Ontology-Based Data Access
and OWL 2 QL

Roman Kontchakov

Department of Computer Science and Information Systems,
Birkbeck, University of London

http://www.dcs.bbk.ac.uk/~roman
Data access in industry

(from Norwegian Petroleum Directorate’s FactPages)

show me the wellbores completed before 2008 where Statoil as a drilling operator sampled less than 10 meters of cores
show me the wellbores completed before 2008 where Statoil as a drilling operator sampled less than 10 meters of cores

5 days later:

```sql
SELECT DISTINCT cores.wlbName, cores.lenghtM, wellbore.wlbDrillingOperator, wellbore.wlbCompletionYear
FROM
  ( (SELECT wlbName, wlbNpdidWellbore, (wlbTotalCoreLength * 0.3048) AS lenghtM
     FROM wellbore_core
     WHERE wlbCoreIntervalUom = 'ft')
   UNION
   (SELECT wlbName, wlbNpdidWellbore, wlbTotalCoreLength AS lenghtM
     FROM wellbore_core
     WHERE wlbCoreIntervalUom = 'm')
  ) as cores,
  ( (SELECT wlbNpdidWellbore, wlbDrillingOperator, wlbCompletionYear
     FROM wellbore_development_all
   UNION
   (SELECT wlbNpdidWellbore, wlbDrillingOperator, wlbCompletionYear
     FROM wellbore_exploration_all )
   UNION
   (SELECT wlbNpdidWellbore, wlbDrillingOperator, wlbCompletionYear
     FROM wellbore_shallow_all )
  ) as wellbore
WHERE wellbore.wlbNpdidWellbore = cores.wlbNpdidWellbore
...
show me the wellbores completed before 2008 where Statoil as a drilling operator sampled less than 10 meters of cores

5 days later:

```
SELECT DISTINCT cores.wlbName, cores.lenghtM, wellbore.wlbDrillingOperator, wellbore.wlbCompletionYear
FROM
  ( (SELECT wlbName, wlbNpdidWellbore, (wlbTotalCoreLength * 0.3048) AS lenghtM
     FROM wellbore_core
     WHERE wlbCoreIntervalUom = '[ft]' )
  UNION
  (SELECT wlbName, wlbNpdidWellbore, wlbTotalCoreLength AS lenghtM
     FROM wellbore_core
     WHERE wlbCoreIntervalUom = '[m]' ) )
) as cores,
  ( (SELECT wlbNpdidWellbore, wlbDrillingOperator, wlbCompletionYear
     FROM wellbore_development_all
  UNION
  (SELECT wlbNpdidWellbore, wlbDrillingOperator, wlbCompletionYear
     FROM wellbore_exploration_all )
  UNION
  (SELECT wlbNpdidWellbore, wlbDrillingOperator, wlbCompletionYear
     FROM wellbore_shallow_all )
 ) as wellbore
WHERE wellbore.wlbNpdidWellbore = cores.wlbNpdidWellbore
...```

In STATOIL:

- 1,000 TB of relational data
- 1,500 tables
- 30–70% of time on data gathering
 Ontology-Based Data Access (OBDA) Poggi et al. (JDS 2008)

```sql
SELECT DISTINCT ?unit ?well
WHERE {
  [] npdv:stratumForWellbore ?wellboreURI ;
  npdv:inLithostratigraphicUnit [ npdv:name ?unit ].
  ?wellboreURI npdv:name ?well ;
  ?core a npdv:WellboreCore ;
  npdv:coreForWellbore ?wellboreURI .
}
```

- gives a high-level conceptual view of the data
- provides a convenient & natural vocabulary for user queries
- enriches incomplete data with background knowledge
Ontology-Based Data Access (OBDA) Poggi et al. (JDS 2008)

SPARQL 1.1 (W3C 2008–13)

```
SELECT DISTINCT ?unit ?well
WHERE {
  [] npdv:stratumForWellbore ?wellboreURI ;
  ?wellboreURI npdv:name ?well ;
  ?core a npdv:WellboreCore ;
    npdv:coreForWellbore ?wellboreURI .
}
```

OWL 2 (W3C 2004–12)

```
CREATE TABLE wellbore_core (
  wlbName varchar(60) NOT NULL,
  wlbCoreNumber int(11) NOT NULL,
  wlbCoreIntervalTop decimal(13,6),
  ...
)
```

R2RML (W3C 2012)

```
[] rdf:type rr:TriplesMap;
  rr:logicalTable "select * from wellbore_core";
  rr:subjectMap [ a rr:TermMap;
    rr:template "&npd-v2;wellbore/{wlbNpdidWellbore}";];
  rr:propertyObjectMap [ rr:property npdv:coreIntervalBottom;
    rr:column "wlbCoreIntervalBottom" ];
  ...
```

RDF 1.1 (W3C 2004–14)

```
data sources
ontology
mappings
```

- gives a high-level conceptual view of the data
- provides a convenient & natural vocabulary for user queries
- enriches incomplete data with background knowledge
Materialisation or ETL (Extract, Transform and Load)

translate mappings into rules:

\[
\text{wellbore\_core}(t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}) \rightarrow \\
\text{npdv:coreIntervalBottom}(\text{URI}("&npdv;wellbore/\{\}/core/\{\}"), t_9, t_2), t_4)
\]
Materialisation or ETL (Extract, Transform and Load)

translate mappings into rules:

\[
\text{wellbore\_core}(t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}) \rightarrow \\
\text{npdv:coreIntervalBottom}(\text{URI("&npdv;wellbore/\{\}/core/\{\}"}, t_9, t_2), t_4)
\]

translate the ontology onto rules:

\[
\text{npdv:production\_wellbore}(x) \rightarrow \text{npdv:wellbore}(x) \quad \text{rdfs:subClassOf}
\]
\[
\text{npdv:coreForWellbore}(x, y) \rightarrow \text{npdv:WellboreCore}(y) \quad \text{rdfs:range}
\]
\[
\text{owl:someValuesFrom} \quad \text{(on the left-hand side of)} \rightarrow
\]

RuleML General Telecon 30.06.17 3
Materialisation or ETL (Extract, Transform and Load)

translate mappings into rules:

\[ \text{wellbore\_core}(t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}) \rightarrow \text{npdv:coreIntervalBottom}(\text{URI}(&\text{npdv:wellbore/}\{\}/\text{core/}\{\}, t_9, t_2), t_4) \]

translate the ontology onto rules:

\[ \text{npdv:production\_wellbore}(x) \rightarrow \text{npdv:wellbore}(x) \quad \text{rdfs:subClassOf} \]
\[ \text{npdv:coreForWellbore}(x, y) \rightarrow \text{npdv:WellboreCore}(y) \quad \text{rdfs:range} \]
\[ \text{owl:someValuesFrom} \quad (\text{on the left-hand side of } \rightarrow) \]

\textbf{not every} OWL 2 axiom can be translated into rules

\[ \text{npdv:WellboreCore}(y) \rightarrow \exists x \text{ npdv:coreForWellbore}(x, y) \quad \text{owl:someValuesFrom} \]
\[ (\text{on the right-hand side of } \rightarrow) \]
Materialisation or ETL (Extract, Transform and Load)

translate mappings into rules:

\[
\text{wellbore\_core}(t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}) \rightarrow \\
\text{npdv:coreIntervalBottom}(\text{URI}("\&npdv;wellbore/\\{\}/core/\\{\}"), t_9, t_2, t_4)
\]

translate the ontology onto rules:

\[
\text{npdv:production\_wellbore}(x) \rightarrow \text{npdv:wellbore}(x) \quad \text{rdfs:subClassOf}
\]

\[
\text{npdv:coreForWellbore}(x, y) \rightarrow \text{npdv:WellboreCore}(y) \quad \text{rdfs:range}
\]

\[
\text{npdv:StratigraphicUnit}(x) \rightarrow \text{npdv:LithostratigraphicUnit}(x) \lor \text{npdv:ChronostratigraphicUnit}(x) \quad \text{owl:unionOf}
\]

not every OWL 2 axiom can be translated into rules

\[
\text{npdv:WellboreCore}(y) \rightarrow \exists x \ \text{npdv:coreForWellbore}(x, y) \quad \text{owl:someValuesFrom}
\]

(on the right-hand side of \(\rightarrow\))

RuleML General Telecon 30.06.17
Forward Chaining and OWL 2 RL

SPARQL query

ontology
npdv:MoveableFacility ⊑ npdv:Facility
...

mapping
npdv:MoveableFacility
(URI("&npdv;facility/{t7}",t7))
:- facility_moveable(t1,...,t6,t7,t8,...,t10)
...

close
derived triples
forward chaining

RDF graph
triples

R2RML processor

database
n-ary relations
chase is defined only for Horn logics (no disjunction)
chase is defined only for Horn logics (no disjunction)

in general, even for Horn ontologies in OWL 2, the chase does not terminate

value invention as in npdv:WellboreCore(y) → ∃x npdv:coreForWellbore(x, y)
Forward Chaining and OWL 2 RL

chase is defined only for Horn logics (no disjunction)

in general, even for Horn ontologies in OWL 2, the chase does not terminate
value invention as in npdv:WellboreCore(y) → ∃x npdv:coreForWellbore(x, y)

**OWL 2 RL** is the largest Horn fragment of OWL 2 without value invention

Grosof et al. (WWW 2003), ter Horst (Web Semantics, 2005)
Backward Chaining and OWL 2 QL

SPARQL query + rewriting + unfolding

ontology
npdv:MoveableFacility ⊑ npdv:Facility
...

mapping
npdv:MoveableFacility (URI("&npdv;facility/{}",t7))
:- facility_moveable(t1,…,t6,t7,t8,…,t10)
...

case
derived triples

virtual RDF graph
triples

database
n-ary relations
Backward Chaining and OWL 2 QL

OWL 2 QL is almost the largest fragment of OWL 2 that supports backward chaining.
**SPARQL query**

**rewriting**

**ontologyn**

npdv:MoveableFacility ⊑ npdv:Facility

... 

**mapping**

npdv:MoveableFacility

(URL("&npdv:facility/{}",t7))

:- facility_moveable(t1,...,t6,t7,t8,...,t10)

...

**database**

n-ary relations

**chase**

derived triples

**virtual RDF graph**

triples

**owl 2 ql** is almost the largest fragment of owl 2 that supports backward chaining

owl 2 ql can encode uml class / er diagrams

artale et el. (er 2007)
OWL 2 QL is almost the largest fragment of OWL 2 that supports backward chaining.

OWL 2 QL can encode UML class / ER diagrams.

Artale et al. (ER 2007)

Data complexity of query answering in OWL 2 QL =

the data complexity of database query evaluation (AC⁰)

+ value invention

− no disjunction, no owl:someValuesFrom on the LHS except for rdfs:domain/range
Forward v Backward Chaining

ontology: production_wellbore(\(x\)) \rightarrow wellbore(\(x\))
data: production_wellbore(\(a_{42}\)), wellbore(\(a_{92}\))
query: \(q(x) \leftarrow wellbore(x)\)
Forward v Backward Chaining

ontology: \( \text{production\_wellbore}(x) \rightarrow \text{wellbore}(x) \)
data: \( \text{production\_wellbore}(a42), \text{wellbore}(a92) \)
query: \( q(x) \leftarrow \text{wellbore}(x) \)

**forward chaining**

1. ‘apply’ ontology to the data to obtain the chase:
   \( \text{production\_wellbore}(a42), \text{wellbore}(a42), \text{wellbore}(a92) \)
2. query the chase
Forward v Backward Chaining

ontology: \( \text{production\_wellbore}(x) \rightarrow \text{wellbore}(x) \)
data: \( \text{production\_wellbore}(a42), \text{wellbore}(a92) \)
query: \( q(x) \leftarrow \text{wellbore}(x) \)

**forward chaining**

1. ‘apply’ ontology to the data to obtain the chase:
   
   \( \text{production\_wellbore}(a42), \text{wellbore}(a42), \text{wellbore}(a92) \)

2. query the chase

**backward chaining**

1. ‘apply’ ontology to the query to obtain its rewriting (a union of CQs, a UCQ):
   
   \( q(x) \leftarrow \text{wellbore}(x) \)
Forward v Backward Chaining

ontology: \( \text{production\_wellbore}(x) \to \text{wellbore}(x) \)
data: \( \text{production\_wellbore}(a42), \text{wellbore}(a92) \)
query: \( q(x) \leftarrow \text{wellbore}(x) \)

forward chaining

1. ‘apply’ ontology to the data to obtain the chase:
   \( \text{production\_wellbore}(a42), \text{wellbore}(a42), \text{wellbore}(a92) \)
2. query the chase

backward chaining

1. ‘apply’ ontology to the query to obtain its rewriting (a union of CQs, a UCQ):
   \( q(x) \leftarrow \text{wellbore}(x) \)
   \( q(x) \leftarrow \text{production\_wellbore}(x) \)
   the head of the rule unifies with a query atom
   \( \implies \) create a copy of the CQ with the atom replaced by the rule body
Forward v Backward Chaining

ontology:  production_wellbore($x$) → wellbore($x$)
data:  production_wellbore($a42$), wellbore($a92$)
query:  $q(x) \leftarrow \text{wellbore}(x)$

**forward chaining**
1. ‘apply’ ontology to the data to obtain the chase:
   production_wellbore($a42$), wellbore($a42$), wellbore($a92$)
2. query the chase

**backward chaining**
1. ‘apply’ ontology to the query to obtain its rewriting (a union of CQs, a UCQ):
   $q(x) \leftarrow \text{wellbore}(x)$
   $q(x) \leftarrow \text{production_wellbore}(x)$ the head of the rule unifies with a query atom
   $\implies$ create a copy of the CQ with the atom replaced by the rule body
2. use the obtained UCQ to query the original data
Query Rewriting: Theory and Practice

UCQ rewritings are exponential $\Rightarrow$ very bad in practice
Query Rewriting: Theory and Practice

UCQ rewritings are exponential $\implies$ very bad in practice
in general, even PE- and NDL-rewritings are exponential
and FO-rewritings are superpolynomial unless $\text{NP/poly} \subseteq \text{NC}^1$
for more details, see Bienvenu et al. (2016)  https://arxiv.org/abs/1605.01207
Query Rewriting: Theory and Practice

UCQ rewritings are exponential \( \Rightarrow \) very bad in practice

in general, even PE- and NDL-rewritings are exponential

and FO-rewritings are superpolynomial unless \( \text{NP/poly} \subseteq \text{NC}^1 \)

for more details, see Bienvenu et al. (2016) https://arxiv.org/abs/1605.01207

using \( \lor \) in UCQs (unions of semiconjunctive queries) helps

to deal with class/property hierarchies in practice
UCQ rewritings are exponential $\Rightarrow$ very bad in practice
in general, even PE- and NDL-rewritings are exponential
and FO-rewritings are superpolynomial unless NP/poly $\subseteq$ NC$^1$
for more details, see Bienvenu et al. (2016)  https://arxiv.org/abs/1605.01207

using $\lor$ in UCQs (unions of semiconjunctive queries) helps
to deal with class/property hierarchies in practice
moreover, hierarchies can be compiled into mappings (T-mappings)
and optimised using database integrity constraints
Query Rewriting: Theory and Practice

UCQ rewritings are exponential $\implies$ very bad in practice
in general, even PE- and NDL-rewritings are exponential
and FO-rewritings are superpolynomial unless $\text{NP/poly} \subseteq \text{NC}^1$
for more details, see Bienvenu et al. (2016) https://arxiv.org/abs/1605.01207

using $\lor$ in UCQs (unions of semiconjunctive queries) helps
to deal with class/property hierarchies in practice
moreover, hierarchies can be compiled into mappings (T-mappings)
and optimised using database integrity constraints

implemented in ontop framework Free University of Bozen-Bolzano
with some help from Birkbeck

Calvanese et al. (Semantic Web, 2017), Rodriguez-Muro et al. (ISWC 2013)