Novel Developments in Ontology-Based Data Access and Integration: Part 3. Extensions

Diego Calvanese, Guohui Xiao

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

17th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2018)
Trento, Italy, 20 November 2018

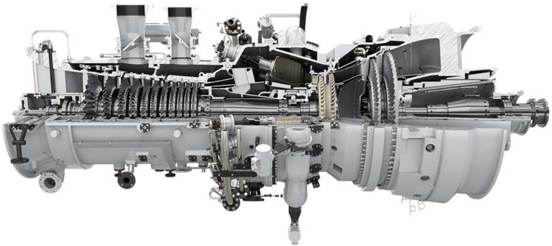
Outline

unibz

Outline

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 present here 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., Burner(`b01`), isMonitoredBy(`b01`, `mf01`)
  - Temporal predicates to represent temporal facts with a validity interval
    E.g., HighRotorSpeed(`rs01`)@[`2017-06-06 12:22:50`, `2017-06-06 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.

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**.

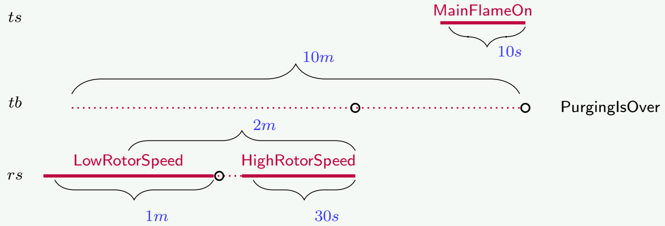
Example: turbine parts monitored by different co-located sensors (e.g., temperature, rotation speed)

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

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:

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

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).

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.

Burner(b01), TemperatureSensor(mf01),
isMonitoredBy(pt01, rs01), isMonitoredBy(b01, mf01).

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

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

- 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
```

$\rightsquigarrow$ 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

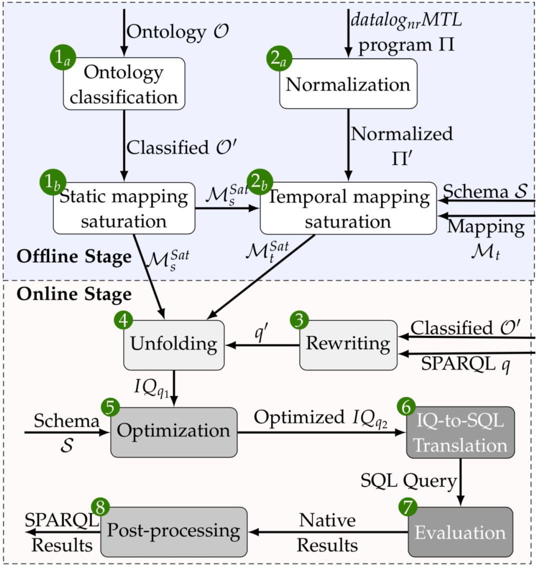
Temporal OBDA specification $\mathcal{P}_t = \langle \Sigma_s, \Sigma_t, \mathcal{O}, \mathcal{R}_s, \mathcal{R}_t, \mathcal{M}_s, \mathcal{M}_t, \mathcal{S} \rangle$

- 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	$\Sigma_s \cup \Sigma_t$	$\mathit{Datalog_{nr}MTL}$
$\mathcal{M}_s$	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.

Outline

1 Temporal Data

2 Ontology-based Integration of Multiple Data Sources
- Issues with Multiple Data Sources
- Canonical IRIs
- Mapping Rewriting
- Experimentation with *Ontop*

3 Conclusions

unibz

Issues when integrating multiple data sources

- Heterogeneity of data sources and data models
  $\leadsto$ Handled through a federation layer, such as Teeid, Denodo, or Exareme.

- Semantic heterogeneity
  $\leadsto$ 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.
  $\leadsto$ **This is what I want to discuss now.**

unibz

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 $\rightsquigarrow$ issues with freshness, privacy, legal aspects.

$\rightsquigarrow$ 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

$\rightsquigarrow$ 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

$\mathtt{canIriOf}\ (\mathtt{WB/2}, \mathtt{NatWB/2\text{-}1})$ $\mathtt{canIriOf}\ (\mathtt{WB/2}, \mathtt{CorpWB/NO\text{-}2\text{-}1})$

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

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

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

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:

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

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 $\mathbf{iri}(\vec{a})$ in ma, if $\mathcal{M}^{can}$ contains a mapping assertion

   $$\mathit{sql}(\vec{b}_0, \vec{b}_1) \rightsquigarrow \texttt{canIriOf}(\mathbf{iri}_c(\vec{b}_0),\ \mathbf{iri}(\vec{b}_1))$$

   then replace $\mathbf{iri}(\vec{a})$ in the target of ma by $\mathbf{iri}_c(\vec{b}_0)$, and

   join the source query of ma with $\mathit{sql}(\vec{b}_0, \vec{b}_1), \vec{a} = \vec{b}_1$.
2. Process IRIs directly occurring in ma in the same way.

Mapping rewriting – Example

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

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

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

1. `SELECT id, natName FROM central.masterTable`
   $\leadsto$ `canIriOf(iri("WB/",id), iri("NatWB/", natName))`
2. `SELECT id, corpName FROM central.masterTable`
   $\leadsto$ `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`
   $\leadsto$ `inField(iri("WB/",id), wlbField),  purpose(iri("WB/",id), opPurp)`
2. `SELECT driStDt, reason, id FROM corp.drillingops, central.masterTable WHERE name = corpName`
   $\leadsto$ `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.

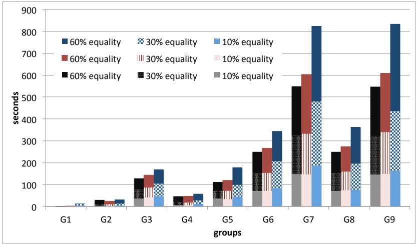
Results:

	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

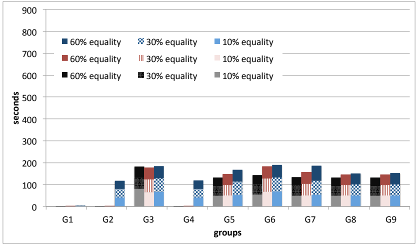
(limit = 100K tuples, timeout = 20 minutes)

Results over benchmark data – Execution times of most expensive queries

2 datasets:

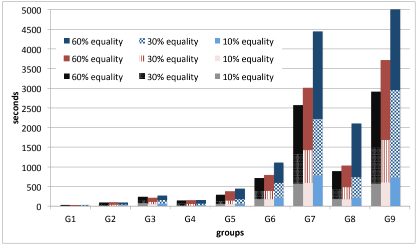


Standard `owl:sameAs`

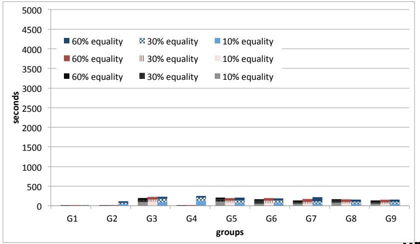


Canonical IRI

3 datasets:



Standard `owl:sameAs`



Canonical IRI

unibz

Outline

1. Temporal Data
2. Ontology-based Integration of Multiple Data Sources
3. Conclusions

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.

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

References I

[1] 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.

[2] 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.

[3] 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.

[4] 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.

unibz

References II

[5] 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.

[6] 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.

[7] 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.

[8] 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.

unibz

References III

[9] 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.

[10] 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.

[11] 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.