INF3580/4580 – Semantic Technologies – Spring 2017

Lecture 14: Introduction to Ontology-based Data Access (OBDA)

Ernesto Jimenez Ruiz

15th May 2017

Department of Informatics

University of Oslo

Today's Plan

Outline

RDF

- Why URIs?
  - URIs naturally have a "global" scope, unique throughout the web.
  - URLs are also addresses.
  - *"A web of data."*
- Why triples?
  - Any information format can be transformed to triples.
  - Relationships are made explicit and are elements in their own right

RDF on the web: Where is it?

- In files:
  - In some serialisation format: XML/RDF, Turtle, ...
  - Typically small RDF graphs, i.e., max. a few 100 triples, e.g.,
    - Vocabularies: `http://xmlns.com/foaf/spec/index.rdf`.
    - Tiny datasets: `http://folk.uio.no/martingi/foaf.rdf`.
- "Behind" *SPARQL endpoints*:
  - Data kept in a *triple store*, i.e., a database of triples.
  - RDF is served from endpoint as results of *SPARQL queries*.
  - Exposes data (in different ways)
    - with endpoint frontends, e.g., `http://dbpedia.org/resource/Norway`, or
    - by direct SPARQL query: `http://dbpedia.org/sparql`.

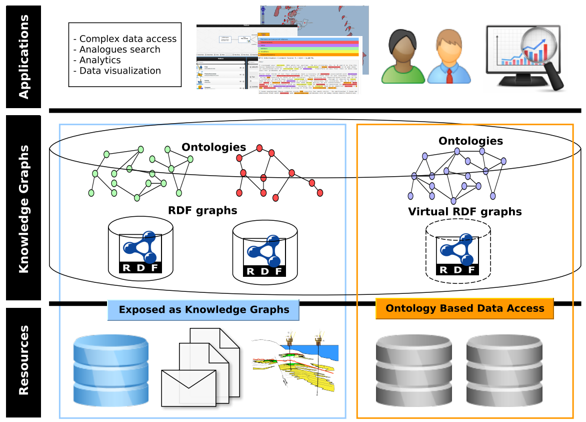
RDF on the web: Where is it?

- "Behind" *OBDA repositories*:
  - OBDA: Ontology-based Data Access
  - Data kept in a traditional *relational database*
  - Access is transparent via SPARQL queries
  - SPARQL queries are "internally" transformed to SQL queries
  - An RDF representation of the relational database is "virtualized" to answer the SPARQL query.

Exposing data as RDF

- **Virtual exposure of data** (OBDA)
  - ✓ End-users' friendly access to "unfriendly" relational data
  - ✓ Pay as you go data integration
  - × Requires an ontology in OWL 2 QL
  - × Data remains in old-fashioned databases
- **Data Export**
  - ✓ Easy to exchange data (over the Web)
  - ✓ Ontology not limited to OWL 2 QL
  - × Data replication
  - × Due to size or privacy it may not be possible to export the data

Exposing data as RDF



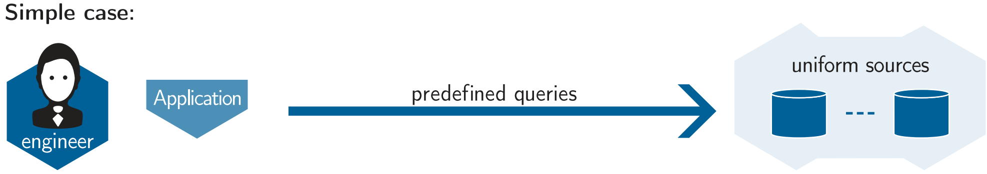
Outline

EU project Optique

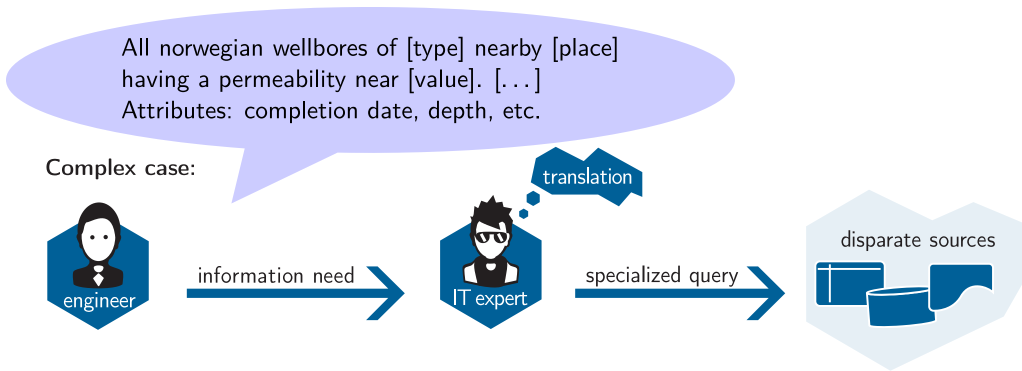
- EU Project from 2012-2016
- Aimed at facilitating **scalable end-user access to big data** in the oil and gas industry.
- Advocated for an **OBDA approach**
  - ontology provides a virtual access to the data
  - mappings connect the ontology with the data source.
- Focused around two demanding use cases provided by the industry partners **Siemens** and **Statoil**
- Currently takes 30-70% of engineers' time (e.g., more than 250 MNOK annually).

Limitations

Simple case:



Limitations



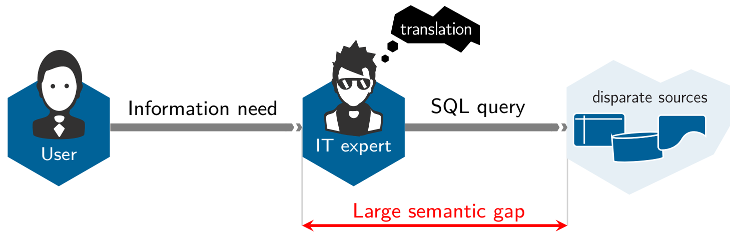
Takes 4 days in average

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))
```


Limitations



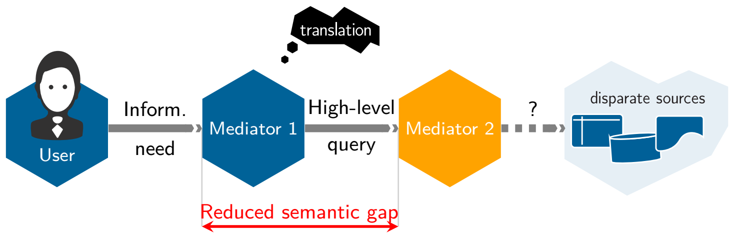
Querying over RDB requires a lot of knowledge about:

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

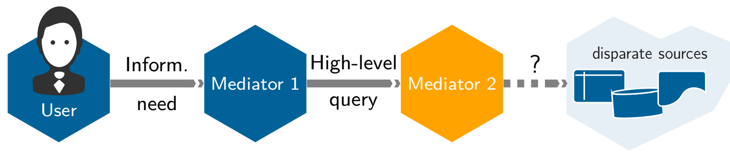
High-level translation solution

General approach: two steps

1. Translate the information needs into a **high-level (formal) query**
    1. *Mediator 1* could be a user, an IT expert or a GUI
    2. Make such a translation easy *(Ideally: IT expertise not required)*
2. Answer the high-level query **automatically** using *Mediator 2*



Two orthogonal choices to be made



Choice 1: Generating a new representation of the data

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

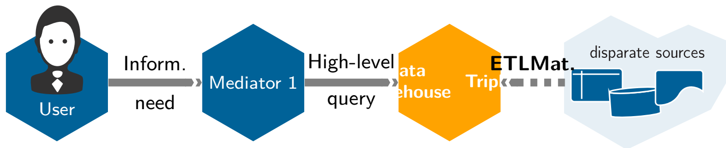
Choice 2: Which data format for the new representation

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) / Materialization

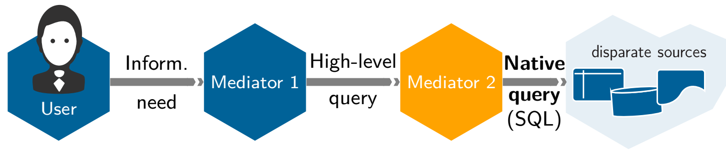
E.g., relational data warehouse, triplestore (RDF)



Generating a new representation of the data

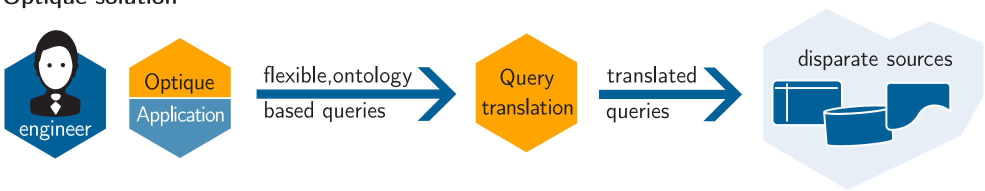
2. Virtual views

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



Optique solution: Ontology-Based Data Access (OBDA)

Optique solution



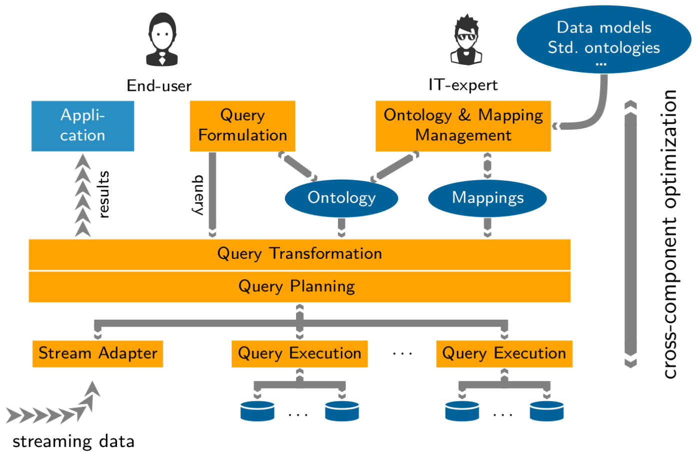
Choice 1: Generating a new representation of the data

1. Virtual views

Choice 2: Which data format for the virtual view

1. Resource Description Framework (RDF)

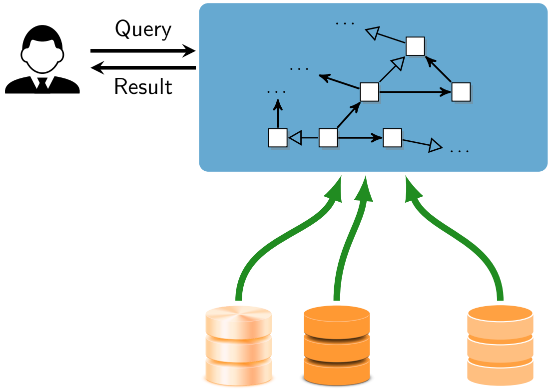
Optique architecture



Outline

Outline

OBDA framework



Ontology
provides global vocabulary and conceptual view

Mappings
how to populate the ontology from the data

Data Sources
external and heterogeneous

OBDA framework

Logical transparency in accessing data:

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

OBDA Ingredients

- Relies on. . .
  - **ontology** to provide a virtual access to the data
  - **mappings** to connect the ontology with the data

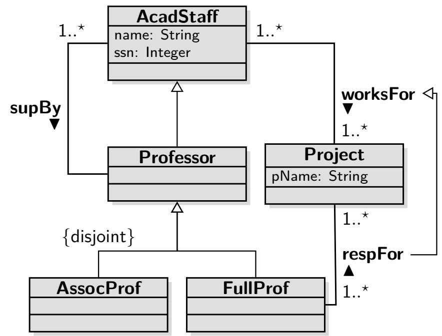
- Required infrastructure
  - **Query formulation system** to express the information needs in SPARQL (Mediator 1)
  - **Query transformation/rewriting** system to covert from SPARQL to (native) SQL (Mediator 2)
  - **Ontology and mapping bootstrapper**

Outline

The Ontology: OWL 2 QL profile

- OWL 2 QL is one of the three standard profiles of OWL 2.
- 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.
- The ontology data (ABox) in an OBDA setting is (usually) implicitly defined through the database and mappings.

Capturing UML class diagrams/ER schemas in OWL 2 QL



$$\begin{aligned}
\text{Professor} &\sqsubseteq \text{AcadStaff}\\
\text{AssocProf} &\sqsubseteq \text{Professor}\\
\text{FullProf} &\sqsubseteq \text{Professor}\\
\text{AssocProf} &\sqsubseteq \neg\text{FullProf}\\
\text{AcadStaff} &\sqsubseteq \exists\text{ssn}\\
\exists\text{ssn} &\sqsubseteq \text{AcadStaff}\\
\exists\text{ssn}^- &\sqsubseteq \texttt{Integer}\\
\exists\text{worksFor} &\sqsubseteq \text{AcadStaff}\\
\exists\text{worksFor}^- &\sqsubseteq \text{Project}\\
\text{AcadStaff} &\sqsubseteq \exists\text{worksFor}\\
\text{Project} &\sqsubseteq \exists\text{worksFor}^-\\
\text{respFor} &\sqsubseteq \text{worksFor}
\end{aligned}$$

Outline

OBDA Mappings

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`)

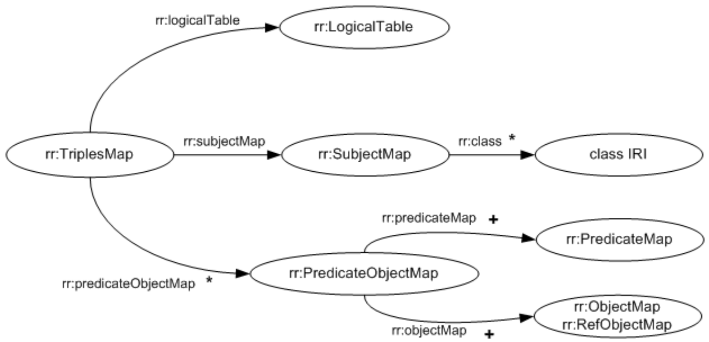
Source	$q(s) \leftarrow \texttt{uni1-student}(s, f, l)$
	 `SELECT s_id`
	 `FROM uni1.student`

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

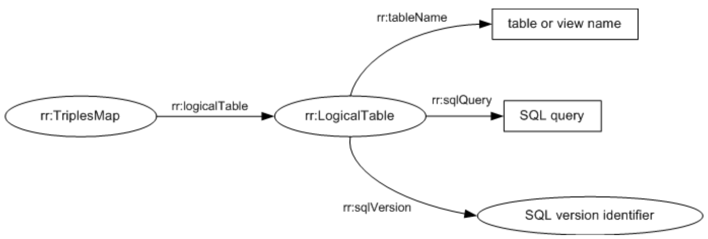
OBDA Mappings: R2RML

- R2RML is a W3C recommended RDB-to-RDF mapping language
  - `https://www.w3.org/TR/r2rml/`
- Generates RDF triples from a relational database based on specific mappings
- The mappings are specified in Turtle syntax
- The R2RML mapping is an RDF graph consisting of several rr:TriplesMaps
  - how to map a logical table in the input relational database into RDF

OBDA Mappings: R2RML



OBDA Mappings: R2RML



OBDA Mappings: R2RML example

Triples map to populate Student class"

```
<TriplesMap1>       a rr:TriplesMapClass;
   rr:logicalTable [rr:SQLQuery "Select s_id, name, from STUDENT"];
   rr:subjectMap
   [
      rr:template "http://example.com/uni1/student/{s_id}";
      rr:class ex:Student;
    ] ;
   rr:predicateObjectMap
   [
      rr:predicate foaf:name;
      rr:objectMap [ rr:column "name" ]
   ] .
```


OBDA Mappings: R2RML example

Table STUDENT:

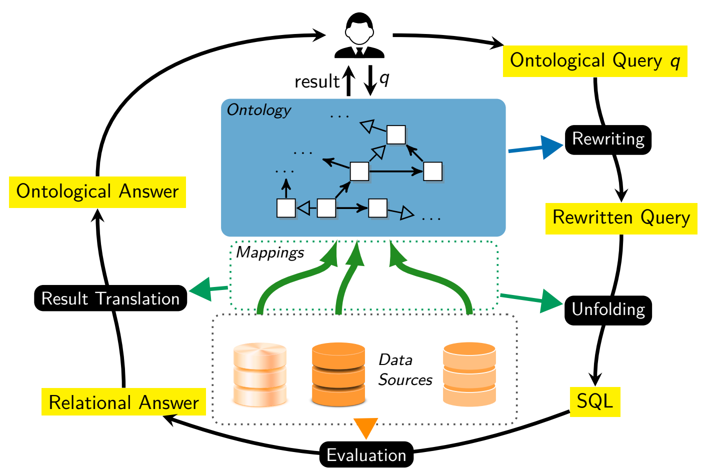
s_id	name
1	Ernesto
2	Martin
3	Leif

Triples:

```
http://example.com/uni1/student/1   foaf:name   Ernesto
http://example.com/uni1/student/2   foaf:name   Martin
http://example.com/uni1/student/3   foaf:name   Leif
```


Outline

Query answering by rewriting

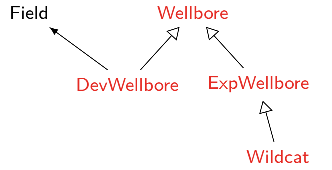


Query answering by rewriting (example)

Database:

```
wlb_dev(name, ...)
wlb_exp(name, purpose, ...)
```

Ontology:



Mappings:

DevWellbore(name) ↦ `SELECT name FROM wlb_dev`

ExpWellbore(name) ↦ `SELECT name FROM wlb_exp`

Wildcat(name) ↦ `SELECT name FROM wlb_exp WHERE purpose = 'WILDCAT'`

Query: *List all wellbores.*

q: Wellbore(x)

q_O: Wellbore(x) $\cup$ DevWellbore(x) $\cup$ ExpWellbore(x) $\cup$ Wildcat(x)

q_{SQL}:

```
SELECT name FROM wlb_dev
UNION
SELECT name FROM wlb_exp
UNION
SELECT name FROM wlb_exp WHERE
purpose = 'WILDCAT'
```

q'_{SQL}:

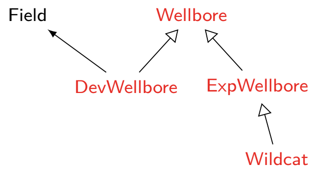
```
SELECT name FROM wlb_dev UNION
SELECT name FROM wlb_exp
```


Query answering by rewriting (example)

Database:

wlb_dev(name, ...)
wlb_exp(name, purpose, ...)

Ontology:



Mappings:

DevWellbore(name) ↦ SELECT name FROM wlb_dev

ExpWellbore(name) ↦ SELECT name FROM wlb_exp

Wildcat(name) ↦ SELECT name FROM wlb_exp WHERE purpose = 'WILDCAT'

Query: *List all wellbores.*

q: Wellbore(x)

q_O: Wellbore(x) $\cup$ DevWellbore(x) $\cup$ ExpWellbore(x) $\cup$ Wildcat(x)

q_{SQL}: SELECT name FROM wlb_dev UNION SELECT name FROM wlb_exp UNION SELECT name FROM wlb_exp WHERE purpose = 'WILDCAT'

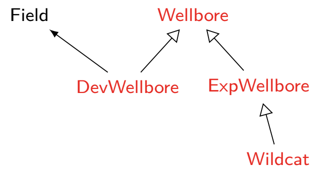
q'_{SQL}: SELECT name FROM wlb_dev UNION SELECT name FROM wlb_exp

Query answering by rewriting (example)

Database:

wlb_dev(name, ...)
wlb_exp(name, purpose, ...)

Ontology:



Mappings:

DevWellbore(name) ↦ SELECT name FROM wlb_dev

ExpWellbore(name) ↦ SELECT name FROM wlb_exp

Wildcat(name) ↦ SELECT name FROM wlb_exp WHERE purpose = 'WILDCAT'

Query: *List all wellbores.*

q: Wellbore(x)

q_O: Wellbore(x) $\cup$ DevWellbore(x) $\cup$ ExpWellbore(x) $\cup$ Wildcat(x)

q_{SQL}: SELECT name FROM wlb_dev UNION SELECT name FROM wlb_exp UNION SELECT name FROM wlb_exp WHERE purpose = 'WILDCAT'

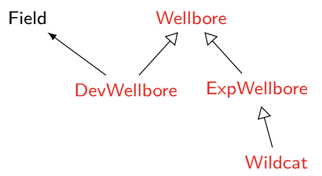
q'_{SQL}: SELECT name FROM wlb_dev UNION SELECT name FROM wlb_exp

Query answering by rewriting (example)

Database:

```
wlb_dev(name, ...)
wlb_exp(name, purpose, ...)
```

Ontology:



Mappings:

DevWellbore(name) $\mapsto$ `SELECT name FROM wlb_dev`

ExpWellbore(name) $\mapsto$ `SELECT name FROM wlb_exp`

Wildcat(name) $\mapsto$ `SELECT name FROM wlb_exp WHERE purpose = 'WILDCAT'`

Query: *List all wellbores.*

q: Wellbore(x)

q_O: Wellbore(x) $\cup$ DevWellbore(x) $\cup$ ExpWellbore(x) $\cup$ Wildcat(x)

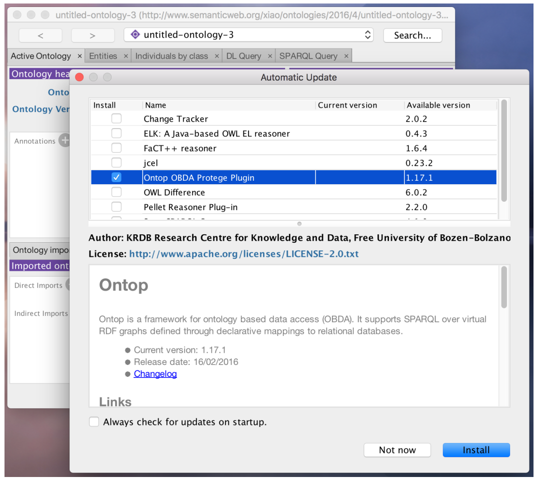
q_{SQL}:
```
SELECT name FROM wlb_dev
UNION
SELECT name FROM wlb_exp
UNION
SELECT name FROM wlb_exp WHERE
purpose = 'WILDCAT'
```

q'_{SQL}:
```
SELECT name FROM wlb_dev UNION
SELECT name FROM wlb_exp
```

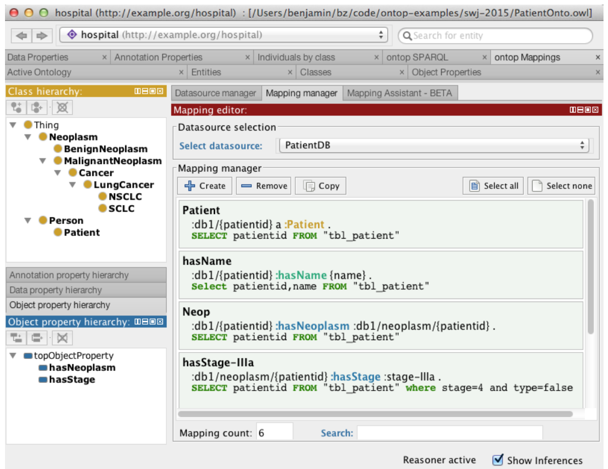

Query answering by rewriting (tool support)

- **Ontop**: state-of-the-art OBDA system. `http://ontop.inf.unibz.it/`
- Compliant with the RDFS, OWL 2 QL, R2RML, and SPARQL standards.
- Supports all major relational DBs
  - Oracle, DB2, MS SQL Server, Postgres, MySQL, Teiid, Exareme, etc.
- Open-source and released under Apache 2 license
- Development of -ontop-:
  - Development started in 2009
  - -ontop- supports (essentially) all features of SPARQL 1.0 and the OWL 2 QL entailment regime of SPARQL 1.1.
  - Other features of SPARQL 1.1 (e.g., aggregates, property path queries, negation) are work in progress.

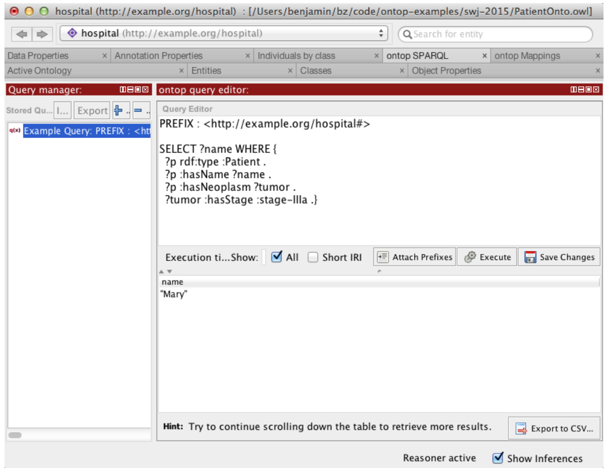
Ontop plugin available from Protégé



Ontop: Mapping editor in Protégé



Ontop: SPARQL query answering in Protégé



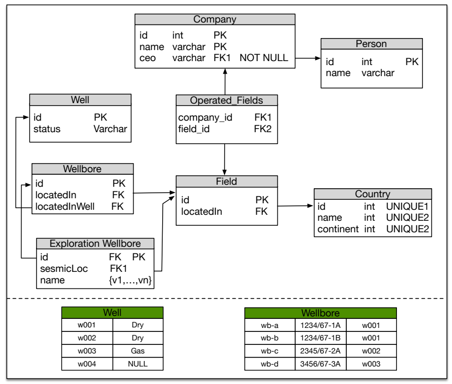
Outline

Bootstrapping overview

- Given a relational database (semi)automatically extracts ontological vocabulary, ontology and R2RML mappings
- Bootstrappers may also accept as input an ontology
  - R2RML mappings will link the given ontoloy to the database, or
  - The given ontology will be *aligned* with the boostrapped ontology
- Type of mappings:
  - W3C direct mapping specification (*schema driven*)
    - To a given or bootstrapped ontology vocabulary
  - Mappings beyond direct ones (*data driven*)
    - Clusters of tuples
    - Joinable tuples

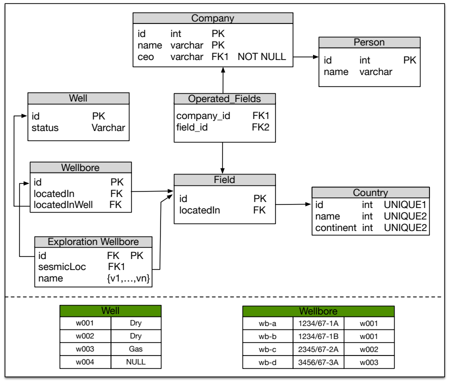
Bootstrapping: Vocabulary Generation

- **W3C directives**
  - Tables → classes
  - Foreign Keys → object properties
  - Data columns → data properties
  - Binary tables → fresh object properties
- **Attribute naming schema:**
  - Unique names (e.g. Person.name)
  - Reusable names (e.g. name)



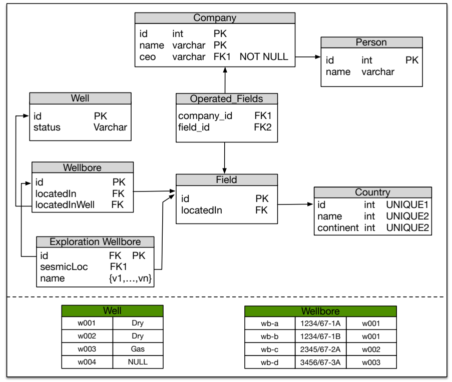
Bootstrapping: Axiom Generation

- **OWL 2 expressiveness**
  - OWL 2 QL (e.g. OBDA/Optique)
  - OWL 2 EL (e.g. EOLO)
  - OWL 2 RL (e.g. RDFox)
  - OWL 2 (e.g. PAGOdA, HermiT)



Bootstrapping: Axiom Generation

- Unique constraints
  - Person **HasKey:** id (OWL 2 RL/EL)
- Global onto. constraints
  - **Functional:** Person.name (OWL 2 RL)
  - name **Domain:** Person (all profiles)
  - Person.name **Range:** xsd:string (all profiles)
- Local onto. constraints
  - Person **subclassOf:** name **some** xsd:string (OWL 2 QL/EL)
  - Person **subclassOf:** name **only** xsd:string (OWL 2 RL)
  - Person **subclassOf:** name **exactly 1** xsd:string (OWL 2)



Bootstrapping: Datatypes

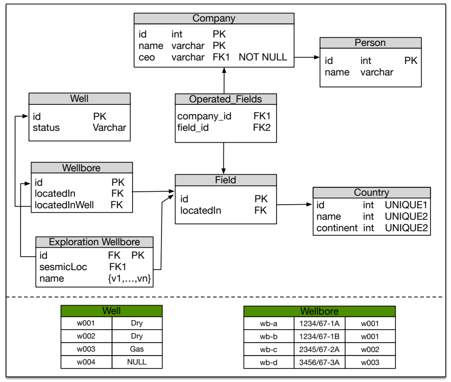
- Clear mapping between SQL and XML schema datatypes
- Not all XML Schema datatypes are included in OWL 2
  - *xsd:date* not in OWL 2
  - *xsd:boolean* and *xsd:double* not in OWL 2 QL/EL
- Value spaces of primitive datatypes are disjoint (e.g. *xsd:double* and *xsd:decimal*)

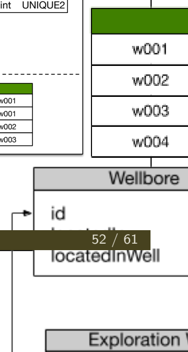
× Problems when materalizing the data or using the ontology for virtual access to data

✓ Solution: *rdfs:Literal*

Bootstrapping: Taxonomy Generation

- **Data driven**
  - Clusters of tuples
  - Joinable tuples
  - e.g. Well_Dry **SubclassOf:** Well (w001, w002)
  - e.g. Well_with_Wellbore **SubclassOf:** Well (w001, w002, w003)
- **Schema driven**
  - A single-column Foreign Key and Primary key
  - e.g. Exploration_Welllbore **SubclassOf:** Wellbore



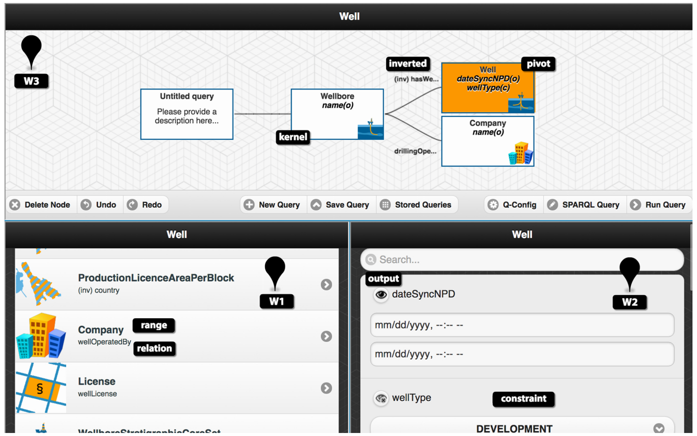


Tool support and Lessons learnt

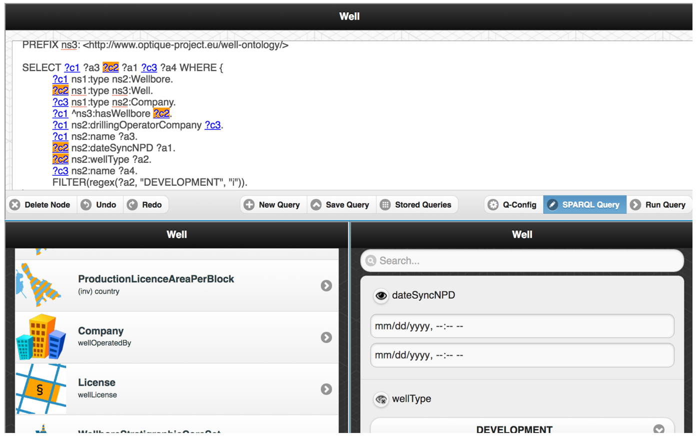
- **BootOX**: `http://www.cs.ox.ac.uk/isg/tools/BootOX/`
- **Feedback from use cases and evaluation. . .**
  - ✓ Good as a first approximation of the ontology and mappings
  - ✓ Competitive results in (academic) benchmarks
  - × For the largest Statoil datasources, the solution is far from perfect
  - × Ontology close to the original database
  - × Large amount of ontology entities and R2RML mappings
  - ✓ Implementation of an incremental/interactive bootstrapping
- **Future work. . .**
  - New ways to exploit the data and schema (e.g. views)

Outline

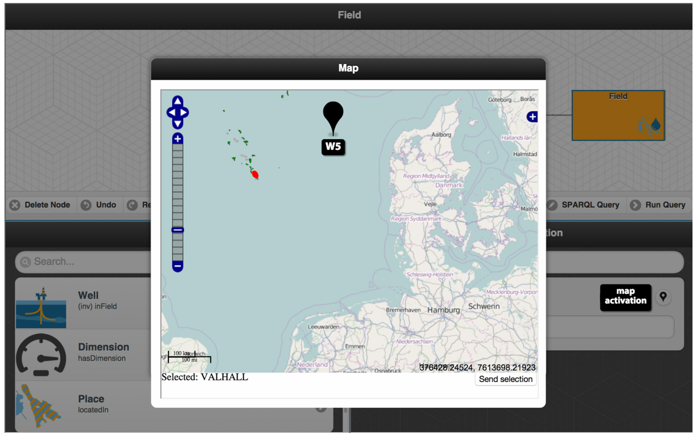
Visual query formulation (OptiqueVQS)



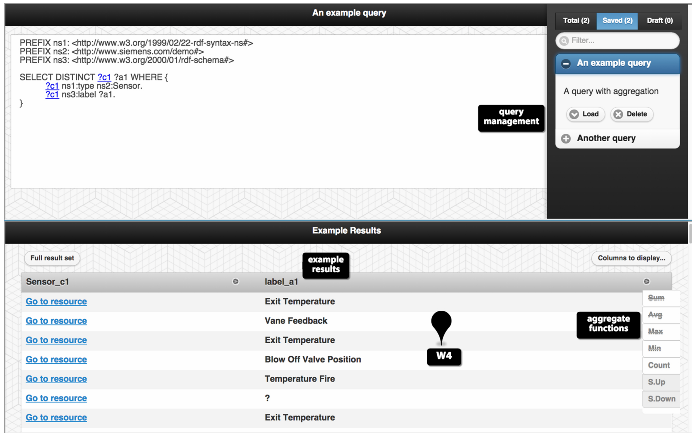
Visual query formulation (OptiqueVQS)



Visual query formulation (OptiqueVQS)



Visual query formulation (OptiqueVQS)



Outline

Optique infrastructure

- Training material:
  - `http://optique-northwind.fluidops.net` (demo/demo)
- OptiqueVQS can be tested online
- Local installation possible (academic license):
  - `https://appcenter.fluidops.com/resource/Search?search=optique`
- What is next?
  - *SIRIUS*: `http://sirius-labs.no/`

Questions?

Ernesto Jiménez-Ruiz (ernestoj@ifi.uio.no)
Office hours: from 9:00 to 16:00 at OJD 8165

Acknowledgements

- -ontop- team (Diego Calvanese, Benjamin Cogrel, Guohui Xiao, etc.) and Dag Hovland for sharing their wonderful slides
  - http://ontop.inf.unibz.it/ekaw-2016-tutorial/
- Ahmet Soylu: main person behind OptiqueVQS
- Optique partners