Novel Developments in Ontology-Based Data Access and Integration: Part 1. Introduction to Semantic Technologies and Data Access

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 of the tutorial

Part 1: Introduction to ontology-based data access and integration (Diego, 60 mins)

1. Motivations
2. Structure of OBDA systems
3. Relevant Semantic Web Technologies

Part 2: Systems and use cases (Guohui, 60 mins)

1. OBDA/I Systems
2. Use-cases for OBDA and OBDI
3. Demo of the OBDA system *Ontop*

Part 3: Latest advancements in OBDA and OBDI (Diego, 45 mins)

1. Temporal OBDA
2. Accessing cross-linked data sources

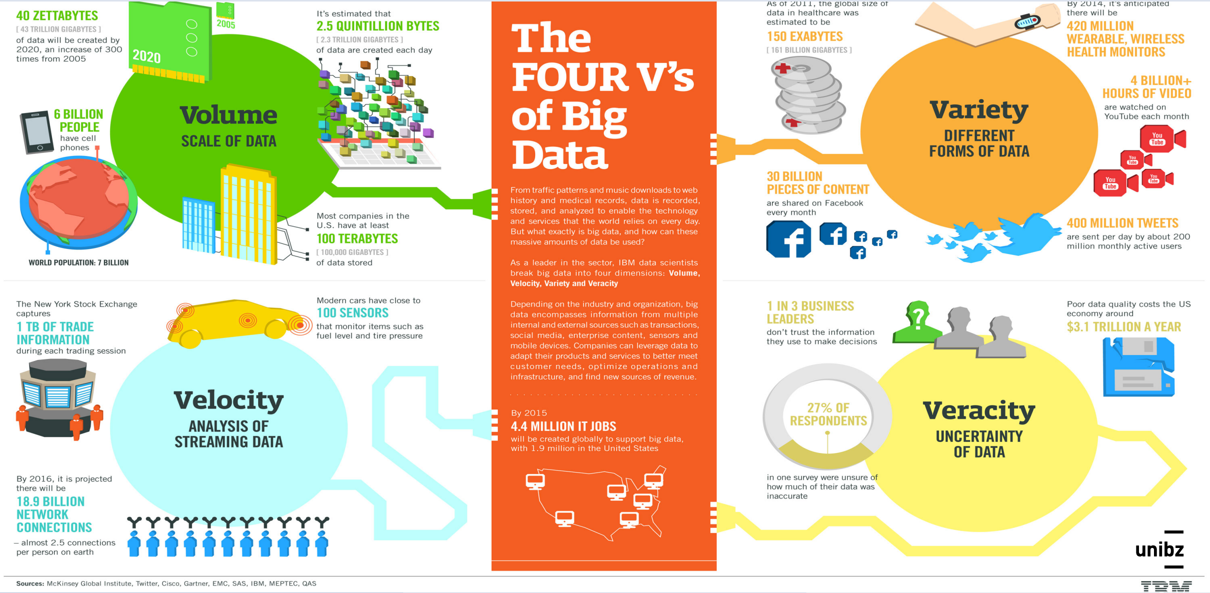
Part 4: Demo of OBDI (Guohui, 30 mins)

unibz

Typical view of Big Data

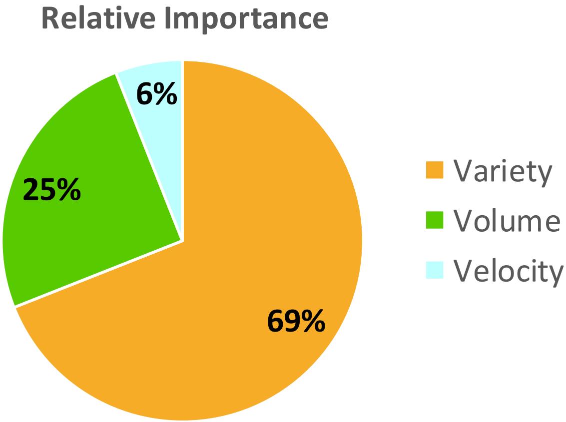
In fact, data has a lot of structure

Challenges in the Big Data era



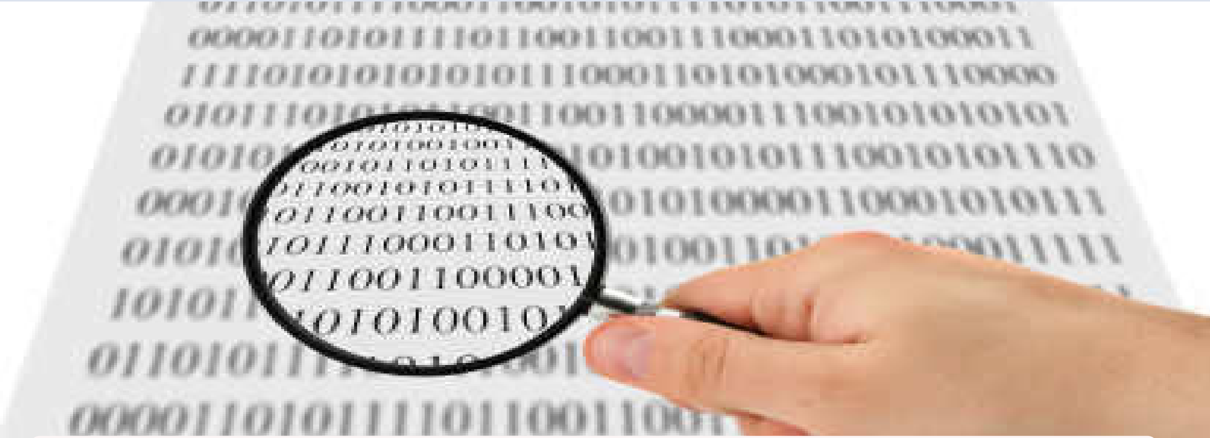
Variety, not volume, is driving Big Data initiatives

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:

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

unibz

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
        LEF
          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'
```

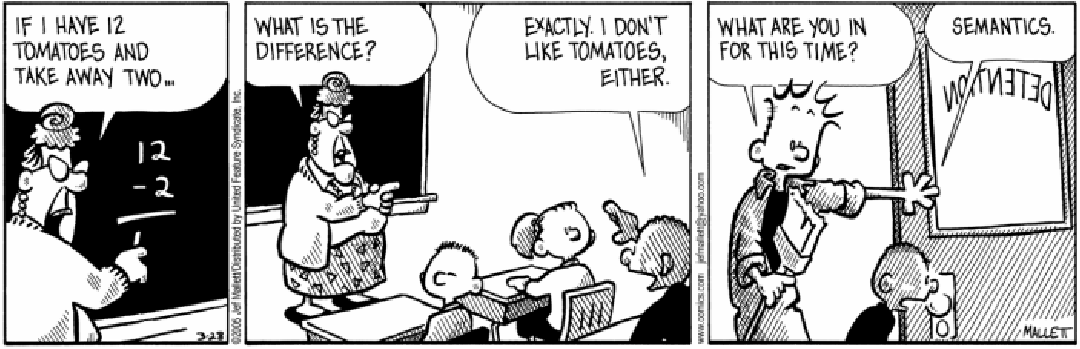
This can be very time consuming, and requires knowledge of the domain of interest, a deep understanding of the database structure, and general IT expertise.

This is also very costly!

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

Idea: 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.

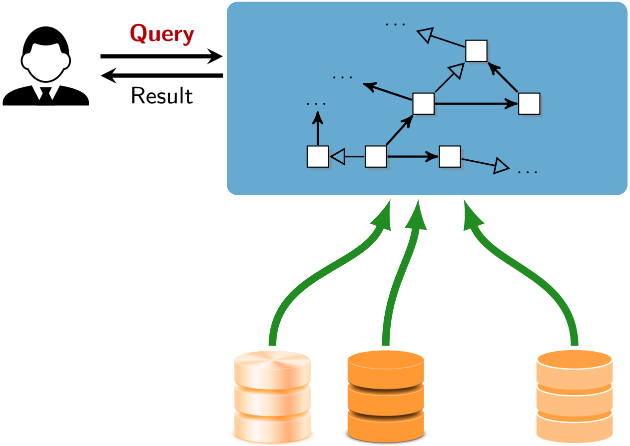
unibz

Outline

Outline

unibz

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

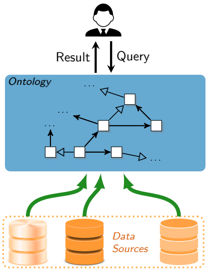
Data Sources $\mathcal{S}$
autonomous and heterogeneous

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

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



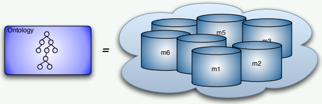
We are in a setting of **incomplete information**!!!

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:

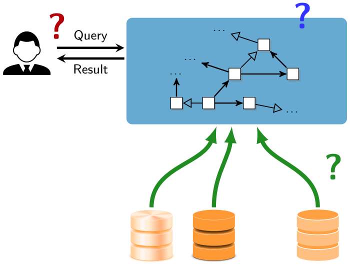
Query answering amounts to **logical inference**, and hence is significantly more challenging.

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

1. Motivation
2. Ontology-based Data Access
3. Representing Data in RDF and RDFS
4. OBDA Framework
5. Query Answering in OBDA

Resource Description Framework (RDF)

RDF provides a description of the domain of interest in terms of **triples**:

Subject | Predicate | Object

<http://slegger.gitlab.io/pf2/WB-16/1-29-S> <http://slegger.gitlab.io/slegge-obda/ontology/subsurface-exploration#name> "16/1-29-S"^^xsd:string

Triple elements: resources denoted by global identifiers (IRIs)

1. Subject: IRI of the described resource
2. Predicate: IRI of the property
3. Object: attribute value or IRI of another resource

Prefixes: useful abbreviations and/or references to external information

```
@prefix expl: <http://slegger.gitlab.io/slegge-obda/ontology/subsurface-exploration#>
@prefix : <http://slegger.gitlab.io/data#>
@base <http://slegger.gitlab.io/>
```

Resource Description Framework (RDF)

RDF provides a description of the domain of interest in terms of **triples**:

Subject	Predicate	Object
<pf2/WB-16/1-29-S>	expl:name	"16/1-29-S"^^xsd:string

Triple elements: resources denoted by **global identifiers** (IRIs)

1. Subject: IRI of the described resource
2. Predicate: IRI of the property
3. Object: attribute value or IRI of another resource

Prefixes: useful abbreviations and/or references to external information

```
@prefix expl: <http://slegger.gitlab.io/slegge-obda/ontology/subsurface-exploration#>
@prefix : <http://slegger.gitlab.io/data#>
@base <http://slegger.gitlab.io/>
```

RDF – Examples

We assume `@prefix : <http://slegger.gitlab.io/slegge-obda/ontology/subsurface-exploration#>`.

Class membership:

Fact	:Wellbore(*pf2/WB-16/1-29-S*)
RDF triple	`<pf2/WB-16/1-29-S> a :Wellbore`

Note: This is an abbreviation for

RDF triple	`<pf2/WB-16/1-29-S> rdf:type :Wellbore`

Attribute of an individual:

Fact	:name(*pf2/WB-16/1-29-S*, "16/1-29-S")
RDF triple	`<pf2/WB-16/1-29-S> :name "16/1-29-S"`

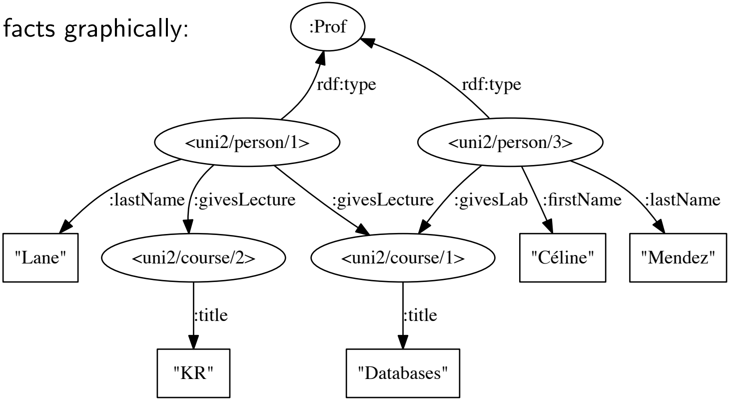
Property of an individual:

Fact	:hasFormationPressure(*pf2/WB-16/1-29-S*, *FP-1249*)
RDF triple	`<pf2/WB-16/1-29-S> :hasFormationPressure <FP-1249>`

RDF graph – Example

```
<uni2/person/1> rdf:type :Prof
<uni2/person/1> foaf:lastName "Lane"
<uni2/person/1> :givesLecture <uni2/course/1>
...
```

We can represent such a set of facts graphically:



Additional RDF features

RDF has additional features that we do not cover here:

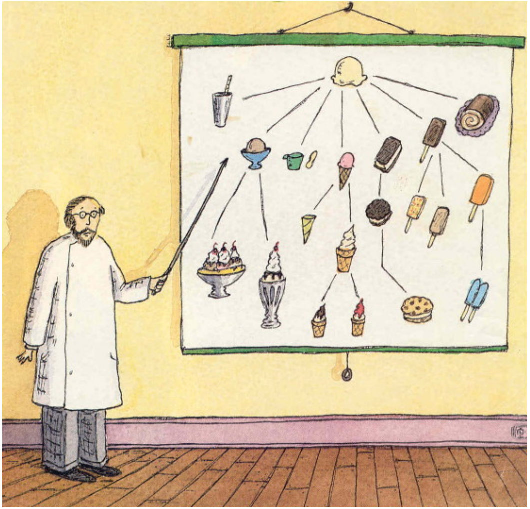
- blank nodes
- named graphs

Outline

Outline

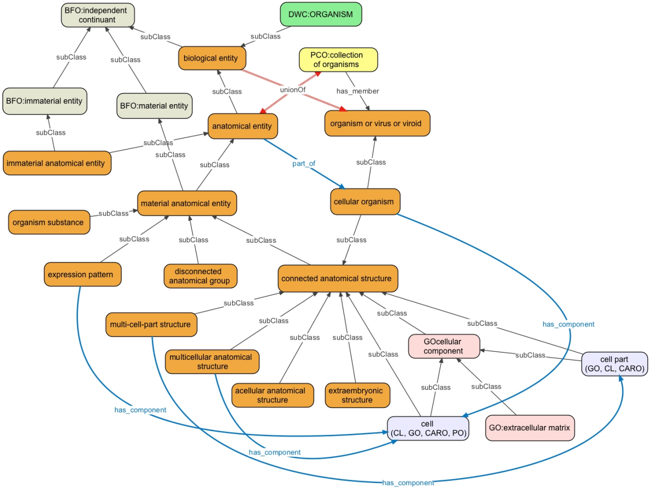
What is an ontology?

- An ontology conceptualizes a domain of interest in terms of **concepts/classes**, (binary) **relations**, and their **properties**.
- It typically organizes the concepts 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 **concepts/classes**, (binary) **relations**, and their **properties**.
- It typically organizes the concepts 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 **concepts/classes**, (binary) **relations**, and their **properties**.
- It typically organizes the concepts 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**.

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

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

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

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

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

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

What is an ontology?

- An ontology conceptualizes a domain of interest in terms of **concepts/classes**, (binary) **relations**, and their **properties**.
- It typically organizes the concepts 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**.

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

The OWL 2 QL ontology language

- **OWL 2 QL** is one of the three standard profiles of OWL 2. [W3C Rec. 2012]
- Derived from the *DL-Lite*$_{\mathcal{R}}$ description logic [Baader et al. 2003] of the *DL-Lite*-family:
  - Groups the domain into classes of objects with common properties.
  - Binary relations between objects are called object properties.
  - Binary relations from objects to values are called data properties.
- Is considered a lightweight ontology language:
  - controlled expressive power
  - efficient inference
- Optimized for accessing large amounts of data (i.e., for data complexity):
  - First-order rewritability of query answering: queries over the ontology can be rewritten into SQL queries over the underlying relational database.
  - Consistency checking is also first-order rewritable.

unibz

OWL 2 QL ontologies

- An OWL 2 QL ontology $\langle \mathcal{T}, \mathcal{A} \rangle$ is constituted by:
  - a TBox $\mathcal{T}$, modeling the intensional level information (i.e., axioms), and
  - an ABox $\mathcal{A}$, modeling the extensional level information (i.e., facts).
- In the OBDA setting, the ABox is (usually) implicitly defined through the database and the mappings.
- Therefore, in the following, we use the term "ontology" to refer to the TBox only.

Constructs of OWL 2 QL / *DL-Lite*$_{\mathcal{R}}$

- Class hierarchies: `rdfs:subClassOf`
- Property hierarchies: `rdfs:subPropertyOf`
- Property domain: `rdfs:domain`
- Property range: `rdfs:range`
- Inverse properties: `owl:inverseOf`
- Class disjointness: `owl:disjointWith`
- Mandatory participation: `owl:someValuesFrom` in superclass expression

RDF Schema (RDFS)

Class hierarchy: rdfs:subClassOf $(A_1 \sqsubseteq A_2)$

Example:

```
:FormationPressure rdfs:subClassOf :Pressure .
<FP-1249> a :FormationPressure .
```

$\Longrightarrow$ `<FP-1249> a :Pressure .`

Property hierarchy: rdfs:subPropertyOf $(P_1 \sqsubseteq P_2)$

Example:

```
:hasFormationPressure rdfs:subPropertyOf :hasMeasurement .
<pf2/WB-16/1-29-S> :hasFormationPressure <FP-1249> .
```

$\Longrightarrow$ `<pf2/WB-16/1-29-S> :hasMeasurement <FP-1249> .`

Domain of properties: rdfs:domain $(\exists P \sqsubseteq A)$

Example:

```
:hasFormationPressure rdfs:domain :Wellbore .
<pf2/WB-16/1-29-S> :hasFormationPressure <FP-1249> .
```

$\Longrightarrow$ `<pf2/WB-16/1-29-S> a :Wellbore .`

Range of properties: rdfs:range $(\exists P^- \sqsubseteq A)$

Example:

```
:hasFormationPressure rdfs:range :FormationPressure .
<pf2/WB-16/1-29-S> :hasFormationPressure <FP-1249> .
```

$\Longrightarrow$ `<FP-1249> a :FormationPressure .`

Other constructs of OWL 2 QL I

Inverse properties: owl:inverseOf $(P_1 \sqsubseteq P_2^- \text{ and } P_2 \sqsubseteq P_1^-)$

Example:

```
:isMeasureOf owl:inverseOf :hasMeasurement .
<pf2/WB-16/1-29-S> :hasMeasurement <FP-1249> .
```

```
⟹   <FP-1249> :isMeasureOf <pf2/WB-16/1-29-S> .
```

Mandatory participation: owl:someValuesFrom in the superclass expression $(A_1 \sqsubseteq \exists R A_2)$

Example:

```
:FormationPressure rdfs:subClassOf
   [ a owl:Restriction ;
     owl:onProperty :hadDepth ;
     owl:someValuesFrom :Depth ] .
<FP-1249> a :FormationPressure .
```

```
⟹   <FP-1249> a
       [ a owl:Restriction ;
         owl:onProperty :hasDepth ;
         owl:someValuesFrom :Depth ] .
```

Other constructs of OWL 2 QL II

Class disjointness: `owl:disjointWith` $(A_1 \sqsubseteq \neg A_2)$

Example:

```
:Wellbore owl:disjointWith :Measurement .
<pf2/WB-16/1-29-S> a :Wellbore .
<pf2/WB-16/1-29-S> a :Measurement .
```

$\Longrightarrow$ Inconsistent RDF graph

Semantics of an OWL 2 QL ontology

The **formal semantics** of OWL 2 QL is given in terms of first-order interpretations.

An **interpretation** $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of:

- a nonempty set $\Delta^\mathcal{I}$, called the interpretation domain (of $\mathcal{I}$), and
- an interpretation function $\cdot^\mathcal{I}$, which maps
  - each class nane A to a subset $A^\mathcal{I}$ of $\Delta^\mathcal{I}$
  - each property name P to a subset $P^\mathcal{I}$ of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$
- The interpretation function is then extended to cover the OWL 2 QL constructs:
  $(P^-)^\mathcal{I} = \{(y,x) \mid (x,y) \in P^\mathcal{I}\}$ $\exists P^\mathcal{I} = \{x \mid \text{there is some } y \text{ such that } (x,y) \in P^\mathcal{I}\}$

The semantics of an ontology is given by specifying when $\mathcal{I}$ **satisfies** an assertion α, denoted $\mathcal{I} \models \alpha$:

$\mathcal{I} \models C_1 \sqsubseteq C_2$ if $C_1^\mathcal{I} \subseteq C_2^\mathcal{I}$; $\mathcal{I} \models R_1 \sqsubseteq R_2$ if $R_1^\mathcal{I} \subseteq R_2^\mathcal{I}$;

$\mathcal{I}$ satisfies an ABox fact, if the fact holds in $\mathcal{I}$.

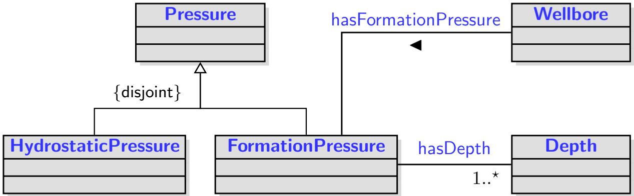
An interpretation that satisfies all assertions of the ontology, is called a **model** of the ontology.

Representing OWL 2 QL ontologies as UML class diagrams/ER schemas

There is a close correspondence between OWL 2 QL and conceptual modeling formalisms [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].

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

An OWL 2 QL ontology can be visualized naturally as a UML class diagram or as an ER schema.



Capturing UML class diagrams/ER schemas in OWL 2 QL

Modeling construct	*DL-Lite*	**FOL formalization**
ISA on classes	$A_1 \sqsubseteq A_2$	$\forall x(A_1(x) \rightarrow A_2(x))$
... and on relations	$R_1 \sqsubseteq R_2$	$\forall x, y(R_1(x, y) \rightarrow R_2(x, y))$
Disjointness of classes	$A_1 \sqsubseteq \neg A_2$	$\forall x(A_1(x) \rightarrow \neg A_2(x))$
... and of relations	$R_1 \sqsubseteq \neg R_2$	$\forall x, y(R_1(x, y) \rightarrow \neg R_2(x, y))$
Domain of relations	$\exists P \sqsubseteq A_1$	$\forall x(\exists y(P(x, y)) \rightarrow A_1(x))$
Range of relations	$\exists P^- \sqsubseteq A_2$	$\forall x(\exists y(P(y, x)) \rightarrow A_2(x))$
Mandatory participation (*min card = 1*)	$A_1 \sqsubseteq \exists P$ $A_2 \sqsubseteq \exists P^-$	$\forall x(A_1(x) \rightarrow \exists y(P(x, y)))$ $\forall x(A_2(x) \rightarrow \exists y(P(y, x)))$

OWL 2 QL/ *DL-Lite*$_{\mathcal{R}}$ **cannot capture**:

- covering constraints – This would require **disjunction**.
- identity between individuals – This would `owl:sameAs`.
- functionality of roles – This would require number restrictions.

Outline

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).

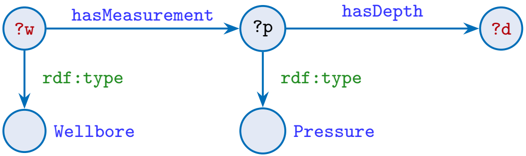
Conjunctive queries – Are concretely represented in **SPARQL**

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)
- ...

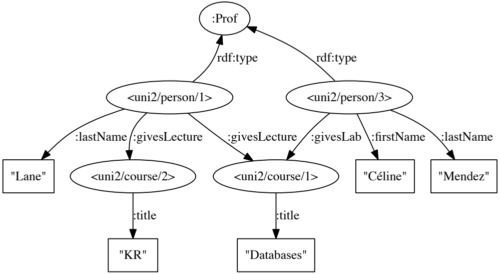
SPARQL Basic Graph Patterns

Basic Graph Pattern (BGP) are the simplest form of SPARQL query, asking for a pattern in the RDF graph.

Example: BGP

```
SELECT ?p ?ln ?c ?t
WHERE {
  ?p :lastName ?ln .
  ?p :givesLecture ?c .
  ?c :title ?t .
}
```

When evaluated over the RDF graph



. . . the query returns:

p	ln	c	t
<uni2/person/1>	"Lane"	<uni2/course/1>	"Databases"
<uni2/person/1>	"Lane"	<uni2/course/2>	"KR"

unibz

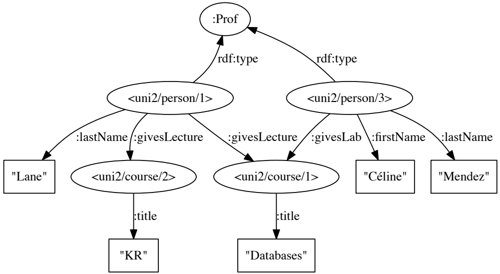
Projecting out variables in a SPARQL query

A query may also return only a subset of the variables used in the BGP.

Example: BGP with projection

```
SELECT ?ln ?t
WHERE {
  ?p :lastName ?ln .
  ?p :givesLecture ?c .
  ?c :title ?t .
}
```

When evaluated over the RDF graph



. . . the query returns:

ln	t
"Lane"	"Databases"
"Lane"	"KR"

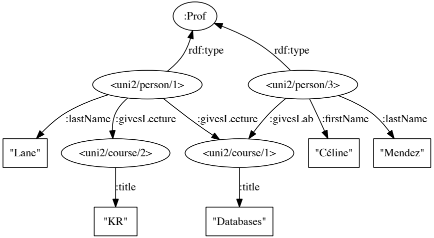
unibz

Union of Basic Graph Patterns

Example: BGPs with UNION

```
SELECT ?p ?ln ?c
WHERE {
  { ?p :lastName ?ln .    ?p :givesLecture ?c . }
  UNION
  { ?p :lastName ?ln .    ?p :givesLab ?c . }
}
```

When evaluated over



. . . the query returns:

p	ln	c
<uni2/person/1>	"Lane"	<uni2/course/1>
<uni2/person/1>	"Lane"	<uni2/course/2>
<uni2/person/3>	"Mendez"	<uni2/course/1>

BGPs vs. conjunctive queries

We can write queries using a more compact and abstract syntax, borrowed from database theory.

Example: BGP

```
SELECT ?p ?ln ?c ?t
WHERE {
  ?p :lastName ?ln .
  ?p :givesLecture ?c .
  ?c :title ?t .
}
```

vs. conjunctive query

$$\boldsymbol{q}(p, ln, c, t) \leftarrow \mathsf{lastName}(p, ln), \mathsf{givesLecture}(p, c), \mathsf{title}(c, t)$$

A **conjunctive query** $\boldsymbol{q}$ has the form $\boldsymbol{q}(\vec{x}) \leftarrow p_1(\vec{y}_1), \ldots, p(\vec{y}_k)$ where

- $\boldsymbol{q}(\vec{x})$ is called the head of $\boldsymbol{q}$,
- $p_1(\vec{y}_1), \ldots, p(\vec{y}_k)$ is a conjunction of atoms called the body of $\boldsymbol{q}$,
- all variables $\vec{x}$ in the head are among $\vec{y}_1, \ldots, \vec{y}_k$, and
- the variables in $\vec{y}_1, \ldots, \vec{y}_k$ that are not among $\vec{x}$ are existentially quantified.

BGPs vs. conjunctive queries (cont.)

Example: BGP with projection

```
SELECT ?ln ?t
WHERE {
  ?p :lastName ?ln .
  ?p :givesLecture ?c .
  ?c :title ?t .
}
```

vs. conjunctive query

$$\boldsymbol{q}(ln, t) \leftarrow \text{lastName}(p, ln), \text{givesLecture}(p, c), \text{title}(c, t)$$

But there is a difference in semantics when we have an ontology:

- In a SPARQL query, all variables, including those that are projected out, must match nodes of the RDF graph.
- In a conjunctive query, the existentially quantified variables can also match nodes that are existentially implied by the axioms of the ontology.

SPARQL UNION vs. unions of CQs

Example: BGP with UNION

```
SELECT ?p ?ln ?c
WHERE {
  { ?p :lastName ?ln .
    ?p :givesLecture ?c .
  }
  UNION
  { ?p :lastName ?ln .
    ?p :givesLab ?c .
  }
}
```

vs. union of CQs (UCQ)

$$\boldsymbol{q}(p, ln, c) \leftarrow \mathsf{lastName}(p, ln), \mathsf{givesLecture}(p, c)$$

$$\boldsymbol{q}(p, ln, c) \leftarrow \mathsf{lastName}(p, ln), \mathsf{givesLab}(p, c)$$

A UCQ is written as a set of CQs, all with the same head.

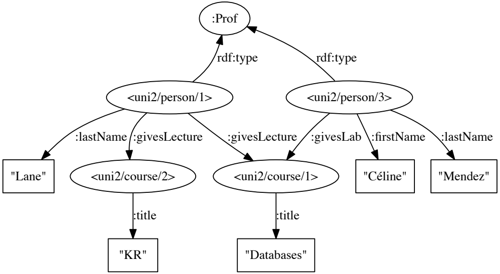
Extending BGPs with OPTIONAL

We might want to add information when available, but **not reject** a solution **when some part of the query does not match**.

Example: BGP with OPTIONAL

```
SELECT ?p ?fn ?ln
WHERE {
  ?p :lastName ?ln .
  OPTIONAL {
    ?p :firstName ?fn .
  }
}
```

When evaluated over the RDF graph



. . . the query returns:

p	fn	ln
<uml2/person/1>		"Lane"
<uml2/person/3>	"Céline"	"Mendez"

SPARQL algebra

We have seen the following features of the SPARQL algebra:

- Basic Graph Patterns
- `UNION`
- `OPTIONAL`

The overall algebra has additional features:

- more complex `FILTER` conditions
- `GROUP BY`, to express aggregations and support aggregation operators
- `MINUS`, to remove possible solutions
- `FILTER NOT EXISTS`, to test for the absence of a pattern

Outline

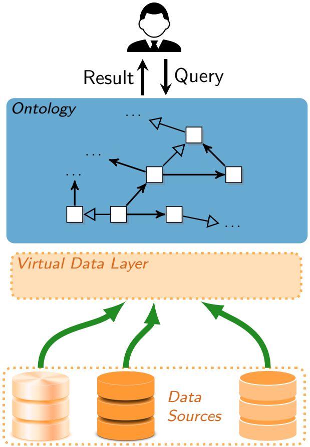
unibz

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.



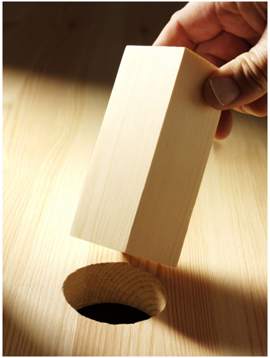
unibz

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.

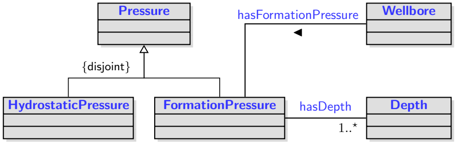
To address the impedance mismatch

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**.

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

Outline

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 $\mathcal{J}$.

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 $\mathcal{D}$,

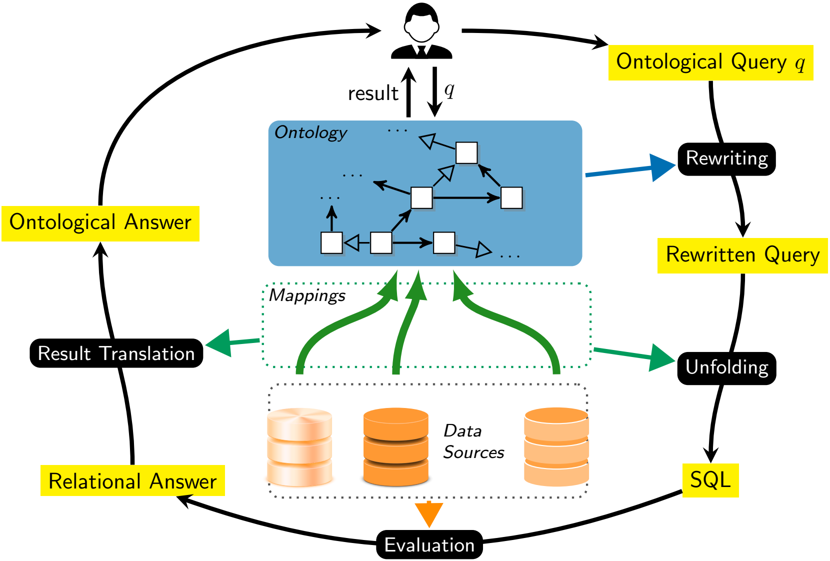
certain answers to $q(\vec{x})$ over $\langle \mathcal{P}, \mathcal{D} \rangle$ $=$ answers to $r(\vec{x})$ over $\mathcal{D}$.

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



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.

unibz

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.

unibz