Exposing data as 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:**
  - XML/RDF, Turtle, …
  - Typically small RDF graphs, i.e., max. a few 100 triples, e.g.,
    - 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.

- **“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.

---

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
Outline

1 Exposing data as RDF
2 Data Access in Statoil: limitations and solutions
3 OBDA Ingredients
   - Overview
   - Ontology
   - Mappings
   - Query rewriting
   - Bootstrapping
   - Visual Query Formulation
   - Optique

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:

<table>
<thead>
<tr>
<th>Application</th>
<th>predefined queries</th>
<th>uniform sources</th>
</tr>
</thead>
</table>

Complex case:

<table>
<thead>
<tr>
<th>engineer</th>
<th>information need</th>
<th>IT expert</th>
<th>specialized query</th>
</tr>
</thead>
<tbody>
<tr>
<td>translation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Takes 4 days in average
### Data Access in Statoil: Limitations and solutions

#### A typical query at Statoil

**Anonymized extract**

```
WHERE db_name.table16 table16,
db_name.table15 table15,
db_name.table14 table14,
db_name.table13 table13,
db_name.table12 table12,
db_name.table11 table11,
db_name.table10a table10a,
db_name.table10b table10b,
db_name.table10c table10c,
db_name.table9 table9,
db_name.table8 table8,
db_name.table7a table7a,
db_name.table7b table7b,
db_name.table6a table6a,
db_name.table6b table6b,
db_name.table5a table5a,
db_name.table5b table5b,
db_name.table4a table4a,
db_name.table4b table4b,
db_name.table4c table4c,
db_name.table4d table4d,
db_name.table4e table4e,
db_name.table4f table4f,
db_name.table3a table3a,
db_name.table3b table3b,
db_name.table3c table3c,
db_name.table3d table3d,
db_name.table2a table2a,
db_name.table2b table2b,
db_name.table1 table1
```

**FROM**

```
SELECT...
```

---

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

- Extract Transform Load (ETL) process
- Virtual views

**Choice 2: Which data format for the new representation**

- New relational schema
- JSON (or XML) documents
- Resource Description Framework (RDF)
Generating a new representation of the data

1. Extract Transform Load (ETL) / Materialization
   E.g., relational data warehouse, triplestore (RDF)

   ![Diagram](image1)

   User → Inform. need → Mediator 1 (High-level query) → Data Warehouse / Tripstore → ETLMat. → disparate sources

2. Virtual views
   E.g., virtual databases (Teiid, Apache Drill, Exareme), OBDA with Optique

   ![Diagram](image2)

   User → Inform. need → Mediator 1 (High-level query) → Mediator 2 (Native query (SQL)) → disparate sources

Optique solution: Ontology-Based Data Access (OBDA)

Optique architecture

Choice 1: Generating a new representation of the data
   - Virtual views

Choice 2: Which data format for the virtual view
   - Resource Description Framework (RDF)
**OBDA Ingredients**

**Outline**

1. Exposing data as RDF
2. Data Access in Statoil: limitations and solutions
3. OBDA Ingredients
   - Overview
   - Ontology
   - Mappings
   - Query rewriting
   - Bootstrapping
   - Visual Query Formulation
   - Optique

**INF3580/4580 :: Spring 2017**

**Lecture 14 :: 15th May**

**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:**
- User does not know where and how the data is stored.
- User 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**

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.

Outline

1. Exposing data as RDF
2. Data Access in Statoil: limitations and solutions
3. OBDA Ingredients
   - Overview
   - Ontology
   - Mappings
   - Query rewriting
   - Bootstrapping
   - Visual Query Formulation
   - Optique

Capturing UML class diagrams/ER schemas in OWL 2 QL

```
Professor ⊑ AcadStaff
AssocProf ⊑ Professor
FullProf ⊑ Professor
AssocProf ⊑ ¬FullProf
AcadStaff ⊑ ssn
FullProf ⊑ ssn
AcadStaff ⊑ Integer

SupBy

Professor

AssocProf FullProf

‘respFor’ ⊑ ‘worksFor’

Professor

‘respFor’ ‘worksFor

Project

AcadStaff ‘worksFor

FullProf

AcadStaff ‘worksFor

Project

FullProf

AssocProf FullProf
```
Outline

1. Exposing data as RDF
2. Data Access in Statoil: limitations and solutions
3. OBDA Ingredients
   - Overview
   - Ontology
   - Mappings
   - Query rewriting
   - Bootstrapping
   - Visual Query Formulation
   - Optique

OBDA Mappings

Global-As-View (GAV) mapping assertion $\phi \rightsquigarrow \psi$
- $\phi$: FO query (over DB predicates)
- $\psi$: atom (over an RDF predicate)
- Open-World Assumption (by default)

Class instance (:Student)

Source

\[
\begin{align*}
q(s) & \leftarrow \text{uniI-student}(s, f, l) \\
\text{SELECT} & \quad \text{\_id} \\
\text{FROM} & \quad \text{uniI}\_\text{student}
\end{align*}
\]

Target

\[
\begin{align*}
\text{Student(URI}_{1}(s)) & \\
\text{ex:uniI/student/}\{\_id\} & \text{ a :Student} .
\end{align*}
\]

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 example

### Triples map to populate Student class

```xml
<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" ]
    ].
</TriplesMap1>
```

---

### Table STUDENT:

<table>
<thead>
<tr>
<th>s_id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ernesto</td>
</tr>
<tr>
<td>2</td>
<td>Martin</td>
</tr>
<tr>
<td>3</td>
<td>Leif</td>
</tr>
</tbody>
</table>

### 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

1. Exposing data as RDF
2. Data Access in Statoil: limitations and solutions
3. OBDA Ingredients
   - Overview
   - Ontology
   - Mappings
   - Query rewriting
   - Bootstrapping
   - Visual Query Formulation
   - Optique
Query answering by rewriting (example)

Database:
- `wlb_dev(name, ...)`
- `wlb_exp(name, purpose, ...)`

Ontology:
- `Wellbore`
- `DevWellbore`
- `ExpWellbore`
- `Wildcat`

Mappings:
- `DevWellbore(name) ↦ \text{SELECT name FROM } wlb_{\text{dev}}`
- `ExpWellbore(name) ↦ \text{SELECT name FROM } wlb_{\text{exp}}`
- `Wildcat(name) ↦ \text{SELECT name FROM } wlb_{\text{exp}} \text{ WHERE purpose = 'WILDCAT'}`

Query: List all wellbores.
\[
q : \text{Wellbore}(x) \\
q : \text{Wellbore}(x) \cup \text{DevWellbore}(x) \cup \text{ExpWellbore}(x) \cup \text{Wildcat}(x)
\]

\[
q_{\text{SQL}} : \text{SELECT name FROM } wlb_{\text{dev}} \cup \text{SELECT name FROM } wlb_{\text{exp}} \cup \text{SELECT name FROM } wlb_{\text{exp}} \text{ WHERE purpose = 'WILDCAT'}
\]

Test case:

```sql
SELECT name FROM wlb_dev UNION SELECT name FROM wlb_exp UNION SELECT name FROM wlb_exp WHERE purpose = 'WILDCAT'
```
Query answering by rewriting (example)

**Database:**
- `wlb_dev(name, ...)`
- `wlb_exp(name, purpose, ...)`

**Ontology:**
- Wellbore
- Field
- DevWellbore
- ExpWellbore
- Wildcat

**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.

\[ q : \text{Wellbore}(x) \]
\[ q_o : \text{Wellbore}(x) \cup \text{DevWellbore}(x) \cup \text{ExpWellbore}(x) \cup \text{Wildcat}(x) \]
\[ q_{SQL} : \text{SELECT name FROM wlb_dev} \]
\[ \quad \cup \text{SELECT name FROM wlb_exp} \]
\[ \quad \cup \text{SELECT name FROM wlb_exp WHERE purpose = 'WILDCAT'} \]

**Query answering by rewriting (tool support)**

- **Ontop:** state-of-the-art OBDA system. [http://ontop.inf.unibz.it/](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**
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 ontology to the database, or
  - The given ontology will be aligned with the bootstrapped 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: 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 materializing the data or using the ontology for virtual access to data
    - Solution: `rdfs:Literal`

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)

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

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
    - Domain: `Mapping`
  - e.g. `Exploration_Wellbore SubclassOf: Wellbore`
Tool support and Lessons learnt

- **BootOX**: [http://www.cs.ox.ac.uk/isg/tools/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**

1. Exposing data as RDF
2. Data Access in Statoil: limitations and solutions
3. OBDA Ingredients
   - Overview
   - Ontology
   - Mappings
   - Query rewriting
   - Bootstrapping
   - Visual Query Formulation
   - Optique

**Visual query formulation (OptiqueVQS)**
Outline

1. Exposing data as RDF
2. Data Access in Statoil: limitations and solutions
3. OBDA Ingredients
   - Overview
   - Ontology
   - Mappings
   - Query rewriting
   - Bootstrapping
   - Visual Query Formulation
   - Optique

Nele

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
- ont-top 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