Ontology-based data access

Gilles Falquet

Semantic Web Technologies

Principle and hypotheses

- View datasets through an ontology, not through a database schema

- Hypotheses/observations
  - the main bottleneck in data access is the complexity of the database schemas
  - the vocabulary of an ontology is more understandable than the table and column names

Reference

- Kharlamov et al. (2016) Ontology Based Data Access in Statoil. Web Semantics: Science, Services and Agents on the World Wide Web 44 (2017) 3–36

Geological data at statoil

Exploration and Production Data Store (EPDS)

- 3000 tables and views

- about 37,000 columns

- naming conventions for schema elements, constraints, and the structure of EPDS's schema are complex and considerable parts of it have limited or no documentation.

- the major challenge with accessing EPDS is the schema complexity

Finding the right data is hard !

Table 1
Database metrics: Showing number of schema constructs for each database schema. Zero-values are removed from the table for readability.

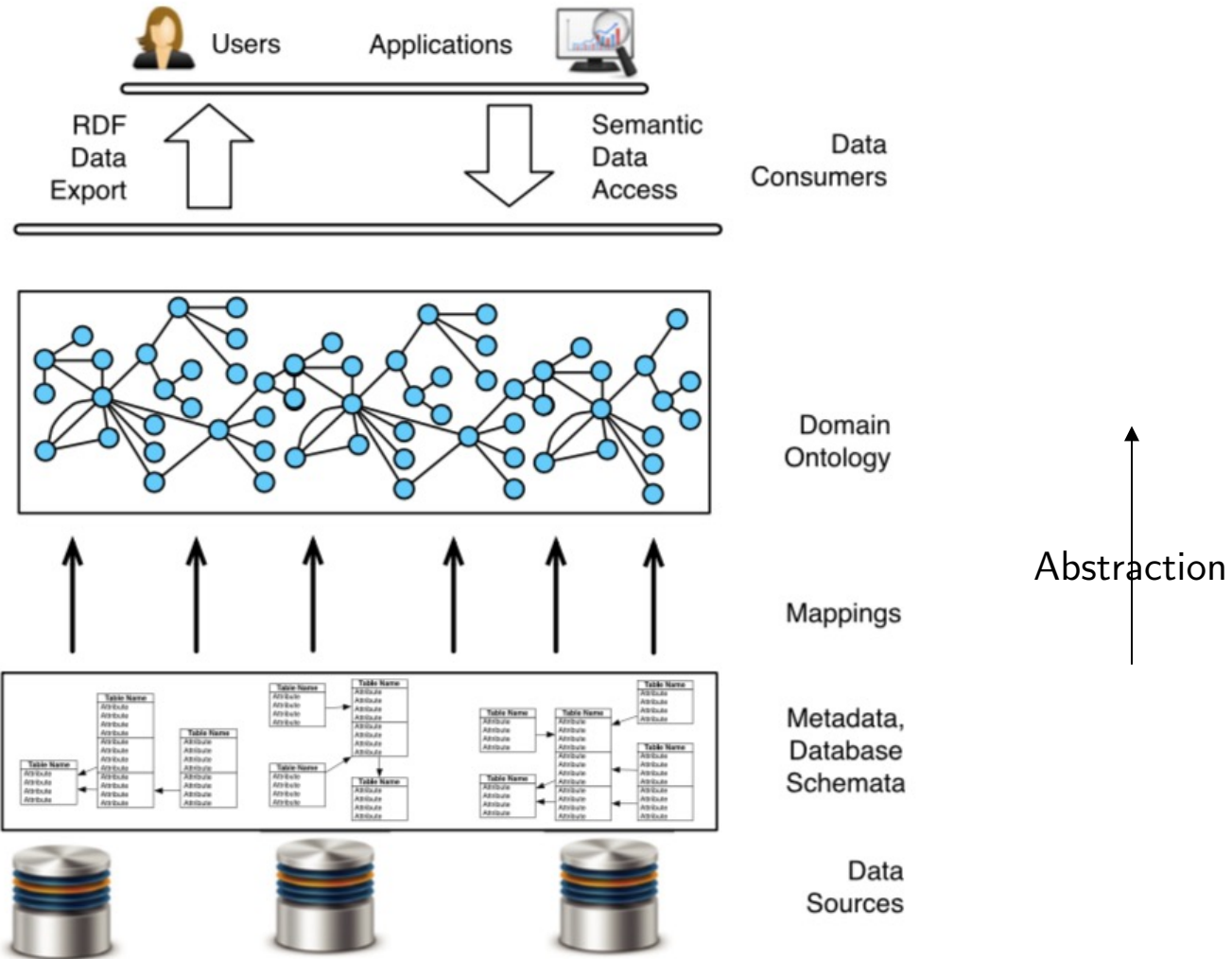|					EPDS	GeoChemDB	Recall	CoreDB	OW	Compass
Overview										
Tables					1595	90	22	15	78	895
Mat. views					27	4				
Views					1703	41	12		1026	1004
Columns					8378	3396	430	63	16668	30638
Tables by no. rows										
	0	rows			1130	3	2		15	512
	1	row			1152	9	2		4	34
1	<	rows	≤	10	135	9	1	4	15	117
10	<	rows	≤	100	83	20	3	2	17	80
100	<	rows	≤	1 000	58	30	3	4	12	87
1 000	<	rows	≤	10 000	63	10	5	1	11	42
10 000	<	rows	≤	100 000	57	4	2	1	2	19
100 000	<	rows	≤	1000 000	35	3	4	2		4
1 000 000	<	rows			12	3	2	1		

Access Points

- Access points are typically based on **materialized** special purpose database **views**.

- The process of making such view consists of the three **ETL** steps:
  - Extracting, Transforming, Loading data.
  - An SQL query for data extraction generated by the extraction tools may contain thousands of words and have 50–200 joins.

- Building an ETL process consists of a myriad of data access and processing steps, many of which require deep knowledge of the data that is being processed and how it is represented.

- IT staff become the de facto mediators between geologists and databases

ETL process in FME
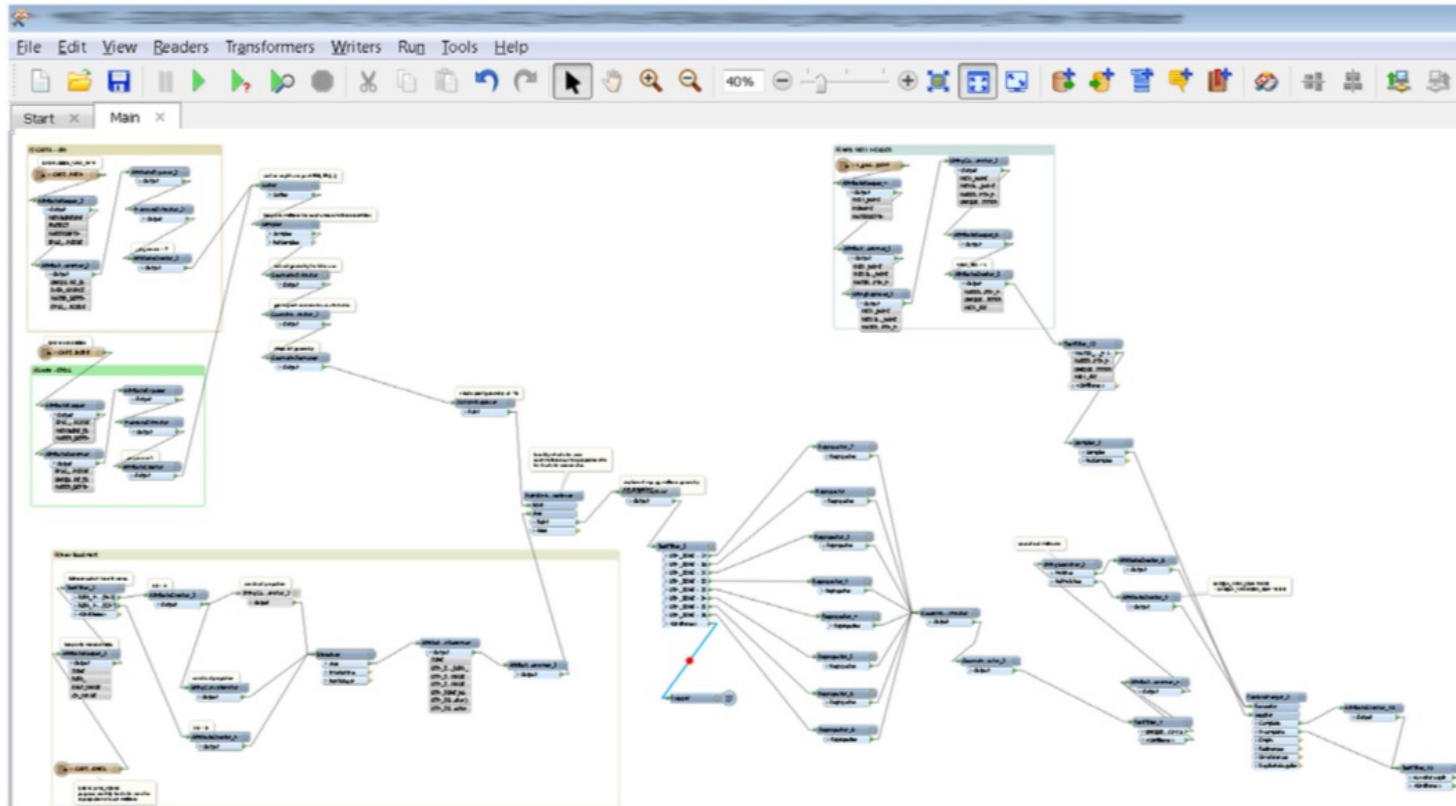
Ontology Based Data Access

- Use a familiar vocabulary

    - Provide the user with access to the data store via the use of a domain specific vocabulary of classes and properties that the user is familiar with.

- This vocabulary is related to the database schema via mappings

    - Technical details of the database schema are hidden from end-users.

Users   Applications

RDF Data Export   Semantic Data Access   Data Consumers

Domain Ontology

Abstraction

Mappings

Metadata, Database Schemata

Data Sources

Principles

- query over the domain vocabulary $\rightarrow$
    - queries over the database schemas
    - executed over the data by DBMS.
- the domain vocabulary enhanced with formal axioms
  $\rightarrow$ an ontology
- exploit ontological axioms to enrich query answers with implicit information
  $\Rightarrow$ ontological reasoning
- data are treated under the Open World Assumption
  $\Rightarrow$ can be incomplete w.r.t. to the ontology axioms.

Query Answer Enrichment

Enrichment of answers is done via logical reasoning:

User query Q over the domain vocabulary

$\rightarrow$ rewriting $\rightarrow$

New query over this vocabulary
- logically equivalent to Q w.r.t. the ontology
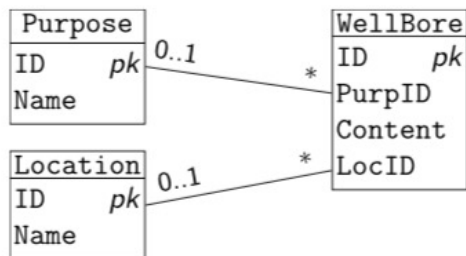- "absorbs" a fragment of the ontology relevant for answering Q

Example

The data may state that the rock layers in the area surrounding the Ekofisk field are chalk, but not their age

The user asks for *wellbores that penetrate rock from the Cretaceous era*

- Direct query: will not return the wellbores of Ekofisk
- Rewritten query: will also ask for wellbores that penetrate chalk, and thus return the Ekofisk's wellbores.
  - Using the knowledge Chalk has been formed in the Cretaaceous era

　　　　Ontology-Based Data Access

DB schema

Purpose	
ID	pk
Name	

Location	
ID	pk
Name	

0..1

0..1

*

*

WellBore	
ID	pk
PurpID	
Content	
LocID	

ExpWBore	
ID	pk
Type	

(a) Database schema.

Location:

ID	Name
L1	Norway
L2	UK

Purpose:

ID	Name
P1	Shallow
P2	Injection

Wellbore:

ID	PurpID	Content	LocID
W1	P1	Dry	L1
W2	P2	Oil	L2

ExpWBore:

ID	Type
E1	Active
E2	Discovery

(b) Database instance.

Ontology

Classes

- Location
- Purpose
- WellBore
- ExplorationWellBore
- ShallowWelleBore
- Properties

Properties

- hasLocation
- hasPurpose
- hasName

Axioms

- ExploratioWellBore ⊑ WellBore
- ShallowWelleBore ⊑ WellBore
- WellBore ⊑ ∃ hasContent . T

Mappings

- "cast" functions
  - fo: Database entity identifier $\rightarrow$ URI   e.g. 'France' $\rightarrow$ http://example.com/country/France
  - fv: Database value $\rightarrow$ Literal

- translation functions
  - Class(fo(x)) $\mapsto$ SQL(x),   // fo(x) rdf:type Class
  - Property(fo(x),fo(y)) $\mapsto$ SQL(x,y),   // fo(x) Property fo(y)
  - Property(fo(x),fv(y)) $\mapsto$ SQL(x,y),   // fo(x) Property fv(y)

  - SQL(x) and SQL(x, y) are SQL queries with respectively one and two output variables

Example

ExplorationWellBore(f(ID))
$\mapsto$ (4)

```
SELECT ID
FROM ExpWBore
```

ShallowWellBore(f(W.ID))
$\mapsto$ (5)

```
SELECT W.ID
FROM WellBore W, Purpose P
WHERE W.PurpID = P.ID
  AND P.Name = "Shallow"
```

hasLocation(f(ID), f(LocID))
$\mapsto$

```
SELECT ID, LocID
FROM WellBore
```

(e) Mappings.

- small SQL query compared to SQL code of ETL processes behind access points.

- SQL code of individual mappings more readable and more maintainable

- can be reused by many data access tasks: they 'connect' ontologies to DBs and then can be used for any query over the ontology,

Query answering over ontologies and in OBDA

```
SELECT ?x WHERE { ?x :hasContent ?y. }
```

- rewritten into a union query s.t. each query of Q' is subsumed by Q .
    - compilation of relevant ontological information, similar to the resolution procedure in Prolog. Equivalent to DL-Lite reasoning (*)

```
SELECT ?x WHERE
{ ?x hasContent ?y. }
UNION { ?x a WellBore. }
UNION { ?x a ExplorationWellBore. }
UNION { ?x a ShallowWellBore. }
```

* D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable reasoning and efficient query answering in description logics: The DL-Lite family, JAR 39 (3) (2007) 385–429.

unfolding

- based on the mappings

```
SELECT f(ID) AS x FROM ExpWBore
UNION
SELECT f(W.ID) AS x FROM WellBore W, Purpose P
WHERE W.PurpID = P.ID AND P.Name = "Shallow".
```

Bootstraping and Importing

- Given a relational database D, generate an instance (D, V, O, M).

  - Vocabulary and Ontology generation: Given D,
    - create a vocabulary V (e.g. using table and column names)
    - create an ontology O over V.

  - Mapping generation: Given D, V, and O create a set of mappings M relating D with V.

- Importing: Given an instance $(D, V, O1, M)$ and an ontology $O2$, return an instance (D, V, O, M), where O is the alignment of $O1$ and $O2$.

Bootstrap ontology

Extracted from the relational schemas

- mapping rules

  - mirror RDB schemata by essentially translating
    - each (non-binary) table into an OWL class;
    - each attribute not involved in a foreign key into an OWL datatype property;
    - each foreign key into an OWL object property.

Alignment

- Extended version of the ontology alignment system LogMap

  - derives OWL 2 equivalence and subclass(subproperty) axioms between the terms from O1's and O2's vocabularies.

- scalable solution that, in the resulting ontology O after O1 and O2 are aligned, minimises the violations of the conservativity principle

  - O does not entail (many) axioms over O1 's and O2 's vocabulary which are not already entailed by O1 or O2.

Wall

equivalent

Wall

isIn

equivalent

belongsTo

GlassWall

equivalent

Window

Window ⊑ ∃ belongsTo . Wall

O1

O2

entails

O1: GlassWall ⊑ ∃ isIn . Wall