Ontology Based Data Access

Theory and Practice

Guohui Xiao

Faculty of Computer Science
Free University of Bozen-Bolzano, Italy

Summer School of the 8th Chinese Semantic Web & Web Science Conference (CSWS 2014)
10 August 2014, Wuhan, China

unibz
Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

Optique™

Outline

unibz

Outline

unibz

Example 1: Statoil Exploration

Experts in geology and geophysics develop stratigraphic models of unexplored areas on the basis of data acquired from previous operations at nearby geographical locations.

Facts:

- 1,000 TB of relational data
- using diverse schemata
- spread over 2,000 tables, over multiple individual data bases

Data Access for Exploration:

- 900 experts in Statoil Exploration.
- up to 4 days for new data access queries, requiring assistance from IT-experts.
- 30–70% of time spent on data gathering.

unibz

Example 1: Statoil Exploration

Experts in geology and geophysics develop stratigraphic models of unexplored areas on the basis of data acquired from previous operations at nearby geographical locations.

Facts:

- 1,000 TB of relational data
- using diverse schemata
- spread over 2,000 tables, over multiple individual data bases

Data Access for Exploration:

- 900 experts in Statoil Exploration.
- up to 4 days for new data access queries, requiring assistance from IT-experts.
- 30–70% of time spent on data gathering.

unibz

Example 2: Siemens Energy Services

Runs service centers for power plants, each responsible for remote monitoring and diagnostics of many thousands of gas/steam turbines and associated components. When informed about potential problems, diagnosis engineers access a variety of raw and processed data.

Facts:

- several TB of time-stamped sensor data
- several GB of event data ("alarm triggered at time T")
- data grows at 30GB per day (sensor data rate 1Hz–1kHz)

Service Requests:

- over 50 service centers worldwide
- 1,000 service requests per center per year
- 80% of time per request used on data gathering

unibz

Example 2: Siemens Energy Services

Runs service centers for power plants, each responsible for remote monitoring and diagnostics of many thousands of gas/steam turbines and associated components. When informed about potential problems, diagnosis engineers access a variety of raw and processed data.

Facts:

- several TB of time-stamped sensor data
- several GB of event data ("alarm triggered at time T")
- data grows at 30GB per day (sensor data rate 1Hz–1kHz)

Service Requests:

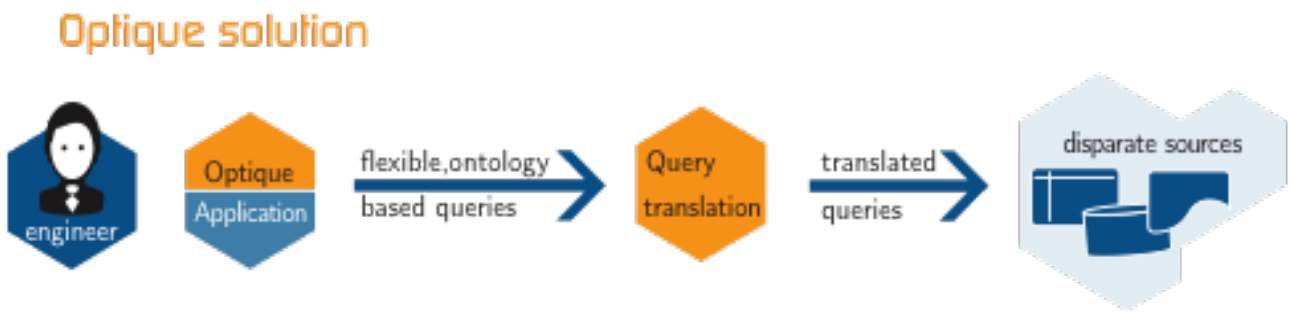
- over 50 service centers worldwide
- 1,000 service requests per center per year
- 80% of time per request used on data gathering

unibz

How to Find the Right Data?

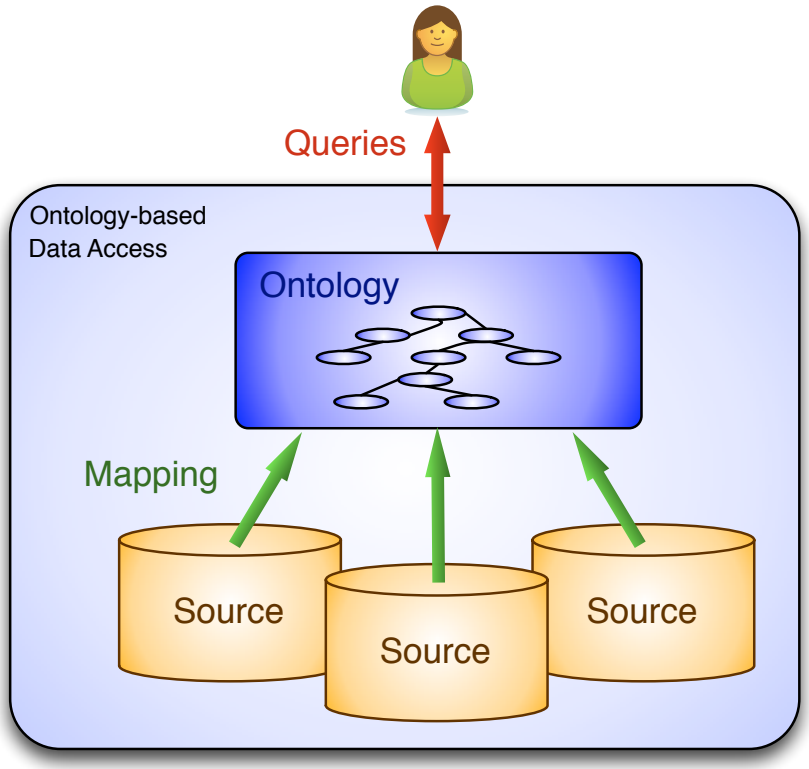
Challenges in complex cases

- You cannot be creative, novel, explorative
- Adding new data sources is painful



unibz

Ontology-based data access: Architecture



Architecture is based on three main components:

- Ontology: provides a unified, conceptual view of the managed information.
- Data source(s): are external and independent (possibly multiple and heterogeneous).
- Mappings: semantically link data at the sources with the ontology.

unibz

Ontology-based data access: Formalization

An **ontology-based access system** is a triple $\mathcal{O} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, where:

- $\mathcal{T}$ is the intensional level of an ontology.
  We consider ontologies formalized in description logics (DLs), hence $\mathcal{T}$ is a DL TBox.
- $\mathcal{S}$ is a (federated) relational database representing the sources;
- $\mathcal{M}$ is a set of mapping assertions, each one of the form

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

  where
  - $\Phi(\vec{x})$ is a FOL query over $\mathcal{S}$, returning tuples of values for $\vec{x}$
  - $\Psi(\vec{x})$ is a FOL query over $\mathcal{T}$ whose free variables are from $\vec{x}$.

Semantics of OBDA system:

We can assign to $\mathcal{O}$ a **first-order logic semantics**, based on interpretations of the TBox $\mathcal{T}$, taking into account data sources $\mathcal{S}$ and mapping $\mathcal{M}$.

Note: the semantics of mappings is captured through material implication, i.e., **data sources** are considered **sound**, but **not necessarily complete**.

Ontology-based data access: Formalization

An **ontology-based access system** is a triple $\mathcal{O} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, where:

- $\mathcal{T}$ is the intensional level of an ontology.
  We consider ontologies formalized in description logics (DLs), hence $\mathcal{T}$ is a DL TBox.
- $\mathcal{S}$ is a (federated) relational database representing the sources;
- $\mathcal{M}$ is a set of mapping assertions, each one of the form

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

  where
  - $\Phi(\vec{x})$ is a FOL query over $\mathcal{S}$, returning tuples of values for $\vec{x}$
  - $\Psi(\vec{x})$ is a FOL query over $\mathcal{T}$ whose free variables are from $\vec{x}$.

Semantics of OBDA system:

We can assign to $\mathcal{O}$ a **first-order logic semantics**, based on interpretations of the TBox $\mathcal{T}$, taking into account data sources $\mathcal{S}$ and mapping $\mathcal{M}$.

Note: the semantics of mappings is captured through material implication, i.e., **data sources** are considered **sound**, but **not necessarily complete**.

Challenges in OBDA

- How to instantiate the abstract framework?
- How to execute queries over the ontology by accessing data in the sources?
- How to deploy such systems using state-of-the-art technology?
- How to optimize the performance of the system?
- How to assess the quality of the constructed system?
- How to provide automated support for key tasks during design and deployment?
  - constructing the ontology;
  - constructing the mappings;
  - formulating queries;
  - characterizing the evolution of the system components;
  - verifying properties over the evolving system.

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 deploy such systems using state-of-the-art technology?
- How to optimize the performance of the system?
- How to assess the quality of the constructed system?
- How to provide automated support for key tasks during design and deployment?
  - constructing the ontology;
  - constructing the mappings;
  - formulating queries;
  - characterizing the evolution of the system components;
  - verifying properties over the evolving system.

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 deploy such systems using state-of-the-art technology?
- How to optimize the performance of the system?
- How to assess the quality of the constructed system?
- How to provide automated support for key tasks during design and deployment?
  - constructing the ontology;
  - constructing the mappings;
  - formulating queries;
  - characterizing the evolution of the system components;
  - verifying properties over the evolving system.

unibz

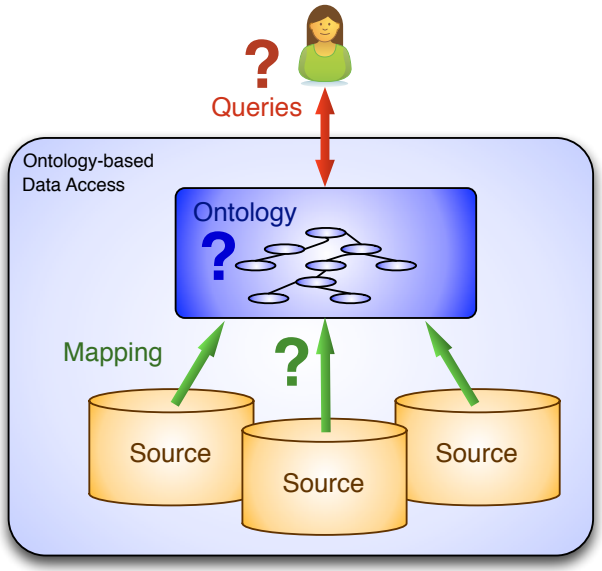
Challenges in OBDA

- How to instantiate the abstract framework?
- How to execute queries over the ontology by accessing data in the sources?
- How to deploy such systems using state-of-the-art technology?
- How to optimize the performance of the system?
- How to assess the quality of the constructed system?
- How to provide automated support for key tasks during design and deployment?
  - constructing the ontology;
  - constructing the mappings;
  - formulating queries;
  - characterizing the evolution of the system components;
  - verifying properties over the evolving system.

unibz

Instantiating the framework

1. Which is the "right" **ontology language**?
2. Which is the "right" **query language**?
3. Which is the "right" **mapping language**?



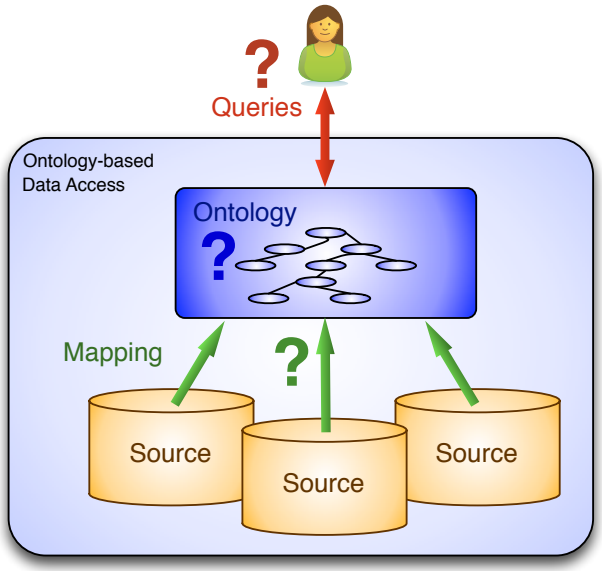
The choices that we make have to take into account the tradeoff between expressive power and efficiency of inference/query answering.

We are in a setting where we want to access large amounts of data, so **efficiency w.r.t. the data** plays an important role.

unibz

Instantiating the framework

1. Which is the "right" **ontology language**?
2. Which is the "right" **query language**?
3. Which is the "right" **mapping language**?



The choices that we make have to take into account the tradeoff between expressive power and efficiency of inference/query answering.

We are in a setting where we want to access large amounts of data, so **efficiency w.r.t. the data** plays an important role.

unibz

Outline

unibz

Use of mappings

In an OBDA system $\mathcal{O} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$, the **mapping** $\mathcal{M}$ is a crucial component:

- $\mathcal{M}$ encodes how the data in the external source(s) $\mathcal{S}$ should be used to populate the elements of $\mathcal{T}$.
- We should talk about **OBDA** only when we are in the presence of a system that includes external sources and mappings.

Virtual data layer

The data sources $\mathcal{S}$ and the mapping $\mathcal{M}$ define a virtual data layer $\mathcal{V} = \mathcal{M}(\mathcal{S})$
(i.e., a virtual ABox, in DL terminology).

- Queries are answered w.r.t. $\mathcal{T}$ and $\mathcal{V}$.
- We do not really materialize the data of $\mathcal{V}$ (that's why it is called virtual).
- Instead, the intensional information in $\mathcal{T}$ and $\mathcal{M}$ is used to translate queries over $\mathcal{T}$ into queries formulated over $\mathcal{S}$.

unibz

Use of mappings

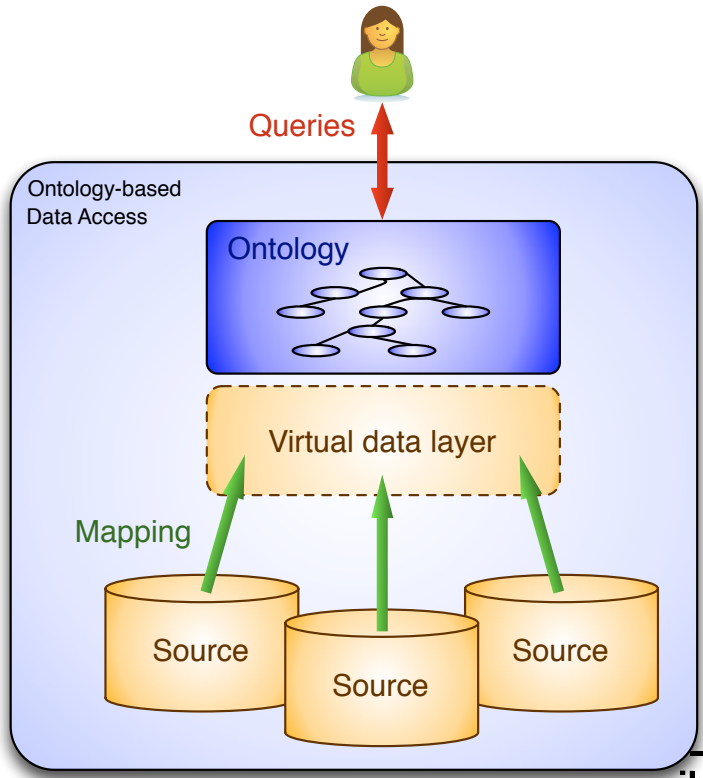
In an OBDA system $\mathcal{O} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$, the **mapping** $\mathcal{M}$ is a crucial component:

- $\mathcal{M}$ encodes how the data in the external source(s) $\mathcal{S}$ should be used to populate the elements of $\mathcal{T}$.
- We should talk about **OBDA** only when we are in the presence of a system that includes external sources and mappings.

Virtual data layer

The data sources $\mathcal{S}$ and the mapping $\mathcal{M}$ define a virtual data layer $\mathcal{V} = \mathcal{M}(\mathcal{S})$
(i.e., a virtual ABox, in DL terminology).

- Queries are answered w.r.t. $\mathcal{T}$ and $\mathcal{V}$.
- We do not really materialize the data of $\mathcal{V}$ (that's why it is called virtual).
- Instead, the intensional information in $\mathcal{T}$ and $\mathcal{M}$ is used to translate queries over $\mathcal{T}$ into queries formulated over $\mathcal{S}$.



unibz

The impedance mismatch problem

We need to address the **impedance mismatch** problem

- In relational databases, information is represented as tuples of values.
- In ontologies, information is represented using both objects and values ...
  - ... with objects playing the main role, ...
  - ... and values playing a subsidiary role as fillers of object attributes.

Proposed solution:

- Use **constructors to create objects** of the ontology from tuples of values in the DB.
- The constructors are modeled through Skolem functions in the query in the rhs of the mapping:

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

- Techniques from partial evaluation of logic programs are adapted for unfolding queries over $\mathcal{T}$, by using $\mathcal{M}$, into queries over $\mathcal{S}$.

unibz

The impedance mismatch problem

We need to address the **impedance mismatch** problem

- In relational databases, information is represented as tuples of values.
- In ontologies, information is represented using both objects and values ...
  - ... with objects playing the main role, ...
  - ... and values playing a subsidiary role as fillers of object attributes.

Proposed solution:

- Use **constructors to create objects** of the ontology from tuples of values in the DB.
- The constructors are modeled through Skolem functions in the query in the rhs of the mapping:

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

- Techniques from partial evaluation of logic programs are adapted for unfolding queries over $\mathcal{T}$, by using $\mathcal{M}$, into queries over $\mathcal{S}$.

unibz

RDB to RDF Mapping Languages

R2RML

- RDB to RDF Mapping Language
- W3C Recommendation 27 September 2012
- `http://www.w3.org/TR/r2rml/`
- R2RML mappings are themselves expressed as RDF graphs and written down in Turtle syntax .

Other Mapping languages

- D2RQ [a]
- SML [b]
- Ontop [c]

[a] `http://d2rq.org/d2rq-language`
[b] `http://sparqlify.org/wiki/Sparqlification_mapping_language`
[c] `https://github.com/ontop/ontop/wiki/ObdalibObdaTurtlesyntax`

unibz

RDB2RDF Example

EMP

EMPNO INTEGER PRIMARY KEY	ENAME VARCHAR(100)	JOB VARCHAR(20)	DEPTNO INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

DEPT

DEPTNO INTEGER PRIMARY KEY	DNAME VARCHAR(30)	LOC VARCHAR(100)
10	APPSERVER	NEW YORK

Example output data

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department <http://data.example.com/department/10>.

<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
<http://data.example.com/department/10> ex:staff 1.
```

Example from W3C R2RML specification `http://www.w3.org/TR/r2rml/`

unibz

R2RML

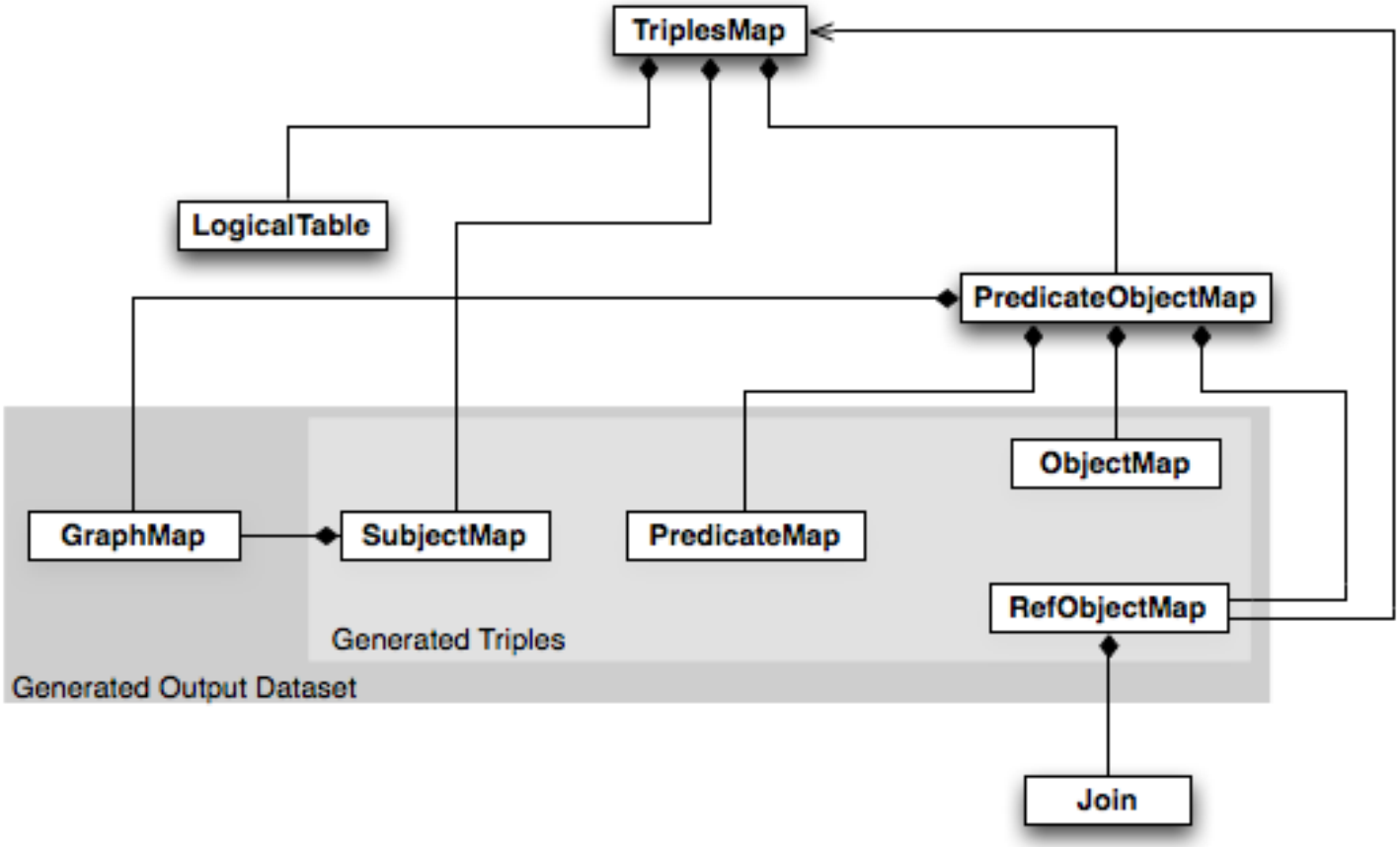


Figure: UML of R2RML Mapping Language

unibz

URI Templates in R2RML

String template

A **string template** is a format string that can be used to build strings from multiple components. It can reference column names by enclosing them in curly braces ("{" and "}").

Example

<http://www.Department{depid}.University{uniid}.edu/UndergraduateStudent{id}>

$\Rightarrow \{\texttt{depid} \rightarrow 12, \texttt{uniid} \rightarrow 54, \texttt{id} \rightarrow 22\}$

<http://www.Department12.University54.edu/GraduateStudent22>

unibz

URI Templates in R2RML

String template

A **string template** is a format string that can be used to build strings from multiple components. It can reference column names by enclosing them in curly braces ("{" and "}").

Example

<http://www.Department{depid}.University{uniid}.edu/UndergraduateStudent{id}>

$\Rightarrow \{\texttt{depid} \rightarrow 12, \texttt{uniid} \rightarrow 54, \texttt{id} \rightarrow 22\}$

<http://www.Department12.University54.edu/GraduateStudent22>

unibz

URI Templates in R2RML

String template

A **string template** is a format string that can be used to build strings from multiple components. It can reference column names by enclosing them in curly braces ("{" and "}").

Example

<http://www.Department{depid}.University{uniid}.edu/UndergraduateStudent{id}>

$\Rightarrow \{\texttt{depid} \rightarrow 12, \texttt{uniid} \rightarrow 54, \texttt{id} \rightarrow 22\}$

<http://www.Department12.University54.edu/GraduateStudent22>

unibz

URI Templates in R2RML

String template

A **string template** is a format string that can be used to build strings from multiple components. It can reference column names by enclosing them in curly braces ("{" and "}").

Example

<http://www.Department{depid}.University{uniid}.edu/UndergraduateStudent{id}>

$\Rightarrow \{\texttt{depid} \rightarrow 12, \texttt{uniid} \rightarrow 54, \texttt{id} \rightarrow 22\}$

<http://www.Department12.University54.edu/GraduateStudent22>

unibz

Example: Mapping a Simple Table

Output RDF

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
```

R2RML Mapping

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns#>.

<#TriplesMap1>
    rr:logicalTable [ rr:tableName "EMP" ];
    rr:subjectMap [
        rr:template "http://data.example.com/employee/{EMPNO}";
        rr:class ex:Employee;
    ];
    rr:predicateObjectMap [
        rr:predicate ex:name;
        rr:objectMap [ rr:column "ENAME" ];
    ].
```

Ontop Mapping

```
mappingId MAP-EMP
target :employee/{EMPNO} a ex:Employee ; ex:name {ENAME} .
source SELECT EMPNO, ENAME FROM EMP
```

Example: Computing a Property with an R2RML View

Output RDF

```
<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
<http://data.example.com/department/10> ex:staff 1.
```

Mapping in R2RML

```
<#DeptTableView> rr:sqlQuery """SELECT DEPTNO, DNAME, LOC,
       (SELECT COUNT(*) FROM EMP WHERE EMP.DEPTNO=DEPT.DEPTNO) AS STAFF FROM DEPT;""".
<#TriplesMap2> rr:logicalTable <#DeptTableView>;
    rr:subjectMap [ rr:template "http://data.example.com/department/{DEPTNO}";
                    rr:class ex:Department;    ];
    rr:predicateObjectMap [ rr:predicate ex:name;
                            rr:objectMap [ rr:column "DNAME" ];  ];
    rr:predicateObjectMap [ rr:predicate ex:location;
                            rr:objectMap [ rr:column "LOC" ];  ];
    rr:predicateObjectMap [ rr:predicate ex:staff;
                            rr:objectMap [ rr:column "STAFF" ]; ].
```

Mapping in Ontop format

```
mappingId MAP-DEPT
target :department/{DEPTNO} a ex:Department; ex:name {DNAME}; ex:location {LOC};
       ex:staff {STAFF}
source  SELECT DEPTNO, DNAME, LOC,
         (SELECT COUNT(*) FROM EMP WHERE EMP.DEPTNO=DEPT.DEPTNO) AS STAFF FROM DEPT;
```

Mapping in Datalog Style

- Syntax
  - Body: (Joins of) Database Tables or Views
  - Head: Triple Patterns
- more concise; used for research
- the internal representation of Mappings in the Ontop system

Example

$$\texttt{ex:Employee}(\mathit{URI}(\text{“:employee/\{}\text{”}, \texttt{EMPNO})) \leftarrow \texttt{EMP}(\texttt{EMPNO}, \texttt{ENAME}, \_, \_)$$

$$\texttt{ex:name}(\texttt{URI}(\text{“:employee/\{}\text{”}, \texttt{EMPNO}), \texttt{ENAME}) \leftarrow \texttt{EMP}(\texttt{EMPNO}, \texttt{ENAME}, \_, \_)$$

unibz

Outline

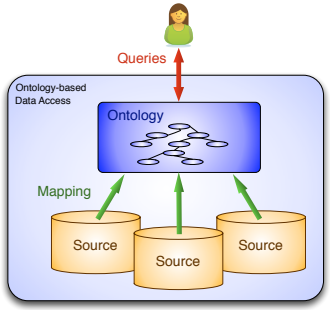
unibz

Incomplete information

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

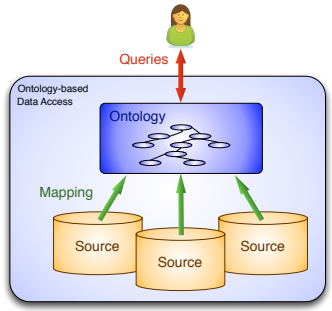
Incompleteness introduced:

- by data source(s), in general assumed to be incomplete;
- by domain constraints encoded in the ontology.



unibz

Incomplete information

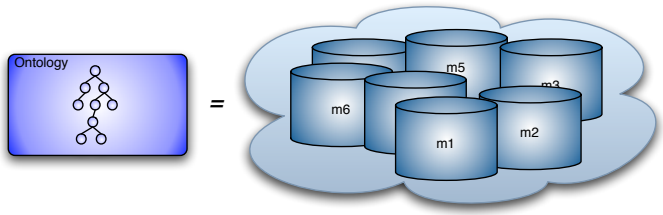


We are in a setting of **incomplete information**!!!

Incompleteness introduced:

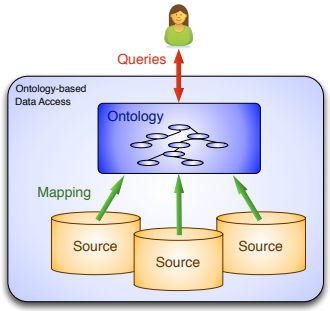
- by data source(s), in general assumed to be incomplete;
- by domain constraints encoded in the ontology.

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



unibz

Incomplete information

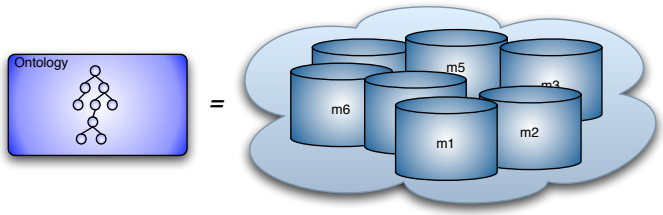


We are in a setting of **incomplete information**!!!

Incompleteness introduced:

- by data source(s), in general assumed to be incomplete;
- by domain constraints encoded in the ontology.

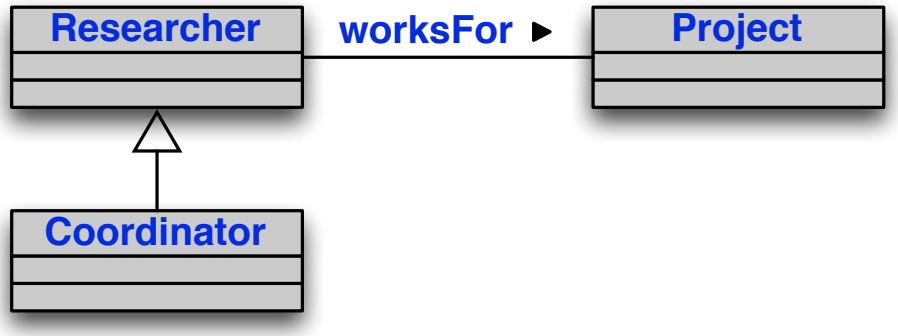
Plus: Ontologies are logical theories, and hence perfectly suited to deal with incomplete information!



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

unibz

Incomplete information – Example 1



We assume that each concept/relationship of the ontology is mapped directly to a database table.

But the database tables may be **incompletely specified**, or even missing for some concepts/relationships.

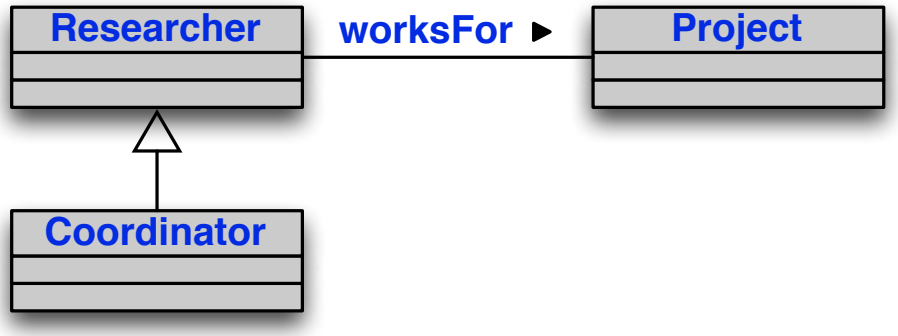
DB: Coordinator $\supseteq$ { `serge`, `marie` }
Project $\supseteq$ { `webdam`, `diadem` }
worksFor $\supseteq$ { (`serge`,`webdam`), (`georg`,`diadem`) }

Query: $q(x) \leftarrow \text{Researcher}(x)$

Answer: ???

unibz

Incomplete information – Example 1



We assume that each concept/relationship of the ontology is mapped directly to a database table.

But the database tables may be **incompletely specified**, or even missing for some concepts/relationships.

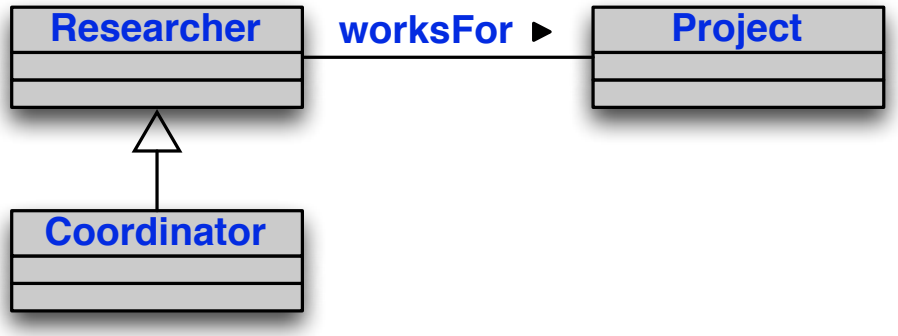
DB: Coordinator $\supseteq$ { `serge`, `marie` }
Project $\supseteq$ { `webdam`, `diadem` }
worksFor $\supseteq$ { (`serge`,`webdam`), (`georg`,`diadem`) }

Query: $q(x) \leftarrow \text{Researcher}(x)$

Answer: ???

unibz

Incomplete information – Example 1



We assume that each concept/relationship of the ontology is mapped directly to a database table.

But the database tables may be **incompletely specified**, or even missing for some concepts/relationships.

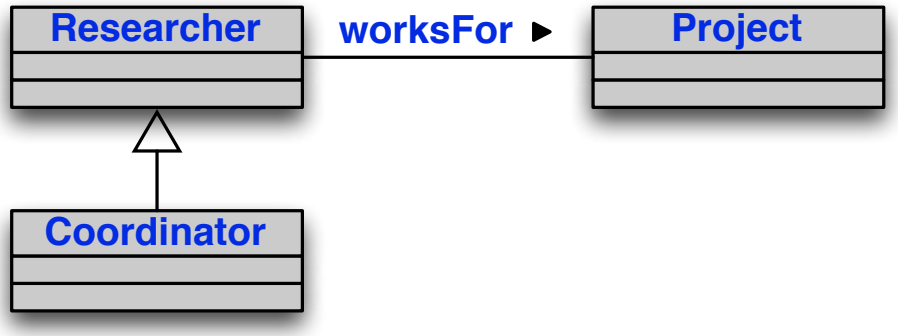
DB: Coordinator $\supseteq$ { serge, marie }
Project $\supseteq$ { webdam, diadem }
worksFor $\supseteq$ { (serge,webdam), (georg,diadem) }

Query: $q(x) \leftarrow \text{Researcher}(x)$

Answer: ???

unibz

Incomplete information – Example 1



We assume that each concept/relationship of the ontology is mapped directly to a database table.

But the database tables may be **incompletely specified**, or even missing for some concepts/relationships.

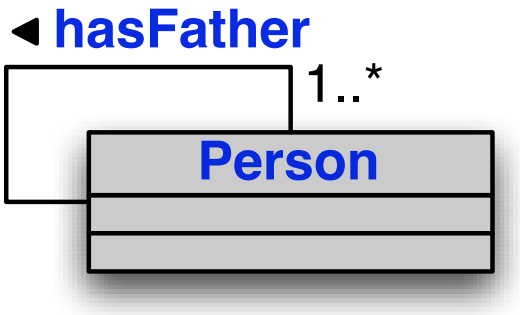
DB: Coordinator $\supseteq$ { serge, marie }
Project $\supseteq$ { webdam, diadem }
worksFor $\supseteq$ { (serge,webdam), (georg,diadem) }

Query: $q(x) \leftarrow \text{Researcher}(x)$

Answer: { serge, marie, georg }

unibz

Incomplete information – Example 2



Each person has a father, who is a person.

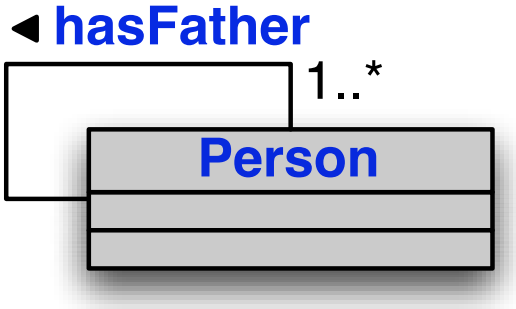
DB: Person $\supseteq$ { `john`, `nick`, `toni` }
hasFather $\supseteq$ { (`john`,`nick`), (`nick`,`toni`) }

Queries: $q_1(x, y) \leftarrow \mathsf{hasFather}(x, y)$
$q_2(x) \leftarrow \exists y.\, \mathsf{hasFather}(x, y)$
$q_3(x) \leftarrow \exists y_1, y_2, y_3.\, \mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3)$
$q_4(x, y_3) \leftarrow \exists y_1, y_2.\, \mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3)$

Answers: to q_1: ???
to q_2: ???
to q_3: ???
to q_4: ???

unibz

Incomplete information – Example 2



Each person has a father, who is a person.

DB: Person $\supseteq$ { john, nick, toni }
hasFather $\supseteq$ { (john,nick), (nick,toni) }

Queries: $q_1(x, y) \leftarrow \text{hasFather}(x, y)$

$q_2(x) \leftarrow \exists y.\, \text{hasFather}(x, y)$

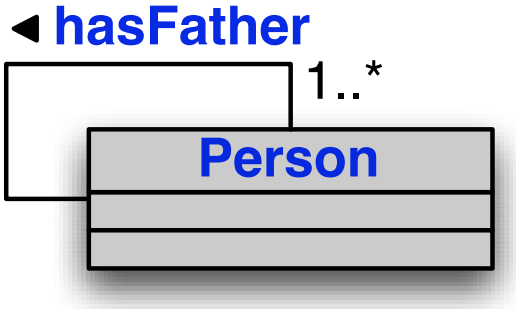
$q_3(x) \leftarrow \exists y_1, y_2, y_3.\, \text{hasFather}(x, y_1) \wedge \text{hasFather}(y_1, y_2) \wedge \text{hasFather}(y_2, y_3)$

$q_4(x, y_3) \leftarrow \exists y_1, y_2.\, \text{hasFather}(x, y_1) \wedge \text{hasFather}(y_1, y_2) \wedge \text{hasFather}(y_2, y_3)$

Answers: to q_1: { (john,nick), (nick,toni) }
to q_2: ???
to q_3: ???
to q_4: ???

unibz

Incomplete information – Example 2



Each person has a father, who is a person.

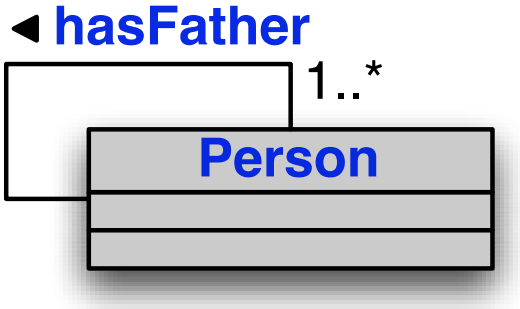
DB: Person $\supseteq$ { john, nick, toni }
hasFather $\supseteq$ { (john,nick), (nick,toni) }

Queries: $q_1(x, y) \leftarrow \mathsf{hasFather}(x, y)$
$q_2(x) \leftarrow \exists y.\, \mathsf{hasFather}(x, y)$
$q_3(x) \leftarrow \exists y_1, y_2, y_3.\, \mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3)$
$q_4(x, y_3) \leftarrow \exists y_1, y_2.\, \mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3)$

Answers: to q_1: { (john,nick), (nick,toni) }
to q_2: ???
to q_3: ???
to q_4: ???

unibz

Incomplete information – Example 2



Each person has a father, who is a person.

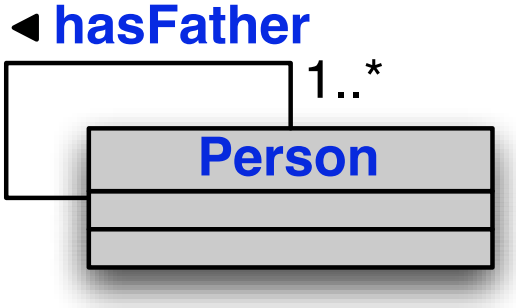
DB: Person $\supseteq$ { john, nick, toni }
hasFather $\supseteq$ { (john,nick), (nick,toni) }

Queries: $q_1(x, y) \leftarrow \mathsf{hasFather}(x, y)$
$q_2(x) \leftarrow \exists y.\, \mathsf{hasFather}(x, y)$
$q_3(x) \leftarrow \exists y_1, y_2, y_3.\, \mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3)$
$q_4(x, y_3) \leftarrow \exists y_1, y_2.\, \mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3)$

Answers: to q_1: { (john,nick), (nick,toni) }
to q_2: { john, nick, toni }
to q_3: ???
to q_4: ???

unibz

Incomplete information – Example 2



Each person has a father, who is a person.

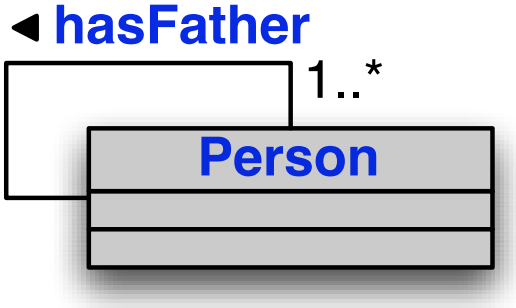
DB: Person $\supseteq$ { john, nick, toni }
hasFather $\supseteq$ { (john,nick), (nick,toni) }

Queries: $q_1(x, y) \leftarrow \mathsf{hasFather}(x, y)$
$q_2(x) \leftarrow \exists y.\, \mathsf{hasFather}(x, y)$
$q_3(x) \leftarrow \exists y_1, y_2, y_3.\, \mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3)$
$q_4(x, y_3) \leftarrow \exists y_1, y_2.\, \mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3)$

Answers: to q_1: { (john,nick), (nick,toni) }
to q_2: { john, nick, toni }
to q_3: ???
to q_4: ???

unibz

Incomplete information – Example 2



Each person has a father, who is a person.

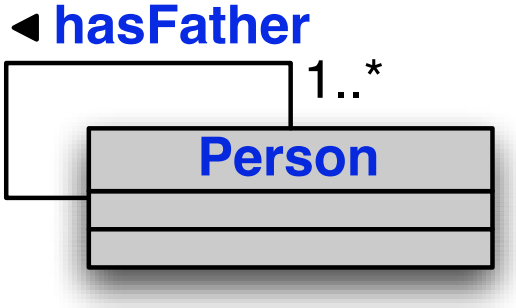
DB: Person $\supseteq$ { john, nick, toni }
hasFather $\supseteq$ { (john,nick), (nick,toni) }

Queries: $q_1(x, y) \leftarrow \mathsf{hasFather}(x, y)$
$q_2(x) \leftarrow \exists y.\, \mathsf{hasFather}(x, y)$
$q_3(x) \leftarrow \exists y_1, y_2, y_3.\, \mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3)$
$q_4(x, y_3) \leftarrow \exists y_1, y_2.\, \mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3)$

Answers: to q_1: { (john,nick), (nick,toni) }
to q_2: { john, nick, toni }
to q_3: { john, nick, toni }
to q_4: ???

unibz

Incomplete information – Example 2



Each person has a father, who is a person.

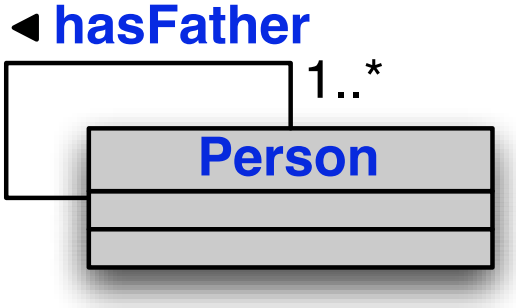
DB: Person $\supseteq$ { john, nick, toni }
hasFather $\supseteq$ { (john,nick), (nick,toni) }

Queries: $q_1(x, y) \leftarrow \mathsf{hasFather}(x, y)$
$q_2(x) \leftarrow \exists y.\, \mathsf{hasFather}(x, y)$
$q_3(x) \leftarrow \exists y_1, y_2, y_3.\, \mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3)$
$q_4(x, y_3) \leftarrow \exists y_1, y_2.\, \mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3)$

Answers: to q_1: { (john,nick), (nick,toni) }
to q_2: { john, nick, toni }
to q_3: { john, nick, toni }
to q_4: ???

unibz

Incomplete information – Example 2



Each person has a father, who is a person.

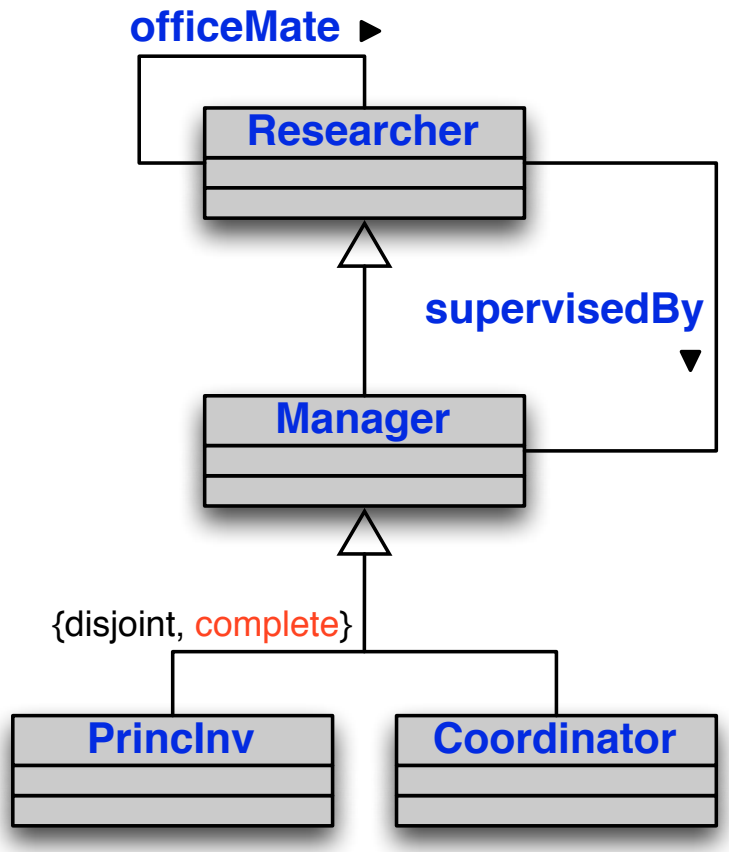
DB: Person $\supseteq$ { john, nick, toni }
hasFather $\supseteq$ { (john,nick), (nick,toni) }

Queries: $q_1(x, y) \leftarrow \mathsf{hasFather}(x, y)$
$q_2(x) \leftarrow \exists y.\, \mathsf{hasFather}(x, y)$
$q_3(x) \leftarrow \exists y_1, y_2, y_3.\, \mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3)$
$q_4(x, y_3) \leftarrow \exists y_1, y_2.\, \mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3)$

Answers: to q_1: { (john,nick), (nick,toni) }
to q_2: { john, nick, toni }
to q_3: { john, nick, toni }
to q_4: { }

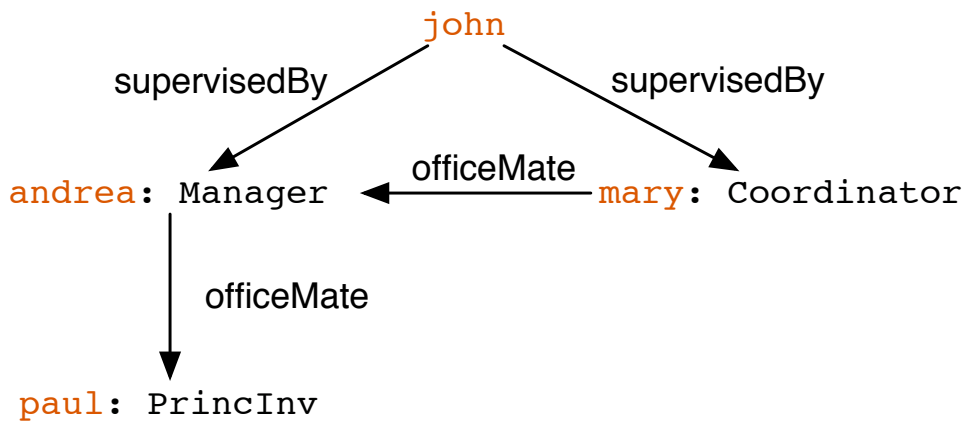
unibz

QA over ontologies – Andrea's Example *



$$\text{Manager} \equiv \text{PrincInv} \sqcup \text{Coordinator}$$

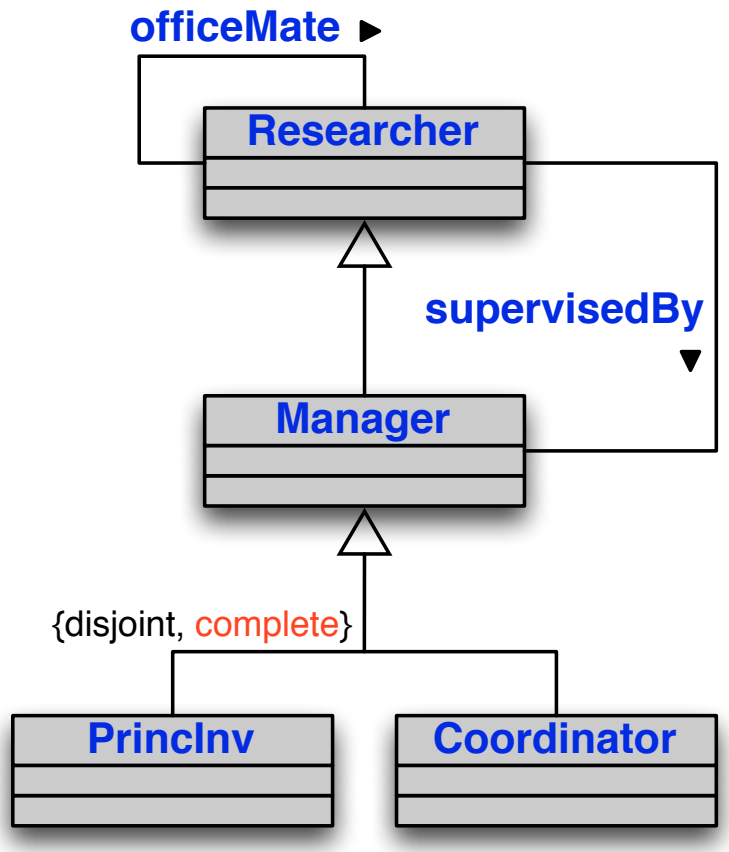
$$\begin{aligned}
\text{Researcher} &\supseteq \{ \texttt{andrea}, \texttt{paul}, \texttt{mary}, \texttt{john} \} \\
\text{Manager} &\supseteq \{ \texttt{andrea}, \texttt{paul}, \texttt{mary} \} \\
\text{PrincInv} &\supseteq \{ \texttt{paul} \} \\
\text{Coordinator} &\supseteq \{ \texttt{mary} \} \\
\text{supervisedBy} &\supseteq \{ (\texttt{john},\texttt{andrea}), (\texttt{john},\texttt{mary}) \} \\
\text{officeMate} &\supseteq \{ (\texttt{mary},\texttt{andrea}), (\texttt{andrea},\texttt{paul}) \}
\end{aligned}$$

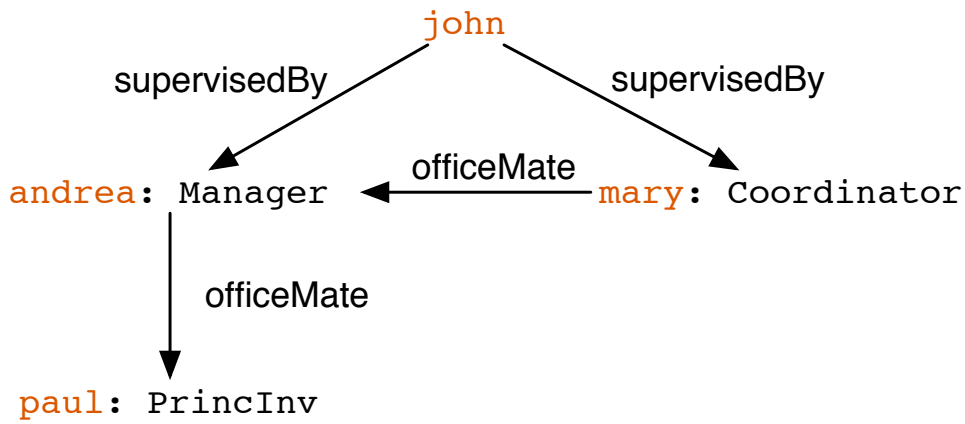


(*) By Andrea Schaerf [PhD Thesis 1994]

unibz

QA over ontologies – Andrea's Example (cont'd)



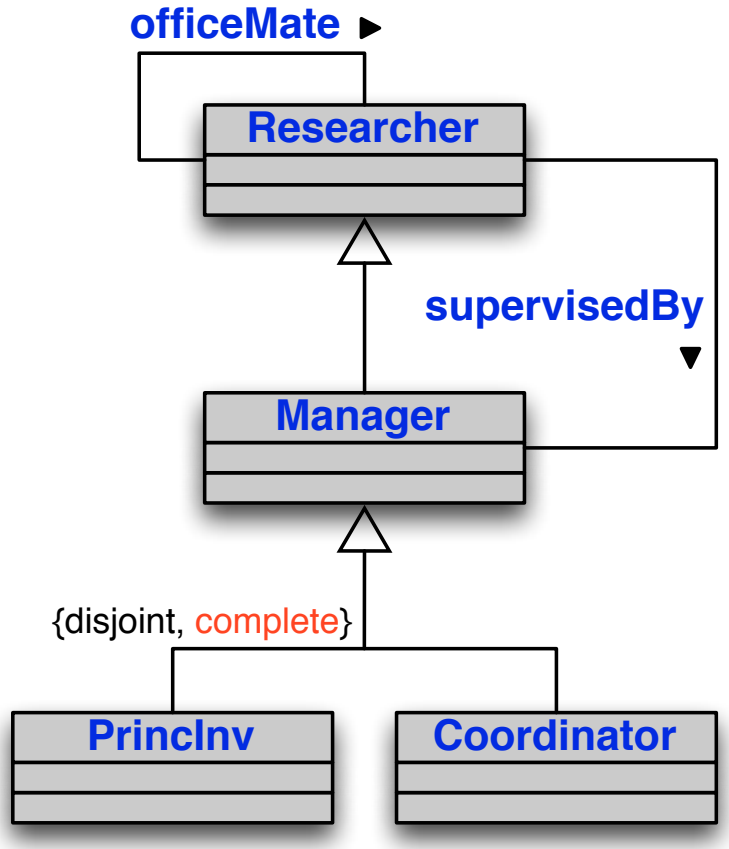


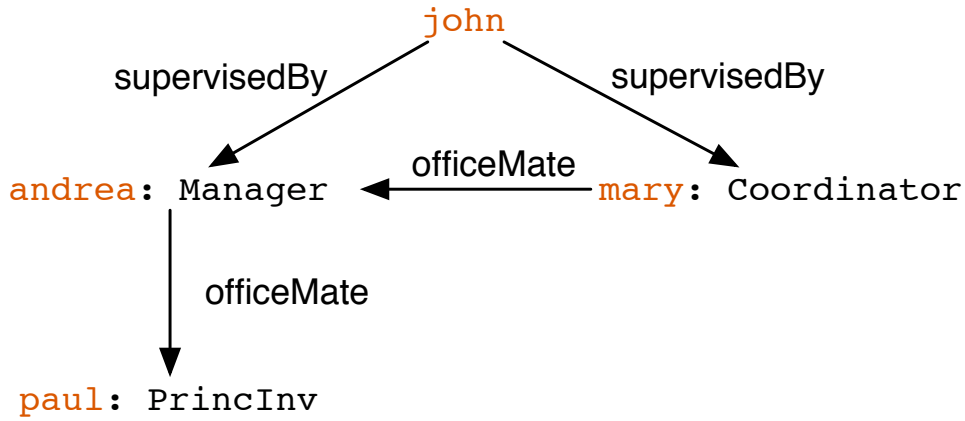
$$q(x) \leftarrow \exists y, z.\ \text{supervisedBy}(x, y),\ \text{Coordinator}(y),\ \text{officeMate}(y, z),\ \text{PrincInv}(z)$$

Answer: ???

unibz

QA over ontologies – Andrea's Example (cont'd)



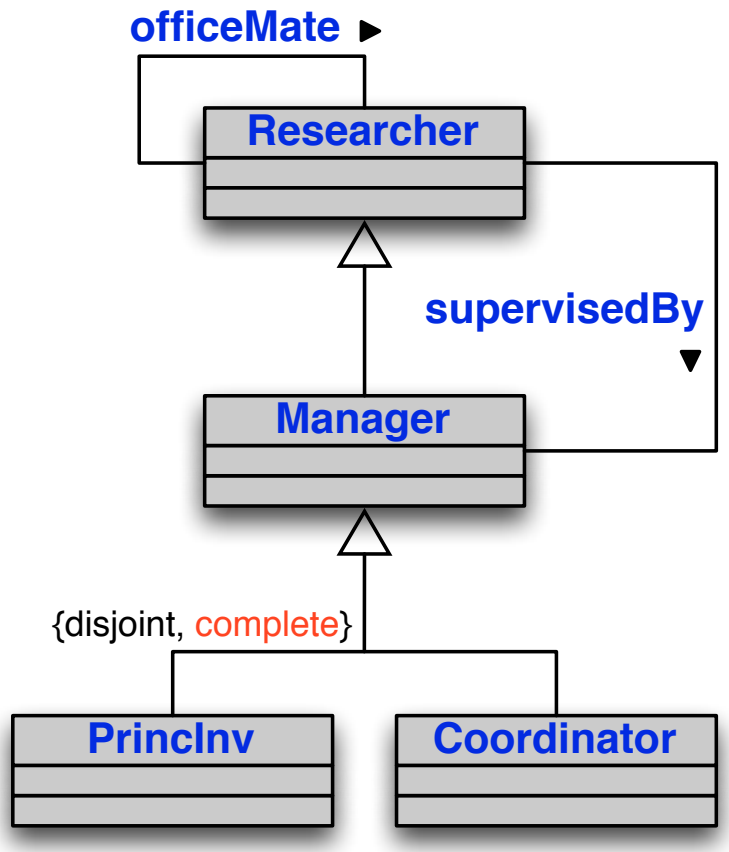


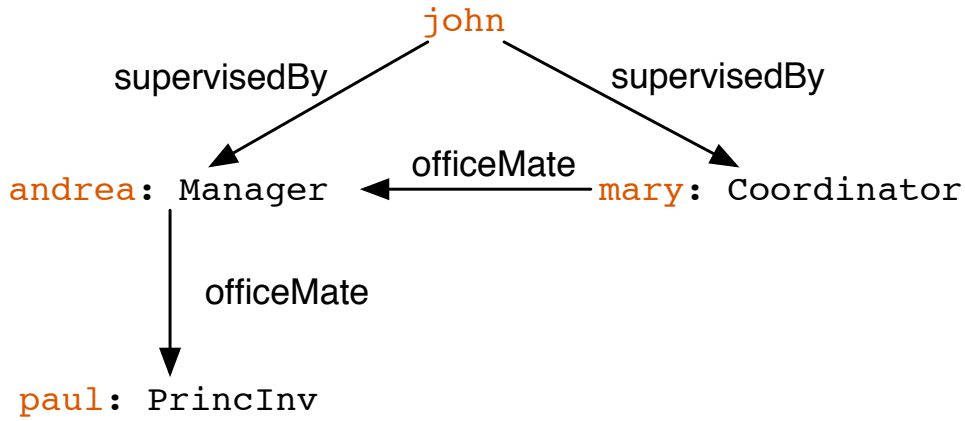
$$q(x) \leftarrow \exists y, z.\ \text{supervisedBy}(x, y),\ \text{Coordinator}(y),\ \text{officeMate}(y, z),\ \text{PrincInv}(z)$$

Answer: ???

unibz

QA over ontologies – Andrea's Example (cont'd)





$$q(x) \leftarrow \exists y, z.\ \text{supervisedBy}(x, y),\ \text{Coordinator}(y),\ \text{officeMate}(y, z),\ \text{PrincInv}(z)$$

Answer: { john }

To obtain this answer, we need to **reason by cases**.

unibz

Query answering

Certain answers

Query answering amounts to finding the **certain answers** $cert(q, \mathcal{O})$ to a query $q(\vec{x})$, i.e., those answers that hold in all models of the OBDA system $\mathcal{O}$.

Two borderline cases for the language to use 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. Full SQL (or equivalently, first-order logic).
   - Problem: in the presence of incomplete information, query answering becomes **undecidable** (FOL validity).

Conjunctive queries

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

Query answering

Certain answers

Query answering amounts to finding the **certain answers** $cert(q, \mathcal{O})$ to a query $q(\vec{x})$, i.e., those answers that hold in all models of the OBDA system $\mathcal{O}$.

Two borderline cases for the language to use 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. Full SQL (or equivalently, first-order logic).
   - Problem: in the presence of incomplete information, query answering becomes **undecidable** (FOL validity).

Conjunctive queries

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

Query answering

Certain answers

Query answering amounts to finding the **certain answers** $cert(q, \mathcal{O})$ to a query $q(\vec{x})$, i.e., those answers that hold in all models of the OBDA system $\mathcal{O}$.

Two borderline cases for the language to use 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. Full SQL (or equivalently, first-order logic).
   - Problem: in the presence of incomplete information, query answering becomes **undecidable** (FOL validity).

Conjunctive queries

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

Query answering

Certain answers

Query answering amounts to finding the **certain answers** $cert(q, \mathcal{O})$ to a query $q(\vec{x})$, i.e., those answers that hold in all models of the OBDA system $\mathcal{O}$.

Two borderline cases for the language to use 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. Full SQL (or equivalently, first-order logic).
   - Problem: in the presence of incomplete information, query answering becomes **undecidable** (FOL validity).

Conjunctive queries

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

Query answering

Certain answers

Query answering amounts to finding the **certain answers** $cert(q, \mathcal{O})$ to a query $q(\vec{x})$, i.e., those answers that hold in all models of the OBDA system $\mathcal{O}$.

Two borderline cases for the language to use 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. Full SQL (or equivalently, first-order logic).
   - Problem: in the presence of incomplete information, query answering becomes **undecidable** (FOL validity).

Conjunctive queries

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

Complexity of conjunctive query answering in DLs

Studied extensively for various ontology languages:

	Combined complexity	Data complexity
Plain databases	NP-complete	in AC^0 (1)
Expressive DLs	$\geq$ 2ExpTime (2)	coNP-hard (3)

(1) This is what we need to scale with the data.

(2) Hardness by [Lutz, 2008; Eiter *et al.*, 2009].
Tight upper bounds obtained for a variety of expressive DLs [C. *et al.*, 1998; Levy and Rousset, 1998; C. *et al.*, 2007c; C. *et al.*, 2008c; Glimm *et al.*, 2008b; Glimm *et al.*, 2008a; Lutz, 2008; Eiter *et al.*, 2008].

(3) Already for an ontology with a single axiom involving disjunction.
However, the complexity does not increase even for very expressive DLs [Ortiz *et al.*, 2006; Ortiz *et al.*, 2008; Glimm *et al.*, 2008a].

unibz

Complexity of conjunctive query answering in DLs

Studied extensively for various ontology languages:

	Combined complexity	Data complexity
Plain databases	NP-complete	in AC^0 (1)
Expressive DLs	$\geq$ 2ExpTime (2)	coNP-hard (3)

(1) This is what we need to scale with the data.

(2) Hardness by [Lutz, 2008; Eiter *et al.*, 2009].
Tight upper bounds obtained for a variety of expressive DLs [C. *et al.*, 1998; Levy and Rousset, 1998; C. *et al.*, 2007c; C. *et al.*, 2008c; Glimm *et al.*, 2008b; Glimm *et al.*, 2008a; Lutz, 2008; Eiter *et al.*, 2008].

(3) Already for an ontology with a single axiom involving disjunction.
However, the complexity does not increase even for very expressive DLs [Ortiz *et al.*, 2006; Ortiz *et al.*, 2008; Glimm *et al.*, 2008a].

unibz

Complexity of conjunctive query answering in DLs

Studied extensively for various ontology languages:

	Combined complexity	Data complexity
Plain databases	NP-complete	in AC^0 (1)
Expressive DLs	$\geq$ 2ExpTime (2)	coNP-hard (3)

(1) This is what we need to scale with the data.

(2) Hardness by [Lutz, 2008; Eiter *et al.*, 2009].
Tight upper bounds obtained for a variety of expressive DLs [C. *et al.*, 1998; Levy and Rousset, 1998; C. *et al.*, 2007c; C. *et al.*, 2008c; Glimm *et al.*, 2008b; Glimm *et al.*, 2008a; Lutz, 2008; Eiter *et al.*, 2008].

(3) Already for an ontology with a single axiom involving disjunction.
However, the complexity does not increase even for very expressive DLs [Ortiz *et al.*, 2006; Ortiz *et al.*, 2008; Glimm *et al.*, 2008a].

unibz

Complexity of conjunctive query answering in DLs

Studied extensively for various ontology languages:

	Combined complexity	Data complexity
Plain databases	NP-complete	in AC^0 (1)
Expressive DLs	$\geq$ 2ExpTime (2)	coNP-hard (3)

(1) This is what we need to scale with the data.

(2) Hardness by [Lutz, 2008; Eiter *et al.*, 2009].
Tight upper bounds obtained for a variety of expressive DLs [C. *et al.*, 1998; Levy and Rousset, 1998; C. *et al.*, 2007c; C. *et al.*, 2008c; Glimm *et al.*, 2008b; Glimm *et al.*, 2008a; Lutz, 2008; Eiter *et al.*, 2008].

(3) Already for an ontology with a single axiom involving disjunction.
However, the complexity does not increase even for very expressive DLs [Ortiz *et al.*, 2006; Ortiz *et al.*, 2008; Glimm *et al.*, 2008a].

unibz

Challenges for query answering in the OBDA setting

Challenges

- Can we find interesting ontology languages for which query answering in OBDA can be done efficiently (i.e., in AC^0)?
- If yes, can we delegate query answering in OBDA to a relational engine?
- If yes, can we obtain acceptable performance in practical scenarios involving large ontologies and large amounts of data?

unibz

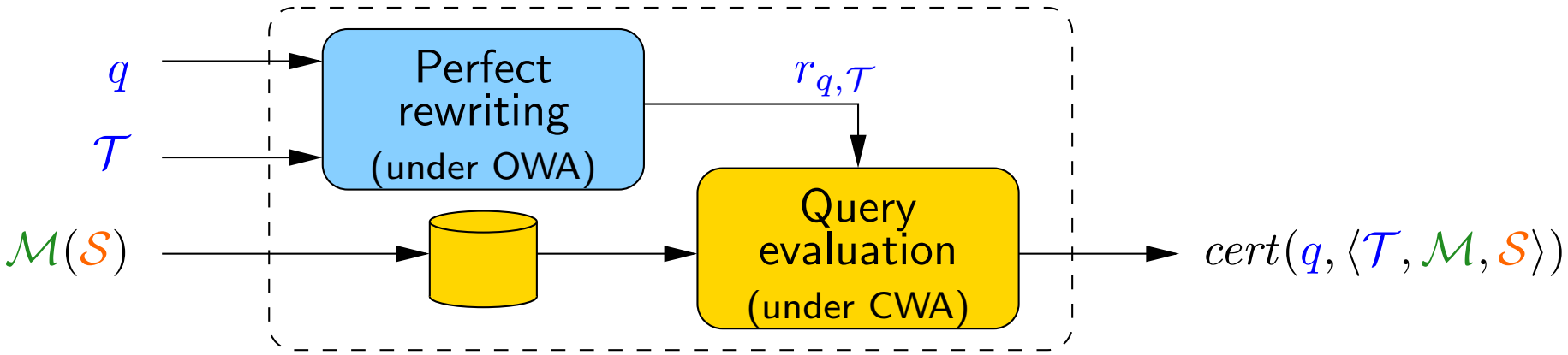
Logical inference for query answering

To be able to deal with data efficiently, we need to separate the contribution of the data $\mathcal{S}$ (accessed via the mapping $\mathcal{M}$) from the contribution of q and $\mathcal{O}$.

$\rightsquigarrow$ Query answering by **query rewriting**.

unibz

Query answering by rewriting



Query answering can **always** be thought as done in two phases:

1. **Perfect rewriting**: produce from q and the ontology TBox $\mathcal{T}$ a new query $r_{q,\mathcal{T}}$ (called the perfect rewriting of q w.r.t. $\mathcal{T}$).
2. **Query evaluation**: evaluate $r_{q,\mathcal{T}}$ over $\mathcal{M}(\mathcal{S})$ seen as a complete database (and without considering $\mathcal{T}$).
   $\rightsquigarrow$ Produces $cert(q, \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle)$.

Note: The "always" holds if we pose no restriction on the language in which to express the rewriting $r_{q,\mathcal{T}}$.

unibz

FOL-rewritability

Let:

- $\mathcal{L}_Q$ be a class of queries (i.e., a query language), and
- $\mathcal{L}_T$ be an ontology TBox language.

$\mathcal{L}_Q$-rewritability of conjunctive query answering

Conjunctive query answering is **$\mathcal{L}_Q$-rewritable** in $\mathcal{L}_T$, if for every TBox $\mathcal{T}$ of $\mathcal{L}_T$ and for every conjunctive query q, the perfect rewriting $r_{q,\mathcal{T}}$ of q w.r.t. $\mathcal{T}$ can be expressed in $\mathcal{L}_Q$.

We are especially interested in **FOL-rewritability**:

- The rewriting can be expressed in FOL, i.e., in SQL.
- Query evaluation can be delegated to a relational DBMS.

This notion was initially proposed in [C. *et al.*, 2005b; 2006; 2007a] and further intensively investigated in the KR and DB community.

unibz

FOL-rewritability

Let:

- $\mathcal{L}_Q$ be a class of queries (i.e., a query language), and
- $\mathcal{L}_T$ be an ontology TBox language.

$\mathcal{L}_Q$-rewritability of conjunctive query answering

Conjunctive query answering is **$\mathcal{L}_Q$-rewritable** in $\mathcal{L}_T$, if for every TBox $\mathcal{T}$ of $\mathcal{L}_T$ and for every conjunctive query q, the perfect rewriting $r_{q,\mathcal{T}}$ of q w.r.t. $\mathcal{T}$ can be expressed in $\mathcal{L}_Q$.

We are especially interested in **FOL-rewritability**:

- The rewriting can be expressed in FOL, i.e., in SQL.
- Query evaluation can be delegated to a relational DBMS.

This notion was initially proposed in [C. *et al.*, 2005b; 2006; 2007a] and further intensively investigated in the KR and DB community.

unibz

Outline

unibz

Description Logics

- **Description Logics (DLs)** stem from early days (70') KR formalisms, and assumed their current form in the late 80's & 90's.
- Are **logics** specifically designed to represent and reason on structured knowledge.
- Technically they can be considered as well-behaved (i.e., decidable) **fragments of first-order logic**.
- Semantics given in terms of first-order interpretations.
- Come in hundreds of variations, with different semantic and computational properties.
- Strongly influenced the W3C standard Web Ontology Language OWL.

unibz

The *DL-Lite* family

- A family of DLs optimized according to the tradeoff between expressive power and **complexity** of query answering, with emphasis on **data**.
  - The same complexity as relational databases.
  - In fact, **query answering is FOL-rewritable** and hence can be delegated to a relational DB engine.
  - The DLs of the *DL-Lite* family are essentially the maximally expressive DLs enjoying these nice computational properties.
- Nevertheless they have the "right" expressive power: capture the essential features of conceptual modeling formalisms.

DL-Lite provides robust foundations for Ontology-Based Data Access.

Note:

- The *DL-Lite* family is at the basis of the OWL 2 QL profile of the W3C standard Web Ontology Language OWL.
- More recently, the *DL-Lite* family has been extended towards n-ary relations and with additional features (see, e.g., [Calì *et al.*, 2009; Baget *et al.*, 2011; Gottlob and Schwentick, 2012; C. *et al.*, 2013]).

unibz

The *DL-Lite* family

- A family of DLs optimized according to the tradeoff between expressive power and **complexity** of query answering, with emphasis on **data**.
  - The same complexity as relational databases.
  - In fact, **query answering is FOL-rewritable** and hence can be delegated to a relational DB engine.
  - The DLs of the *DL-Lite* family are essentially the maximally expressive DLs enjoying these nice computational properties.
- Nevertheless they have the "right" expressive power: capture the essential features of conceptual modeling formalisms.

DL-Lite provides robust foundations for Ontology-Based Data Access.

Note:

- The *DL-Lite* family is at the basis of the OWL 2 QL profile of the W3C standard Web Ontology Language OWL.
- More recently, the *DL-Lite* family has been extended towards n-ary relations and with additional features (see, e.g., [Calì *et al.*, 2009; Baget *et al.*, 2011; Gottlob and Schwentick, 2012; C. *et al.*, 2013]).

unibz

The *DL-Lite* family

- A family of DLs optimized according to the tradeoff between expressive power and **complexity** of query answering, with emphasis on **data**.
  - The same complexity as relational databases.
  - In fact, **query answering is FOL-rewritable** and hence can be delegated to a relational DB engine.
  - The DLs of the *DL-Lite* family are essentially the maximally expressive DLs enjoying these nice computational properties.
- Nevertheless they have the "right" expressive power: capture the essential features of conceptual modeling formalisms.

DL-Lite provides robust foundations for Ontology-Based Data Access.

Note:

- The *DL-Lite* family is at the basis of the OWL 2 QL profile of the W3C standard Web Ontology Language OWL.
- More recently, the *DL-Lite* family has been extended towards n-ary relations and with additional features (see, e.g., [Calì *et al.*, 2009; Baget *et al.*, 2011; Gottlob and Schwentick, 2012; C. *et al.*, 2013]).

unibz

DL-Lite ontologies (essential features)

Concept and role language:

- Roles R: either atomic: P
  or an inverse role: P^-
- Concepts C: either atomic: A
  or the projection of a role on one component: $\exists P$, $\exists P^-$

TBox assertions: encode terminological knowledge about the domain

Role inclusion: $R_1 \sqsubseteq R_2$ Concept inclusion: $C_1 \sqsubseteq C_2$
Role disjointness: $R_1 \sqsubseteq \neg R_2$ Concept disjointness: $C_1 \sqsubseteq \neg C_2$
Role functionality: $(\textbf{funct}\ R)$

ABox assertions: encode knowledge about individuals
$A(c)$, $P(c_1, c_2)$, with c_1, c_2 constants

Note: *DL-Lite* distinguishes also between abstract objects and data values (ignored here).

unibz

DL-Lite ontologies (essential features)

Concept and role language:

- Roles R: either atomic: P
  or an inverse role: P^-
- Concepts C: either atomic: A
  or the projection of a role on one component: $\exists P$, $\exists P^-$

TBox assertions: encode terminological knowledge about the domain

Role inclusion:	$R_1 \sqsubseteq R_2$	Concept inclusion:	$C_1 \sqsubseteq C_2$
Role disjointness:	$R_1 \sqsubseteq \neg R_2$	Concept disjointness:	$C_1 \sqsubseteq \neg C_2$
Role functionality:	(**funct** R)		

ABox assertions: encode knowledge about individuals

$A(c)$, $P(c_1, c_2)$, with c_1, c_2 constants

Note: *DL-Lite* distinguishes also between abstract objects and data values (ignored here).

unibz

DL-Lite ontologies (essential features)

Concept and role language:

- Roles R: either atomic: P
  or an inverse role: P^-
- Concepts C: either atomic: A
  or the projection of a role on one component: $\exists P$, $\exists P^-$

TBox assertions: encode terminological knowledge about the domain

Role inclusion:	$R_1 \sqsubseteq R_2$	Concept inclusion:	$C_1 \sqsubseteq C_2$
Role disjointness:	$R_1 \sqsubseteq \neg R_2$	Concept disjointness:	$C_1 \sqsubseteq \neg C_2$
Role functionality:	$(\textbf{funct}\ R)$		

ABox assertions: encode knowledge about individuals
$A(c)$, $P(c_1, c_2)$, with c_1, c_2 constants

Note: *DL-Lite* distinguishes also between abstract objects and data values (ignored here).

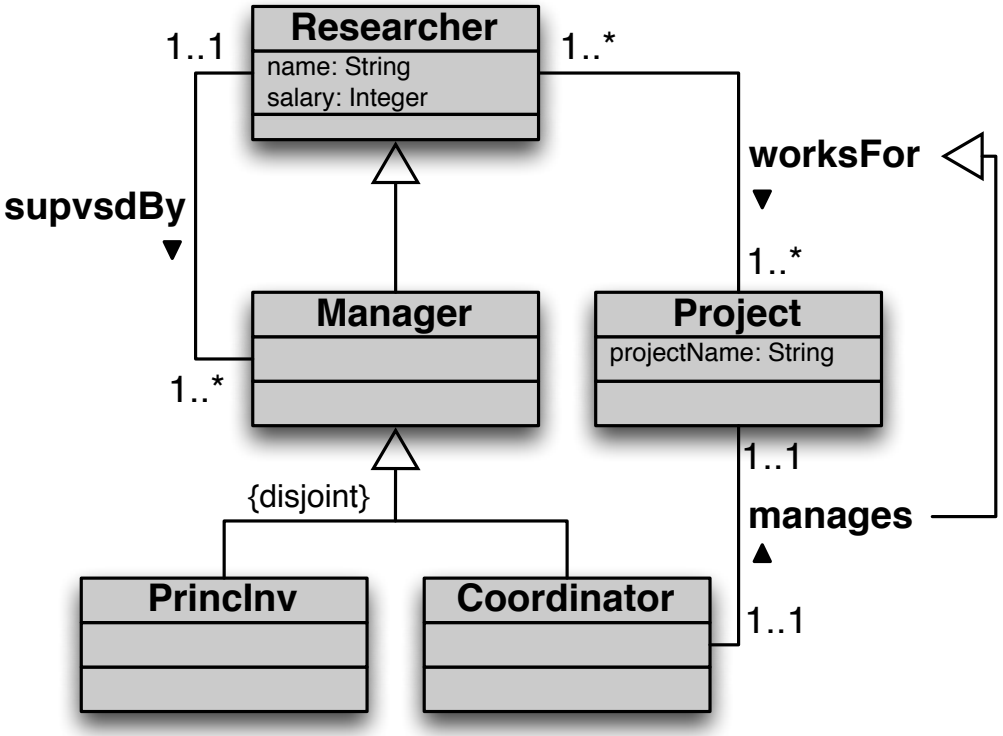
unibz

DL-Lite captures conceptual modeling formalisms

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)*	$A_1 \sqsubseteq \exists P$ $A_2 \sqsubseteq \exists P^-$	$\forall x(A_1(x) \rightarrow \exists y(P(x, y)))$ $\forall x(A_2(x) \rightarrow \exists y(P(y, x)))$
Functionality *(max card = 1)*	(**funct** P) (**funct** P^-)	$\forall x, y, y'(P(x, y) \wedge P(x, y') \rightarrow y = y')$ $\forall x, x', y(P(x, y) \wedge P(x', y) \rightarrow x = x')$
...	...	...

unibz

Capturing UML class diagrams/ER schemas in *DL-Lite*



$$\begin{aligned}
\mathit{Manager} &\sqsubseteq \mathit{Researcher}\\
\mathit{PrincInv} &\sqsubseteq \mathit{Manager}\\
\mathit{Coordinator} &\sqsubseteq \mathit{Manager}\\
\mathit{PrincInv} &\sqsubseteq \neg \mathit{Coordinator}
\end{aligned}$$

$$\begin{aligned}
\mathit{Researcher} &\sqsubseteq \exists \mathit{salary}\\
\exists \mathit{salary}^- &\sqsubseteq \texttt{xsd:int}\\
&(\textbf{funct}\ \mathit{salary})
\end{aligned}$$

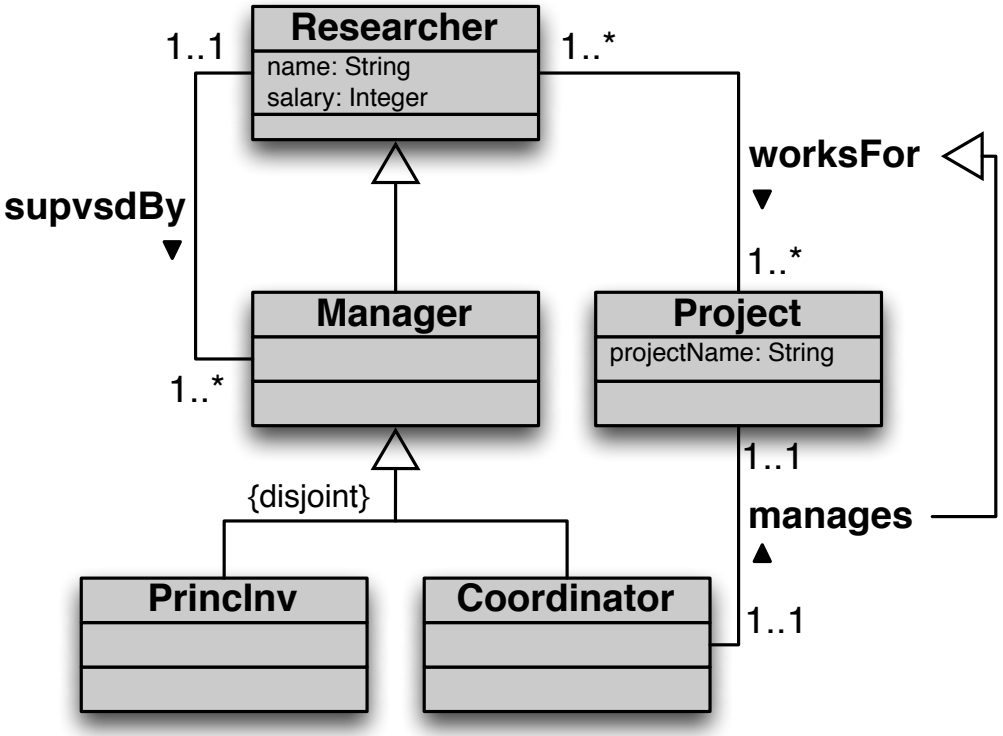
$$\begin{aligned}
\exists \mathit{worksFor} &\sqsubseteq \mathit{Researcher}\\
\exists \mathit{worksFor}^- &\sqsubseteq \mathit{Project}\\
\mathit{Researcher} &\sqsubseteq \exists \mathit{worksFor}\\
\mathit{Project} &\sqsubseteq \exists \mathit{worksFor}^-
\end{aligned}$$

$$\begin{aligned}
\exists \mathit{manages} &\sqsubseteq \mathit{Coordinator}\\
\exists \mathit{manages}^- &\sqsubseteq \mathit{Project}\\
\mathit{Coordinator} &\sqsubseteq \exists \mathit{manages}\\
\mathit{Project} &\sqsubseteq \exists \mathit{manages}^-\\
\mathit{manages} &\sqsubseteq \mathit{worksFor}\\
&(\textbf{funct}\ \mathit{manages})\\
&(\textbf{funct}\ \mathit{manages}^-)\\
&\ldots
\end{aligned}$$

Note: DL-Lite cannot capture completeness of a hierarchy. This would require **disjunction** (i.e., **OR**).

unibz

Capturing UML class diagrams/ER schemas in *DL-Lite*



$$\begin{aligned}
\mathit{Manager} &\sqsubseteq \mathit{Researcher}\\
\mathit{PrincInv} &\sqsubseteq \mathit{Manager}\\
\mathit{Coordinator} &\sqsubseteq \mathit{Manager}\\
\mathit{PrincInv} &\sqsubseteq \neg \mathit{Coordinator}
\end{aligned}$$

$$\begin{aligned}
\mathit{Researcher} &\sqsubseteq \exists \mathit{salary}\\
\exists \mathit{salary}^- &\sqsubseteq \texttt{xsd:int}\\
&(\textbf{funct}\ \mathit{salary})
\end{aligned}$$

$$\begin{aligned}
\exists \mathit{worksFor} &\sqsubseteq \mathit{Researcher}\\
\exists \mathit{worksFor}^- &\sqsubseteq \mathit{Project}\\
\mathit{Researcher} &\sqsubseteq \exists \mathit{worksFor}\\
\mathit{Project} &\sqsubseteq \exists \mathit{worksFor}^-
\end{aligned}$$

$$\begin{aligned}
\exists \mathit{manages} &\sqsubseteq \mathit{Coordinator}\\
\exists \mathit{manages}^- &\sqsubseteq \mathit{Project}\\
\mathit{Coordinator} &\sqsubseteq \exists \mathit{manages}\\
\mathit{Project} &\sqsubseteq \exists \mathit{manages}^-\\
\mathit{manages} &\sqsubseteq \mathit{worksFor}\\
&(\textbf{funct}\ \mathit{manages})\\
&(\textbf{funct}\ \mathit{manages}^-)\\
&\cdots
\end{aligned}$$

Note: *DL-Lite* cannot capture completeness of a hierarchy. This would require **disjunction** (i.e., **OR**).

unibz

Query answering in *DL-Lite*

Query answering via query rewriting

Given a (U)CQ q and an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$:

1. Compute the perfect rewriting of q w.r.t. $\mathcal{T}$, which is a FOL query.
2. Evaluate the perfect rewriting over $\mathcal{A}$. (We have ignored the mapping.)

Step 1 requires to iterate over:

- rewriting steps that involving inclusion assertions, and
- unification steps.

Note: disjointness assertions and functionalities play a role in ontology satisfiability, but can be ignored during query rewriting (i.e., we have **separability**).

unibz

Query answering in *DL-Lite*

Query answering via query rewriting

Given a (U)CQ q and an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$:

1. Compute the perfect rewriting of q w.r.t. $\mathcal{T}$, which is a FOL query.
2. Evaluate the perfect rewriting over $\mathcal{A}$. (We have ignored the mapping.)

Step 1 requires to iterate over:

- rewriting steps that involving inclusion assertions, and
- unification steps.

Note: disjointness assertions and functionalities play a role in ontology satisfiability, but can be ignored during query rewriting (i.e., we have **separability**).

unibz

Query answering in *DL-Lite*

Query answering via query rewriting

Given a (U)CQ q and an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$:

1. Compute the perfect rewriting of q w.r.t. $\mathcal{T}$, which is a FOL query.
2. Evaluate the perfect rewriting over $\mathcal{A}$. (We have ignored the mapping.)

Step 1 requires to iterate over:

- rewriting steps that involving inclusion assertions, and
- unification steps.

Note: disjointness assertions and functionalities play a role in ontology satisfiability, but can be ignored during query rewriting (i.e., we have **separability**).

unibz

Query rewriting step: Basic idea

Intuition: an inclusion assertion corresponds to a logic programming rule.

Basic rewriting step:

When an atom in the query unifies with the **head** of the rule, generate a new query by substituting the atom with the **body** of the rule.

We say that the inclusion assertion **applies to** the atom.

Example

The inclusion assertion $\mathsf{Coordinator} \sqsubseteq \mathsf{Researcher}$

corresponds to the logic programming rule $\mathsf{Researcher}(z) \leftarrow \mathsf{Coordinator}(z)$.

Consider the query $\mathsf{q}(x) \leftarrow \mathsf{Researcher}(x)$.

By applying the inclusion assertion to the atom $\mathsf{Researcher}(x)$, we generate:

$$\mathsf{q}(x) \leftarrow \mathsf{Coordinator}(x)$$

unibz

Query rewriting step: Basic idea

Intuition: an inclusion assertion corresponds to a logic programming rule.

Basic rewriting step:

When an atom in the query unifies with the **head** of the rule, generate a new query by substituting the atom with the **body** of the rule.

We say that the inclusion assertion **applies to** the atom.

Example

The inclusion assertion $\text{Coordinator} \sqsubseteq \text{Researcher}$

corresponds to the logic programming rule $\text{Researcher}(z) \leftarrow \text{Coordinator}(z)$.

Consider the query $\text{q}(x) \leftarrow \text{Researcher}(x)$.

By applying the inclusion assertion to the atom $\text{Researcher}(x)$, we generate:

$$\text{q}(x) \leftarrow \text{Coordinator}(x)$$

unibz

Query rewriting step: Basic idea

Intuition: an inclusion assertion corresponds to a logic programming rule.

Basic rewriting step:

When an atom in the query unifies with the **head** of the rule, generate a new query by substituting the atom with the **body** of the rule.

We say that the inclusion assertion **applies to** the atom.

Example

The inclusion assertion $\text{Coordinator} \sqsubseteq \text{Researcher}$
corresponds to the logic programming rule $\text{Researcher}(z) \leftarrow \text{Coordinator}(z)$.

Consider the query $\text{q}(x) \leftarrow \text{Researcher}(x)$.

By applying the inclusion assertion to the atom $\text{Researcher}(x)$, we generate:

$$\text{q}(x) \leftarrow \text{Coordinator}(x)$$

unibz

Query rewriting step: Basic idea

Intuition: an inclusion assertion corresponds to a logic programming rule.

Basic rewriting step:

When an atom in the query unifies with the **head** of the rule, generate a new query by substituting the atom with the **body** of the rule.

We say that the inclusion assertion **applies to** the atom.

Example

The inclusion assertion $\text{Coordinator} \sqsubseteq \text{Researcher}$
corresponds to the logic programming rule $\text{Researcher}(z) \leftarrow \text{Coordinator}(z)$.

Consider the query $\text{q}(x) \leftarrow \text{Researcher}(x)$.

By applying the inclusion assertion to the atom $\text{Researcher}(x)$, we generate:

$$\text{q}(x) \leftarrow \text{Coordinator}(x)$$

unibz

Query rewriting

To compute the perfect rewriting of a query q, start from q, iteratively get a CQ q' to be processed, and do one of the following:

- Apply to some atom of q' an inclusion assertion in $\mathcal{T}$ as follows:

$$
\begin{array}{rlcl}
A_1 \sqsubseteq A_2 & \ldots, A_2(x), \ldots & \rightsquigarrow & \ldots, A_1(x), \ldots \\
\exists P \sqsubseteq A & \ldots, A(x), \ldots & \rightsquigarrow & \ldots, P(x, \_), \ldots \\
\exists P^- \sqsubseteq A & \ldots, A(x), \ldots & \rightsquigarrow & \ldots, P(\_, x), \ldots \\
A \sqsubseteq \exists P & \ldots, P(x, \_), \ldots & \rightsquigarrow & \ldots, A(x), \ldots \\
A \sqsubseteq \exists P^- & \ldots, P(\_, x), \ldots & \rightsquigarrow & \ldots, A(x), \ldots \\
\exists P_1 \sqsubseteq \exists P_2 & \ldots, P_2(x, \_), \ldots & \rightsquigarrow & \ldots, P_1(x, \_), \ldots \\
P_1 \sqsubseteq P_2 & \ldots, P_2(x, y), \ldots & \rightsquigarrow & \ldots, P_1(x, y), \ldots \\
\ldots & & &
\end{array}
$$

('_' denotes a variable that appears only once)

- Choose two atoms of q' that unify, and apply the unifier to q'.

Each time, the result of the above step is added to the queries to be processed.

Note: Unifying atoms can make rules applicable that were not so before, and is required for completeness of the method [C. *et al.*, 2007a].

The UCQ resulting from this process is the **perfect rewriting** $r_{q,\mathcal{T}}$.

unibz

Query rewriting

To compute the perfect rewriting of a query q, start from q, iteratively get a CQ q' to be processed, and do one of the following:

- Apply to some atom of q' an inclusion assertion in $\mathcal{T}$ as follows:

$$
\begin{array}{rlcl}
A_1 \sqsubseteq A_2 & \ldots, A_2(x), \ldots & \rightsquigarrow & \ldots, A_1(x), \ldots \\
\exists P \sqsubseteq A & \ldots, A(x), \ldots & \rightsquigarrow & \ldots, P(x, \_), \ldots \\
\exists P^- \sqsubseteq A & \ldots, A(x), \ldots & \rightsquigarrow & \ldots, P(\_, x), \ldots \\
A \sqsubseteq \exists P & \ldots, P(x, \_), \ldots & \rightsquigarrow & \ldots, A(x), \ldots \\
A \sqsubseteq \exists P^- & \ldots, P(\_, x), \ldots & \rightsquigarrow & \ldots, A(x), \ldots \\
\exists P_1 \sqsubseteq \exists P_2 & \ldots, P_2(x, \_), \ldots & \rightsquigarrow & \ldots, P_1(x, \_), \ldots \\
P_1 \sqsubseteq P_2 & \ldots, P_2(x, y), \ldots & \rightsquigarrow & \ldots, P_1(x, y), \ldots \\
\ldots & & &
\end{array}
$$

('_' denotes a variable that appears only once)

- Choose two atoms of q' that unify, and apply the unifier to q'.

Each time, the result of the above step is added to the queries to be processed.

Note: Unifying atoms can make rules applicable that were not so before, and is required for completeness of the method [C. *et al.*, 2007a].

The UCQ resulting from this process is the **perfect rewriting** $r_{q,\mathcal{T}}$.

unibz

Query rewriting

To compute the perfect rewriting of a query q, start from q, iteratively get a CQ q' to be processed, and do one of the following:

- Apply to some atom of q' an inclusion assertion in $\mathcal{T}$ as follows:

$$
\begin{array}{rlcl}
A_1 \sqsubseteq A_2 & \ldots, A_2(x), \ldots & \rightsquigarrow & \ldots, A_1(x), \ldots \\
\exists P \sqsubseteq A & \ldots, A(x), \ldots & \rightsquigarrow & \ldots, P(x, \_), \ldots \\
\exists P^- \sqsubseteq A & \ldots, A(x), \ldots & \rightsquigarrow & \ldots, P(\_, x), \ldots \\
A \sqsubseteq \exists P & \ldots, P(x, \_), \ldots & \rightsquigarrow & \ldots, A(x), \ldots \\
A \sqsubseteq \exists P^- & \ldots, P(\_, x), \ldots & \rightsquigarrow & \ldots, A(x), \ldots \\
\exists P_1 \sqsubseteq \exists P_2 & \ldots, P_2(x, \_), \ldots & \rightsquigarrow & \ldots, P_1(x, \_), \ldots \\
P_1 \sqsubseteq P_2 & \ldots, P_2(x, y), \ldots & \rightsquigarrow & \ldots, P_1(x, y), \ldots \\
\ldots & & &
\end{array}
$$

('_' denotes a variable that appears only once)

- Choose two atoms of q' that unify, and apply the unifier to q'.

Each time, the result of the above step is added to the queries to be processed.

Note: Unifying atoms can make rules applicable that were not so before, and is required for completeness of the method [C. *et al.*, 2007a].

The UCQ resulting from this process is the **perfect rewriting** $r_{q,\mathcal{T}}$.

unibz

Query answering in *DL-Lite* – Example

TBox:

$\mathsf{Coordinator} \sqsubseteq \mathsf{Researcher}$	$\mathsf{Coordinator}(x) \rightarrow \mathsf{Researcher}(x)$
$\mathsf{Researcher} \sqsubseteq \exists \mathsf{worksFor}$	$\mathsf{Researcher}(x) \rightarrow \exists y(\mathsf{worksFor}(x, y))$
$\exists \mathsf{worksFor}^- \sqsubseteq \mathsf{Project}$	$\mathsf{worksFor}(y, x) \rightarrow \mathsf{Project}(x)$

Query: $\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, y), \mathsf{Project}(y)$

Perfect rewriting: $\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, y), \mathsf{Project}(y)$
$\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, y), \mathsf{worksFor}(\_, y)$
$\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, \_)$
$\mathsf{q}(x) \leftarrow \mathsf{Researcher}(x)$
$\mathsf{q}(x) \leftarrow \mathsf{Coordinator}(x)$

ABox: $\mathsf{worksFor}(\mathtt{serge}, \mathtt{webdam})$ $\mathsf{Coordinator}(\mathtt{serge})$
$\mathsf{worksFor}(\mathtt{georg}, \mathtt{diadem})$ $\mathsf{Coordinator}(\mathtt{marie})$

Evaluating the perfect rewriting over the ABox (seen as a DB) produces as answer $\{\mathtt{serge}, \mathtt{georg}, \mathtt{marie}\}$.

unibz

Query answering in *DL-Lite* – Example

TBox:

$\mathsf{Coordinator} \sqsubseteq \mathsf{Researcher}$	$\mathsf{Coordinator}(x) \rightarrow \mathsf{Researcher}(x)$
$\mathsf{Researcher} \sqsubseteq \exists \mathsf{worksFor}$	$\mathsf{Researcher}(x) \rightarrow \exists y(\mathsf{worksFor}(x, y))$
$\exists \mathsf{worksFor}^- \sqsubseteq \mathsf{Project}$	$\mathsf{worksFor}(y, x) \rightarrow \mathsf{Project}(x)$

Query: $\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, y), \mathsf{Project}(y)$

Perfect rewriting: $\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, y), \mathsf{Project}(y)$
$\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, y), \mathsf{worksFor}(\_, y)$
$\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, \_)$
$\mathsf{q}(x) \leftarrow \mathsf{Researcher}(x)$
$\mathsf{q}(x) \leftarrow \mathsf{Coordinator}(x)$

ABox: $\mathsf{worksFor}(\mathtt{serge}, \mathtt{webdam})$ $\mathsf{Coordinator}(\mathtt{serge})$
$\mathsf{worksFor}(\mathtt{georg}, \mathtt{diadem})$ $\mathsf{Coordinator}(\mathtt{marie})$

Evaluating the perfect rewriting over the ABox (seen as a DB) produces as answer $\{\mathtt{serge}, \mathtt{georg}, \mathtt{marie}\}$.

unibz

Query answering in *DL-Lite* – Example

TBox:

$$\begin{aligned} \text{Coordinator} &\sqsubseteq \text{Researcher} & \text{Coordinator}(x) &\rightarrow \text{Researcher}(x) \\ \text{Researcher} &\sqsubseteq \exists \text{worksFor} & \text{Researcher}(x) &\rightarrow \exists y(\text{worksFor}(x, y)) \\ \exists \text{worksFor}^- &\sqsubseteq \text{Project} & \text{worksFor}(y, x) &\rightarrow \text{Project}(x) \end{aligned}$$

Query: $\text{q}(x) \leftarrow \text{worksFor}(x, y), \text{Project}(y)$

Perfect rewriting:
$\text{q}(x) \leftarrow \text{worksFor}(x, y), \text{Project}(y)$
$\text{q}(x) \leftarrow \text{worksFor}(x, y), \text{worksFor}(\_, y)$
$\text{q}(x) \leftarrow \text{worksFor}(x, \_)$
$\text{q}(x) \leftarrow \text{Researcher}(x)$
$\text{q}(x) \leftarrow \text{Coordinator}(x)$

ABox:
$\text{worksFor}(\texttt{serge}, \texttt{webdam})$ $\quad$ $\text{Coordinator}(\texttt{serge})$
$\text{worksFor}(\texttt{georg}, \texttt{diadem})$ $\quad$ $\text{Coordinator}(\texttt{marie})$

Evaluating the perfect rewriting over the ABox (seen as a DB) produces as answer $\{\texttt{serge}, \texttt{georg}, \texttt{marie}\}$.

unibz

Query answering in *DL-Lite* – Example

TBox:

$$\begin{aligned} \text{Coordinator} &\sqsubseteq \text{Researcher} & \text{Coordinator}(x) &\rightarrow \text{Researcher}(x) \\ \text{Researcher} &\sqsubseteq \exists \text{worksFor} & \text{Researcher}(x) &\rightarrow \exists y(\text{worksFor}(x, y)) \\ \exists \text{worksFor}^- &\sqsubseteq \text{Project} & \text{worksFor}(y, x) &\rightarrow \text{Project}(x) \end{aligned}$$

Query: $\text{q}(x) \leftarrow \text{worksFor}(x, y), \text{Project}(y)$

Perfect rewriting:

$$\begin{aligned} &\text{q}(x) \leftarrow \text{worksFor}(x, y), \text{Project}(y) \\ &\text{q}(x) \leftarrow \text{worksFor}(x, y), \text{worksFor}(\_, y) \\ &\text{q}(x) \leftarrow \text{worksFor}(x, \_) \\ &\text{q}(x) \leftarrow \text{Researcher}(x) \\ &\text{q}(x) \leftarrow \text{Coordinator}(x) \end{aligned}$$

ABox:

$$\begin{aligned} &\text{worksFor}(\texttt{serge}, \texttt{webdam}) & &\text{Coordinator}(\texttt{serge}) \\ &\text{worksFor}(\texttt{georg}, \texttt{diadem}) & &\text{Coordinator}(\texttt{marie}) \end{aligned}$$

Evaluating the perfect rewriting over the ABox (seen as a DB) produces as answer $\{\texttt{serge}, \texttt{georg}, \texttt{marie}\}$.

unibz

Query answering in *DL-Lite* – Example

TBox:

$$\begin{aligned} \text{Coordinator} &\sqsubseteq \text{Researcher} & \text{Coordinator}(x) &\rightarrow \text{Researcher}(x) \\ \text{Researcher} &\sqsubseteq \exists \text{worksFor} & \text{Researcher}(x) &\rightarrow \exists y(\text{worksFor}(x, y)) \\ \exists \text{worksFor}^- &\sqsubseteq \text{Project} & \text{worksFor}(y, x) &\rightarrow \text{Project}(x) \end{aligned}$$

Query: $\text{q}(x) \leftarrow \text{worksFor}(x, y), \text{Project}(y)$

Perfect rewriting:

$$\begin{aligned} &\text{q}(x) \leftarrow \text{worksFor}(x, y), \text{Project}(y) \\ &\text{q}(x) \leftarrow \text{worksFor}(x, y), \text{worksFor}(\_, y) \\ &\text{q}(x) \leftarrow \text{worksFor}(x, \_) \\ &\text{q}(x) \leftarrow \text{Researcher}(x) \\ &\text{q}(x) \leftarrow \text{Coordinator}(x) \end{aligned}$$

ABox:

$$\begin{aligned} &\text{worksFor}(\texttt{serge}, \texttt{webdam}) & &\text{Coordinator}(\texttt{serge}) \\ &\text{worksFor}(\texttt{georg}, \texttt{diadem}) & &\text{Coordinator}(\texttt{marie}) \end{aligned}$$

Evaluating the perfect rewriting over the ABox (seen as a DB) produces as answer $\{\texttt{serge}, \texttt{georg}, \texttt{marie}\}$.

unibz

Query answering in *DL-Lite* – Example

TBox:

$\mathsf{Coordinator} \sqsubseteq \mathsf{Researcher}$	$\mathsf{Coordinator}(x) \rightarrow \mathsf{Researcher}(x)$
$\mathsf{Researcher} \sqsubseteq \exists \mathsf{worksFor}$	$\mathsf{Researcher}(x) \rightarrow \exists y(\mathsf{worksFor}(x, y))$
$\exists \mathsf{worksFor}^- \sqsubseteq \mathsf{Project}$	$\mathsf{worksFor}(y, x) \rightarrow \mathsf{Project}(x)$

Query: $\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, y), \mathsf{Project}(y)$

Perfect rewriting:
$\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, y), \mathsf{Project}(y)$
$\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, y), \mathsf{worksFor}(\_, y)$
$\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, \_)$
$\mathsf{q}(x) \leftarrow \mathsf{Researcher}(x)$
$\mathsf{q}(x) \leftarrow \mathsf{Coordinator}(x)$

ABox: $\mathsf{worksFor}(\mathtt{serge}, \mathtt{webdam})$ $\mathsf{Coordinator}(\mathtt{serge})$
$\mathsf{worksFor}(\mathtt{georg}, \mathtt{diadem})$ $\mathsf{Coordinator}(\mathtt{marie})$

Evaluating the perfect rewriting over the ABox (seen as a DB) produces as answer $\{\mathtt{serge}, \mathtt{georg}, \mathtt{marie}\}$.

unibz

Query answering in *DL-Lite* – Example

TBox:

$$\begin{aligned} \text{Coordinator} &\sqsubseteq \text{Researcher} & \text{Coordinator}(x) &\rightarrow \text{Researcher}(x) \\ \text{Researcher} &\sqsubseteq \exists \text{worksFor} & \text{Researcher}(x) &\rightarrow \exists y(\text{worksFor}(x, y)) \\ \exists \text{worksFor}^- &\sqsubseteq \text{Project} & \text{worksFor}(y, x) &\rightarrow \text{Project}(x) \end{aligned}$$

Query: $\text{q}(x) \leftarrow \text{worksFor}(x, y), \text{Project}(y)$

Perfect rewriting:

$$\begin{aligned} \text{q}(x) &\leftarrow \text{worksFor}(x, y), \text{Project}(y) \\ \text{q}(x) &\leftarrow \text{worksFor}(x, y), \text{worksFor}(\_, y) \\ \text{q}(x) &\leftarrow \text{worksFor}(x, \_) \\ \text{q}(x) &\leftarrow \text{Researcher}(x) \\ \text{q}(x) &\leftarrow \text{Coordinator}(x) \end{aligned}$$

ABox:

$$\begin{aligned} &\text{worksFor}(\texttt{serge}, \texttt{webdam}) & &\text{Coordinator}(\texttt{serge}) \\ &\text{worksFor}(\texttt{georg}, \texttt{diadem}) & &\text{Coordinator}(\texttt{marie}) \end{aligned}$$

Evaluating the perfect rewriting over the ABox (seen as a DB) produces as answer $\{\texttt{serge}, \texttt{georg}, \texttt{marie}\}$.

unibz

Complexity of query answering in *DL-Lite*

Ontology satisfiability and all classical DL reasoning tasks are:

- Efficiently tractable in the size of the TBox (i.e., PTIME).
- Very efficiently tractable in the size of the ABox (i.e., AC^0).

In fact, reasoning can be done by constructing suitable FOL/SQL queries and evaluating them over the ABox (FOL-rewritability).

Query answering for CQs and UCQs is:

- PTIME in the size of the TBox.
- AC^0 in the size of the ABox.
- Exponential in the size of the **query**, more precisely NP-complete.

  In theory this is not bad, since this is precisely the complexity of evaluating CQs in plain relational DBs.

unibz

Complexity of query answering in *DL-Lite*

Ontology satisfiability and all classical DL reasoning tasks are:

- Efficiently tractable in the size of the TBox (i.e., PTIME).
- Very efficiently tractable in the size of the ABox (i.e., AC^0).

In fact, reasoning can be done by constructing suitable FOL/SQL queries and evaluating them over the ABox (FOL-rewritability).

Query answering for CQs and UCQs is:

- PTIME in the size of the TBox.
- AC^0 in the size of the ABox.
- Exponential in the size of the **query**, more precisely NP-complete.

  In theory this is not bad, since this is precisely the complexity of evaluating CQs in plain relational DBs.

unibz

Tracing the expressivity boundary

	Lhs concept	Rhs concept	funct.	Relation incl.	Data complexity of query answering
0	*DL-Lite*		$\surd^*$	$\surd^*$	in AC^0
1	$A \mid \exists P.A$	A	–	–	NLogSpace-hard
2	A	$A \mid \forall P.A$	–	–	NLogSpace-hard
3	A	$A \mid \exists P.A$	$\surd$	–	NLogSpace-hard
4	$A \mid \exists P.A \mid A_1 \sqcap A_2$	A	–	–	PTime-hard
5	$A \mid A_1 \sqcap A_2$	$A \mid \forall P.A$	–	–	PTime-hard
6	$A \mid A_1 \sqcap A_2$	$A \mid \exists P.A$	$\surd$	–	PTime-hard
7	$A \mid \exists P.A \mid \exists P^-.A$	$A \mid \exists P$	–	–	PTime-hard
8	$A \mid \exists P \mid \exists P^-$	$A \mid \exists P \mid \exists P^-$	$\surd$	$\surd$	PTime-hard
9	$A \mid \neg A$	A	–	–	coNP-**hard**
10	A	$A \mid A_1 \sqcup A_2$	–	–	coNP-**hard**
11	$A \mid \forall P.A$	A	–	–	coNP-**hard**

From [C. *et al.*, 2006; Artale *et al.*, 2009].

Notes:

- Data complexity beyond AC^0 means that query answering in **not FOL rewritable**, hence cannot be delegated to a relational DBMS.
- These results pose strict bounds on the expressive power of the ontology language that can be used in OBDA.

unibz

Outline

unibz

Experimentations and experiences

Several experimentations:

- Monte dei Paschi di Siena (led by Sapienza Univ. of Rome)
- Selex: world leading radar producer
- National Accessibility Portal of South Africa
- Horizontal Gene Transfer data and ontology
- Stanford's "Resource Index" comprising 200 ontologies from BioPortal
- Experiments on artificial data ongoing

Observations:

- Approach highly effective for bridging impedance mismatch between data sources and ontology.
- Rewriting technique effective against incompleteness in the data.

However, performance is a major issue that still prevents large-scale deployment of this technology.

Experimentations and experiences

Several experimentations:

- Monte dei Paschi di Siena (led by Sapienza Univ. of Rome)
- Selex: world leading radar producer
- National Accessibility Portal of South Africa
- Horizontal Gene Transfer data and ontology
- Stanford's "Resource Index" comprising 200 ontologies from BioPortal
- Experiments on artificial data ongoing

Observations:

- Approach highly effective for bridging impedance mismatch between data sources and ontology.
- Rewriting technique effective against incompleteness in the data.

However, performance is a major issue that still prevents large-scale deployment of this technology.

Experimentations and experiences

Several experimentations:

- Monte dei Paschi di Siena (led by Sapienza Univ. of Rome)
- Selex: world leading radar producer
- National Accessibility Portal of South Africa
- Horizontal Gene Transfer data and ontology
- Stanford's "Resource Index" comprising 200 ontologies from BioPortal
- Experiments on artificial data ongoing

Observations:

- Approach highly effective for bridging impedance mismatch between data sources and ontology.
- Rewriting technique effective against incompleteness in the data.

However, performance is a major issue that still prevents large-scale deployment of this technology.

Traditional OBDA Architecture

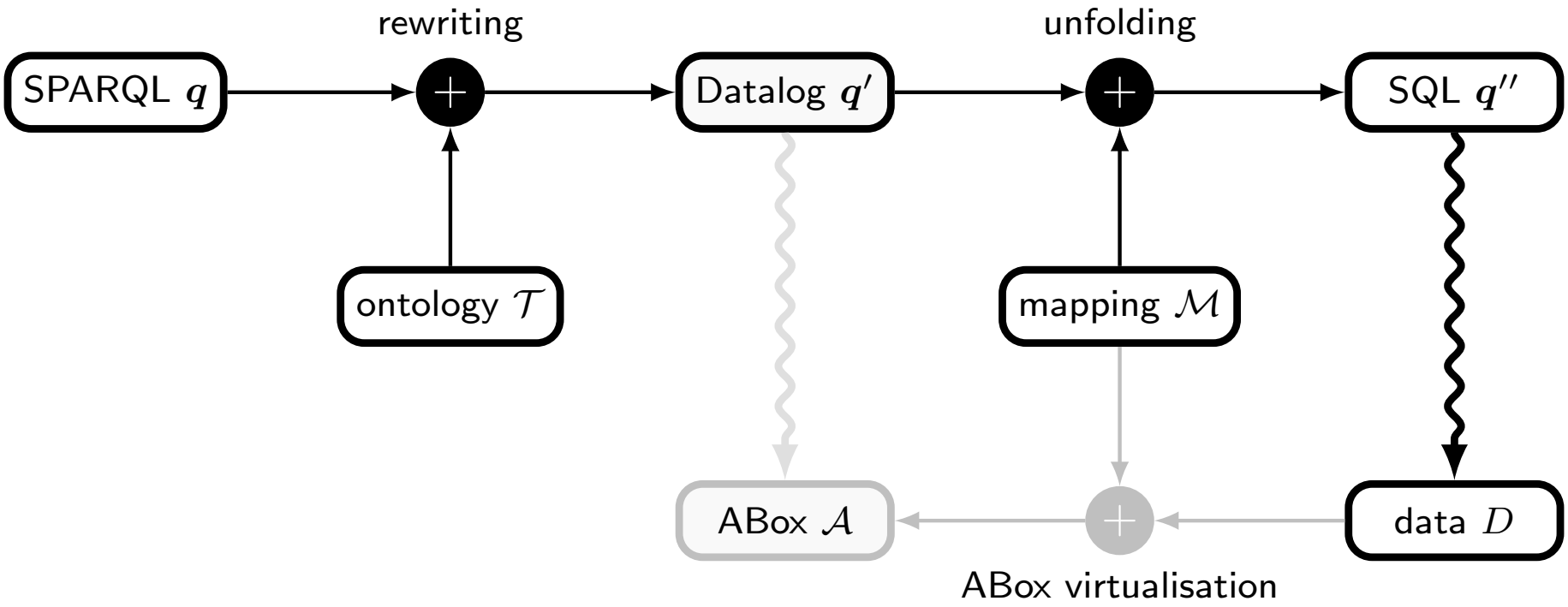


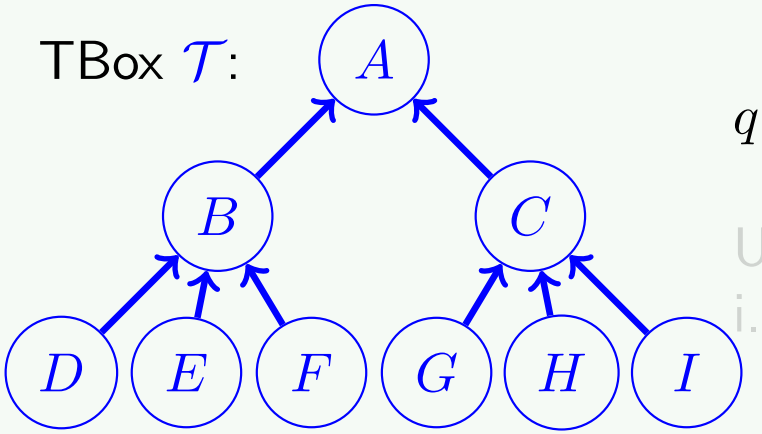
Figure: Query processing in an OBDA system

unibz

Size of the rewriting

Example

TBox $\mathcal{T}$:



$$q(x) \leftarrow A(x), P(x, y), A(y), P(y, z), A(z)$$

UCQ rewriting of q w.r.t. $\mathcal{T}$ contains 729 CQs
i.e., a UNION of 729 SPJ SQL queries

The size of UCQ rewritings may become very large

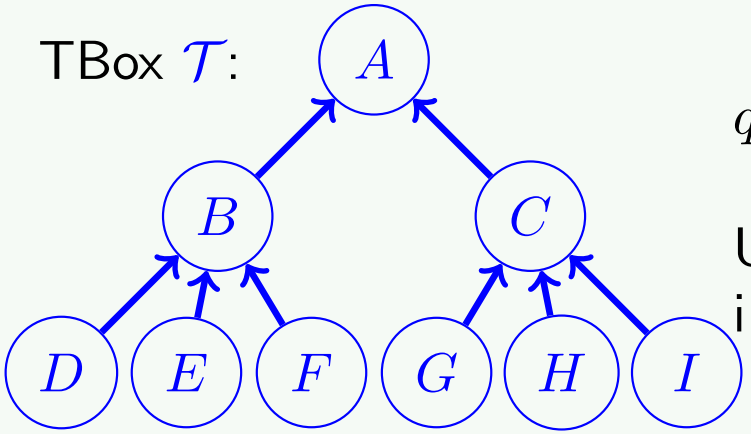
- In the worst case, it may be $O((|\mathcal{T}| \cdot |q|)^{|q|})$, i.e., **exponential in** $|q|$.
- Unfortunately, this **blowup occurs also in practice**.

unibz

Size of the rewriting

Example

TBox $\mathcal{T}$:



$q(x) \leftarrow A(x), P(x, y), A(y), P(y, z), A(z)$

UCQ rewriting of q w.r.t. $\mathcal{T}$ contains 729 CQs
i.e., a UNION of 729 SPJ SQL queries

The size of UCQ rewritings may become very large

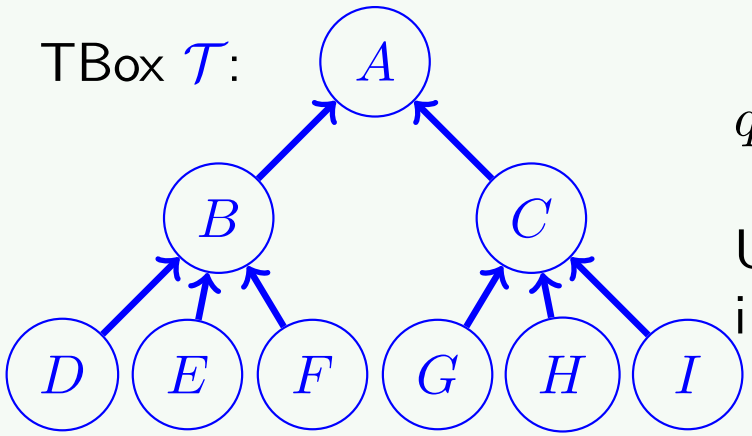
- In the worst case, it may be $O((|\mathcal{T}| \cdot |q|)^{|q|})$, i.e., **exponential in** $|q|$.
- Unfortunately, this **blowup occurs also in practice**.

unibz

Size of the rewriting

Example

TBox $\mathcal{T}$:



$q(x) \leftarrow A(x), P(x, y), A(y), P(y, z), A(z)$

UCQ rewriting of q w.r.t. $\mathcal{T}$ contains 729 CQs
i.e., a UNION of 729 SPJ SQL queries

The size of UCQ rewritings may become very large

- In the worst case, it may be $O((|\mathcal{T}| \cdot |q|)^{|q|})$, i.e., **exponential in** $|q|$.
- Unfortunately, this **blowup occurs also in practice**.

unibz

Taming the size of the rewriting

Note: It is not possible to avoid rewriting altogether, since this would require in general to materialize an infinite database [C. *et al.*, 2007a].

Several techniques have been proposed recently to limit the size of the rewriting:

- Alternative rewriting techniques [Pérez-Urbina *et al.*, 2010]: more efficient algorithm based on resolution, but produces still an exponential UCQ.
- Combined approach [Kontchakov *et al.*, 2010]: combines partial materialization with rewriting:
  - When $\mathcal{T}$ contains no role inclusions rewriting is polynomial.
  - But in general rewriting is exponential.
  - Materialization requires control over the data sources and might not be applicable in an OBDA setting.
- Rewriting into non-recursive Datalog:
  - Presto system [Rosati and Almatelli, 2010]: still worst-case exponential.
  - Polynomial rewriting for Datalog$^{\pm}$ [Gottlob and Schwentick, 2012]: rewriting uses polynomially many new existential variables and "guesses" a relevant portion of the canonical model for the TBox.

See [Kikot *et al.*, 2012a; Kikot *et al.*, 2012b] for discussion and further results.

unibz

Taming the size of the rewriting

Note: It is not possible to avoid rewriting altogether, since this would require in general to materialize an infinite database [C. *et al.*, 2007a].

Several techniques have been proposed recently to limit the size of the rewriting:

- Alternative rewriting techniques [Pérez-Urbina *et al.*, 2010]: more efficient algorithm based on resolution, but produces still an exponential UCQ.
- Combined approach [Kontchakov *et al.*, 2010]: combines partial materialization with rewriting:
  - When $\mathcal{T}$ contains no role inclusions rewriting is polynomial.
  - But in general rewriting is exponential.
  - Materialization requires control over the data sources and might not be applicable in an OBDA setting.
- Rewriting into non-recursive Datalog:
  - Presto system [Rosati and Almatelli, 2010]: still worst-case exponential.
  - Polynomial rewriting for Datalog$^{\pm}$ [Gottlob and Schwentick, 2012]: rewriting uses polynomially many new existential variables and "guesses" a relevant portion of the canonical model for the TBox.

See [Kikot *et al.*, 2012a; Kikot *et al.*, 2012b] for discussion and further results.

unibz

Taming the size of the rewriting

Note: It is not possible to avoid rewriting altogether, since this would require in general to materialize an infinite database [C. *et al.*, 2007a].

Several techniques have been proposed recently to limit the size of the rewriting:

- Alternative rewriting techniques [Pérez-Urbina *et al.*, 2010]: more efficient algorithm based on resolution, but produces still an exponential UCQ.
- Combined approach [Kontchakov *et al.*, 2010]: combines partial materialization with rewriting:
  - When $\mathcal{T}$ contains no role inclusions rewriting is polynomial.
  - But in general rewriting is exponential.
  - Materialization requires control over the data sources and might not be applicable in an OBDA setting.
- Rewriting into non-recursive Datalog:
  - Presto system [Rosati and Almatelli, 2010]: still worst-case exponential.
  - Polynomial rewriting for Datalog$^{\pm}$ [Gottlob and Schwentick, 2012]: rewriting uses polynomially many new existential variables and "guesses" a relevant portion of the canonical model for the TBox.

See [Kikot *et al.*, 2012a; Kikot *et al.*, 2012b] for discussion and further results.

unibz

Taming the size of the rewriting

Note: It is not possible to avoid rewriting altogether, since this would require in general to materialize an infinite database [C. *et al.*, 2007a].

Several techniques have been proposed recently to limit the size of the rewriting:

- Alternative rewriting techniques [Pérez-Urbina *et al.*, 2010]: more efficient algorithm based on resolution, but produces still an exponential UCQ.
- Combined approach [Kontchakov *et al.*, 2010]: combines partial materialization with rewriting:
  - When $\mathcal{T}$ contains no role inclusions rewriting is polynomial.
  - But in general rewriting is exponential.
  - Materialization requires control over the data sources and might not be applicable in an OBDA setting.
- Rewriting into non-recursive Datalog:
  - Presto system [Rosati and Almatelli, 2010]: still worst-case exponential.
  - Polynomial rewriting for Datalog$^{\pm}$ [Gottlob and Schwentick, 2012]: rewriting uses polynomially many new existential variables and "guesses" a relevant portion of the canonical model for the TBox.

See [Kikot *et al.*, 2012a; Kikot *et al.*, 2012b] for discussion and further results.

unibz

Taming the size of the rewriting

Note: It is not possible to avoid rewriting altogether, since this would require in general to materialize an infinite database [C. *et al.*, 2007a].

Several techniques have been proposed recently to limit the size of the rewriting:

- Alternative rewriting techniques [Pérez-Urbina *et al.*, 2010]: more efficient algorithm based on resolution, but produces still an exponential UCQ.
- Combined approach [Kontchakov *et al.*, 2010]: combines partial materialization with rewriting:
  - When $\mathcal{T}$ contains no role inclusions rewriting is polynomial.
  - But in general rewriting is exponential.
  - Materialization requires control over the data sources and might not be applicable in an OBDA setting.
- Rewriting into non-recursive Datalog:
  - Presto system [Rosati and Almatelli, 2010]: still worst-case exponential.
  - Polynomial rewriting for Datalog$^{\pm}$ [Gottlob and Schwentick, 2012]: rewriting uses polynomially many new existential variables and "guesses" a relevant portion of the canonical model for the TBox.

See [Kikot *et al.*, 2012a; Kikot *et al.*, 2012b] for discussion and further results.

unibz

A holistic approach to optimization

Recall our main objective

Given an OBDA system $\mathcal{O} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ and a set of queries, **compute the certain answers** of such queries w.r.t. $\mathcal{O}$ **as efficiently as possible**.

Observe:

- The size of the rewriting is only one coordinate in the problem space.
- Optimizing rewriting is necessary but not sufficient, since the more compact rewritings are in general much more difficult to evaluate.
- In fact, the **efficiency of the query evaluation by the DBMS** is the crucial factor.

Hence, a **holistic approach** is required, that considers all components of an OBDA system, i.e.:

- the TBox $\mathcal{T}$,
- the mappings $\mathcal{M}$,
- the data sources $\mathcal{S}$ with their dependencies, and
- the query load.

unibz

A holistic approach to optimization

Recall our main objective

Given an OBDA system $\mathcal{O} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ and a set of queries, **compute the certain answers** of such queries w.r.t. $\mathcal{O}$ **as efficiently as possible**.

Observe:

- The size of the rewriting is only one coordinate in the problem space.
- Optimizing rewriting is necessary but not sufficient, since the more compact rewritings are in general much more difficult to evaluate.
- In fact, the **efficiency of the query evaluation by the DBMS** is the crucial factor.

Hence, a **holistic approach** is required, that considers all components of an OBDA system, i.e.:

- the TBox $\mathcal{T}$,
- the mappings $\mathcal{M}$,
- the data sources $\mathcal{S}$ with their dependencies, and
- the query load.

unibz

A holistic approach to optimization

Recall our main objective

Given an OBDA system $\mathcal{O} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ and a set of queries, **compute the certain answers** of such queries w.r.t. $\mathcal{O}$ **as efficiently as possible**.

Observe:

- The size of the rewriting is only one coordinate in the problem space.
- Optimizing rewriting is necessary but not sufficient, since the more compact rewritings are in general much more difficult to evaluate.
- In fact, the **efficiency of the query evaluation by the DBMS** is the crucial factor.

Hence, a **holistic approach** is required, that considers all components of an OBDA system, i.e.:

- the TBox $\mathcal{T}$,
- the mappings $\mathcal{M}$,
- the data sources $\mathcal{S}$ with their dependencies, and
- the query load.

unibz

Why Rewritings may become "Intractable"

There are 3 main reasons:

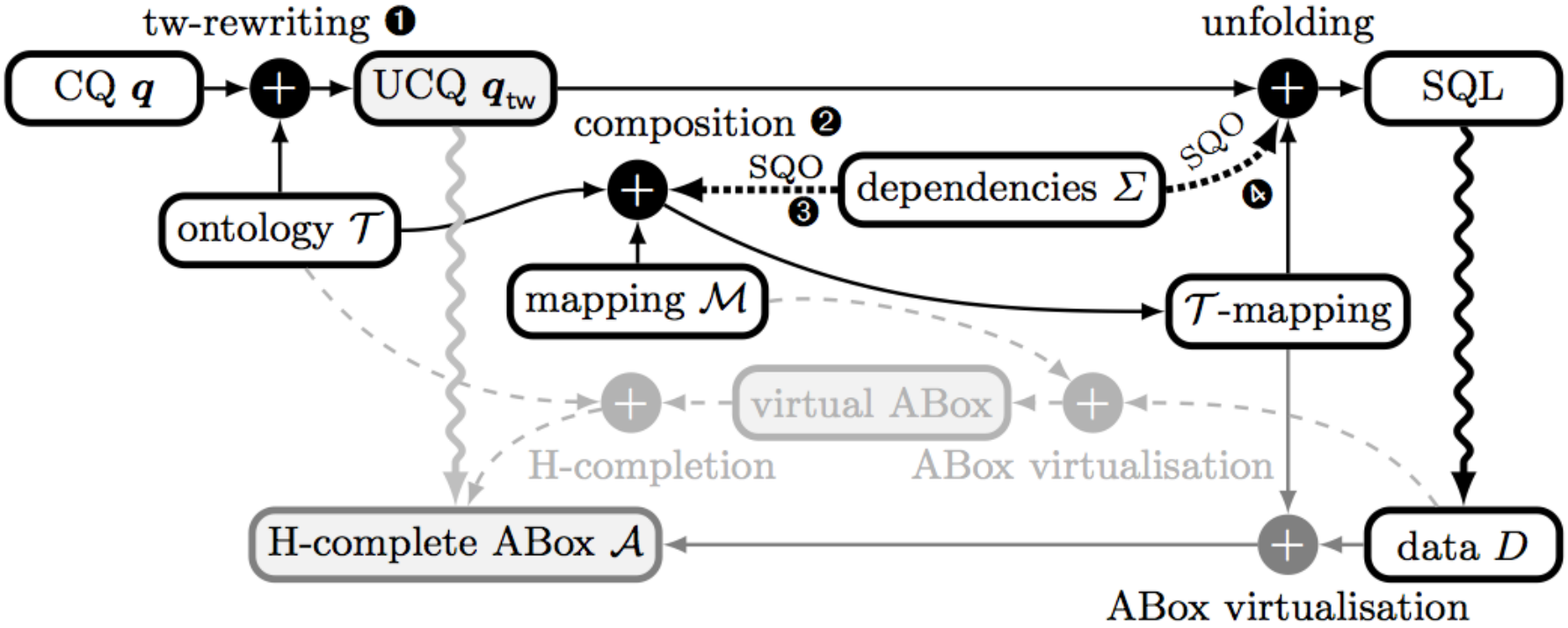
(E) Sub-queries of q with **existentially quantified variables** might lead in general to exponentially large rewritings.

(H) Classes / properties occurring in q can have **many subclasses / subproperties** according to $\mathcal{T}$, which all have to be included in the rewriting q'.

(M) The mapping $\mathcal{M}$ can have **multiple definitions of the ontology terms**, which may result in a further exponential blowup

unibz

Optimizations in Ontop [Rodriguez-Muro *et al.*, 2013]



1. the tree-witness rewriting over H-complete ABoxes,
2. $\mathcal{T}$-mappings combining $\mathcal{M}$ and $\mathcal{T}$
3. The $\mathcal{T}$-mapping is simplified using the Semantic Query Optimisation (SQO)
4. the optimized unfolding algorithm

unibz

H-complete ABoxes and T-Mapping

H-complete ABox

An ABox $\mathcal{A}$ is H-complete with respect to $\mathcal{T}$ if it satisfies the following conditions:

$$A(a) \in \mathcal{A} \text{ if } A'(a) \in \mathcal{A}, \mathcal{T} \models A' \sqsubseteq A$$
$$\text{or } R(a,b) \in \mathcal{A}, \mathcal{T} \models \exists R \sqsubseteq A$$
$$P(a,b) \in A \text{ if } R(a,b) \in \mathcal{A} \text{ and } \mathcal{T} \models R \sqsubseteq P$$

H-complete does not guarantee completeness for existential reasoning

H-complete ABox might be incomplete for axioms $A \sqsubseteq \exists R$

$\mathcal{T}$-Mapping

Given a TBox $\mathcal{T}$ and a virtual ABox $\mathcal{A} = \mathcal{M}(D)$, a $\mathcal{T}$-mapping $\mathcal{M}^{\mathcal{T}}$ for $\mathcal{T}$ w.r.t. $\mathcal{A}$ is a virtual ABox $\mathcal{A}' = \mathcal{M}^{\mathcal{T}}(D)$, such that $\mathcal{A}'$ is H-complete w.r.t. $\mathcal{T}$.

Intuition of the $\mathcal{T}$-Mapping

Embedding the inference of the ontology $\mathcal{T}$ into the mapping $\mathcal{M}$

Optimizing $\mathcal{T}$-Mappings

A naive way of constructing $\mathcal{T}$-mapping is to take the composition $\mathcal{M}^{\mathcal{T}}$ of $\mathcal{M}$ and the inclusions in $\mathcal{T}$ given by

$$A(x) \leftarrow \phi(x, z) \text{ if } A'(x) \leftarrow \phi(x, z) \in \mathcal{M} \text{ and } \mathcal{T} \models A' \sqsubseteq A,$$
$$A(x) \leftarrow \phi(x, y, z) \text{ if } R(x, y) \leftarrow \phi(x, y, z) \in \mathcal{M} \text{ and } \mathcal{T} \models \exists R \sqsubseteq A,$$
$$P(x, y) \leftarrow \phi(x, y, z) \text{ if } R(x, y) \leftarrow \phi(x, y, z) \in \mathcal{M} \text{ and } \mathcal{T} \models R \sqsubseteq P$$

This $\mathcal{T}$-Mapping can be simplified a lot using the dependencies in the DB and the ontology

unibz

Running Example

simplified IMDB (www.imdb.com/interfaces)

- The schema contains relations
  - $title[m, t, y]$ with information about movies (ID, title, production year), and
  - $castinfo[p, m, r]$ with information about movie casts (person ID, movie ID, person role),
- Ontology

$$mo{:}Movie \equiv \exists\, mo{:}title, \qquad mo{:}Movie \sqsubseteq \exists\, mo{:}year,$$
$$mo{:}Movie \equiv \exists\ mo{:}cast, \qquad \exists\ mo{:}cast^{-} \sqsubseteq mo{:}Person.$$

- Mappings

$$mo{:}Movie(m), mo{:}title(m, t), mo{:}year(m, y) \leftarrow title(m, t, y)$$
$$mo{:}cast(m, p), mo{:}Person(p) \leftarrow castinfo(p, m, r).$$

unibz

T-Mapping Optimizations – Inclusion Dependencies

Since $\mathcal{T} \models \exists mo{:}cast \sqsubseteq mo{:}Movie$, $\mathcal{T}^{MO}$ contains

$$m_1 : \quad mo{:}Movie(m) \leftarrow title(m, t, y),$$
$$m_2 : \quad mo{:}Movie(m) \leftarrow castinfo(p, m, r)$$

If there is a foreign key constraint in DB:

$$\forall m(\exists p, r \ castinfo(p, m, r) \rightarrow \exists t, y \ title(m, t, y)$$

Then m_2 is redundant.

unibz

T-Mapping Optimizations – Disjunctions in SQL

For example, the mapping $\mathcal{M}$ for IMDb and MO contains six rules for subclasses of mo:Person:

$$mo{:}Actor(p) \leftarrow castinfo(c, p, m, r), (r = 1),$$
$$\dots$$
$$mo{:}Editor(p) \leftarrow castinfo(c, p, m, r), (r = 6).$$

Then the composition $\mathcal{M}^{MO}$ contains six rules for mo:Person that differ only in the last condition $(r = k), 1 \leq k \leq 6$. These can be reduced to a single rule:

$$mo{:}Person(p) \leftarrow castinfo(c, p, m, r), (r = 1) \vee \dots \vee (r = 6).$$

unibz

Unfolding with Semantic Query Optimisation (SQO)

The CQ

$$q(t, y) \leftarrow mo{:}Movie(m), mo{:}title(m, t), mo{:}year(m, y), (y > 2010).$$

Unfolded:

$$q'(t, y) \leftarrow title(m, t_0, y_0), title(m, t, y_1), title(m, t_2, y), (y > 2010),$$

Primary key:

$$\forall m \forall t_1 \forall t_2 (\exists y\, title(m, t_1, y) \wedge \exists y\, title(m, t_2, y) \rightarrow (t_1 = t_2)),$$
$$\forall m \forall y_1 \forall y_2 (\exists t\, title(m, t, y1) \wedge \exists t\, title(m, t, y2) \rightarrow (y_1 = y_2))$$

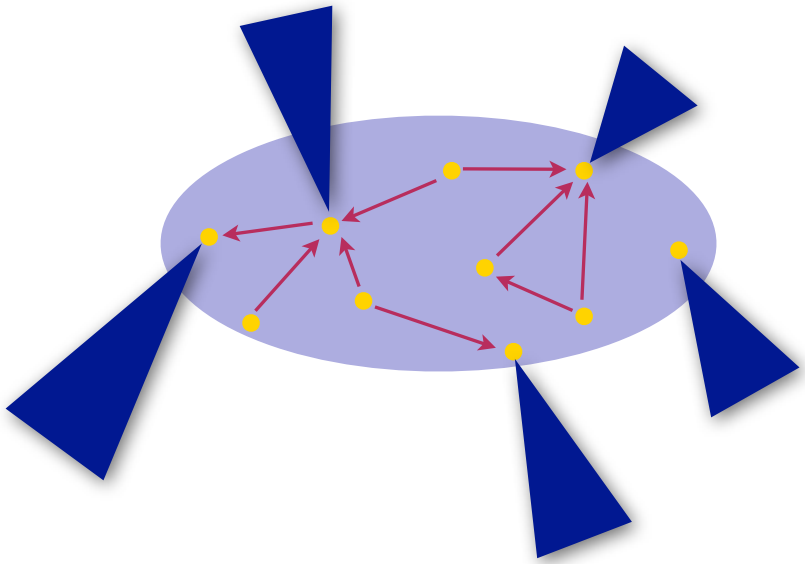
Eliminating expensive self-joins:

$$q''(t, y) \leftarrow title(m, t, y), (y > 2010).$$

unibz

Tree-witness Rewrting

Fact: every consistent DL-Lite KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ has a **canonical model** $\mathcal{I}_\mathcal{K}$ which **gives the right answers to all CQs**, i.e. $\mathsf{cert}(q, \mathcal{K}) = \mathsf{ans}(q, \mathcal{I}_\mathcal{K})$



- The core part can be handled by $\mathcal{T}$-Mapping
- The anonymous part can be handled by Tree-witness rewriting

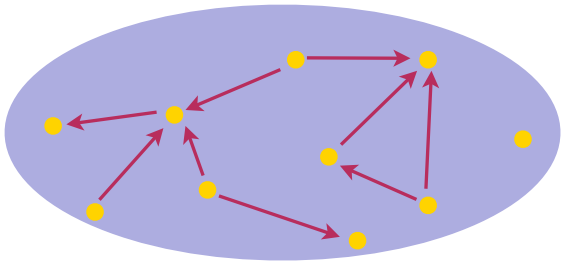
unibz

Tree-witness Rewrting

Fact: every consistent DL-Lite KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ has a **canonical model** $\mathcal{I}_\mathcal{K}$ which **gives the right answers to all CQs**, i.e. $\text{cert}(q, \mathcal{K}) = \text{ans}(q, \mathcal{I}_\mathcal{K})$

Core

individuals from $\mathcal{A}$

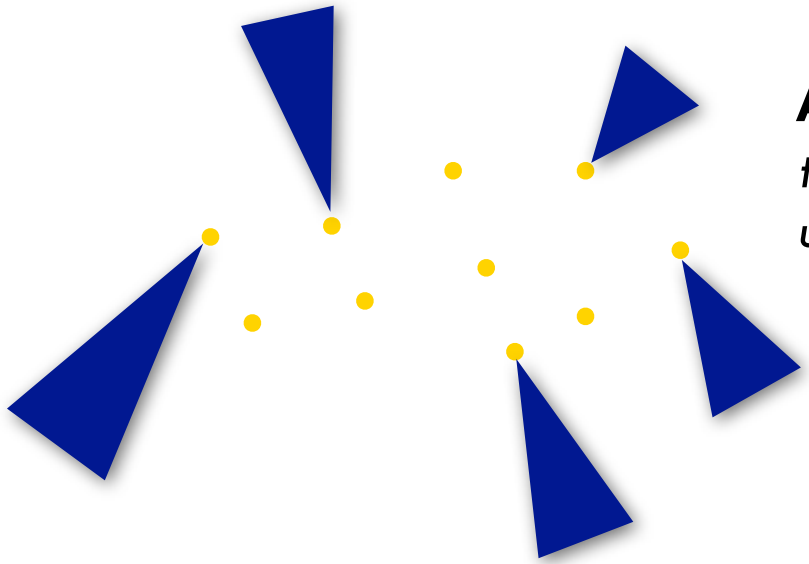


- The core part can be handled by $\mathcal{T}$-Mapping
- The anonymous part can be handled by Tree-witness rewriting

unibz

Tree-witness Rewrting

Fact: every consistent DL-Lite KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ has a **canonical model** $\mathcal{I}_\mathcal{K}$ which **gives the right answers to all CQs**, i.e. $\text{cert}(q, \mathcal{K}) = \text{ans}(q, \mathcal{I}_\mathcal{K})$



Anonymous part

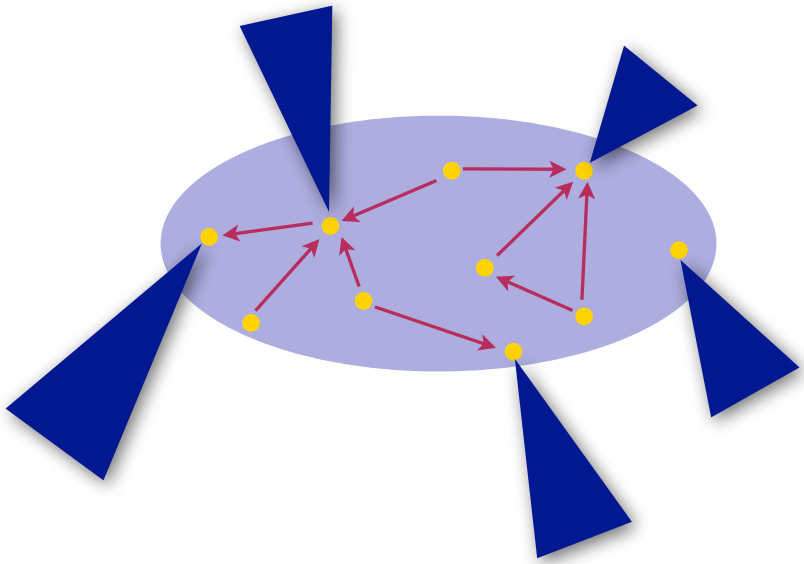
trees rooted at individuals, using unnamed objects

- The core part can be handled by $\mathcal{T}$-Mapping
- The anonymous part can be handled by Tree-witness rewriting

unibz

Tree-witness Rewrting

Fact: every consistent DL-Lite KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ has a **canonical model** $\mathcal{I}_\mathcal{K}$ which **gives the right answers to all CQs**, i.e. $\text{cert}(q, \mathcal{K}) = \text{ans}(q, \mathcal{I}_\mathcal{K})$

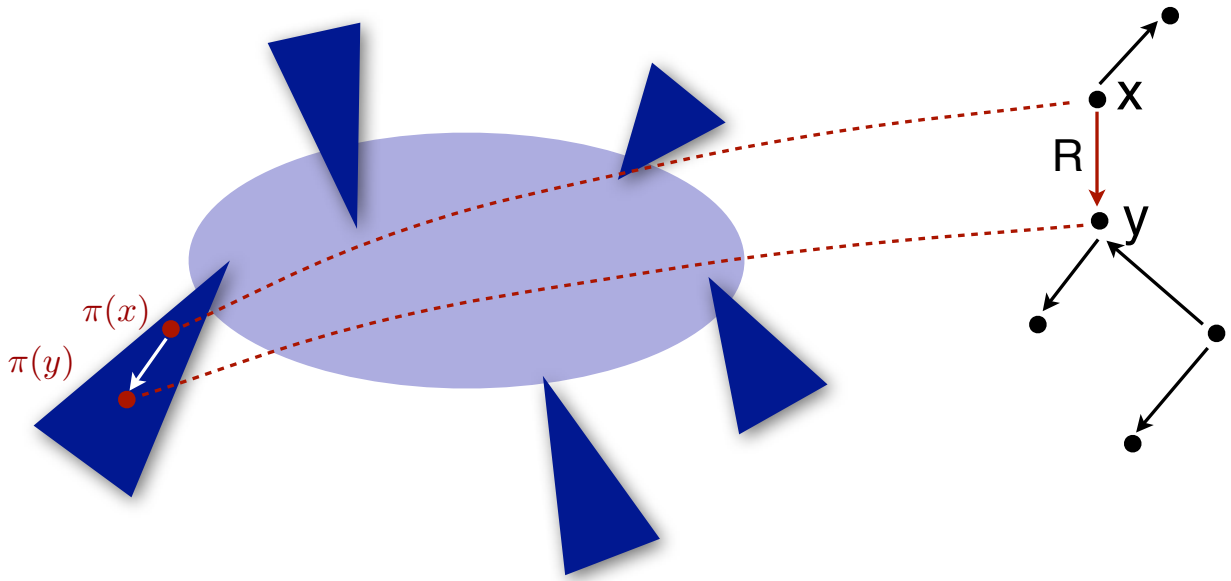


- The core part can be handled by $\mathcal{T}$-Mapping
- The anonymous part can be handled by Tree-witness rewriting

unibz

Tree-witness Rewrting

Fact: every consistent DL-Lite KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ has a **canonical model** $\mathcal{I}_\mathcal{K}$ which **gives the right answers to all CQs**, i.e. $\text{cert}(q, \mathcal{K}) = \text{ans}(q, \mathcal{I}_\mathcal{K})$

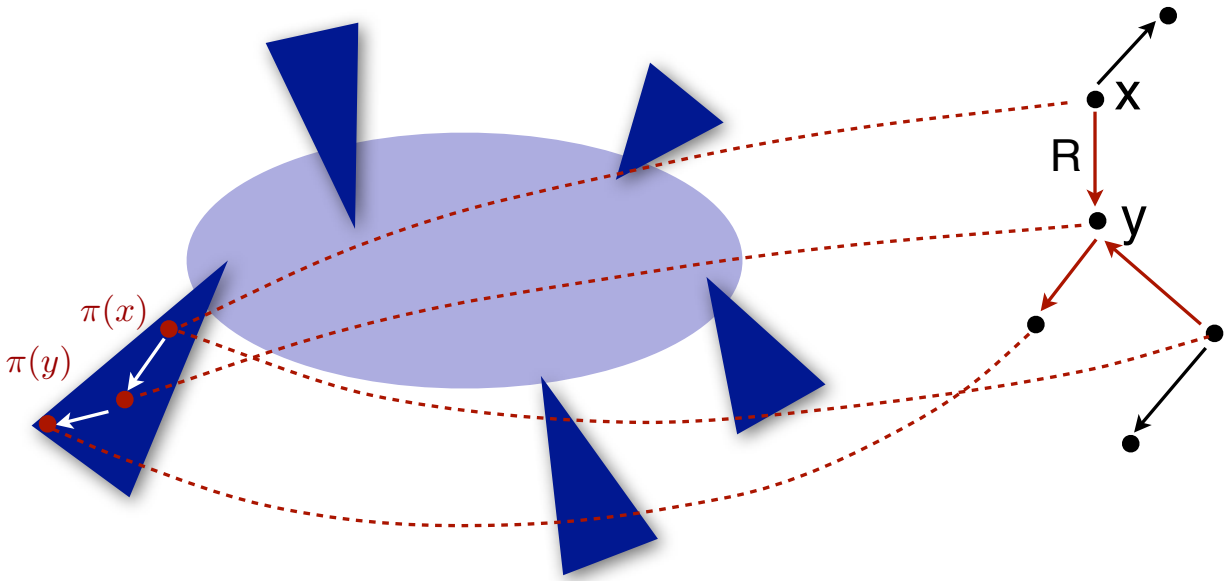


- The core part can be handled by $\mathcal{T}$-Mapping
- The anonymous part can be handled by Tree-witness rewriting

unibz

Tree-witness Rewrting

Fact: every consistent DL-Lite KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ has a **canonical model** $\mathcal{I}_\mathcal{K}$ which **gives the right answers to all CQs**, i.e. $\text{cert}(q, \mathcal{K}) = \text{ans}(q, \mathcal{I}_\mathcal{K})$



- The core part can be handled by $\mathcal{T}$-Mapping
- The anonymous part can be handled by Tree-witness rewriting

unibz

Outline

unibz

the Ontop Framework

"Stay on top of your data with semantics"

Features of Ontop

- Query language: SPARQL 1.0 (and part of 1.1) support
- Mapping languages:
  - Intuitive Ontop mapping language
  - W3C R2RML
- Ontology language: OWL 2 QL (DL-Lite)
- Database: Support for free and commercial DBMS
  - PostgreSQL, MySQL, H2, DB2, ORACLE, MS SQL SERVER, TEIID, ADP
- Java library/providers for Sesame and OWLAPI
  - Sesame: "a de-facto standard framework for processing RDF data"
  - OWLAPI: "Java API and reference implementation for OWL Ontologies"
- Integrated with Protege 4.x
- SPARQL end-point (via Sesame Workbench)
- Apache License

Infrastructure for the Development

- Homepage
  - `http://ontop.inf.unibz.it/`
- Git & Github
  - Branch Model
  - `https://github.com/ontop/ontop`
- Travis-CI
  - `https://travis-ci.org/ontop/ontop`
- Documents
  - `https://github.com/ontop/ontop/wiki`
- Maven
  - Central
  - Bolzano

unibz

Outline

unibz

Demo of Ontop

- Database: IMDb `http://www.imdb.com/`
- Ontology: MO `http://www.movieontology.org/`
- Mapping: Created by students of UNIBZ
- More info:
  `https://github.com/ontop/ontop/wiki/Example_MovieOntology`

unibz

IMDb

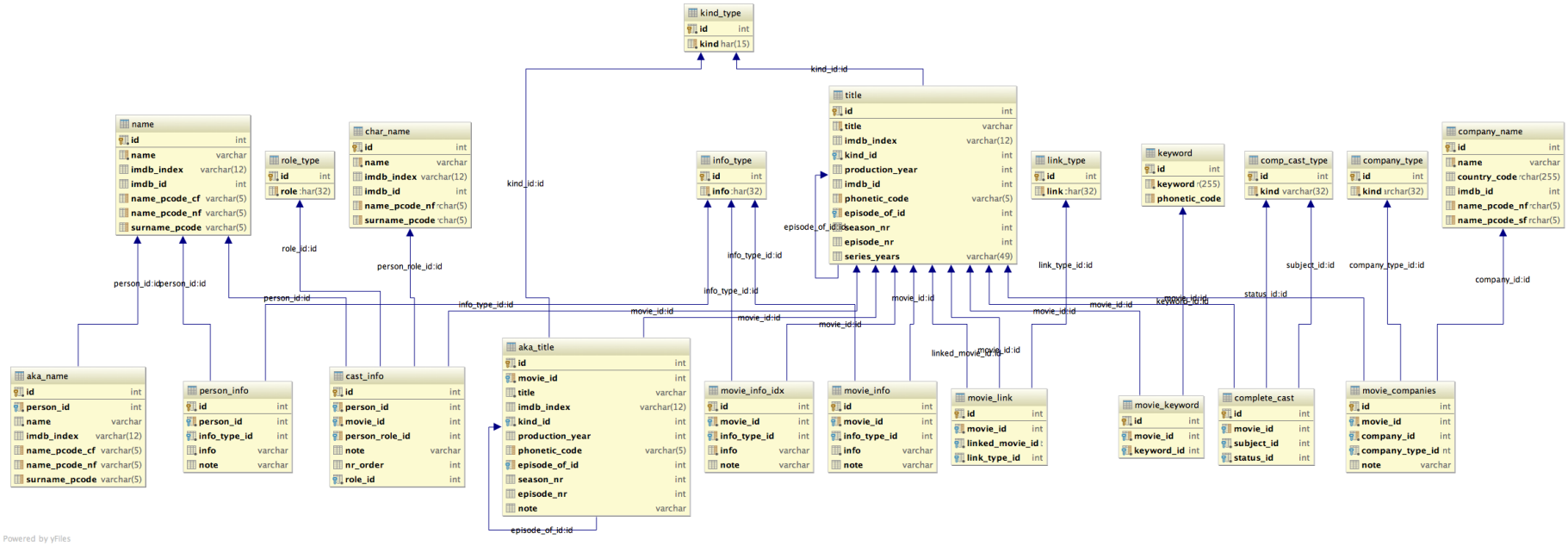


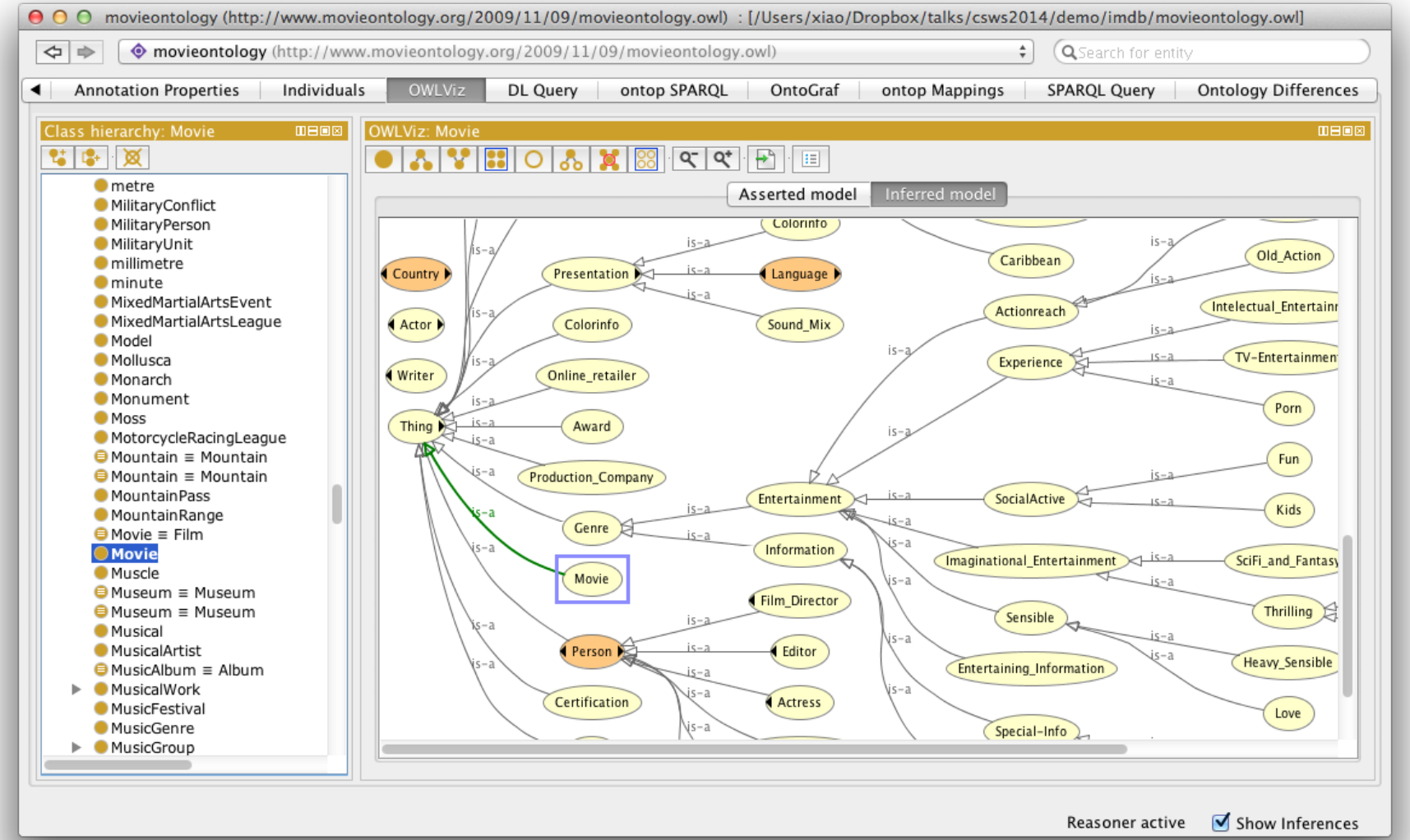
Figure: Postgres

- Movie database about movies, people, characters and companies.
- Dataset of 2010
- Processed by IMDbPY: `http://imdbpy.sourceforge.net/`
- $\sim$ 6GB on the disk

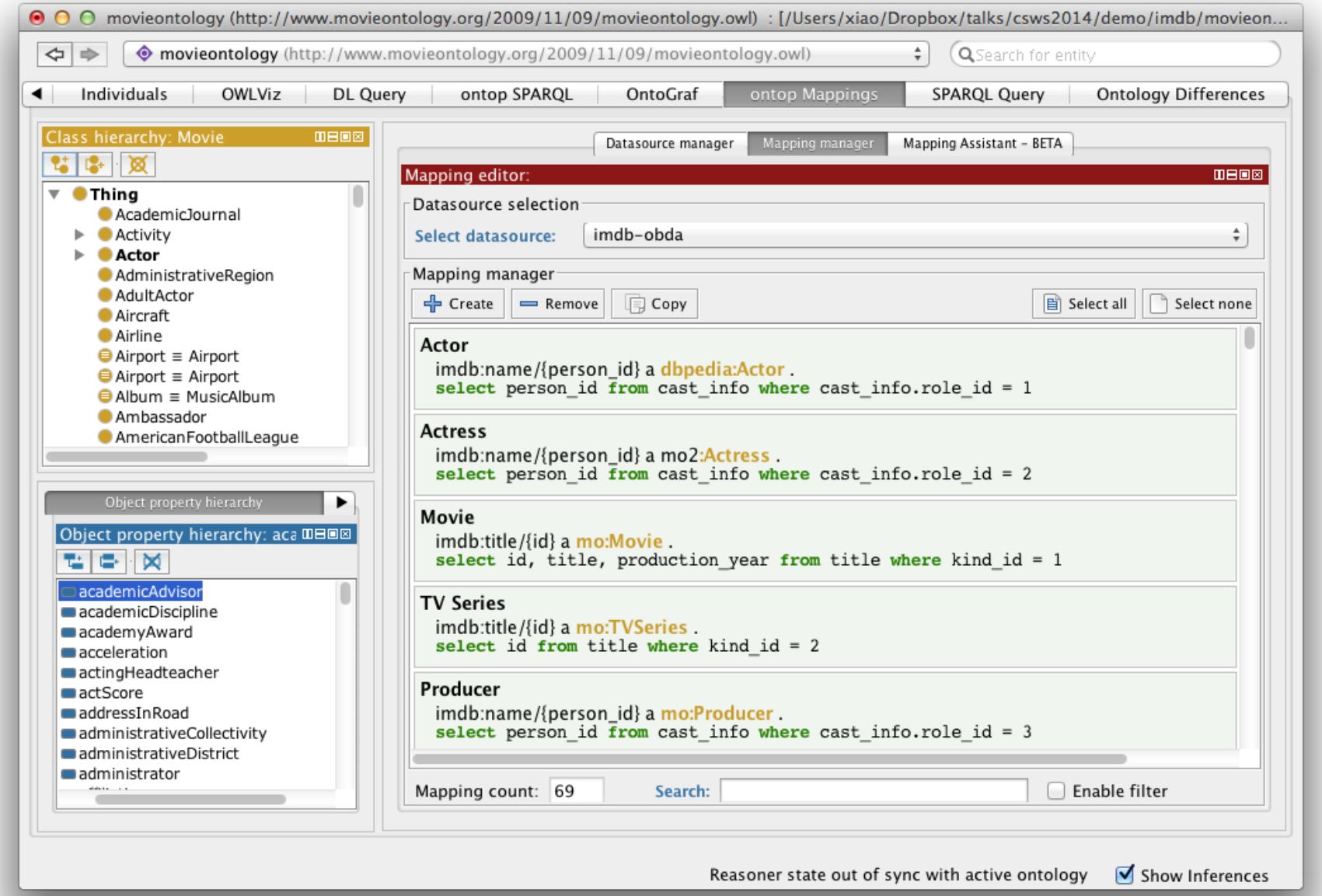
unibz

Movie Ontology

- Adapted from MO `http://www.movieontology.org/`

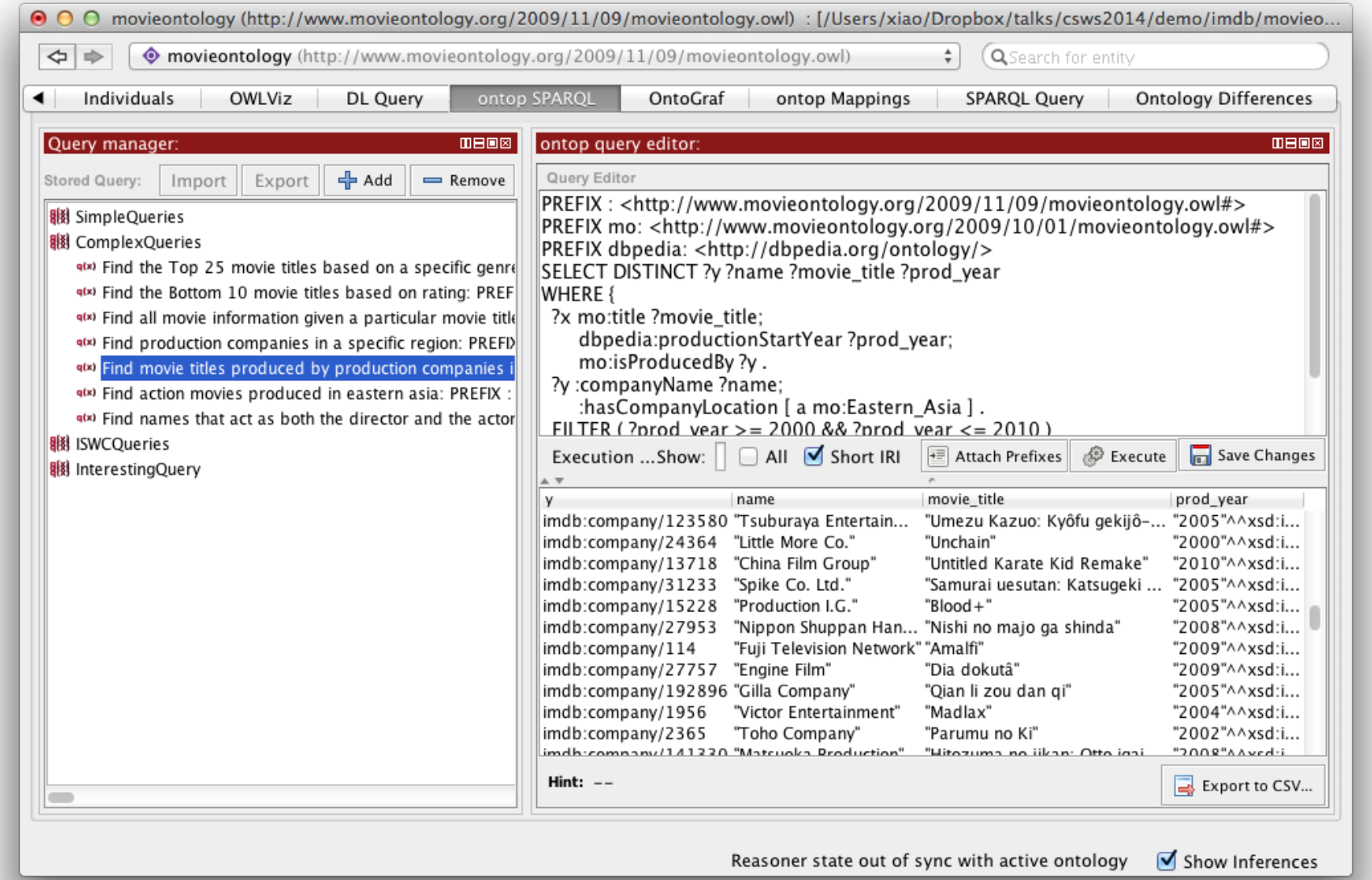


Mappings


movieontology (http://www.movieontology.org/2009/11/09/movieontology.owl) : [/Users/xiao/Dropbox/talks/csws2014/demo/imdb/movieon...
movieontology (http://www.movieontology.org/2009/11/09/movieontology.owl)
Search for entity
Individuals
OWLViz
DL Query
ontop SPARQL
OntoGraf
ontop Mappings
SPARQL Query
Ontology Differences
Class hierarchy: Movie
Thing
AcademicJournal
Activity
Actor
AdministrativeRegion
AdultActor
Aircraft
Airline
Airport ≡ Airport
Airport ≡ Airport
Album ≡ MusicAlbum
Ambassador
AmericanFootballLeague
Object property hierarchy
Object property hierarchy: aca
academicAdvisor
academicDiscipline
academyAward
acceleration
actingHeadteacher
actScore
addressInRoad
administrativeCollectivity
administrativeDistrict
administrator
Datasource manager
Mapping manager
Mapping Assistant - BETA
Mapping editor:
Datasource selection
Select datasource:
imdb-obda
Mapping manager
Create
Remove
Copy
Select all
Select none
Actor
imdb:name/{person_id} a dbpedia:Actor .
select person_id from cast_info where cast_info.role_id = 1
Actress
imdb:name/{person_id} a mo2:Actress .
select person_id from cast_info where cast_info.role_id = 2
Movie
imdb:title/{id} a mo:Movie .
select id, title, production_year from title where kind_id = 1
TV Series
imdb:title/{id} a mo:TVSeries .
select id from title where kind_id = 2
Producer
imdb:name/{person_id} a mo:Producer .
select person_id from cast_info where cast_info.role_id = 3
Mapping count: 69
Search:
Enable filter
Reasoner state out of sync with active ontology
Show Inferences

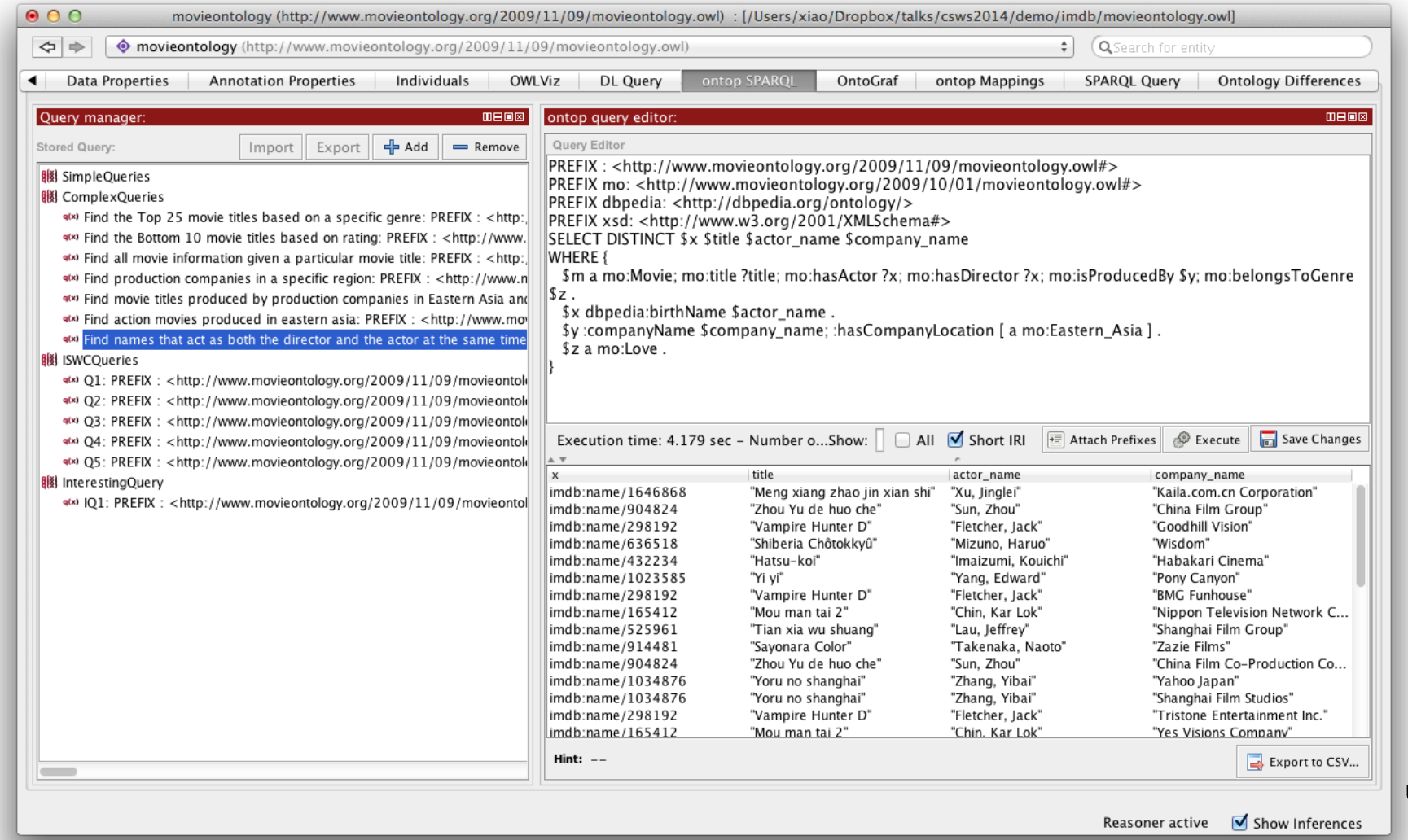
Example Query

- Find movie titles produced by production companies in Eastern Asia



Example Query

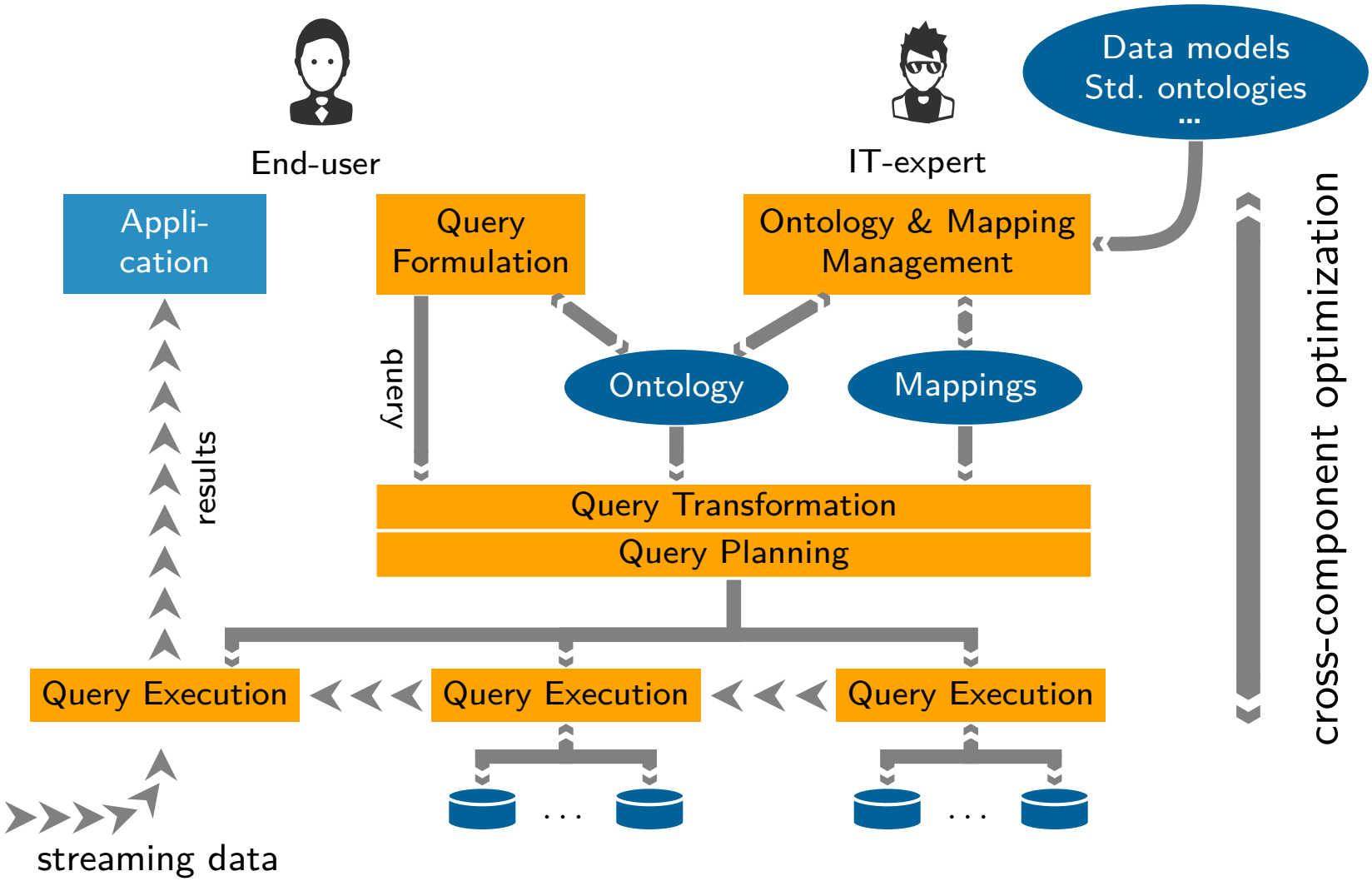
- Find names that act as both the director and the actor at the same time produced in Eastern Asia



Outline

unibz

Optique **Architecture**



Conclusions

- Ontology-based data access provides challenging problems with great practical relevance.
- In this setting, the size of the data is a critical parameter that must guide technological choices.
- Theoretical foundations provide a solid basis for system development.
- Practical deployment of this technology in real world scenarios is ongoing, but requires further research.
- Adoption of a holistic approach, considering all components of OBDA systems seems the only way to cope with real-world challenges.

unibz

Further research directions

- Dealing with inconsistency in the ontology.
- Ontology-based update.
- Coping with evolution of data in the presence of ontological constraints.
- Dealing with different kinds of data, besides relational sources: XML, graph-structured data, RDF and linked data.
- Close connection to work carried out in the Semantic Web on Triple Stores.
- Management of Mappinps and ontologies
- Natural language queries

unibz

Thanks

This presentation is based on the slides of Prof. Diego Calvanese.

Thanks to the many people that contributed to this work:

- Alessandro Artale
- Elena Botoeva
- Giuseppe De Giacomo
- Roman Kontschakov
- Domenico Lembo
- Maurizio Lenzerini
- Antonella Poggi
- Mariano Rodriguez Muro
- Martin Rezk
- Riccardo Rosati
- Michael Zakhariaschev
- many students

unibz

References I

[Artale *et al.*, 2009] Alessandro Artale, Diego C., Roman Kontchakov, and Michael Zakharyaschev.

The *DL-Lite* family and relations.

J. of Artificial Intelligence Research, 36:1–69, 2009.

[Baget *et al.*, 2011] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat.

On rules with existential variables: Walking the decidability line.

Artificial Intelligence, 175(9–10):1620–1654, 2011.

[Berardi *et al.*, 2005] Daniela Berardi, Diego C., and Giuseppe De Giacomo.

Reasoning on UML class diagrams.

Artificial Intelligence, 168(1–2):70–118, 2005.

[Bienvenu *et al.*, 2013] Meghyn Bienvenu, Magdalena Ortiz, Mantas Simkus, and Guohui Xiao.

Tractable queries for lightweight description logics.

In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. IJCAI/AAAI, 2013.

unibz

References II

[C. *et al.*, 1998] Diego C., Giuseppe De Giacomo, and Maurizio Lenzerini.

On the decidability of query containment under constraints.

In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[C. *et al.*, 2004] Diego C., Giuseppe De Giacomo, Maurizio Lenzerini, Riccardo Rosati, and Guido Vetere.

DL-Lite: Practical reasoning for rich DLs.

In *Proc. of the 17th Int. Workshop on Description Logic (DL 2004)*, volume 104 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2004.

[C. *et al.*, 2005a] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.

Tailoring OWL for data intensive ontologies.

In *Proc. of the 1st Int. Workshop on OWL: Experiences and Directions (OWLED 2005)*, volume 188 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2005.

unibz

References III

[C. *et al.*, 2005b] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.

DL-Lite: Tractable description logics for ontologies.

In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.

[C. *et al.*, 2006] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.

Data complexity of query answering in description logics.

In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 260–270, 2006.

[C. *et al.*, 2007a] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.

Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family.

J. of Automated Reasoning, 39(3):385–429, 2007.

[C. *et al.*, 2007b] Diego C., Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati.

Actions and programs over description logic ontologies.

In *Proc. of the 20th Int. Workshop on Description Logic (DL 2007)*, volume 250 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, pages 29–40, 2007.

unibz

References IV

[C. *et al.*, 2007c] Diego C., Thomas Eiter, and Magdalena Ortiz.

Answering regular path queries in expressive description logics: An automata-theoretic approach.

In *Proc. of the 22nd AAAI Conf. on Artificial Intelligence (AAAI 2007)*, pages 391–396, 2007.

[C. *et al.*, 2008a] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Riccardo Rosati, and Marco Ruzzi.

Data integration through *DL-Lite*$_{\mathcal{A}}$ ontologies.

In Klaus-Dieter Schewe and Bernhard Thalheim, editors, *Revised Selected Papers of the 3rd Int. Workshop on Semantics in Data and Knowledge Bases (SDKB 2008)*, volume 4925 of *Lecture Notes in Computer Science*, pages 26–47. Springer, 2008.

[C. *et al.*, 2008b] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.

Path-based identification constraints in description logics.

In *Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2008)*, pages 231–241, 2008.

unibz

References V

[C. *et al.*, 2008c] Diego C., Giuseppe De Giacomo, and Maurizio Lenzerini.

Conjunctive query containment and answering under description logics constraints.

ACM Trans. on Computational Logic, 9(3):22.1–22.31, 2008.

[C. *et al.*, 2013] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.

Data complexity of query answering in description logics.

Artificial Intelligence, 195:335–360, 2013.

[Calì *et al.*, 2009] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz.

Datalog$^{\pm}$: a unified approach to ontologies and integrity constraints.

In *Proc. of the 12th Int. Conf. on Database Theory (ICDT 2009)*, pages 14–30, 2009.

[Eiter *et al.*, 2008] Thomas Eiter, Georg Gottlob, Magdalena Ortiz, and Mantas Šimkus.

Query answering in the description logic Horn-$\mathcal{SHIQ}$.

In *Proc. of the 11th Eur. Conference on Logics in Artificial Intelligence (JELIA 2008)*, pages 166–179, 2008.

unibz

References VI

[Eiter *et al.*, 2009] Thomas Eiter, Carsten Lutz, Magdalena Ortiz, and Mantas Šimkus.

Query answering in description logics with transitive roles.

In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009)*, pages 759–764, 2009.

[Eiter *et al.*, 2012] T. Eiter, M. Ortiz, M. Simkus, T.K. Tran, and G. Xiao.

Query rewriting for Horn-SHIQ plus rules.

In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2012), July 22-26, 2012, Toronto, Ontario, Canada*. AAAI, AAAI Press, 2012.

[Glimm *et al.*, 2008a] Birte Glimm, Ian Horrocks, Carsten Lutz, and Uli Sattler.

Conjunctive query answering for the description logic $\mathcal{SHIQ}$.

J. of Artificial Intelligence Research, 31:151–198, 2008.

[Glimm *et al.*, 2008b] Birte Glimm, Ian Horrocks, and Ulrike Sattler.

Unions of conjunctive queries in $\mathcal{SHOQ}$.

In *Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2008)*, pages 252–262, 2008.

unibz

References VII

[Gottlob and Schwentick, 2012] Georg Gottlob and Thomas Schwentick.

Rewriting ontological queries into small nonrecursive Datalog programs.

In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2012)*, pages 254–263, 2012.

[Kikot *et al.*, 2012a] Stanislav Kikot, Roman Kontchakov, V. Podolskii, and Michael Zakharyaschev.

Long rewritings, short rewritings.

In *Proc. of the 25th Int. Workshop on Description Logic (DL 2012)*, volume 846 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2012.

[Kikot *et al.*, 2012b] Stanislav Kikot, Roman Kontchakov, and Michael Zakharyaschev.

Conjunctive query answering with OWL 2 QL.

In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2012)*, pages 275–285, 2012.

[Kontchakov *et al.*, 2010] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev.

The combined approach to query answering in *DL-Lite*.

In *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010)*, pages 247–257, 2010.

unibz

References VIII

[Kontchakov *et al.*, 2014] Roman Kontchakov, Martin Rezk, Mariano Rodriguez-Muro, Guohui Xiao, and Michael Zakharyaschev.

Answering SPARQL queries over databases under OWL 2 QL entailment regime.

In *Proc. of International Semantic Web Conference (ISWC 2014)*, Lecture Notes in Computer Science. Springer, 2014.

[Levy and Rousset, 1998] Alon Y. Levy and Marie-Christine Rousset.

Combining Horn rules and description logics in CARIN.

Artificial Intelligence, 104(1–2):165–209, 1998.

[Lutz, 2008] Carsten Lutz.

The complexity of conjunctive query answering in expressive description logics.

In *Proc. of the 4th Int. Joint Conf. on Automated Reasoning (IJCAR 2008)*, volume 5195 of *Lecture Notes in Artificial Intelligence*, pages 179–193. Springer, 2008.

[Ortiz *et al.*, 2006] Maria Magdalena Ortiz, Diego C., and Thomas Eiter.

Characterizing data complexity for conjunctive query answering in expressive description logics.

In *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006)*, pages 275–280, 2006.

unibz

References IX

[Ortiz *et al.*, 2008] Magdalena Ortiz, Diego C., and Thomas Eiter.

Data complexity of query answering in expressive description logics via tableaux.

J. of Automated Reasoning, 41(1):61–98, 2008.

[Pérez-Urbina *et al.*, 2010] Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks.

Tractable query answering and rewriting under description logic constraints.

J. of Applied Logic, 8(2):186–209, 2010.

[Poggi *et al.*, 2008] Antonella Poggi, Domenico Lembo, Diego C., Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati.

Linking data to ontologies.

J. on Data Semantics, X:133–173, 2008.

[Rodriguez-Muro *et al.*, 2013] Mariano Rodriguez-Muro, Roman Kontchakov, and Michael Zakharyaschev.

Ontology-based data access: Ontop of databases.

In *International Semantic Web Conference (1)*, volume 8218, pages 558–573. Springer, 2013.

unibz

References X

[Rodríguez-Muro, 2010] Mariano Rodríguez-Muro.

Tools and Techniques for Ontology Based Data Access in Lightweight Description Logics.

PhD thesis, KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano, 2010.

[Rosati and Almatelli, 2010] Riccardo Rosati and Alessandro Almatelli.

Improving query answering over *DL-Lite* ontologies.

In *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010)*, pages 290–300, 2010.

unibz