Ontology-based Data Access and Integration: Relational Data and Beyond

Diego Calvanese

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

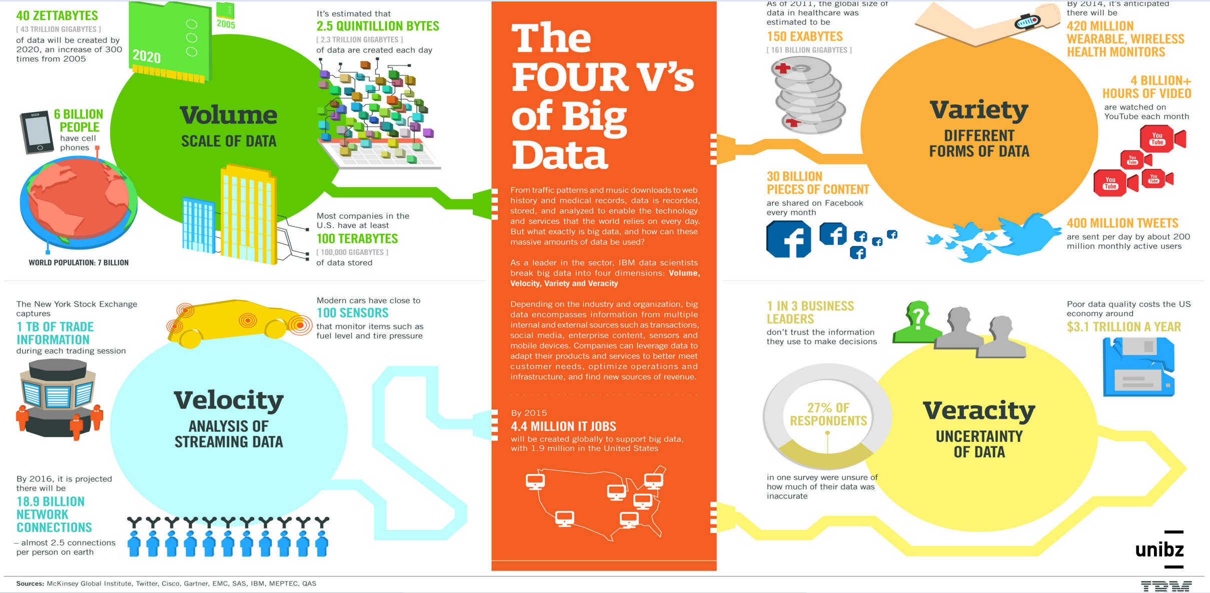
44th Latin American Computing Conference (CLEI)
13th Latin American Conference on Learning Technologies (LACLO)
Sãu Paulo, Brazil, 1–5 October 2018

Typical view of Big Data

But data has structure

In fact, data has a lot of structure

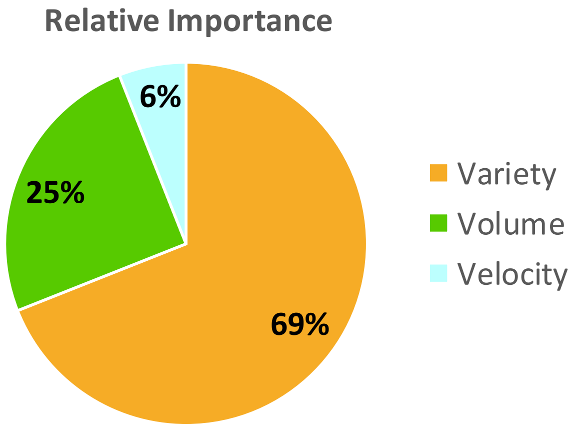
Challenges in the Big Data era



unibz

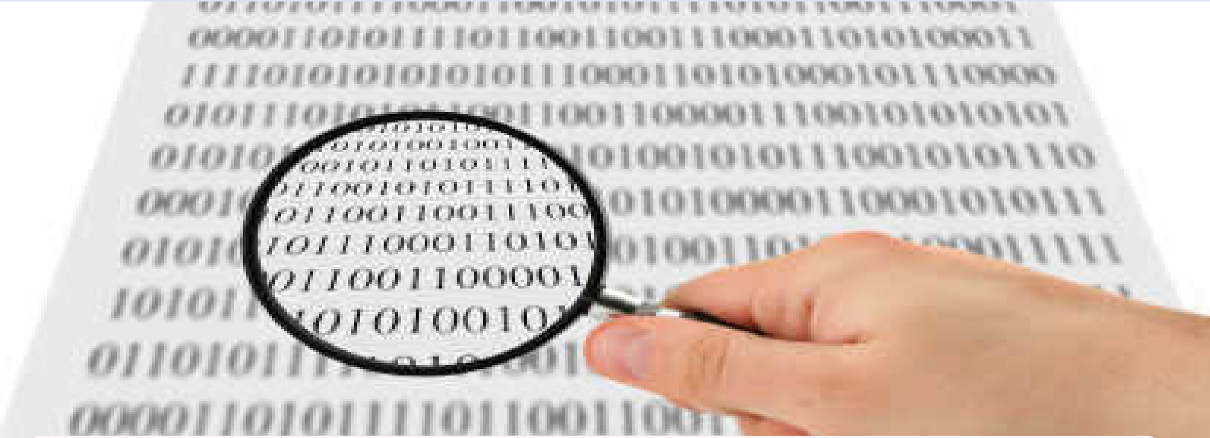
Variety, not volume, is driving Big Data initiatives

MIT Sloan Management Review (28 March 2016)



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

How much time is spent searching for the right data?



Important problem: searching for data and establishing its quality

Example: in oil&gas, engineers spend 30–70% of their time on this (Crompton, 2008)

Challenge: Accessing heterogeneous data

Statoil (now Equinor) Exploration

Geologists at Statoil, prior to making decisions on drilling new wellbores, need to gather relevant information about previous drillings.

Slegge relational database:

- 1,000 TB of relational data
- 1,545 tables and 1727 views
- each with dozens of attributes
- consulted by 900 geologists

Problem: Translating information needs

Information need expressed by geologists

In my geographical area of interest, return all pressure data tagged with key stratigraphy information with understandable quality control attributes, and suitable for further filtering.

To obtain the answer, this needs to be translated into SQL[1]:

- main table for wellbores has 38 columns (with cryptic names)
- to obtain pressure data requires a 4-table join with two additional filters
- to obtain stratigraphic information requires a join with 5 more tables

[1] BTW, SQL is the standard DB query language.

Problem: Translating information needs

We would obtain the following SQL query:

```
SELECT WELLBORE.IDENTIFIER, PTY_PRESSURE.PTY_PRESSURE_S,
       STRATIGRAPHIC_ZONE.STRAT_COLUMN_IDENTIFIER, STRATIGRAPHIC_ZONE.STRAT_UNIT_IDENTIFIER
FROM WELLBORE,
     PTY_PRESSURE,
     ACTIVITY FP_DEPTH_DATA
        LEFT JOIN (PTY_LOCATION_1D FP_DEPTH_PT1_LOC
           INNER JOIN PICKED_STRATIGRAPHIC_ZONES ZS
              ON ZS.STRAT_ZONE_ENTRY_MD $<=$ FP_DEPTH_PT1_LOC.DATA_VALUE_1_O AND
                 ZS.STRAT_ZONE_EXIT_MD $>=$ FP_DEPTH_PT1_LOC.DATA_VALUE_1_O AND
                 ZS.STRAT_ZONE_DEPTH_UOM = FP_DEPTH_PT1_LOC.DATA_VALUE_1_OU
           INNER JOIN STRATIGRAPHIC_ZONE
              ON   ZS.WELLBORE = STRATIGRAPHIC_ZONE.WELLBORE AND
                 ZS.STRAT_COLUMN_IDENTIFIER = STRATIGRAPHIC_ZONE.STRAT_COLUMN_IDENTIFIER AND
                 ZS.STRAT_INTERP_VERSION = STRATIGRAPHIC_ZONE.STRAT_INTERP_VERSION   AND
                 ZS.STRAT_ZONE_IDENTIFIER = STRATIGRAPHIC_ZONE.STRAT_ZONE_IDENTIFIER)
           ON FP_DEPTH_DATA.FACILITY_S = ZS.WELLBORE AND
              FP_DEPTH_DATA.ACTIVITY_S  = FP_DEPTH_PT1_LOC.ACTIVITY_S,
     ACTIVITY_CLASS FORM_PRESSURE_CLASS
WHERE WELLBORE.WELLBORE_S = FP_DEPTH_DATA.FACILITY_S AND
      FP_DEPTH_DATA.ACTIVITY_S = PTY_PRESSURE.ACTIVITY_S AND
      FP_DEPTH_DATA.KIND_S = FORM_PRESSURE_CLASS.ACTIVITY_CLASS_S AND
      WELLBORE.REF_EXISTENCE_KIND = 'actual' AND
      FORM_PRESSURE_CLASS.NAME = 'formation pressure depth data'
```

Problem: Translating information needs

We would obtain the following SQL query:

```
SELECT WELLBORE.IDENTIFIER, PTY_PRESSURE.PTY_PRESSURE_S
       STRA
FROM WELLBO
     PTY_PR
     ACTIVI
        LEN
              INNER JOIN STRATIGRAPHIC_ZONE
                 ON   ZS.WELLBORE = STRATIGRAPHIC_ZONE.WELLBORE AND
                      ZS.STRAT_COLUMN_IDENTIFIER = STRATIGRAPHIC_ZONE.STRAT_COLUMN_IDENTIFIER AND
     ACTIVI
WHERE WELLB
      FP_DE
      FP_DE
      WELLBORE.REF_EXISTENCE_KIND = 'actual' AND
      FORM_PRESSURE_CLASS.NAME = 'formation pressure depth data'
```

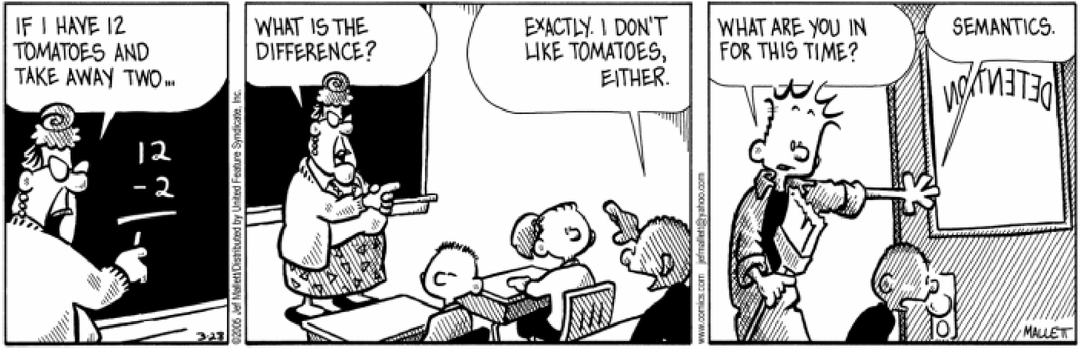
This can be very time consuming, and requires knowledge of the domain of interest, a deep understanding of the database structure, and general IT expertise.

This is also very costly!

Statoil loses **50.000.000€** per year only due to this problem!!

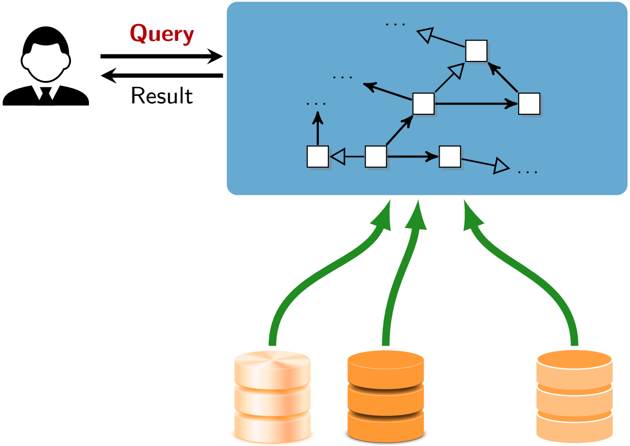
Solution: Exploit semantics of data

Spring 2015 issue of AI Magazine is devoted to Semantics for Big Data.



FRAZZ: © Jeff Mallett/Dist. by United Feature Syndicate, Inc.

Solution: Ontology-based data access (OBDA)



Ontology $\mathcal{O}$
conceptual view of data, convenient vocabulary

Mapping $\mathcal{M}$
how to populate the ontology from the data

Data Sources $\mathcal{S}$
autonomous and heterogeneous

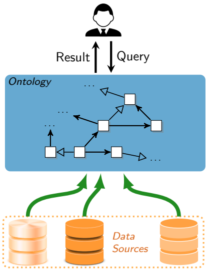
Reduces the time for translating information needs into queries from days to minutes.

unibz

Challenges in OBDA

- How to instantiate the abstract framework?
- How to execute queries over the ontology by accessing data in the sources?
- How to address the expressivity – efficiency tradeoff?
- How to optimize performance with big data and large ontologies?
- How to deal with heterogeneity in the data?
- How to deal with different types of data sources?
- How to provide automated support for key tasks during design and deployment?
- How to assess the quality of the constructed system?

Incomplete information



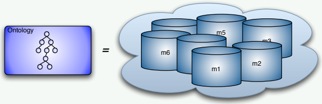
We are in a setting of **incomplete information**!!!

Incompleteness is introduced:

- by data sources, in general assumed to be incomplete;
- by domain constraints encoded in the ontology.

Plus:

Ontologies are logical theories, and hence perfectly suited to deal with incomplete information!



Minus:

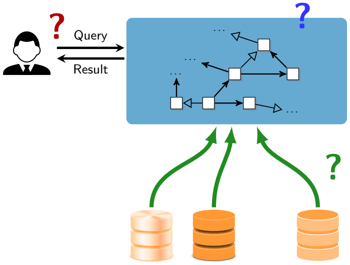
Query answering amounts to **logical inference**, and hence is significantly more challenging.

unibz

OBDA framework – Which languages to use?

The choice of the right languages needs to take into account the tradeoff between expressive power and efficiency of query answering.

Note: We are in a setting where data plays a prominent role, so **efficiency with respect to the data** is the key factor.



The W3C has standardized languages that are suitable for OBDA:

1. Ontology $\mathcal{O}$: expressed in **OWL 2 QL** [W3C Rec. 2012]
2. Query: expressed in **SPARQL** [W3C Rec. 2013] (v1.1)
3. Mapping $\mathcal{M}$: expressed in **R2RML** [W3C Rec. 2012]

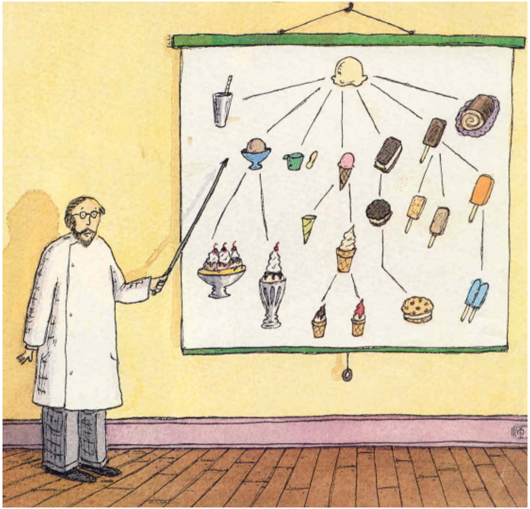
unibz

Outline

Outline

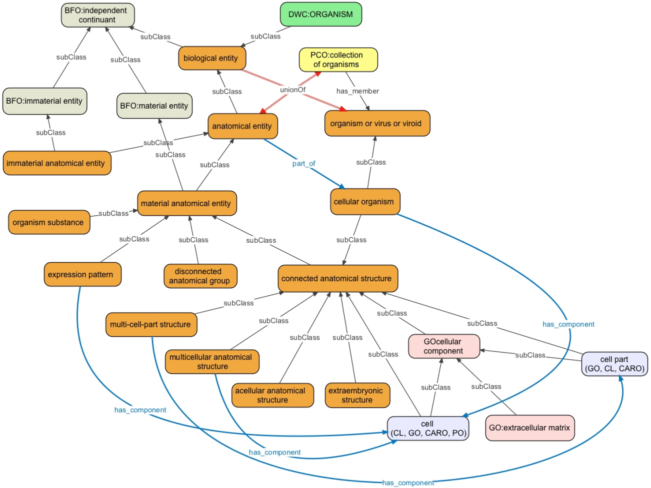
What is an ontology?

- An ontology conceptualizes a domain of interest in terms of **classes**, (binary) **relations**, and their **properties**.
- It typically organizes the classes in a hierarchical structure.
- Ontologies are often represented as graphs.
- However, we consider an ontology as a **logical theory**, expressed in a suitable fragment of first-order logic, or better, in **description logics**.



What is an ontology?

- An ontology conceptualizes a domain of interest in terms of **classes**, (binary) **relations**, and their **properties**.
- It typically organizes the classes in a hierarchical structure.
- Ontologies are often represented as graphs.
- However, we consider an ontology as a **logical theory**, expressed in a suitable fragment of first-order logic, or better, in **description logics**.



What is an ontology?

- An ontology conceptualizes a domain of interest in terms of **classes**, (binary) **relations**, and their **properties**.
- It typically organizes the classes in a hierarchical structure.
- Ontologies are often represented as graphs.
- However, we consider an ontology as a **logical theory**, expressed in a suitable fragment of first-order logic, or better, in **description logics**.

$\forall x.\, \mathsf{Pressure}(x) \rightarrow \mathsf{Measurement}(x)$
$\forall x.\, \mathsf{Porosity}(x) \rightarrow \mathsf{Measurement}(x)$
$\forall x.\, \mathsf{Permeability}(x) \rightarrow \mathsf{Measurement}(x)$
$\forall x.\, \mathsf{Temperature}(x) \rightarrow \mathsf{Measurement}(x)$
$\forall x.\, \mathsf{Pressure}(x) \rightarrow \neg\mathsf{Porosity}(x) \wedge \neg\mathsf{Permeability}(x) \wedge \neg\mathsf{Temperature}(x)$
$\forall x.\, \mathsf{Porosity}(x) \rightarrow \neg\mathsf{Permeability}(x) \wedge \neg\mathsf{Temperature}(x)$
$\forall x.\, \mathsf{Permeability}(x) \rightarrow \neg\mathsf{Temperature}(x)$

$\forall x.\, \mathsf{HydrostaticPressure}(x) \rightarrow \mathsf{Pressure}(x)$
$\forall x.\, \mathsf{FormationPressure}(x) \rightarrow \mathsf{Pressure}(x)$
$\forall x.\, \mathsf{PorePressure}(x) \rightarrow \mathsf{Pressure}(x)$
$\forall x.\, \mathsf{HydrostaticPressure}(x) \rightarrow \neg\mathsf{FormationPressure}(x) \wedge \neg\mathsf{PorePressure}(x)$
$\forall x.\, \mathsf{FormationPressure}(x) \rightarrow \neg\mathsf{PorePressure}(x)$

$\forall x, y.\, \mathsf{hasFormationPressure}(x, y) \rightarrow \mathsf{Wellbore}(x) \wedge \mathsf{FormationPressure}(y)$
$\forall x, y.\, \mathsf{hasDepth}(x, y) \rightarrow \mathsf{FormationPressure}(x) \wedge \mathsf{Depth}(y)$
$\forall x.\, \mathsf{FormationPressure}(x) \rightarrow \exists y.\, \mathsf{hasDepth}(x, y)$

$\forall x, y.\, \mathsf{hasFormationPressure}(x, y) \rightarrow \mathsf{hasMeasurement}(x, y)$

$\forall x, y.\, \mathsf{completionDate}_{\mathsf{Wellbore}}(x, y) \rightarrow \mathsf{Wellbore}(x) \wedge \mathsf{xsd{:}dateTime}(y)$
$\forall x.\, \mathsf{Wellbore}(x) \rightarrow (\sharp\{y \mid \mathsf{completionDate}_{\mathsf{Wellbore}}(x, y)\} \leq 1)$
$\forall x, y.\, \mathsf{wellboreTrack}_{\mathsf{Wellbore}}(x, y) \rightarrow \mathsf{Wellbore}(x) \wedge \mathsf{xsd{:}string}(y)$
$\forall x.\, \mathsf{Wellbore}(x) \rightarrow (\sharp\{y \mid \mathsf{wellboreTrack}_{\mathsf{Wellbore}}(x, y)\} \leq 1)$

$\forall x, y.\, \mathsf{hasCoreSample}(x, y) \rightarrow \mathsf{Core}(x) \wedge \mathsf{CoreSample}(y)$
$\forall x.\, \mathsf{CoreSample}(x) \rightarrow \exists y.\, \mathsf{hasCoreSample}(y, x) \wedge \mathsf{Core}(y)$
$\ldots$

unibz

What is an ontology?

- An ontology conceptualizes a domain of interest in terms of **classes**, (binary) **relations**, and their **properties**.
- It typically organizes the classes in a hierarchical structure.
- Ontologies are often represented as graphs.
- However, we consider an ontology as a **logical theory**, expressed in a suitable fragment of first-order logic, or better, in **description logics**.

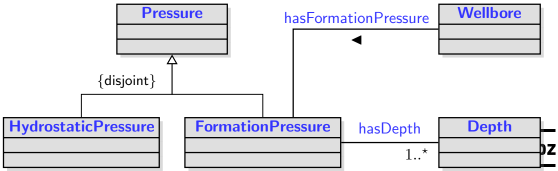
$$
\begin{array}{rcl}
\mathsf{Pressure} & \sqsubseteq & \mathsf{Measurement}\\
\mathsf{Porosity} & \sqsubseteq & \mathsf{Measurement}\\
\mathsf{Permeability} & \sqsubseteq & \mathsf{Measurement}\\
\mathsf{Temperature} & \sqsubseteq & \mathsf{Measurement}\\
\mathsf{Pressure} & \sqsubseteq & \neg\mathsf{Porosity} \sqcap \neg\mathsf{Permeability} \sqcap \neg\mathsf{Temperature}\\
\mathsf{Porosity} & \sqsubseteq & \neg\mathsf{Permeability} \sqcap \neg\mathsf{Temperature}\\
\mathsf{Permeability} & \sqsubseteq & \neg\mathsf{Temperature}\\
\mathsf{HydrostaticPressure} & \sqsubseteq & \mathsf{Pressure}\\
\mathsf{FormationPressure} & \sqsubseteq & \mathsf{Pressure}\\
\mathsf{PorePressure} & \sqsubseteq & \mathsf{Pressure}\\
\mathsf{HydrostaticPressure} & \sqsubseteq & \neg\mathsf{FormationPressure} \sqcap \neg\mathsf{PorePressure}\\
\mathsf{FormationPressure} & \sqsubseteq & \neg\mathsf{PorePressure}\\
\exists\mathsf{hasFormationPressure} & \sqsubseteq & \mathsf{Wellbore}\\
\exists\mathsf{hasFormationPressure}^- & \sqsubseteq & \mathsf{FormationPressure}\\
\exists\mathsf{hasDepth} & \sqsubseteq & \mathsf{FormationPressure}\\
\exists\mathsf{hasDepth}^- & \sqsubseteq & \mathsf{Depth}\\
\mathsf{FormationPressure} & \sqsubseteq & \exists\mathsf{hasDepth}\\
\mathsf{hasFormationPressure} & \sqsubseteq & \mathsf{hasMeasurement}\\
\exists\mathsf{completionDate}_{\mathsf{Wellbore}} & \sqsubseteq & \mathsf{Wellbore}\\
\exists\mathsf{completionDate}^-_{\mathsf{Wellbore}} & \sqsubseteq & \mathsf{xsd{:}dateTime}\\
\mathsf{Wellbore} & \sqsubseteq & (\leq 1\ \mathsf{completionDate}_{\mathsf{Wellbore}})\\
\exists\mathsf{wellboreTrack}_{\mathsf{Wellbore}} & \sqsubseteq & \mathsf{Wellbore}\\
& \dots &
\end{array}
$$

The OWL 2 QL ontology language

- **OWL 2 QL** is one of the three profiles of OWL 2. [W3C Rec. 2012]
- Based on the *DL-Lite* family of descriptions logics. [Baader, C., et al. 2003]

$$\begin{array}{rcll}
\mathsf{FormationPressure} & \sqsubseteq & \mathsf{Pressure} & \text{subclass}\\
\mathsf{FormationPressure} & \sqsubseteq & \neg\mathsf{HydrostaticPressure} & \text{disjointness}\\
\exists\mathsf{hasFormationPressure} & \sqsubseteq & \mathsf{Wellbore} & \text{domain}\\
\exists\mathsf{hasFormationPressure}^- & \sqsubseteq & \mathsf{FormationPressure} & \text{range}\\
\mathsf{FormationPressure} & \sqsubseteq & \exists\mathsf{hasDepth} & \text{mandatory participation}\\
\mathsf{hasFormationPressure} & \sqsubseteq & \mathsf{hasMeasurement} & \text{sub-association}\\
 & \cdots & &
\end{array}$$

OWL 2 QL captures conceptual modeling formalisms (UML class diagrams, ER schemas) [Lenzerini and Nobili 1990; Bergamaschi and Sartori 1992; Borgida 1995; C., Lenzerini, and Nardi 1999; Borgida and Brachman 2003; Berardi, C., and De Giacomo 2005; Queralt et al. 2012].



Query answering – Which query language to use

Querying under incomplete information

Query answering is not simply query evaluation, but a form of logical inference, and requires reasoning.

Two borderline cases for choosing the language for querying ontologies:

1. Use the **ontology language** as query language.
   - Ontology languages are tailored for capturing intensional relationships.
   - They are quite poor as query languages.
2. Use **Full SQL** (or equivalently, first-order logic).
   - Problem: in a setting with incomplete information, query answering is undecidable (FOL validity).

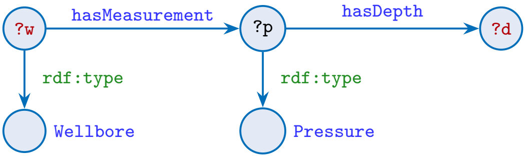
Conjunctive queries – Are concretely represented in **SPARQL**

A good tradeoff is to use conjunctive queries (CQs) or unions of CQs (UCQs), corresponding to SQL/relational algebra (union) select-project-join queries.

SPARQL query language

- Is the standard query language for RDF data. [W3C Rec. 2008, 2013]
- Core query mechanism is based on **graph matching**.

```
SELECT ?w ?d
WHERE { ?w rdf:type Wellbore .
        ?w hasMeasurement ?p .
        ?p rdf:type Pressure .
        ?p hasDepth ?d
      }
```



Additional language features (SPARQL 1.1):

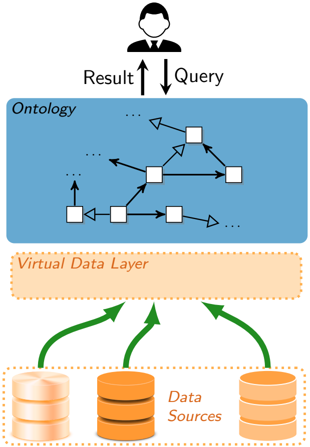
- `UNION`: matches one of alternative graph patterns
- `OPTIONAL`: produces a match even when part of the pattern is missing
- complex `FILTER` conditions
- `GROUP BY`, to express aggregations
- `MINUS`, to remove possible solutions
- property paths (regular expressions)
- ...

Use of mappings

In OBDA, the **mapping** $\mathcal{M}$ encodes how the data $\mathcal{D}$ in the sources should be used to populate the elements of the ontology $\mathcal{O}$.

Virtual data layer $\mathcal{V} = \mathcal{M}(\mathcal{D})$ defined from $\mathcal{M}$ and $\mathcal{D}$

- Queries are answered with respect to $\mathcal{O}$ and $\mathcal{V}$.
- The data of $\mathcal{V}$ is not materialized (it is virtual!).
- Instead, the information in $\mathcal{O}$ and $\mathcal{M}$ is used to translate queries over $\mathcal{O}$ into queries formulated over the sources.



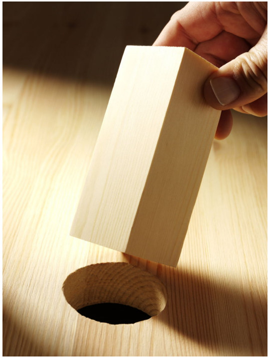
unibz

Mismatch between data layer and ontology

Impedance mismatch

- Relational databases store values.
- Ontologies represent both objects and values.

We need to construct the ontology objects from the database values.



Proposed solution

The specification of **how to construct the ontology objects** that populate the virtual data layer from the database values **is embedded in the mapping** between the data sources and the ontology.

Mapping language

The **mapping** consists of a set of assertions of the form

$$\Phi(\vec{x}) \rightsquigarrow \Psi(\vec{t}, \vec{x})$$

where

- $\Phi(\vec{x})$ is the source query in SQL,
- $\Psi(\vec{t}, \vec{x})$ is the target query, consisting of atoms in the ontology vocabulary.

To address the impedance mismatch

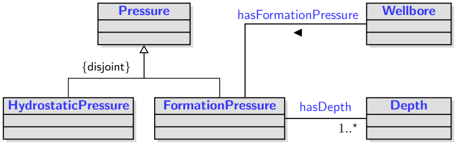
In the target query, we make use of a function `iri`, which constructs object identifiers (IRIs) from database values and string constants by concatenation.

We call a term making use of the `iri` function, an **IRI-template**.

unibz

Mapping language – Example

Ontology $\mathcal{O}$:



Database $\mathcal{D}$:

WELLBORE		
IDENTIFIER	*REF_EXISTENCE_KIND*	...
16/1-29_S	actual	...
30/8-5	actual	...
33/10-12	planned	...

Mapping $\mathcal{M}$:

```
SELECT IDENTIFIER FROM WELLBORE
WHERE REF_EXISTENCE_KIND = 'actual'
   ⇝ Wellbore(iri("wb-", IDENTIFIER))
```

We obtain the virtual data layer $\mathcal{M}(\mathcal{D})$: Wellbore(wb-16/1-29_S)
Wellbore(wb-30/8-5)

unibz

Concrete mapping languages

Several proposals for concrete languages to map a relational DB to an ontology:

- They assume that the ontology is populated in terms of RDF triples.
- Some template mechanism is used to specify the triples to instantiate.

Examples: D2RQ[2], SML[3], Ontop[4]

R2RML

- Most popular RDB to RDF mapping language
- W3C Recommendation 27 Sep. 2012, `http://www.w3.org/TR/r2rml/`
- R2RML mappings are themselves expressed as RDF graphs and written in Turtle syntax.

[2] http://d2rq.org/d2rq-language
[3] http://sparqlify.org/wiki/Sparqlification_mapping_language
[4] https://github.com/ontop/ontop/wiki/ontopOBDAModel#Mapping_axioms

unibz

Formalizing OBDA

OBDA specification $\mathcal{P} = \langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle$ and **OBDA instance** $\langle \mathcal{P}, \mathcal{D} \rangle$

- $\mathcal{O}$ is an ontology (expressed in OWL 2 QL),
- $\mathcal{M}$ is a set of (R2RML) mapping assertions,
- $\mathcal{S}$ is a (relational) database schema with integrity constraints,
- $\mathcal{D}$ is a database conforming to $\mathcal{S}$.

Semantics:

A first-order interpretation $\mathcal{I}$ of the ontology predicates is a **model** of $\langle \mathcal{P}, \mathcal{D} \rangle$ if

- it satisfies all axioms in $\mathcal{O}$, and
- contains all facts in $\mathcal{M}(\mathcal{D})$, i.e., retrieved through $\mathcal{M}$ from $\mathcal{D}$.

Note:

- In general, $\langle \mathcal{P}, \mathcal{D} \rangle$ has infinitely many models, and some of these might be infinite.
- However, for query answering, we do not need to compute such models.

Query answering in OBDA – Certain answers

In OBDA, we want to answer queries formulated over the ontology, by using the data provided by the data sources through the mapping.

Consider our formalization of OBDA and an OBDA instance $\mathcal{J} = \langle \mathcal{P}, \mathcal{D} \rangle$.

Certain answers

Given an OBDA instance $\mathcal{J}$ and a query q over $\mathcal{J}$, the certain answers to q are those answers that hold in all models of $\mathcal{J}$.

unibz

First-order rewritability

To make computing certain answers viable in practice, OBDA relies on reducing it to evaluating SQL (i.e., first-order logic) queries over the data.

Consider an OBDA specification $\mathcal{P} = \langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle$.

First-order rewritability

A query $r(\vec{x})$ is a **first-order rewriting** of a query $q(\vec{x})$ with respect to $\mathcal{P}$ if, for every source DB $\mathcal{D}$,

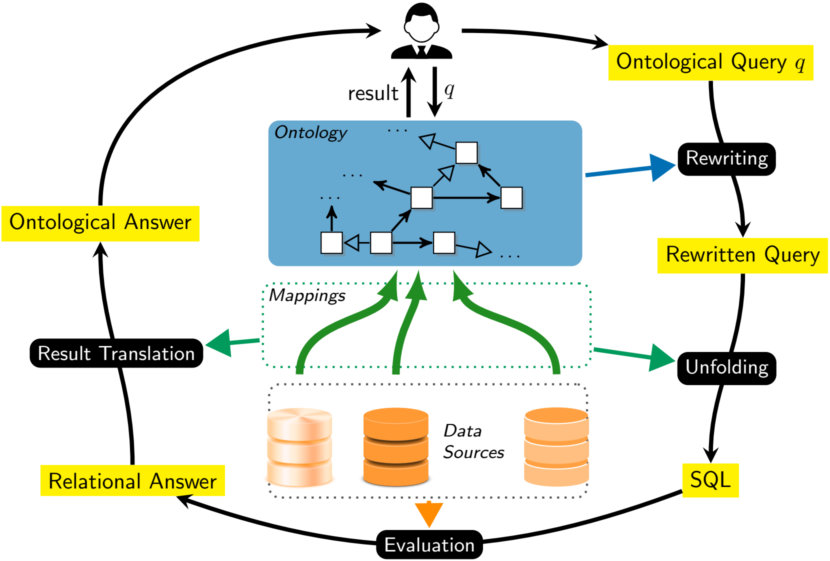
certain answers to $q(\vec{x})$ over $\langle \mathcal{P}, \mathcal{D} \rangle$ $=$ answers to $r(\vec{x})$ over $\mathcal{D}$.

For OWL 2 QL ontologies and R2RML mappings,
(core) SPARQL queries are first-order rewritable.

In other words, **in OBDA, we can compute the certain answers to a SPARQL query by evaluating over the sources its rewriting, which is an SQL query.**

unibz

Query answering by query rewriting



OBDA is by now a mature technology

Ontology-based querying of relational data sources is supported by several systems, both open-source and commercial:

- Mastro [C., De Giacomo, et al. 2011] [5]
  Sapienza Università di Roma & OBDA systems SRL, Italy
- Morph [Priyatna, Corcho, and Sequeda 2014] [6]
  Technical University of Madrid, Spain
- Ontop [C., Cogrel, et al. 2017] [7]
  Free University of Bozen-Bolzano, Italy
- Stardog [8], Stardog Union, US
- Ultrawrap [Sequeda and Miranker 2013] [9], Capsenta, US
- Oracle Spatial and Graph [10]

---

[5] http://www.obdasystems.com/it/mastro
[6] https://github.com/oeg-upm/morph-rdb
[7] http://ontop.inf.unibz.it
[8] http://www.stardog.com
[9] https://capsenta.com/ultrawrap
[10] http://www.oracle.com/technetwork/database/options/spatialandgraph

unibz

Ontop

ontop

http://ontop.inf.unibz.it/

- State-of-the-art OBDA system developed at the Free University of Bozen-Bolzano.
- Compliant with all relevant Semantic Web standards:
  RDF, RDFS, OWL 2 QL, R2RML, and SPARQL
- 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
  - already well established and widely adopted:
    +200 members in the mailing list
    +9000 downloads in last 2 years
  - main development was carried out in the context of the EU project Optique

unibz

Some use cases of *Ontop*

- EU FP7 Project Optique: Scalable End-user Access to Big Data
  - November 2012 – October 2016, 10 Partners
  - *Ontop* is core component of the Optique platform
  - Industrial Partners: Statoil, Siemens, DNV
- Siemens Corportate Technologie is experimenting with managing temporal and streaming data
- EU ERC Advanced Grant EPNet Project in the cultural heritage domain
  - EPNet: "Production and distribution of food during the Roman Empire: Economics and Political Dynamics"
- German BMBF Project EMSec
  - EMSec: Real-time Services for the Maritime Security
  - Collaborated with Airbus Defence & Space
- IBM is using *Ontop* for several internal projects
- Commercial adoption
  - Stardog by Complexible Inc.
  - Fluidops Information Workbench (Optique platform)
  - Metaphacts semantic data management platform

unibz

Extending OBDA

In several real-world application domains, data access is more complex!

- data are not always structured in relations:
  graphs; trees/noSQL; csv-files; textual data, possibly annotated
- **temporal data**
- geospatial data
- streaming data
- **multiple heterogeneous data sources**

Notably, **open data** typically combines many of the above.

Users have also additional requests:

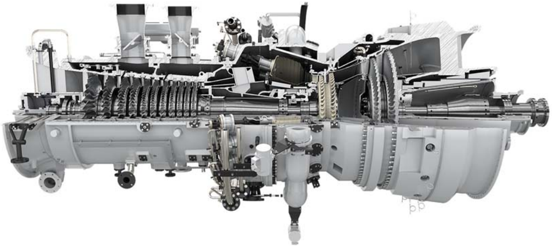
- richer querying capabilities, including aggregation and analytics
- more expressive power in the ontology, with reasoning support
- improved performance
- friendly interfaces
- data management (e.g., updates), mediated by the ontology

Outline

unibz

Siemens Energy Services

- Monitor gas and steam turbines.
- Collect data from 50 remote diagnostic centers around the world.
- Centers linked to a common central DB.
- Turbines are highly complex, with 5 000–50 000 sensors each.



Objective: retrospective diagnostics

i.e., detect abnormal or potentially dangerous events.

Events

- Involve a number of sensor measurements.
- Have a certain temporal duration.
- Occur in a certain temporal sequence.

Example request

Find the gas turbines deployed in the train with ID T001, and the time periods of their accomplished purgings.

To capture such a complex scenario . . .

. . . we need to **enrich OBDA with temporal features**.

Approaches proposed in the literature:

1. Use standard ontologies and extend queries with temporal operators

[Gutiérrez-Basulto and Klarman 2012; Baader, Borgwardt, and Lippmann 2013; Klarman and Meyer 2014; Özçep and Möller 2014; Kharlamov et al. 2016]

However:

- Query language gets significantly more complicated.
- Effort is shifted from design time to query time.

2. Extend both query and ontology with linear temporal logic (LTL) operators

[Artale, Kontchakov, Wolter, et al. 2013; Artale, Kontchakov, Kovtunova, et al. 2015]

However:

- LTL is not suited to deal with metric temporal information.

We propose a different approach to temporal OBDA

- At the ontology level, we have both **static** and **temporal predicates**:
  - Static predicates to represent ordinary facts.
    E.g., $\mathsf{Burner}(\mathtt{b01})$, $\mathsf{isMonitoredBy}(\mathtt{b01}, \mathtt{mf01})$
  - Temporal predicates to represent temporal facts with a validity interval
    E.g., $\mathsf{HighRotorSpeed}(\mathtt{rs01})@[\mathtt{2017\text{-}06\text{-}06\ 12{:}22{:}50},\ \mathtt{2017\text{-}06\text{-}06\ 12{:}23{:}40})$
    We consider both open and closed intervals:
    $A(d)@(t_1, t_2)$, $A(d)@[t_1, t_2)$, $A(d)@(t_1, t_2]$, $A(d)@[t_1, t_2]$
- The ontology is expressed in OWL 2 QL $\leadsto$ First-order rewritability.
- We enrich it with static and temporal rules.
- We extend the mapping mechanisms so as to retrieve also temporal information from the data, i.e., both static and temporal facts.

unibz

Formal framework for temporal OBDA

A **traditional OBDA specification** is a triple $\mathcal{P} = \langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle$

- $\mathcal{O}$ is an ontology.
- $\mathcal{M}$ is a set of mapping assertions between ontology and data sources.
- $\mathcal{S}$ is a database schema.

Temporal OBDA builds on traditional OBDA.

A **temporal OBDA specification** is a tuple $\mathcal{P}_t = \langle \Sigma_s, \Sigma_t, \mathcal{O}, \mathcal{R}_s, \mathcal{R}_t, \mathcal{M}_s, \mathcal{M}_t, \mathcal{S} \rangle$

- Σ_s is a static vocabulary.
- $\mathcal{O}$ is an ontology.
- $\mathcal{R}_s$ is a set of static rules.
- $\mathcal{M}_s$ is a set of static mapping assertions.
- $\mathcal{S}$ is a database schema.
- Σ_t is a temporal vocabulary.
- $\mathcal{R}_t$ is a set of temporal rules.
- $\mathcal{M}_t$ is a set of temporal mapping assertions.

Static ontology – Example

We use an **ontology** to model the **static knowledge** about

- machines and their deployment profiles
- component hierarchies
- sensor configurations
- functional profiles

We still use **OWL 2 QL** as the static ontology language.

Devices consist of parts, and these are monitored by many different kinds of sensors (temperature, pressure, vibration etc.).

$$
\begin{aligned}
\mathsf{GasTurbine} &\sqsubseteq \mathsf{Turbine} & \exists\mathsf{isDeployedIn} &\sqsubseteq \mathsf{Turbine}\\
\mathsf{SteamTurbine} &\sqsubseteq \mathsf{Turbine} & \exists\mathsf{isDeployedIn}^- &\sqsubseteq \mathsf{Train}\\
\mathsf{PowerTurbine} &\sqsubseteq \mathsf{TurbinePart} & \exists\mathsf{isPartOf} &\equiv \mathsf{TurbinePart}\\
\mathsf{Burner} &\sqsubseteq \mathsf{TurbinePart} & \exists\mathsf{isPartOf}^- &\sqsubseteq \mathsf{Turbine}\\
\mathsf{RotationSpeedSensor} &\sqsubseteq \mathsf{Sensor} & \exists\mathsf{isMonitoredBy} &\sqsubseteq \mathsf{TurbinePart}\\
\mathsf{TemperatureSensor} &\sqsubseteq \mathsf{Sensor} & \exists\mathsf{isMonitoredBy}^- &\sqsubseteq \mathsf{Sensor}
\end{aligned}
$$

Static rules

However, OWL 2 QL is not able to capture all the static knowledge required, e.g., in the Siemens use case.

We complement this ontology with **nonrecursive Datalog static rules**.

Example: turbine parts monitored by different co-located sensors (e.g., temperature, rotation speed)

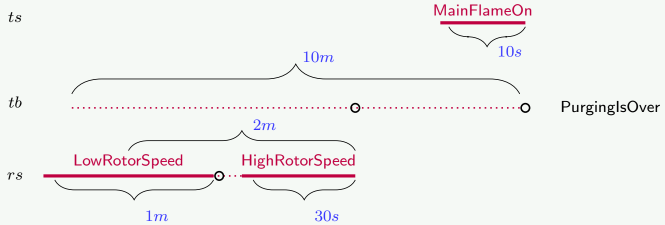
$$\begin{aligned}\mathsf{ColocSensors}(tb, ts, rs) \leftarrow\ & \mathsf{Turbine}(tb),\ \mathsf{isPartOf}(pt, tb),\\ & \mathsf{isMonitoredBy}(pt, ts),\ \mathsf{TemperatureSensor}(ts),\\ & \mathsf{isMonitoredBy}(pt, rs),\ \mathsf{RotationSpeedSensor}(rs).\end{aligned}$$

unibz

Temporal rules

Siemens is interested in detecting abnormal situations, and monitoring running tasks.

“**Purging is Over**” is a complex event of a turbine



We model this situation with metric temporal rules:

$$
\begin{aligned}
\mathsf{PurgingIsOver}(tb) \;\leftarrow\;& \boxminus_{[0s,10s]}\mathsf{MainFlameOn}(ts) \land \\
& \diamondminus_{(0,10m]}\big(\boxminus_{(0,30s]}\mathsf{HighRotorSpeed}(rs) \land \\
& \qquad \diamondminus_{(0,2m]}\boxminus_{(0,1m]}\mathsf{LowRotorSpeed}(rs)\big) \land \\
& \mathsf{ColocTempRotSensors}(tb, ts, rs). \\
\mathsf{HighRotorSpeed}(tb) \;\leftarrow\;& \mathsf{rotorSpeed}(tb, v) \land v > 1260. \\
\mathsf{LowRotorSpeed}(tb) \;\leftarrow\;& \mathsf{rotorSpeed}(tb, v) \land v < 1000.
\end{aligned}
$$

We use *DatalogMTL*

DatalogMTL is a Horn fragment of Metric Temporal Logic (MTL).

A **DatalogMTL** *program* is a finite set of rules of the form

$$A^+ \leftarrow A_1 \wedge \cdots \wedge A_k \qquad \text{or} \qquad \bot \leftarrow A_1 \wedge \cdots \wedge A_k,$$

where

- each A_i is either $\tau \neq \tau'$, or defined by the grammar

$$A \ ::= \ P(\tau_1, \ldots, \tau_m) \ \mid \ \boxplus_\varrho A \ \mid \ \boxminus_\varrho A \ \mid \ \diamondplus_\varrho A \ \mid \ \diamondminus_\varrho A$$

  where ϱ denotes a (left/right open or closed) interval with non-negative endpoints,
- A^+ does not contain $\diamondplus_\varrho$ or $\diamondminus_\varrho$ (since this would lead to undecidability).

unibz

Query evaluation in *DatalogMTL*

Theorem ([Brandt et al. 2017])

Answering *DatalogMTL* queries is EXPSPACE-complete in combined complexity.

We consider the nonrecursive fragment $Datalog_{nr}MTL$ of *DatalogMTL*:

- sufficient expressive power for many real-world situations
- computationally well-behaved

Answering $Datalog_{nr}MTL$ queries:

- Is PSPACE-complete in combined complexity.
- Is **in AC^0 in data complexity**.
- The problem can be reduced to SQL query evaluation.

Hence, $Datalog_{nr}MTL$ is well suited as a temporal rule language for OBDA.

Data sources: schema and data

Data sources often contain temporal information in the form of time-stamps.

Example data schema $\mathcal{S}$ for the Siemens data

It includes time-stamped sensor measurements and deployment details:

tb_measurement(timestamp, sensor_id, value),
tb_sensors(sensor_id, sensor_type, mnted_part, mnted_tb),
tb_components(turbine_id, component_id, component_type).

A corresponding data instance $\mathcal{D}_0$:

tb_measurement

timestamp	*sensor_id*	*value*
2017-06-06 12:20:00	rs01	570
2017-06-06 12:22:50	rs01	1278
2017-06-06 12:23:40	rs01	1310
...	...	...
2017-06-06 12:32:30	mf01	2.3
2017-06-06 12:32:50	mf01	1.8
2017-06-06 12:33:40	mf01	0.9
...	...	...

tb_sensors

sensor_id	*sensor_type*	*mnted_part*	*mnted_tb*
rs01	0	pt01	tb01
mf01	1	b01	tb01
...	...	...	...

tb_components

turbine_id	*component_id*	*component_type*
tb01	pt01	0
tb01	b01	1
...	...	...

Static mapping assertions in $\mathcal{M}_s$

Static mapping assertions: $\Phi(\vec{x}) \rightsquigarrow \Psi(\vec{x})$

- $\Phi(\vec{x})$ is a query over the source schema $\mathcal{S}$
- $\Psi(\vec{x})$ is an atom with predicate in Σ_s

Example

```
SELECT sensor_id AS X FROM tb_sensors
WHERE sensor_type = 1                                        ⇝ TemperatureSensor(X)

SELECT component_id AS X FROM tb_components
WHERE component_type = 1                                     ⇝ Burner(X)

SELECT mnted_part AS X, sensor_id AS Y FROM tb_sensors       ⇝ isMonitoredBy(X, Y)
```

These mappings retrieve from the database ordinary facts.

$$\mathsf{Burner}(\mathtt{b01}), \quad \mathsf{TemperatureSensor}(\mathtt{mf01}),$$
$$\mathsf{isMonitoredBy}(\mathtt{pt01}, \mathtt{rs01}), \quad \mathsf{isMonitoredBy}(\mathtt{b01}, \mathtt{mf01}).$$

Temporal mapping assertions in $\mathcal{M}_t$

Temporal mapping assertions: $\Phi(\vec{x}, \mathtt{begin}, \mathtt{end}) \leadsto \Psi(\vec{x})@\langle t_{\mathtt{begin}}, t_{\mathtt{end}}\rangle$

- begin and end are variables returning a date/time.
- '$\langle$' is either '(' or '[', and similarly for '$\rangle$'.
- $\Psi(\vec{x})$ is an atom with predicate in Σ_t.
- $t_{\mathtt{begin}}$ is either begin or a date-time constant, and similarly for $t_{\mathtt{end}}$.

Example

```
SELECT * FROM (
  SELECT sensor_id, value, timestamp AS begin,
         LEAD(timestamp,1) OVER W AS end
  FROM tb_measurement, tb_sensors
  WINDOW W AS (PARTITION BY sensor_id ORDER BY timestamp)
  WHERE tb_measurement.sensor_id = tb_sensors.sensor_id AND sensor_type = 0
) SUBQ WHERE value > 1260
```

$\leadsto$ HighRotorSpeed(sensor_id)@[begin,end)

These mappings retrieve from the database temporal facts.

HighRotorSpeed(rs01)@[2017-06-06 12:22:50, 2017-06-06 12:23:40)

Concrete syntax for temporal OBDA specifications

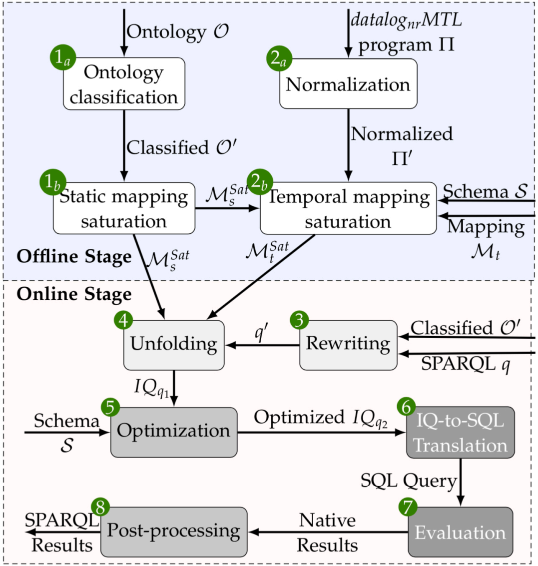
Temporal OBDA specification $\mathcal{P}_t = \langle \Sigma_s, \Sigma_t, \mathcal{O}, \mathcal{R}_s, \mathcal{R}_t, \mathcal{M}_s, \mathcal{M}_t, \mathcal{S} \rangle$

- Σ_s is a static vocabulary,
- Σ_t is a temporal vocabulary,
- $\mathcal{O}$ is an ontology,
- $\mathcal{R}_s$ is a set of static rules,
- $\mathcal{R}_t$ is a set of temporal rules,
- $\mathcal{M}_s$ is a set of static mapping assertions,
- $\mathcal{M}_t$ is a set of temporal mapping assertions,
- $\mathcal{S}$ is a database schema.

Component	defines predicates in	in terms of predicates in	Adopted language
$\mathcal{O}$	Σ_s	Σ_s	OWL 2 QL
$\mathcal{R}_s$	Σ_s	Σ_s	non-recursive Datalog
$\mathcal{R}_t$	Σ_t	$\Sigma_s \cup \Sigma_t$	$Datalog_{nr}MTL$
$\mathcal{M}_s$	Σ_s	$\mathcal{S}$	R2RML / Ontop
$\mathcal{M}_t$	Σ_t	$\mathcal{S}$	R2RML / Ontop

unibz

System workflow for temporal OBDA in *Ontop*



We are currently working on the implementation:

- already available in *Ontop*: 1_a, 1_b, 7, 8
- new components are being implemented: 2_a, 2_b
- components need to be extended: 3, 4, 5, 6.

unibz

Outline

unibz

Issues when integrating multiple data sources

- Heterogeneity of data sources and data models
  $\rightsquigarrow$ Handled through a federation layer, such as Teeid, Denodo, or Exareme.
- Semantic heterogeneity
  $\rightsquigarrow$ Can in part be handled through the mapping layer. Might require meta-modeling capabilities in the ontology [Lenzerini, Lepore, and Poggi 2016],
- Heterogeneity in the representation of real-world entities, hence there is need for object/entity matching.
  $\rightsquigarrow$ **This is what I want to discuss now.**

Problems when integrating multiple data sources

The information about one real-world entity can be distributed over several data sources.

Entity resolution

Understand which records actually represent the same real world entity.

We assume that this information is available and/or known to the integration system designer.

Need for Integrated querying

Answer queries that require to integrate data items representing the same entity, but coming from different data sources.

OBDI – Example

Consider two databases `nat` and `corp` with one table each (keys in red):

nat.wellbore		
name	*wbField*	*opPurpose*
2-1	BLANE	WILDCAT
3-1		WILDCAT
3-10	OSELVAR	APPRAISAL
4-2	EKOFISK	WILDCAT

corp.drillingops		
name	*driStDt*	*reason*
NO-2-1	20-03-1989	WILDCAT
NO-3-1	06-07-1968	WILDCAT
NO-3-A	22-07-2011	PRODUCTION
NO-4-2	18-09-1969	

Mapping assertions make use of **different IRI-templates**

```
SELECT name, wbField, opPurpose FROM nat.wellbore
  ⇝ inField(iri("NatWB/",name), wbField), purpose(iri("NatWB/",name), opPurpose)
SELECT name, driStDt, reason FROM corp.drillingops
  ⇝ drillingStarted(iri("CorpWB/",name), driStDt), purpose(iri("CorpWB/",name), reason)
```

Some fact obtained in the virtual data layer by the DBs and mapping

inField(NatWB/2-1, BLANE), purpose(NatWB/2-1, WILDCAT), ...
drillingStarted(CorpWB/NO-2-1, 20-03-1989), purpose(CorpWB/NO-2-1, WILDCAT), ...

Integrated querying – Example

nat.wellbore		
name	*wbField*	*opPurpose*
2-1	BLANE	WILDCAT
3-1		WILDCAT
3-10	OSELVAR	APPRAISAL
4-2	EKOFISK	WILDCAT

corp.drillingops		
name	*driStDt*	*reason*
NO-2-1	20-03-1989	WILDCAT
NO-3-1	06-07-1968	WILDCAT
NO-3-A	22-07-2011	PRODUCTION
NO-4-2	18-09-1969	

Some fact obtained in the virtual data layer by the DBs and mapping

inField(NatWB/2-1, BLANE), purpose(NatWB/2-1, WILDCAT), ...
drillingStarted(CorpWB/NO-2-1, 20-03-1989), purpose(CorpWB/NO-2-1, WILDCAT), ...

Intuitively, 2-1 in nat.wellbore and NO-2-1 in corp.drillingops represent the same wellbore.

Hence the SPARQL query

```
SELECT ?w ?f ?d WHERE { ?w inField ?f .  ?w drillingStarted ?d }
```

should return some answers, e.g., the triple (NatWB/2-1, BLANE, 20-3-1989).

Integrated querying in OBDI

Can be achieved by merging the data.

Physically merge the data (as done in ETL).

- Requires full control over the data sources.
- Requires to move the data $\leadsto$ issues with freshness, privacy, legal aspects.

$\leadsto$ Not possible in many real world scenarios!

Virtually merge the data using the standard `sameAs` construct of the OWL language, and mappings [Calvanese et al. 2015, ISWC].

- `sameAs` is the standard way of dealing with identity resolution in OWL.
- Semantics of `sameAs` may cause an exponential number of query results:
  - detrimental for performance
  - redundancy makes query answers difficult to understand

$\leadsto$ Not feasible or desirable in practice!

Approach based on canonical IRIs

Canonical IRIs

- Each entity may have several IRIs, but only **a single canonical representation**.
- This breaks the symmetry between the different representations, and avoids the exponential blowup.

We want to achieve that the virtual data layer $\mathcal{M}(\mathcal{D})$ contains **canonical IRI assertions**, which relate IRIs to their canonical representation using the binary predicate `canIriOf`.

Example canonical IRI assertions

`canIriOf (WB/2, NatWB/2-1)`   `canIriOf (WB/2, CorpWB/NO-2-1)`

We need to ensure that **each IRI has at most one canonical IRI.**

Formally: `canIriOf` is inverse functional in $\mathcal{M}(\mathcal{D})$:

$$\{\ \texttt{canIriOf}(c_1, o),\ \texttt{canIriOf}(c_2, o)\ \} \subseteq \mathcal{M}(\mathcal{D}) \quad \text{implies} \quad c_1 = c_2.$$

Query answering under canonical IRIs

To deal with canonical IRIs efficiently, we would like to resort to query rewriting:

- One can formalize the semantics of `canIriOf` and relate it to that of `sameAs` (technically, one defines a suitable SPARQL entailment regime [Xiao et al. 2018, ESWC].
- However, the canonical IRI entailment regime is non-monotonic, hence the rewritten query needs to contain some form of negation.
- A rewriting can indeed be constructed by using `NOT EXISTS`.
- However, the resulting query would contain a `NOT EXISTS` clause for each variable in the original query, and would be rather inefficient.

unibz

Handling canonical IRI statements in OBDI

- We propose a practical approach for canonical IRI semantics in OBDI.
- We assume that the mapping $\mathcal{M}$ includes **assertions $\mathcal{M}^{can}$ that populate** `canIriOf`.
- The mapping $\mathcal{M}^{can}$ may be fed from master tables, typical of many corporate scenarios.
- However, we do not rely on master tables, and may use arbitrary SQL queries to ordinary tables.

Example master table and mapping

central.masterTable

id	natName	corpName
2	2-1	NO-2-1
3	3-1	NO-3-1
4	4-2	NO-4-2
5		NO-3-A
6	3-10	

```
SELECT id, natName FROM central.masterTable
  ⇝ canIriOf(iri("WB/",id), iri("NatWB/",natName))

SELECT id, corpName FROM central.masterTable
  ⇝ canIriOf(iri("WB/",id), iri("CorpWB/",corpName))
```

Mapping rewriting to deal with canonical IRIs

- We propose a practical method based on compiling the consequences of canonical IRI semantics into mappings $\rightsquigarrow$ **Mapping rewriting**
- Inspired by the mapping saturation algorithm in classical OBDA.
- We need to ensure inverse functionality of `canIriOf`.

Assumption on the mappings

For each IRI template **iri**, at most one mapping assertion in $\mathcal{M}^{can}$ of the form:

$$\mathtt{sql}(\vec{a},\vec{b}) \rightsquigarrow \mathtt{canIriOf}(\mathbf{iri}_c(\vec{a}),\ \mathbf{iri}(\vec{b}))$$

Note:

- This assumption suffices: if $\mathcal{M}^{can}$ satisfies it, then for every database $\mathcal{D}$, `canIriOf` is inverse functional in the extracted (virtual) data layer $\mathcal{M}^{can}(\mathcal{D})$.
- Is stronger than inverse functionality of `canIriOf`.
- But is reasonable in practice.

unibz

Mapping rewriting algorithm

To rewrite the mapping, we replace individuals and IRI-templates in the mapping by their canonical representation.

Let $\mathcal{M} = \mathcal{M}^{orig} \cup \mathcal{M}^{can}$ be a set of mapping assertions.

Canonical-iri rewriting $cm(\mathcal{M}^{orig}, \mathcal{M}^{can})$ of $\mathcal{M}$

Is obtained by processing each mapping assertion $ma \in \mathcal{M}^{orig}$ as follows:

1. For each IRI template $\textbf{iri}(\vec{a})$ in ma, if $\mathcal{M}^{can}$ contains a mapping assertion
$$sql(\vec{b}_0, \vec{b}_1) \rightsquigarrow \texttt{canIriOf}(\textbf{iri}_c(\vec{b}_0), \textbf{iri}(\vec{b}_1))$$
then replace $\textbf{iri}(\vec{a})$ in the target of ma by $\textbf{iri}_c(\vec{b}_0)$, and
join the source query of ma with $sql(\vec{b}_0, \vec{b}_1), \vec{a} = \vec{b}_1$.
2. Process IRIs directly occurring in ma in the same way.

unibz

Mapping rewriting – Example

Mapping $\mathcal{M}^{orig}$

1. `SELECT name, wbField, opPurpose FROM nat.wellbore`
   $\rightsquigarrow$ `inField(iri("NatWB/",name), wbField),  purpose(iri("NatWB/",name), opPurpose)`
2. `SELECT name, driStDt, reason FROM corp.drillingops`
   $\rightsquigarrow$ `drillingStarted(iri("CorpWB/",name), driStDt),  purpose(iri("CorpWB/",name), reason)`

Mapping $\mathcal{M}^{can}$

1. `SELECT id, natName FROM central.masterTable`
   $\rightsquigarrow$ `canIriOf(iri("WB/",id), iri("NatWB/", natName))`
2. `SELECT id, corpName FROM central.masterTable`
   $\rightsquigarrow$ `canIriOf(iri("WB/",id), iri("CorpWB/", corpName))`

Canonical-iri rewriting $cm(\mathcal{M}^{orig}, \mathcal{M}^{can})$ of $\mathcal{M}^{orig} \cup \mathcal{M}^{can}$

1. `SELECT wlbFld, opPurp, id FROM nat.wellbore, central.masterTable WHERE name = natName`
   $\rightsquigarrow$ `inField(iri("WB/",id), wlbField),  purpose(iri("WB/",id), opPurp)`
2. `SELECT driStDt, reason, id FROM corp.drillingops, central.masterTable WHERE name = corpName`
   $\rightsquigarrow$ `drillingStarted(iri("WB/",id), driStDt),  purpose(iri("WB/",id), reason)`

Correctness of mapping rewriting

- Let $\mathcal{M}^{orig}$ be a traditional mapping.
- Let $\mathcal{M}^{can}$ be a mapping for `canIriOf`.

The mapping rewriting algorithm cm preserves the semantics of $\mathcal{M}^{orig} \cup \mathcal{M}^{can}$, i.e., for every database $\mathcal{D}$:

$cm(\mathcal{M}^{orig}, \mathcal{M}^{can})(\mathcal{D})$ is the set of facts of $\mathcal{M}^{orig}(\mathcal{D})$, but where each individual is replaced by its canonical representative according to $\mathcal{M}^{can}(\mathcal{D})$.

It follows that queries can be answered with respect to the rewritten mapping $cm(\mathcal{M}^{o}, \mathcal{M}^{can})$, using standard OBDA query answering.

unibz

Results for *Ontop* over Statoil query catalog

We have implemented the approach in *Ontop*, and applied it to the Statoil use case:

- 7 data sources: DDR, Compass, Slegge, Recall, CoreDB, GeoChemDB, and OpenWorks
- We have exploited existing master tables.
- The mappings for canonical IRIs are simple mappings into these tables.
- Query catalog with 76 challenging SPARQL queries constructed from information needs by geologists and geoscientists.

Results:

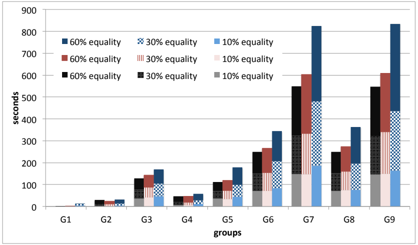
	sameAs	canonical IRI
Total queries	76	76
Timeouts	31	11
Successful	45	65
Success %	59%	85%
Min exec. time	12s	0.50s
Mean exec. time	11m	4.3m
Median exec. time	11m	0.77m

(limit = 100K tuples, timeout = 20 minutes)

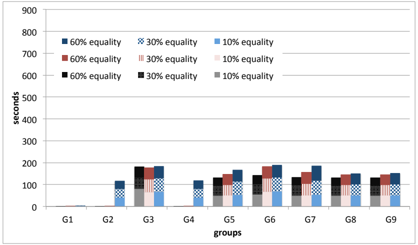
unibz

Results over benchmark data – Execution times of most expensive queries

2 datasets:

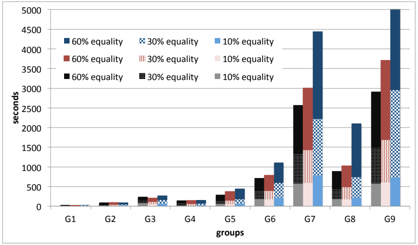


Standard `owl:sameAs`

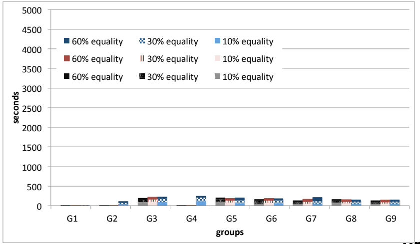


Canonical IRI

3 datasets:



Standard `owl:sameAs`



Canonical IRI

Outline

unibz

Conclusions

- OBDA/I is by now a mature technology to address the data wrangling and data preparation problems.
- However, it has been well-investigated and applied in real-world scenarios mostly for the case of relational data sources.
- Also in that setting, performance and scalability w.r.t. larger datasets (volume), larger and more complex ontologies (variety, veracity), and multiple heterogeneous data sources (variety, volume) is a challenge.
- Only recently OBDA has been investigated for alternative types of data, such as **temporal data**, **noSQL** and tree structured data, **streaming data** (velocity), **linked open data**, and **geo-spatial data**.
- Performance and scalability are even more critical for these more complex domains.

unibz

Further research directions

Theoretical investigations:

- Dealing with data provenance and explanation.
- Dealing with data inconsistency and incompleteness – Data quality!
- Ontology-based update.
- More expressive queries, supporting analytical tasks.
- Coping with evolution of data in the presence of ontological constraints.

From a practical point of view, supporting technologies need to be developed to make the OBDA/I technology easier to adopt:

- Improving the support for multiple, heterogeneous data sources.
- Techniques for (semi-)automatic extraction/learning of ontology axioms and mapping assertions.
- Techniques and tools for efficient management of mappings and ontology axioms, to support design, maintenance, and evolution.
- User-friendly ontology querying modalities (graphical query languages, natural language querying).

unibz

Thanks

Thank you for your attention!

. . . and thanks to many people who contributed to this work:

- Elena Botoeva (unibz)
- Benjamin Cogrel (unibz)
- Julien Corman (unibz)
- Giuseppe De Giacomo (Uniroma1)
- Elem Güzel Kalayci (unibz)
- Sarah Komla-Ebri (unibz)
- Roman Kontchakov (Birkbeck)
- Davide Lanti (unibz)
- Domenico Lembo (Uniroma1)
- Maurizio Lenzerini (Uniroma1)
- Antonella Poggi (Uniroma1)
- Mariano Rodriguez Muro (unibz, IBM, Google)
- Riccardo Rosati (Uniroma1)
- Vladislav Ryzhikov (unibz, Birkbeck)
- Guohui Xiao (unibz)
- Michael Zakharyaschev (Birkbeck)

OBDA framework developed in Bolzano

ontop

ontop.inf.unibz.it/

EU IP Project

Optique

(Nov. 2012 – Oct. 2016)

unibz

References I

[1] Franz Baader, Diego C., Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, eds. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[2] Maurizio Lenzerini and Paolo Nobili. “On the Satisfiability of Dependency Constraints in Entity-Relationship Schemata”. In: *Information Systems* 15.4 (1990), pp. 453–461.

[3] Sonia Bergamaschi and Claudio Sartori. “On Taxonomic Reasoning in Conceptual Design”. In: *ACM Trans. on Database Systems* 17.3 (1992), pp. 385–422.

[4] Alexander Borgida. “Description Logics in Data Management”. In: *IEEE Trans. on Knowledge and Data Engineering* 7.5 (1995), pp. 671–682.

[5] Diego C., Maurizio Lenzerini, and Daniele Nardi. “Unifying Class-Based Representation Formalisms”. In: *J. of Artificial Intelligence Research* 11 (1999), pp. 199–240.

[6] Alexander Borgida and Ronald J. Brachman. “Conceptual Modeling with Description Logics”. In: *The Description Logic Handbook: Theory, Implementation and Applications*. Ed. by Franz Baader, Diego C., Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. Cambridge University Press, 2003. Chap. 10, pp. 349–372.

References II

[7] Daniela Berardi, Diego C., and Giuseppe De Giacomo. "Reasoning on UML Class Diagrams". In: *Artificial Intelligence* 168.1–2 (2005), pp. 70–118.

[8] Anna Queralt, Alessandro Artale, Diego C., and Ernest Teniente. "OCL-Lite: Finite Reasoning on UML/OCL Conceptual Schemas". In: *Data and Knowledge Engineering* 73 (2012), pp. 1–22.

[9] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. "The Mastro System for Ontology-Based Data Access". In: *Semantic Web J.* 2.1 (2011), pp. 43–53.

[10] Freddy Priyatna, Oscar Corcho, and Juan F. Sequeda. "Formalisation and Experiences of R2RML-based SPARQL to SQL Query Translation Using morph". In: *Proc. of the 23rd Int. World Wide Web Conf. (WWW)*. 2014, pp. 479–490. DOI: 10.1145/2566486.2567981.

[11] Diego C., Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. "Ontop: Answering SPARQL Queries over Relational Databases". In: *Semantic Web J.* 8.3 (2017), pp. 471–487. DOI: 10.3233/SW-160217.

unibz

References III

[12] Juan F. Sequeda and Daniel P. Miranker. "Ultrawrap: SPARQL Execution on Relational Data". In: *J. of Web Semantics* 22 (2013), pp. 19–39.

[13] Victor Gutiérrez-Basulto and Szymon Klarman. "Towards a Unifying Approach to Representing and Querying Temporal Data in Description Logics". In: *Proc. of the 6th Int. Conf. on Web Reasoning and Rule Systems (RR)*. Vol. 7497. Lecture Notes in Computer Science. Springer, 2012, pp. 90–105. DOI: 10.1007/978-3-642-33203-6_8.

[14] Franz Baader, Stefan Borgwardt, and Marcel Lippmann. "Temporalizing Ontology-based Data Access". In: *Proc. of the 24th Int. Conf. on Automated Deduction (CADE)*. Vol. 7898. Lecture Notes in Computer Science. Springer, 2013, pp. 330–344. DOI: 10.1007/978-3-642-38574-2_23.

[15] Szymon Klarman and Thomas Meyer. "Querying Temporal Databases via OWL 2 QL". In: *Proc. of the 8th Int. Conf. on Web Reasoning and Rule Systems (RR)*. Vol. 8741. Lecture Notes in Computer Science. Springer, 2014, pp. 92–107. DOI: 10.1007/978-3-319-11113-1_7.

unibz

References IV

[16] Özgür Lütfü Özçep and Ralf Möller. "Ontology Based Data Access on Temporal and Streaming Data". In: *Reasoning Web: Reasoning on the Web in the Big Data Era – 10th Int. Summer School Tutorial Lectures (RW)*. Vol. 8714. Lecture Notes in Computer Science. Springer, 2014, pp. 279–312.

[17] Evgeny Kharlamov et al. "Ontology-Based Integration of Streaming and Static Relational Data with Optique". In: *Proc. of the 37th ACM Int. Conf. on Management of Data (SIGMOD)*. 2016, pp. 2109–2112. DOI: 10.1145/2882903.2899385.

[18] Alessandro Artale, Roman Kontchakov, Frank Wolter, and Michael Zakharyaschev. "Temporal Description Logic for Ontology-Based Data Access". In: *Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI)*. AAAI Press, 2013, pp. 711–717.

[19] Alessandro Artale, Roman Kontchakov, Alisa Kovtunova, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. "First-order Rewritability of Temporal Ontology-mediated Queries". In: *Proc. of the 24th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. AAAI Press, 2015, pp. 2706–2712.

References V

[20] Sebastian Brandt, Elem Güzel Kalayci, Roman Kontchakov, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyaschev. "Ontology-Based Data Access with a Horn Fragment of Metric Temporal Logic". In: *Proc. of the 31st AAAI Conf. on Artificial Intelligence (AAAI)*. AAAI Press, 2017, pp. 1070–1076.

[21] Maurizio Lenzerini, Lorenzo Lepore, and Antonella Poggi. "A Higher-Order Semantics for Metaquerying in OWL 2 QL". In: *Proc. of the 15th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*. AAAI Press, 2016, pp. 577–580.

[22] Diego Calvanese, Martin Giese, Dag Hovland, and Martin Rezk. "Ontology-based Integration of Cross-linked Datasets". In: *Proc. of the 14th Int. Semantic Web Conf. (ISWC)*. Vol. 9366. Lecture Notes in Computer Science. Springer, 2015, pp. 199–216. DOI: 10.1007/978-3-319-25007-6_12.

[23] Guohui Xiao, Dag Hovland, Dimitris Bilidas, Martin Rezk, Martin Giese, and Diego C. "Efficient Ontology-Based Data Integration with Canonical IRIs". In: *Proc. of the 15th Extended Semantic Web Conf. (ESWC)*. Vol. 10843. Lecture Notes in Computer Science. Springer, 2018, pp. 697–713.

unibz