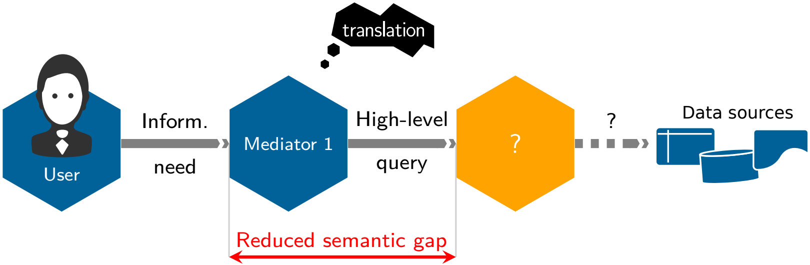
High-level translation

Main bottleneck: translation

- of the information needs
- ... into a **formal query**

Goal

Make such a translation easy
(Ideally: IT expertise not required)

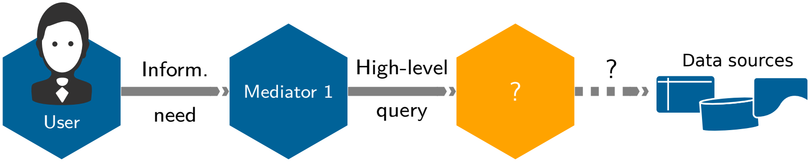


Mediator 1 could be a user, an IT expert or a GUI

General approach: two steps

1. Translate the information needs into a **high-level query**
2. Answer the high-level query **automatically**

Two orthogonal choices to be made



Choice 1: Generating a new representation of the data

1. Extract Transform Load (ETL) process
2. Virtual views

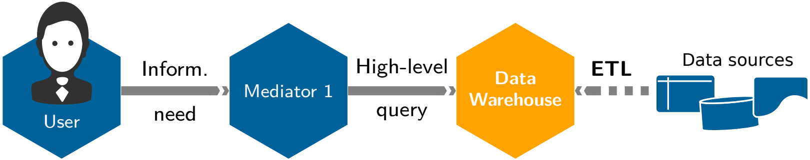
Choice 2: Which data format for the new representation

1. New relational schema
2. JSON (or XML) documents
3. Resource Description Framework (RDF)

Generating a new representation of the data

1. Extract Transform Load (ETL)

E.g., relational data warehouse

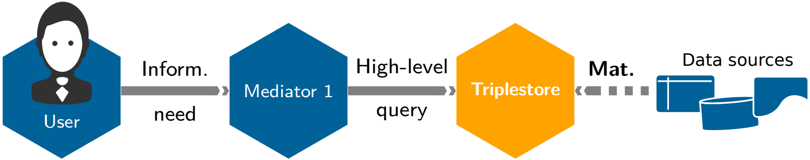


unibz

Generating a new representation of the data

1. Extract Transform Load (ETL) / Materialization

E.g., relational data warehouse, triplestore

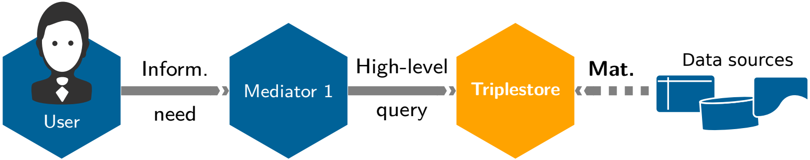


unibz

Generating a new representation of the data

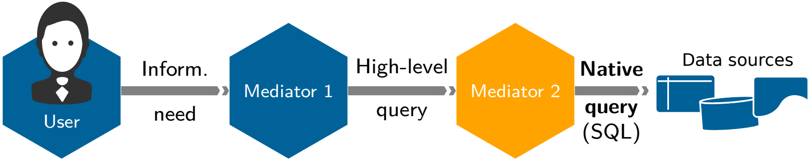
1. Extract Transform Load (ETL) / Materialization

E.g., relational data warehouse, triplestore



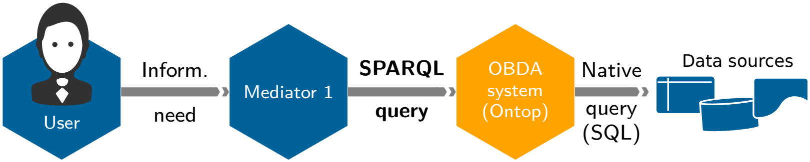
2. Virtual views

E.g., virtual databases (Teiid, Apache Drill, Exareme), **OBDA** (Ontop)



unibz

Ontology-Based Data Access (OBDA)



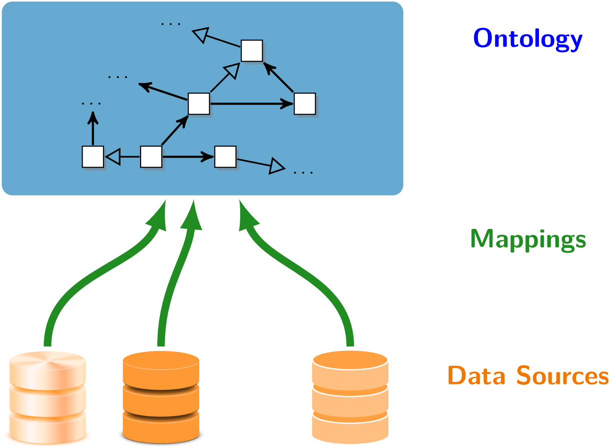
Choice 1: Generating a new representation of the data

1. Extract Transform Load (ETL) process
2. **Virtual views**

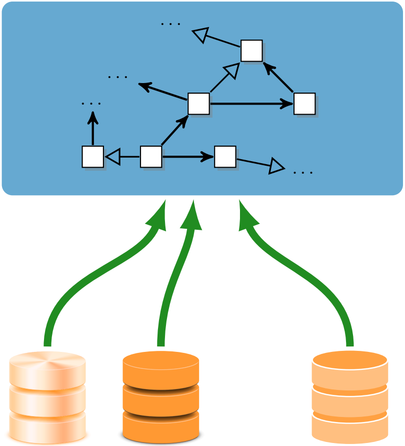
Choice 2: Which data format for the virtual view

1. New relational schema
2. JSON or XML documents
3. **Resource Description Framework (RDF)**

OBDA framework



OBDA framework



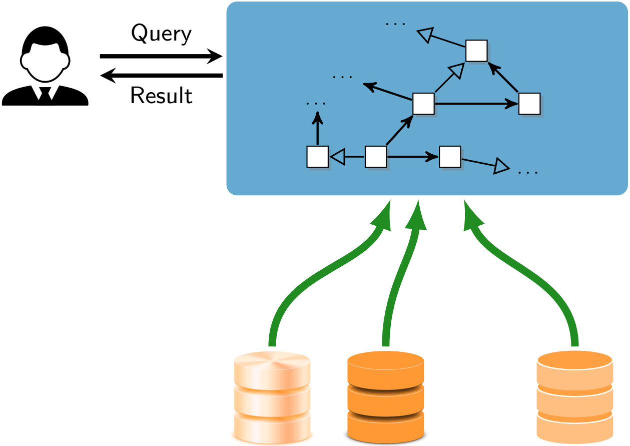
Ontology
provides global vocabulary and conceptual view

Mappings
how to populate the ontology from the data

Data Sources
external and heterogeneous

unibz

OBDA framework



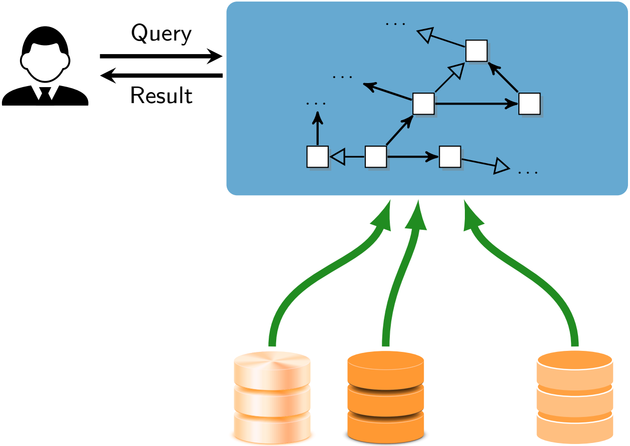
Ontology
provides global vocabulary and conceptual view

Mappings
how to populate the ontology from the data

Data Sources
external and heterogeneous

unibz

OBDA framework



Ontology
provides
global vocabulary
and
conceptual view

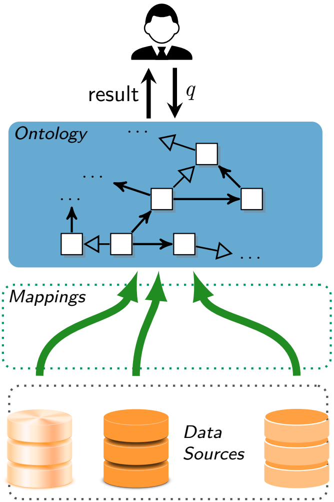
Mappings
how to populate
the ontology
from the data

Data Sources
external and
heterogeneous

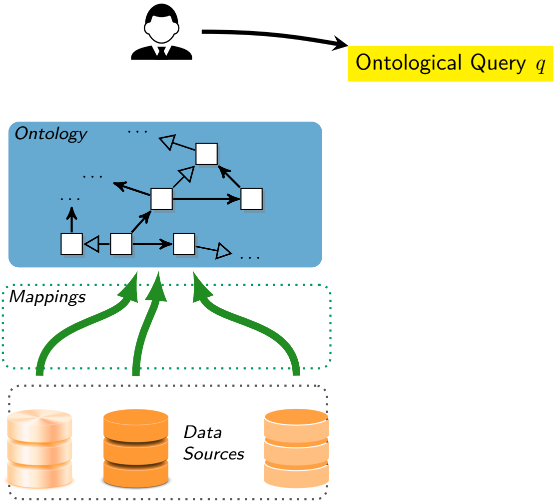
Logical transparency in accessing data:

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

Query answering by rewriting (conceptual framework)

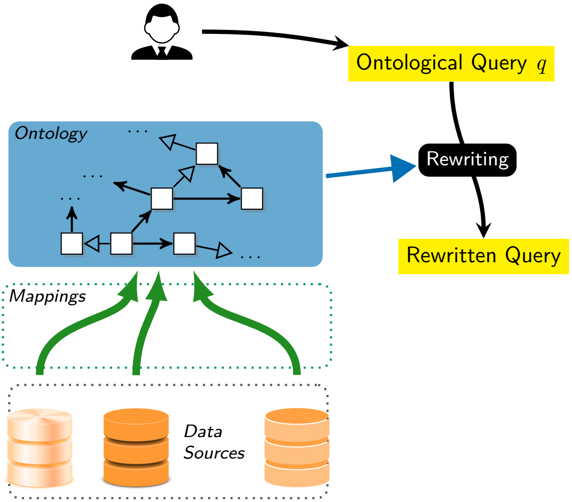


Query answering by rewriting (conceptual framework)

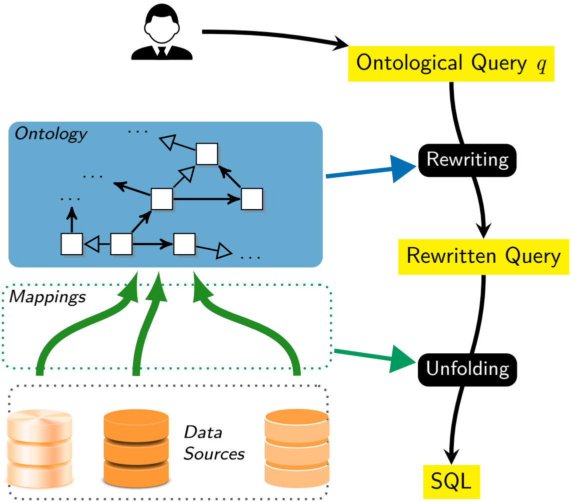


unibz

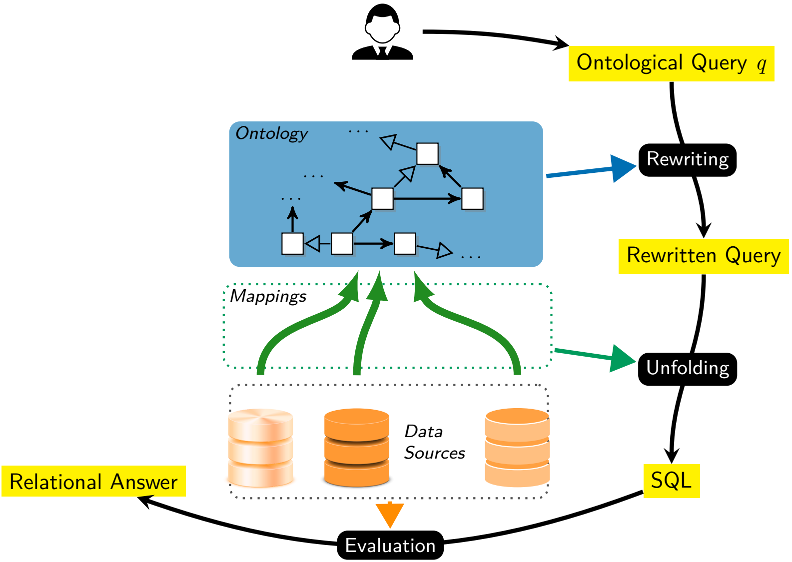
Query answering by rewriting (conceptual framework)



Query answering by rewriting (conceptual framework)

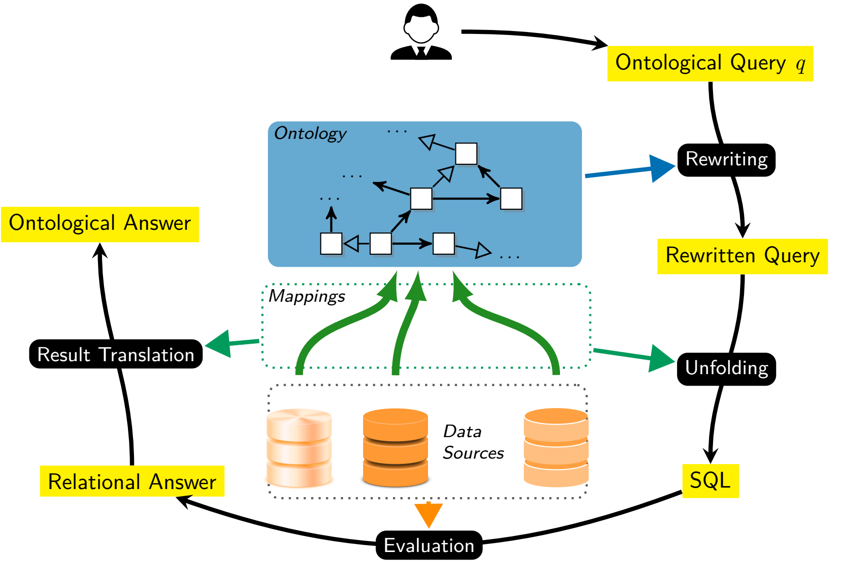


Query answering by rewriting (conceptual framework)

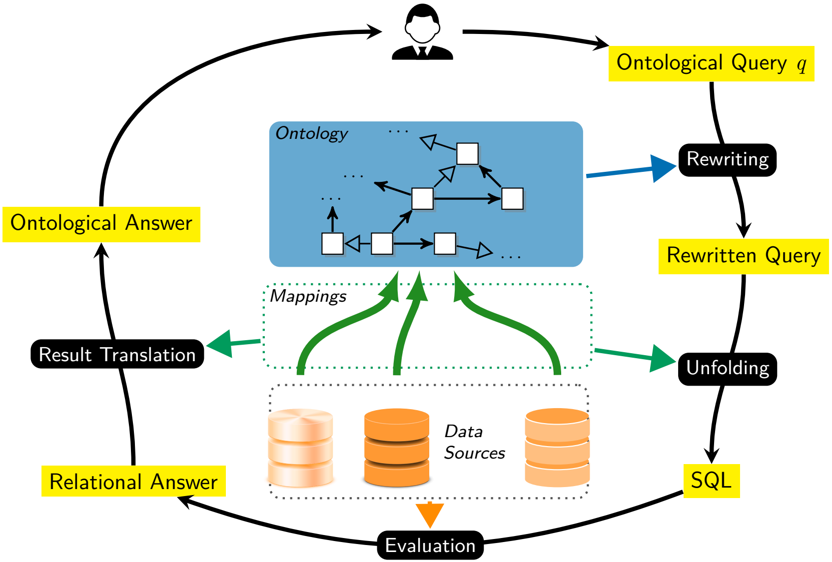


unibz

Query answering by rewriting (conceptual framework)



Query answering by rewriting (conceptual framework)



unibz

Outline

Semantic Web standards

1. RDF (Resource Description Framework): format for Semantic Web data
2. SPARQL: query language for RDF data
3. RDFS and OWL 2 QL: ontology languages
4. Mapping languages

unibz

Outline

unibz

Resource Description Framework (RDF)

RDF provides a description of the domain in terms of **triples**:

Subject

<http://example.org/uni2/person/3>

Predicate

<http://xmlns.com/foaf/0.1/firstName>

Object

"Céline"^^xsd:string

Triple elements: resources denoted by global identifiers (IRIs)

1. Subject: IRI of the described resource
2. Predicate: IRI of the property
3. Object: attribute value or IRI of another resource

Resource Description Framework (RDF)

RDF provides a description of the domain in terms of **triples**:

Subject

Predicate

Object

<http://example.org/uni2/person/3>

<http://xmlns.com/foaf/0.1/firstName>

“Céline”^^xsd:string

Triple elements: resources denoted by global identifiers (IRIs)

1. Subject: IRI of the described resource
2. Predicate: IRI of the property
3. Object: attribute value or IRI of another resource

Prefixes: useful abbreviations and/or references to external information

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>
```

Resource Description Framework (RDF)

RDF provides a description of the domain in terms of **triples**:

Subject → `<http://example.org/uni2/person/3>`

Predicate → `foaf:firstName`

Object → "Céline"`^^xsd:string`

Triple elements: resources denoted by **global identifiers** (IRIs)

1. Subject: IRI of the described resource
2. Predicate: IRI of the property
3. Object: attribute value or IRI of another resource

Prefixes: useful abbreviations and/or references to external information

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>
```

Resource Description Framework (RDF)

RDF provides a description of the domain in terms of **triples**:

Subject → `<http://example.org/uni2/person/3>`

Predicate → `foaf:firstName`

Object → "Céline"`^^xsd:string`

Triple elements: resources denoted by global identifiers (IRIs)

1. Subject: IRI of the described resource
2. Predicate: IRI of the property
3. Object: attribute value or IRI of another resource

Prefixes: useful abbreviations and/or references to external information

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>
@prefix : <http://example.org/voc#>
```

Resource Description Framework (RDF)

RDF provides a description of the domain in terms of **triples**:

Subject — `<http://example.org/uni2/person/3>`

Predicate — `foaf:firstName`

Object — “Céline”`^^xsd:string`

Triple elements: resources denoted by global identifiers (IRIs)

1. Subject: IRI of the described resource
2. Predicate: IRI of the property
3. Object: attribute value or IRI of another resource

Prefixes: useful abbreviations and/or references to external information

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>
@prefix : <http://example.org/voc#>
@base <http://example.org/>
```

Resource Description Framework (RDF)

RDF provides a description of the domain in terms of **triples**:

Subject	Predicate	Object
<uni2/person/3>	foaf:firstName	"Céline"^^xsd:string

Triple elements: resources denoted by global identifiers (IRIs)

1. Subject: IRI of the described resource
2. Predicate: IRI of the property
3. Object: attribute value or IRI of another resource

Prefixes: useful abbreviations and/or references to external information

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>
@prefix : <http://example.org/voc#>
@base <http://example.org/>
```

RDF – Examples

Class membership:

Fact	Prof(*uni2/person/1*)
RDF triple	`<uni2/person/1> a :Prof`

RDF – Examples

Class membership:

Fact	Prof(*uni2/person/1*)
RDF triple	`<uni2/person/1> a :Prof`

Note: This is an abbreviation for

RDF triple	`<uni2/person/1> rdf:type :Prof`

RDF – Examples

Class membership:

Fact	Prof(*uni2/person/1*)
RDF triple	`<uni2/person/1> a :Prof`

Note: This is an abbreviation for

RDF triple	`<uni2/person/1> rdf:type :Prof`

Attribute of an individual:

Fact	lastName(*uni2/person/3*, 'Mendez')
RDF triple	`<uni2/person/3> foaf:lastName "Mendez"`

unibz

RDF – Examples

Class membership:

Fact	Prof(*uni2/person/1*)
RDF triple	`<uni2/person/1> a :Prof`

Note: This is an abbreviation for

RDF triple	`<uni2/person/1> rdf:type :Prof`

Attribute of an individual:

Fact	lastName(*uni2/person/3*, 'Mendez')
RDF triple	`<uni2/person/3> foaf:lastName "Mendez"`

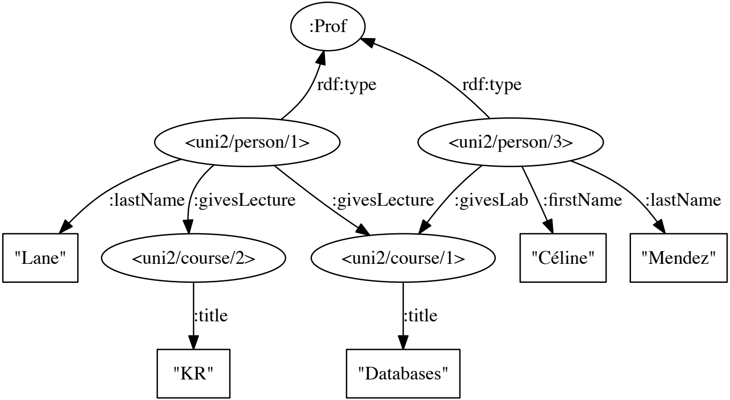
Property of an individual:

Fact	givesLab(*uni2/person/3*, *uni2/course/1*)
RDF triple	`<uni2/person/3> :givesLab <uni2/course/1>`

RDF graph – Example

```
<uni2/person/1> rdf:type :Prof
<uni2/person/1> foaf:lastName "Lane"
<uni2/person/1> :givesLecture <uni2/course/1>
...
```

We can represent such a set of facts graphically:



Additional RDF features

RDF has additional features not covered here:

- blank nodes
- named graphs

Outline

unibz

SPARQL Basic Graph Patterns

- SPARQL is the standard query language for RDF.
- **Basic Graph Pattern (BGP)**: simplest form of SPARQL query, asking for a pattern in the RDF graph

Ex.: BGP

```
SELECT ?p ?ln ?c ?t
WHERE {
  ?p :lastName ?ln .
  ?p :givesLecture ?c .
  ?c :title ?t .
}
```

unibz

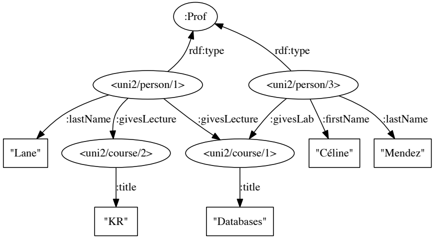
SPARQL Basic Graph Patterns

- SPARQL is the standard query language for RDF.
- **Basic Graph Pattern (BGP)**: simplest form of SPARQL query, asking for a pattern in the RDF graph

Ex.: BGP

```
SELECT ?p ?ln ?c ?t
WHERE {
  ?p :lastName ?ln .
  ?p :givesLecture ?c .
  ?c :title ?t .
}
```

When evaluated over the RDF graph



... the query returns:

p	ln	c	t
<uni2/person/1>	"Lane"	<uni2/course/1>	"Databases"
<uni2/person/1>	"Lane"	<uni2/course/2>	"KR"

Projecting out variables in a SPARQL query

A query may also return only a subset of the variables used in the BGP.

Ex.: BGP with projection

```
SELECT ?ln ?t
WHERE {
  ?p :lastName ?ln .
  ?p :givesLecture ?c .
  ?c :title ?t .
}
```

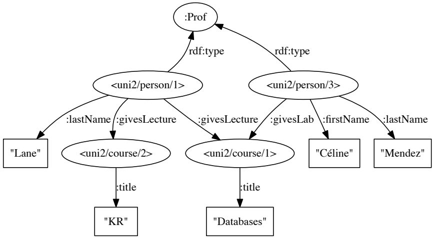
Projecting out variables in a SPARQL query

A query may also return only a subset of the variables used in the BGP.

Ex.: BGP with projection

```
SELECT ?ln ?t
WHERE {
  ?p :lastName ?ln .
  ?p :givesLecture ?c .
  ?c :title ?t .
}
```

When evaluated over the RDF graph



... the query returns:

ln	t
"Lane"	"Databases"
"Lane"	"KR"

Union of Basic Graph Patterns

Ex.: BGPs with UNION

```
SELECT ?p ?ln ?c
WHERE {
  { ?p :lastName ?ln .    ?p :givesLecture ?c . }
  UNION
  { ?p :lastName ?ln .    ?p :givesLab ?c . }
}
```

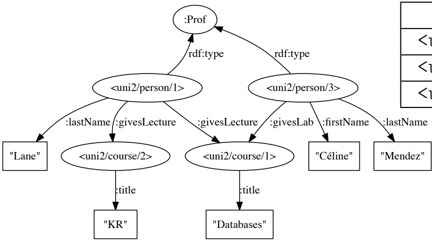
unibz

Union of Basic Graph Patterns

Ex.: BGPs with UNION

```
SELECT ?p ?ln ?c
WHERE {
  { ?p :lastName ?ln .    ?p :givesLecture ?c . }
  UNION
  { ?p :lastName ?ln .    ?p :givesLab ?c . }
}
```

When evaluated over



. . . the query returns:

p	ln	c
<uni2/person/1>	"Lane"	<uni2/course/1>
<uni2/person/1>	"Lane"	<uni2/course/2>
<uni2/person/3>	"Mendez"	<uni2/course/1>

BGPs vs. conjunctive queries

We can write queries using a more compact and abstract syntax, borrowed from database theory.

Ex.: BGP

```
SELECT ?p ?ln ?c ?t
WHERE {
  ?p :lastName ?ln .
  ?p :givesLecture ?c .
  ?c :title ?t .
}
```

vs. conjunctive query

$$\boldsymbol{q}(p, ln, c, t) \leftarrow \begin{array}{l} \mathsf{lastName}(p, ln), \\ \mathsf{givesLecture}(p, c), \\ \mathsf{title}(c, t) \end{array}$$

unibz

BGPs vs. conjunctive queries

We can write queries using a more compact and abstract syntax, borrowed from database theory.

Ex.: BGP

```
SELECT ?p ?ln ?c ?t
WHERE {
  ?p :lastName ?ln .
  ?p :givesLecture ?c .
  ?c :title ?t .
}
```

vs. conjunctive query

$$\boldsymbol{q}(p, ln, c, t) \leftarrow \text{lastName}(p, ln), \text{givesLecture}(p, c), \text{title}(c, t)$$

A **conjunctive query** $\boldsymbol{q}$ has the form $\boldsymbol{q}(\vec{x}) \leftarrow p_1(\vec{y}_1), \ldots, p(\vec{y}_k)$ where

- $\boldsymbol{q}(\vec{x})$ is called the head of $\boldsymbol{q}$,
- $p_1(\vec{y}_1), \ldots, p(\vec{y}_k)$ is a conjunction of atoms called the body of $\boldsymbol{q}$,
- all variables $\vec{x}$ in the head are among $\vec{y}_1, \ldots, \vec{y}_k$, and
- the variables in $\vec{y}_1, \ldots, \vec{y}_k$ not among $\vec{x}$ are existentially quantified.

BGPs vs. conjunctive queries (cont.)

Ex.: BGP with projection

```
SELECT ?ln ?t
WHERE {
  ?p :lastName ?ln .
  ?p :givesLecture ?c .
  ?c :title ?t .
}
```

vs. conjunctive query

$$\boldsymbol{q}(ln, t) \leftarrow \mathsf{lastName}(p, ln), \mathsf{givesLecture}(p, c), \mathsf{title}(c, t)$$

BGPs vs. conjunctive queries (cont.)

Ex.: BGP with projection

```
SELECT ?ln ?t
WHERE {
  ?p :lastName ?ln .
  ?p :givesLecture ?c .
  ?c :title ?t .
}
```

vs. conjunctive query

$$\boldsymbol{q}(ln, t) \leftarrow \mathsf{lastName}(p, ln), \mathsf{givesLecture}(p, c), \mathsf{title}(c, t)$$

But there is a difference in semantics when we have an ontology.

SPARQL UNION vs. unions of CQs

Ex.: BGP with UNION

```
SELECT ?p ?ln ?c
WHERE {
  { ?p :lastName ?ln .
    ?p :givesLecture ?c .
  }
  UNION
  { ?p :lastName ?ln .
    ?p :givesLab ?c .
  }
}
```

vs. union of CQs (UCQ)

$$\begin{aligned}\boldsymbol{q}(p, ln, c) &\leftarrow \mathsf{lastName}(p, ln), \\ &\quad\ \mathsf{givesLecture}(p, c) \\ \boldsymbol{q}(p, ln, c) &\leftarrow \mathsf{lastName}(p, ln), \\ &\quad\ \mathsf{givesLab}(p, c)\end{aligned}$$

A UCQ is written as a set of CQs, all with the same head.

unibz

Extending BGPs with OPTIONAL

We might want to add information when available, but **not reject** a solution **when some part of the query does not match**.

Ex.: BGP with OPTIONAL

```
SELECT ?p ?fn ?ln
WHERE {
  ?p :lastName ?ln .
  OPTIONAL {
    ?p :firstName ?fn .
  }
}
```

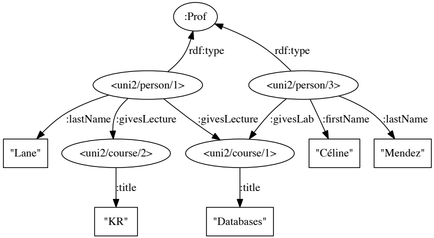
Extending BGPs with OPTIONAL

We might want to add information when available, but **not reject** a solution **when some part of the query does not match**.

Ex.: BGP with OPTIONAL

```
SELECT ?p ?fn ?ln
WHERE {
  ?p :lastName ?ln .
  OPTIONAL {
    ?p :firstName ?fn .
  }
}
```

When evaluated over the RDF graph



... the query returns:

p	fn	ln
<uml2/person/1>		"Lane"
<uml2/person/3>	"Céline"	"Mendez"

unibz

SPARQL algebra

We have seen the following features of the SPARQL algebra:

- Basic Graph Patterns
- `UNION`
- `OPTIONAL`

SPARQL algebra

We have seen the following features of the SPARQL algebra:

- Basic Graph Patterns
- `UNION`
- `OPTIONAL`

The overall algebra has additional features:

- more complex `FILTER` conditions
- `GROUP BY`, to express aggregations and support aggregation operators
- `MINUS`, to remove possible solutions
- `FILTER NOT EXISTS`, to test for the absence of a pattern

unibz

Outline

The OWL 2 QL profile

- OWL 2 QL is one of the three standard profiles of OWL 2.
- Derived from the ***DL-Lite*** $_{\mathcal{R}}$ description logic of the *DL-Lite*-family:
  - Groups the domain into classes of objects with common properties.
  - Binary relations between objects (object properties).
  - Binary relations from objects to values (data properties).

The OWL 2 QL profile

- OWL 2 QL is one of the three standard profiles of OWL 2.
- Derived from the ***DL-Lite***$_{\mathcal{R}}$ description logic of the *DL-Lite*-family:
  - Groups the domain into classes of objects with common properties.
  - Binary relations between objects (object properties).
  - Binary relations from objects to values (data properties).
- Is considered a lightweight ontology language:
  - controlled expressive power
  - efficient inference
- Optimized for accessing large amounts of data (i.e., for data complexity):
  - First-order rewritability of query answering: queries over the ontology can be rewritten into SQL queries over the underlying relational database.
  - Consistency checking is also first-order rewritable.

unibz

Constructs of OWL 2 QL/ $DL\text{-}Lite_{\mathcal{R}}$

- Class hierarchies: `rdfs:subClassOf`
- Property hierarchies: `rdfs:subPropertyOf`
- Property domain: `rdfs:domain`
- Property range: `rdfs:range`
- Inverse properties: `owl:inverseOf`
- Class disjointness: `owl:disjointWith`
- Mandatory participation: `owl:someValuesFrom` in superclass expression

unibz

RDF Schema (RDFS)

rdfs:subClassOf $(A \sqsubseteq B)$

```
:FullProf rdfs:subClassOf :Professor .
<uni1/academic/1> a :FullProf .
⟹ <uni1/academic/1> a :Professor .
```

rdfs:subPropertyOf $(P \sqsubseteq R)$

```
:givesLecture rdfs:subPropertyOf :teaches .
<uni2/academic/2> :givesLecture <uni2/course/1> .
⟹ <uni2/academic/2> :teaches <uni2/course/1> .
```

rdfs:domain $(\exists P \sqsubseteq A)$

```
:teaches rdfs:domain :Teacher .
<uni2/academic/2> :teaches <uni2/course/1> .
⟹ <uni2/academic/2> a :Teacher .
```

rdfs:range $(\exists P^{-} \sqsubseteq A)$

```
:teaches rdfs:range :Course .
<uni2/academic/2> :teaches <uni2/course/1> .
⟹ <uni2/course/1> a :Course .
```

Other constructs of OWL 2 QL I

owl:inverseOf $(P^- \sqsubseteq R,\ R^- \sqsubseteq P)$

```
:isTaughtBy owl:inverseOf :teaches .
<uni2/academic/2> :teaches <uni2/course/1> .
```

```
⟹   <uni2/course/1> :isTaughtBy <uni2/academic/2> .
```

owl:someValuesFrom in the superclass expression $(A \sqsubseteq \exists R.B)$

```
:GraduateStudent rdfs:subClassOf
   [ a owl:Restriction ;
     owl:onProperty :isSupervisedBy ;
     owl:someValuesFrom :Professor ] .
<uni2/person/10> a :GraduateStudent .
```

```
        <uni2/person/10> a
          [ a owl:Restriction ;
⟹           owl:onProperty :isSupervisedBy ;
             owl:someValuesFrom :Professor ] .
```

unibz

Other constructs of OWL 2 QL II

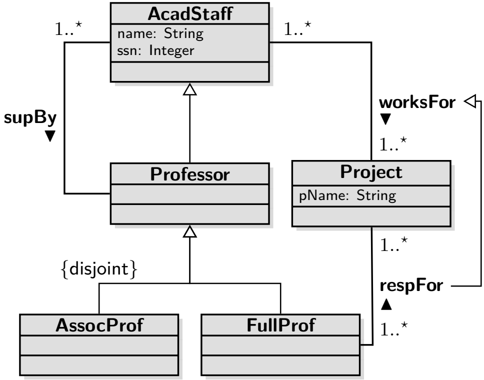
owl:disjointWith $(A \sqsubseteq \neg B,\ B \sqsubseteq \neg A)$

```
:Student owl:disjointWith :Professor .
<uni1/academic/19> a :Professor .
<uni1/academic/19> a :Student .
```

$\Longrightarrow$ Inconsistent RDF graph

unibz

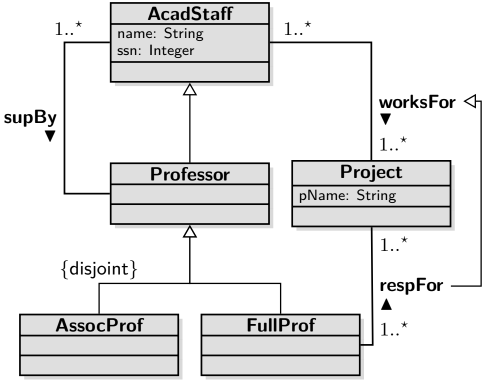
Capturing UML class diagrams/ER schemas in *DL-Lite*$_\mathcal{R}$



$$
\begin{array}{rcl}
\mathsf{Professor} & \sqsubseteq & \mathsf{AcadStaff} \\
\mathsf{AssocProf} & \sqsubseteq & \mathsf{Professor} \\
\mathsf{FullProf} & \sqsubseteq & \mathsf{Professor} \\
\mathsf{AssocProf} & \sqsubseteq & \neg\mathsf{FullProf} \\
\mathsf{AcadStaff} & \sqsubseteq & \exists\mathsf{ssn} \\
\exists\mathsf{ssn} & \sqsubseteq & \mathsf{AcadStaff} \\
\exists\mathsf{ssn}^- & \sqsubseteq & \mathtt{Integer} \\
\exists\mathsf{worksFor} & \sqsubseteq & \mathsf{AcadStaff} \\
\exists\mathsf{worksFor}^- & \sqsubseteq & \mathsf{Project} \\
\mathsf{AcadStaff} & \sqsubseteq & \exists\mathsf{worksFor} \\
\mathsf{Project} & \sqsubseteq & \exists\mathsf{worksFor}^- \\
\mathsf{respFor} & \sqsubseteq & \mathsf{worksFor} \\
 & \vdots &
\end{array}
$$

unibz

Capturing UML class diagrams/ER schemas in *DL-Lite*$_\mathcal{R}$



$$
\begin{array}{rcl}
\mathsf{Professor} & \sqsubseteq & \mathsf{AcadStaff}\\
\mathsf{AssocProf} & \sqsubseteq & \mathsf{Professor}\\
\mathsf{FullProf} & \sqsubseteq & \mathsf{Professor}\\
\mathsf{AssocProf} & \sqsubseteq & \neg\mathsf{FullProf}\\
\mathsf{AcadStaff} & \sqsubseteq & \exists\mathsf{ssn}\\
\exists\mathsf{ssn} & \sqsubseteq & \mathsf{AcadStaff}\\
\exists\mathsf{ssn}^- & \sqsubseteq & \mathtt{Integer}\\
\exists\mathsf{worksFor} & \sqsubseteq & \mathsf{AcadStaff}\\
\exists\mathsf{worksFor}^- & \sqsubseteq & \mathsf{Project}\\
\mathsf{AcadStaff} & \sqsubseteq & \exists\mathsf{worksFor}\\
\mathsf{Project} & \sqsubseteq & \exists\mathsf{worksFor}^-\\
\mathsf{respFor} & \sqsubseteq & \mathsf{worksFor}\\
 & \vdots &
\end{array}
$$

DL-Lite$_\mathcal{R}$ / OWL 2 QL **cannot capture**:

- covering constraints – This would require **disjunction**
- identity between individuals – This would `owl:sameAs` – see later
- functionality of roles – This would require number restrictions

unibz

DL-Lite$_\mathcal{R}$ captures conceptual modeling formalisms

Modeling construct	*DL-Lite*	FOL formalization
ISA on classes	$A_1 \sqsubseteq A_2$	$\forall x(A_1(x) \rightarrow A_2(x))$
. . . and on relations	$R_1 \sqsubseteq R_2$	$\forall x, y(R_1(x, y) \rightarrow R_2(x, y))$
Disjointness of classes	$A_1 \sqsubseteq \neg A_2$	$\forall x(A_1(x) \rightarrow \neg A_2(x))$
. . . and of relations	$R_1 \sqsubseteq \neg R_2$	$\forall x, y(R_1(x, y) \rightarrow \neg R_2(x, y))$
Domain of relations	$\exists P \sqsubseteq A_1$	$\forall x(\exists y(P(x, y)) \rightarrow A_1(x))$
Range of relations	$\exists P^- \sqsubseteq A_2$	$\forall x(\exists y(P(y, x)) \rightarrow A_2(x))$
Mandatory participation *(min card = 1)*	$A_1 \sqsubseteq \exists P$ $A_2 \sqsubseteq \exists P^-$	$\forall x(A_1(x) \rightarrow \exists y(P(x, y)))$ $\forall x(A_2(x) \rightarrow \exists y(P(y, x)))$
. . .	. . .	. . .

unibz

Outline

1. Motivation
2. Semantic Web standards
   - Resource Description Framework
   - SPARQL
   - OWL 2 QL
   - Mapping assertions
3. OBDA framework
4. Ontop system and its ecosystem
5. Use Cases
6. Conclusion

Mapping assertions – RDB-RDF

Global-As-View (GAV) mapping assertion $\varphi \leadsto \psi$

- φ: FO query
- ψ: atom
- Open-World Assumption (by default)

Mapping assertions – RDB-RDF

Global-As-View (GAV) mapping assertion $\varphi \rightsquigarrow \psi$

- φ: FO query (over DB predicates)
- ψ: atom (over an RDF predicate)
- Open-World Assumption (by default)

unibz

Mapping assertions – RDB-RDF

Global-As-View (GAV) mapping assertion $\varphi \rightsquigarrow \psi$

- φ: FO query (over DB predicates)
- ψ: atom (over an RDF predicate)
- Open-World Assumption (by default)

Class instance (`:Student`)

Source	$\boldsymbol{q}(s) \leftarrow \texttt{uni1-student}(s, f, l)$
	
	`SELECT s_id`
	`FROM uni1.student`

Target	$\text{Student}(\text{URI}_1(s))$ `ex:uni1/student/{s_id} a :Student .`

unibz

Mapping assertions RDB-RDF

Ontop native format (similar to the R2RML standard)

Data property (`foaf:firstName`)

Source (SQL)	`SELECT s_id, firstName, lastName FROM uni1.student`
Target (RDF)	`ex:uni1/student/{s_id} foaf:firstName "{firstName}"^^xsd:string ; foaf:lastName "{lastName}"^^xsd:string .`

unibz

Mapping assertions RDB-RDF

Ontop native format (similar to the R2RML standard)

Data property (`foaf:firstName`)

Source (SQL)	`SELECT s_id, firstName, lastName FROM uni1.student`
Target (RDF)	`ex:uni1/student/{s_id} foaf:firstName "{firstName}"^^xsd:string ; foaf:lastName "{lastName}"^^xsd:string .`

Object property (`:teaches`)

Source	`SELECT * FROM "uni1"."teaching"`
Target (RDF)	`ex:uni1/academic/{a_id} :teaches ex:uni1/course/{c_id} .`

Mapping assertions RDB-RDF

Ontop native format (similar to the R2RML standard)

Data property (`foaf:firstName`)

Source (SQL)	`SELECT s_id, firstName, lastName FROM uni1.student`
Target (RDF)	`ex:uni1/student/{s_id} foaf:firstName "{firstName}"^^xsd:string ; foaf:lastName "{lastName}"^^xsd:string .`

Object property (`:teaches`)

Source	`SELECT * FROM "uni1"."teaching"`
Target (RDF)	`ex:uni1/academic/{a_id} :teaches ex:uni1/course/{c_id} .`

Magic number

Source	`SELECT * FROM "uni1"."academic" WHERE "position" = 1`
Target (RDF)	`ex:uni1/academic/{a_id} a :FullProf .`

Outline

1. Motivation
2. Semantic Web standards
3. OBDA framework
   - OBDA architecture
4. Ontop system and its ecosystem
5. Use Cases
6. Conclusion

unibz

Ontology-based data access: Formalization

To formalize OBDA, we distinguish between the intensional and the extensional level information.

An **OBDA specification** is a triple $\mathcal{P} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, where:

- $\mathcal{T}$ is the intensional level of an ontology.
  We consider ontologies formalized in description logics (DLs), hence the intensional level is a DL TBox.
- $\mathcal{S}$ is a (possibly federated) relational database schema for the data source(s), possibly with constraints;
- $\mathcal{M}$ is a set of mapping assertions between $\mathcal{T}$ and $\mathcal{S}$.

Ontology-based data access: Formalization

To formalize OBDA, we distinguish between the intensional and the extensional level information.

An **OBDA specification** is a triple $\mathcal{P} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, where:

- $\mathcal{T}$ is the intensional level of an ontology.
  We consider ontologies formalized in description logics (DLs), hence the intensional level is a DL TBox.
- $\mathcal{S}$ is a (possibly federated) relational database schema for the data source(s), possibly with constraints;
- $\mathcal{M}$ is a set of mapping assertions between $\mathcal{T}$ and $\mathcal{S}$.

An **OBDA instance** is a pair $\mathcal{O} = \langle \mathcal{P}, \mathcal{D} \rangle$, where

- $\mathcal{P} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ is an OBDA specification, and
- $\mathcal{D}$ is a relational database compliant with $\mathcal{S}$.

Semantics of OBDA: Intuition

In an OBDA instance $\mathcal{O} = \langle\langle\mathcal{T}, \mathcal{M}, \mathcal{S}\rangle, \mathcal{D}\rangle$, the **mapping** $\mathcal{M}$ encodes how the data $\mathcal{D}$ in the source(s) $\mathcal{S}$ should be used to populate the elements of $\mathcal{T}$.

unibz

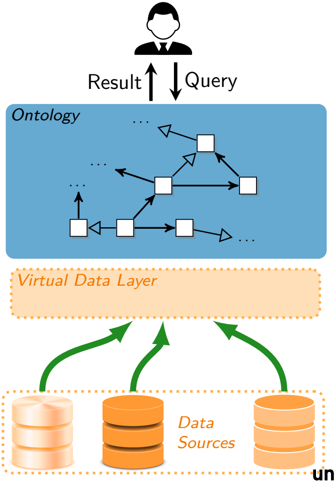
Semantics of OBDA: Intuition

In an OBDA instance $\mathcal{O} = \langle\langle\mathcal{T}, \mathcal{M}, \mathcal{S}\rangle, \mathcal{D}\rangle$, the **mapping** $\mathcal{M}$ encodes how the data $\mathcal{D}$ in the source(s) $\mathcal{S}$ should be used to populate the elements of $\mathcal{T}$.

Virtual data layer

The data $\mathcal{D}$ and the mapping $\mathcal{M}$ define a virtual data layer $\mathcal{V} = \mathcal{M}(\mathcal{D})$

- Queries are answered w.r.t. $\mathcal{T}$ and $\mathcal{V}$.
- We do not really materialize the data of $\mathcal{V}$ (it is virtual!).
- Instead, the intensional information in $\mathcal{T}$ and $\mathcal{M}$ is used to translate queries over $\mathcal{T}$ into queries formulated over $\mathcal{S}$.



unibz

Semantics of mappings

To formally define the semantics of an OBDA instance $\mathcal{O} = \langle \mathcal{P}, \mathcal{D} \rangle$, where $\mathcal{P} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, we first need to define the semantics of mappings.

Satisfaction of a mapping assertion with respect to a database

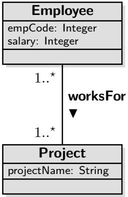
An interpretation $\mathcal{I}$ **satisfies** a mapping assertion $\Phi(\vec{x}) \rightsquigarrow \Psi(\vec{x})$ in $\mathcal{M}$ **with respect to a database** $\mathcal{D}$ **for** $\mathcal{S}$, if the following FOL sentence is true in $\mathcal{I} \cup \mathcal{D}$:

$$\forall \vec{x}.\, \Phi(\vec{x}) \rightarrow \Psi(\vec{x})$$

Intuitively, $\mathcal{I}$ **satisfies** $\Phi \rightsquigarrow \Psi$ w.r.t. $\mathcal{D}$ if all facts obtained by evaluating Φ over $\mathcal{D}$ and then propagating the answers to Ψ, hold in $\mathcal{I}$.

unibz

Semantics of mappings – Example



D_1:

SSN	*PrName*
23AB	optique
...	...

D_2:

Code	*Salary*
e23	1500
...	...

D_3:

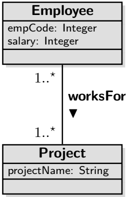
Code	*SSN*
e23	23AB
...	...

m_1:
```
SELECT SSN, PrName
FROM D1
```
$\rightsquigarrow$ Employee(**pers**(*SSN*)),
Project(**proj**(*PrName*)),
projectName(**proj**(*PrName*), *PrName*),
worksFor(**pers**(*SSN*), **proj**(*PrName*))

m_2:
```
SELECT SSN, Salary
FROM D2, D3
WHERE D2.Code = D3.Code
```
$\rightsquigarrow$ Employee(**pers**(*SSN*)),
salary(**pers**(*SSN*), *Salary*)

unibz

Semantics of mappings – Example



D_1:

SSN	*PrName*
23AB	optique
...	...

D_2:

Code	*Salary*
e23	1500
...	...

D_3:

Code	*SSN*
e23	23AB
...	...

The following interpretation $\mathcal{I}$ satisfies the mapping assertions m_1 and m_2 with respect to the above database:

$$\mathcal{I} : \Delta_O^{\mathcal{I}} = \{\textbf{pers}(\texttt{23AB}), \textbf{proj}(\texttt{optique}), \ldots\}, \quad \Delta_V^{\mathcal{I}} = \{\texttt{optique}, \texttt{1500}, \ldots\}$$

$$\text{Employee}^{\mathcal{I}} = \{\textbf{pers}(\texttt{23AB}), \ldots\}, \quad \text{Project}^{\mathcal{I}} = \{\textbf{proj}(\texttt{optique}), \ldots\},$$

$$\text{projectName}^{\mathcal{I}} = \{(\textbf{proj}(\texttt{optique}), \texttt{optique}), \ldots\},$$

$$\text{worksFor}^{\mathcal{I}} = \{(\textbf{pers}(\texttt{23AB}), \textbf{proj}(\texttt{optique})), \ldots\},$$

$$\text{salary}^{\mathcal{I}} = \{(\textbf{pers}(\texttt{23AB}), \texttt{1500}), \ldots\}$$

m_1:

```
SELECT SSN, PrName
FROM D1
```

$\rightsquigarrow$ Employee(**pers**(*SSN*)), Project(**proj**(*PrName*)), projectName(**proj**(*PrName*), *PrName*), worksFor(**pers**(*SSN*), **proj**(*PrName*))

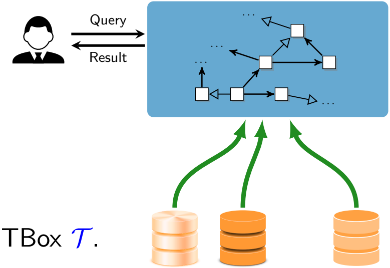
m_2:

```
SELECT SSN, Salary
FROM D2, D3
WHERE D2.Code = D3.Code
```

$\rightsquigarrow$ Employee(**pers**(*SSN*)), salary(**pers**(*SSN*), *Salary*)

unibz

Semantics of an OBDA instance



Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation of the TBox $\mathcal{T}$.

Model of an OBDA instance

$\mathcal{I}$ is a **model** of $\mathcal{O} = \langle \mathcal{P}, \mathcal{D} \rangle$, with $\mathcal{P} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ if:

- $\mathcal{I}$ is a model of $\mathcal{T}$, and
- $\mathcal{I}$ satisfies $\mathcal{M}$ w.r.t. $\mathcal{D}$, i.e., it satisfies every assertion in $\mathcal{M}$ w.r.t. $\mathcal{D}$.

An OBDA instance $\mathcal{O}$ is **satisfiable** if it admits at least one model.

Outline

unibz

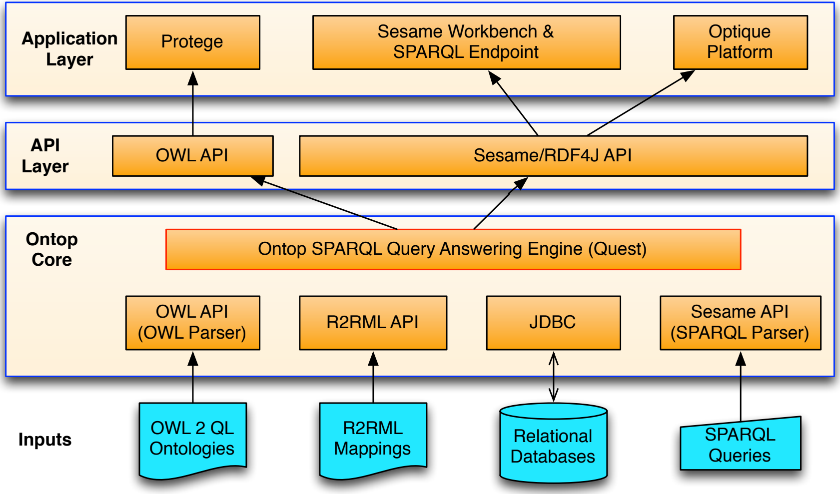
Ontop

ontop

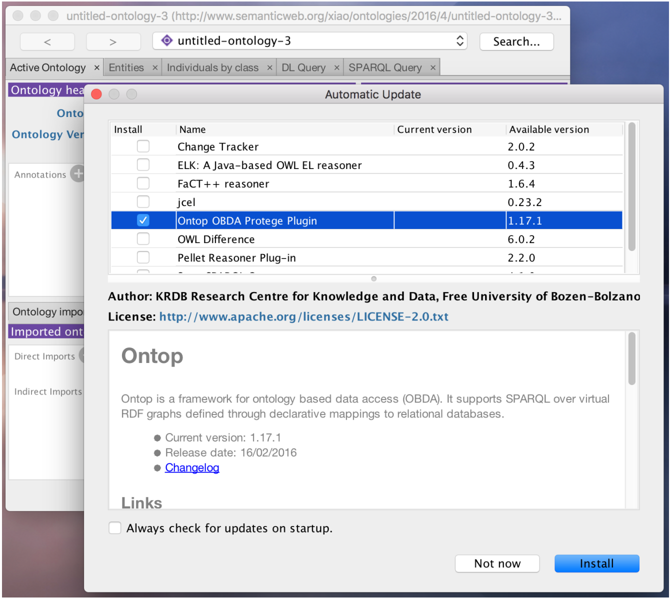
- State-of-the-art OBDA system
- Compliant with the RDFS, OWL 2 QL, R2RML, and SPARQL standards.
- Supports all major relational DBs
- Open-source and released under Apache 2 license
- Development of *Ontop*:
  - development started 6 years ago
  - already well established:
    - +200 members in the mailing list
    - +7000 downloads in last 18 months
  - main development carried out in the context of the EU project Optique

unibz

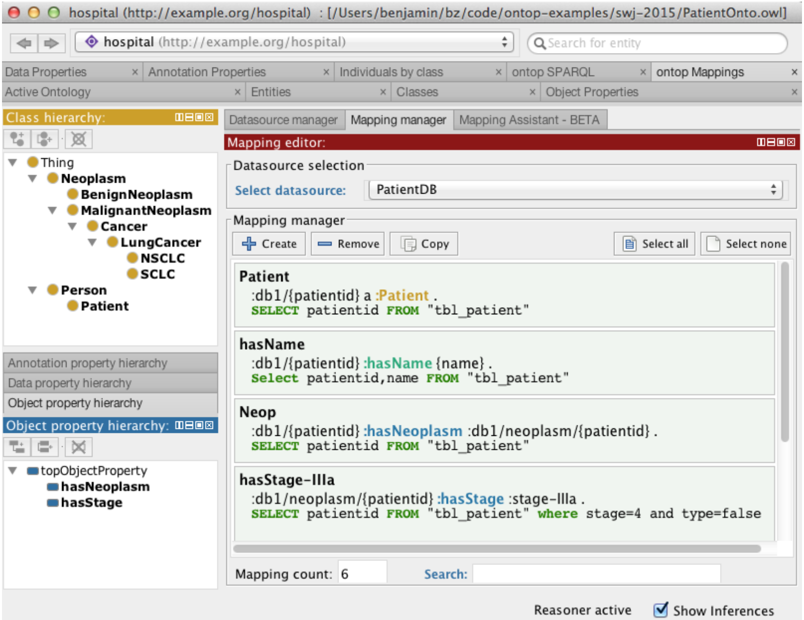
Architecture of *Ontop*



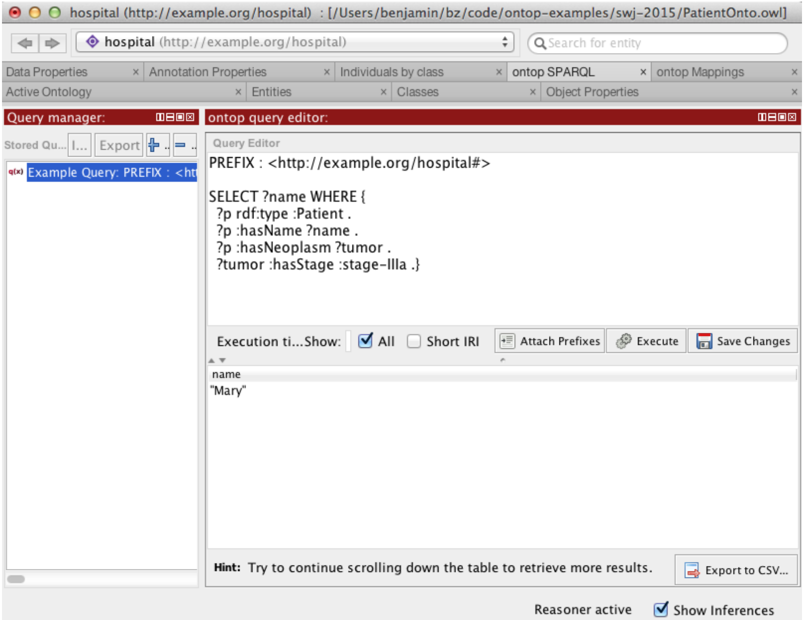
Ontop plugin available from Protégé plugin repository



Mapping editor in Protégé

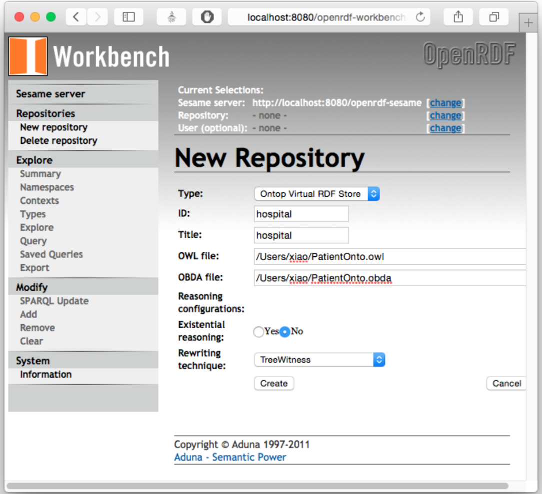


SPARQL query answering in Protégé



Sesame/RDF4J Workbench and SPARQL endpoint

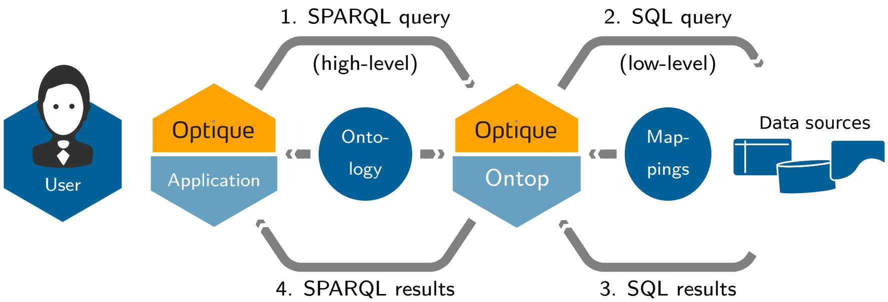
- Sesame Workbench extension to create and manage *Ontop* repositories
- Standard HTTP SPARQL endpoints.



Optique system

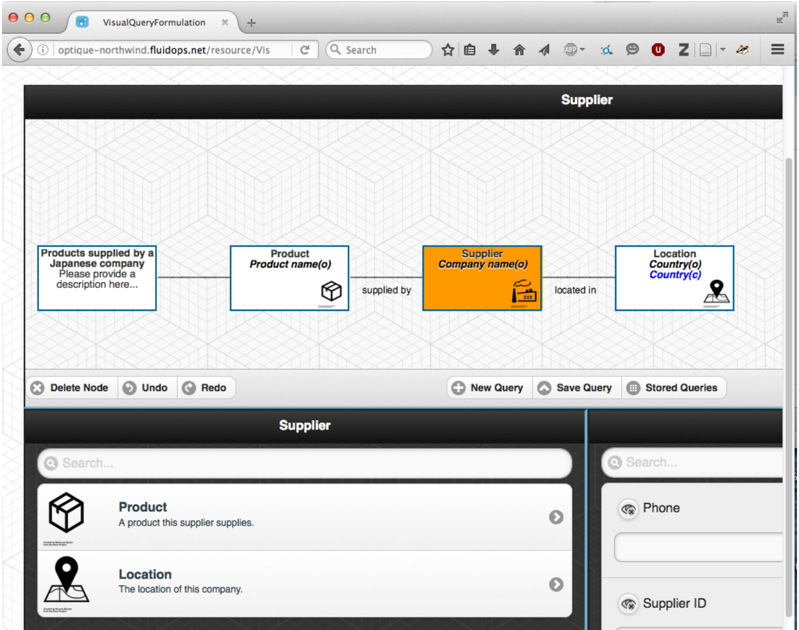
EU FP7 Project Optique: *Scalable End-user Access to Big Data*

- November 2012 – October 2016
- 10 Partners, including industrial partners Statoil, Siemens, DNV
- *Ontop* is core component of the Optique platform



Visual query formulation (Optique VQS)

`http://optique-northwind.fluidops.net` demo/demo



Commercial adoption

Ontop is adopted by several commercial tools.

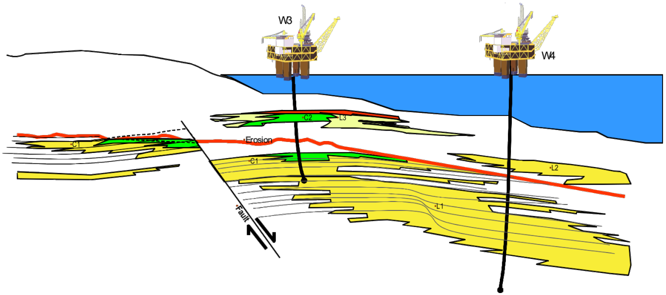
- Stardog by Complexible Inc.
- Fluidops Information Workbench
- Metaphacts semantic data management platform

Outline

1. Motivation
2. Semantic Web standards
3. OBDA framework
4. Ontop system and its ecosystem
5. Use Cases
6. Conclusion

Statoil use case [DBLP:conf/semweb/KharlamovHJLLPR15]

- Optique use case partner
- Exploration domain: analyse existing relevant data in order to find exploitable accumulations of oil or gas
- Improve the efficiency of the information gathering routine for geologists
- Efficient, creative data collection from multiple large volume data sources

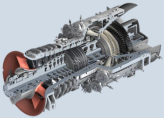


unibz

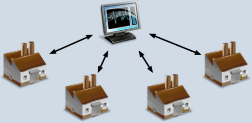
Siemens use case [KharlamovSOZHLRSW-iswc14]

- Optique use case partner
- Siemens energy department
- streaming and temporal data

- Siemens
  - produces huge appliances, e.g., turbines
  - installs them in plants

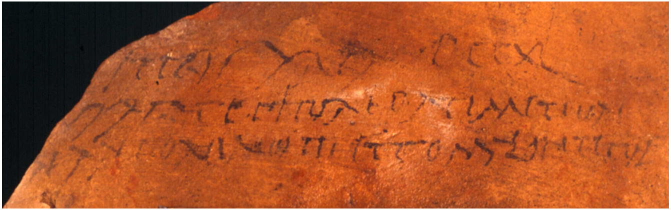


- Siemens service centers
  - offers constant monitoring and diagnostics service
  - over 50 service centers world wide
  - each SC is responsible for several thousands appliances
  - their job: monitoring and diagnostic of turbine
- Monitoring and diagnostic tasks
  - reactive and preventive diagnostics
    - offline, after an issue is detected
  - predictive analyses
    - real-time, to avoid issues



EPNet project [EAAI-2016]

- Ontology-based data integration for humanities and archaeologists
- ERC advanced grant EPNet "Production and distribution of food during the Roman Empire: Economics and Political Dynamics".
- Linking three datasets:
    1. the EPNet relational repository
    2. the Epigraphic Database Heidelberg
    3. the Pleiades dataset



Melodies project

- EU FP7 Melodies project: working with Open Data, 16 partners.
- Geospatial extension *Ontop*-spatial used for accessing geospatial data
- Use cases: urban development, land management, disaster management

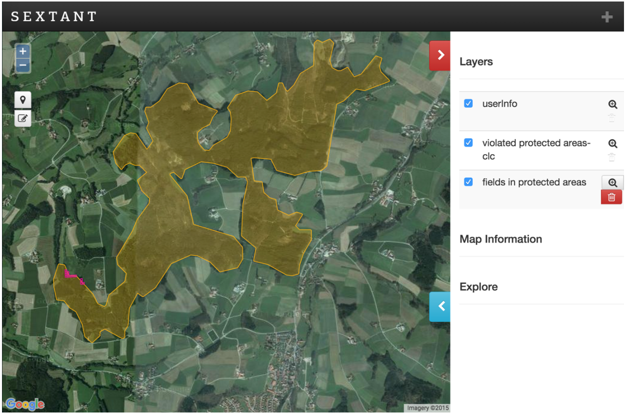


Figure: Visualization of violated protected areas in land management

unibz

EMSec project [2016-eswc-maritime]

- German BMBF project EMSec: real-time services for maritime security
- SPARQL federation to access different kinds of data sources:
  - SPARQL endpoints of *Ontop* over *in situ* data
  - open SPARQL endpoints: Geonames, DBPedia

unibz

Outline

Conclusion

Main message: we need high-level access to data

1. SQL queries over tables can be difficult to write manually (low-level)
2. OBDA is a powerful solution for high-level data access
3. Ontop is an open-source OBDA framework

Ontop Team

In Bolzano

- Diego Calvanese (Scientific coordinator)
- Guohui Xiao (Leader of the development team)
- Benjamin Cogrel
- Elena Botoeva
- Vladislav Ryzhikov
- Davide Lanti (PhD student)
- Elem Güzel (PhD student)
- Sarah Komla Ebri
- Ugur Donmez

External collaborators

- Konstantina Bereta (National and Kapodistrian Univ. of Athens, Greece)
- Dag Hovland (University of Oslo, Norway)
- Roman Kontchakov (Birkbeck College London, UK)
- Martin Rezk (Rakuten, Japan)
- Mariano Rodríguez-Muro (IBM Watson Research Center, USA)
- Michael Zakharyaschev (Birkbeck College London, UK)

Semantic web journal paper

- **Ontop: Answering SPARQL Queries over Relational Databases.** D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao. Semantic Web Journal. 2016

Figure: The Semantic Web journal 2016 Outstanding Paper Award

unibz

- *Ontop* website: `http://ontop.inf.unibz.it/`
- Github: `http://github.com/ontop/ontop/`
- Facebook: `https://www.facebook.com/obdaontop/`
- Twitter: @ontop4obda
- *Optique* website: `http://optique-project.eu/`
- Tutorial: `https://github.com/ontop/ontop-examples/tree/master/ekaw-tutorial-2016`

References I