# Ontology-based Data Management

*Maurizio Lenzerini*

Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

SAPIENZA
UNIVERSITÀ DI ROMA

*Semantic Days 2013 – Business Intelligence and Semantics*

Stavanger, Norway, 28-30 May 2013

# Today in many organizations...

Fragment of a relational table in a Bank Information system:

| CUC | TS_START | TS_END | ID_GRUP | FLAG_CP | FLAG_CF | FATTURATO | FLAG_FATT |
|---|---|---|---|---|---|---|---|
| 124589 | 30-lug-2004 | 1-gen-9999 | 92736 | S | N | 195000,00 | N |
| 140904 | 15-mag-2001 | 15-giu-2005 | 35060 | N | N | 230600,00 | N |
| 124589 | 5-mag-2001 | 30-lug-2004 | 92736 | N | S | 195000,00 | S |
| -452901 | 13-mag-2001 | 27-lug-2004 | 92770 | S | N | 392000,00 | N |
| 129008 | 10-mag-2001 | 1-gen-9999 | 62010 | N | S | 247000,00 | S |
| -472900 | 10-mag-2001 | 1-gen-9999 | 62010 | S | N | 0 00 | N |
| 130976 | 7-mag-2001 | 9-lug-2003 | 75680 | | | | |

# Today in many organizations ...

*Negative value denotes a holding*

| CUC | TS_START | TS_END | ID_GRUP | FLAG_CP | FLAG_CF | FATTURATO | FLAG_FATT |
|---|---|---|---|---|---|---|---|
| 124589 | 30-lug-2004 | 1-gen-9999 | 92736 | S | N | 195000,00 | N |
| 140904 | 15-mag-2001 | 15-giu-2005 | 35060 | N | N | 230600,00 | N |
| 124589 | 5-mag-2001 | 30-lug-2004 | 92736 | N | S | 195000,00 | S |
| -452901 | 13-mag-2001 | 27-lug-2004 | 92770 | S | N | 392000,00 | N |
| 129008 | 10-mag-2001 | 1-gen-9999 | 62010 | N | S | 247000,00 | S |
| -472900 | 10-mag-2001 | 1-gen-9999 | 62010 | S | N | 0 00 | N |
| 130976 | 7-mag-2001 | 9-lug-2003 | 75680 | | | | |

# Today in many organizations ...

*S means that the customer is the leader of the group it belongs to*

*S means that the customer is the head of the group it belongs to*

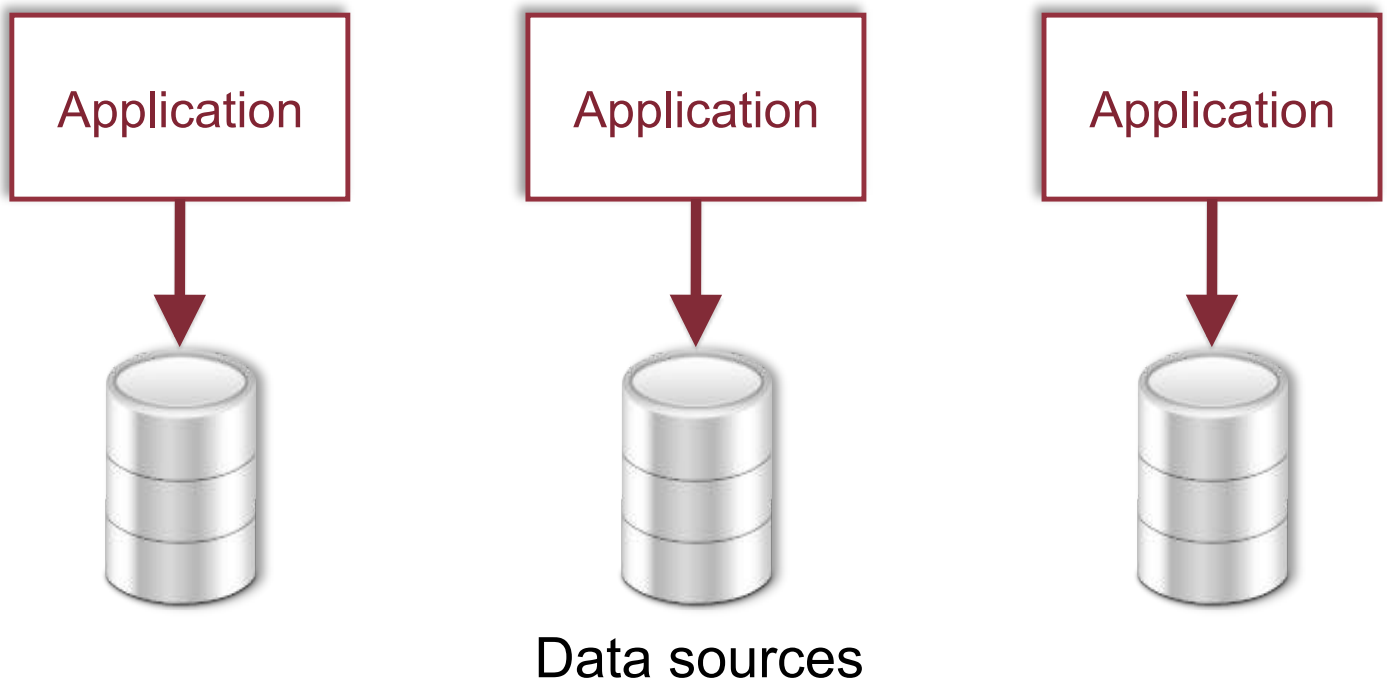
| CUC | TS_START | TS_END | ID_GRUP | FLAG_CP | FLAG_CF | FATTURATO | FLAG_FATT |
|---|---|---|---|---|---|---|---|
| 124589 | 30-lug-2004 | 1-gen-9999 | 92736 | S | N | 195000,00 | N |
| 140904 | 15-mag-2001 | 15-giu-2005 | 35060 | N | N | 230600,00 | N |
| 124589 | 5-mag-2001 | 30-lug-2004 | 92736 | N | S | 195000,00 | S |
| -452901 | 13-mag-2001 | 27-lug-2004 | 92770 | S | N | 392000,00 | N |
| 129008 | 10-mag-2001 | 1-gen-9999 | 62010 | N | S | 247000,00 | S |
| -472900 | 10-mag-2001 | 1-gen-9999 | 62010 | S | N | 0 00 | N |
| 130976 | 7-mag-2001 | 9-lug-2003 | 75680 | | | | |

# Today in many organizations ...

*N means that the FATTURATO field is not valid*

| CUC | TS_START | TS_END | ID_GRUP | FLAG_CP | FLAG_CF | FATTURATO | FLAG_FATT |
|---|---|---|---|---|---|---|---|
| 124589 | 30-lug-2004 | 1-gen-9999 | 92736 | S | N | 195000,00 | N |
| 140904 | 15-mag-2001 | 15-giu-2005 | 35060 | N | N | 230600,00 | N |
| 124589 | 5-mag-2001 | 30-lug-2004 | 92736 | N | S | 195000,00 | S |
| -452901 | 13-mag-2001 | 27-lug-2004 | 92770 | S | N | 392000,00 | N |
| 129008 | 10-mag-2001 | 1-gen-9999 | 62010 | N | S | 247000,00 | S |
| -472900 | 10-mag-2001 | 1-gen-9999 | 62010 | S | N | 0 00 | N |
| 130976 | 7-mag-2001 | 9-lug-2003 | 75680 | | | | |

# Today in many organizations ...




- Distributed, redundant, application-dependent, and mutually incoherent data
- Desperate need of a coherent, conceptual, unified view of data

# Information integration

From [Bernstein & Haas, CACM Sept. 2008]:

- Large enterprises spend a great deal of time and money on information integration (e.g., 40% of information-technology shops' budget).
- Market for information integration software estimated to grow from \$1.87 billion in 2011 to \$2.79 billion in 2015 (+15% per year) [Gartner, 2012]
- Data integration is a large and growing part of software development, computer science, and specific applications settings, such as scientific computing, semantic web, "big data" processing etc..

Basing the information system on a clean, rich and abstract conceptual representation of the data has always been both a goal and a challenge [Mylopoulos et al 1984]

# Ontology-based data management: our program

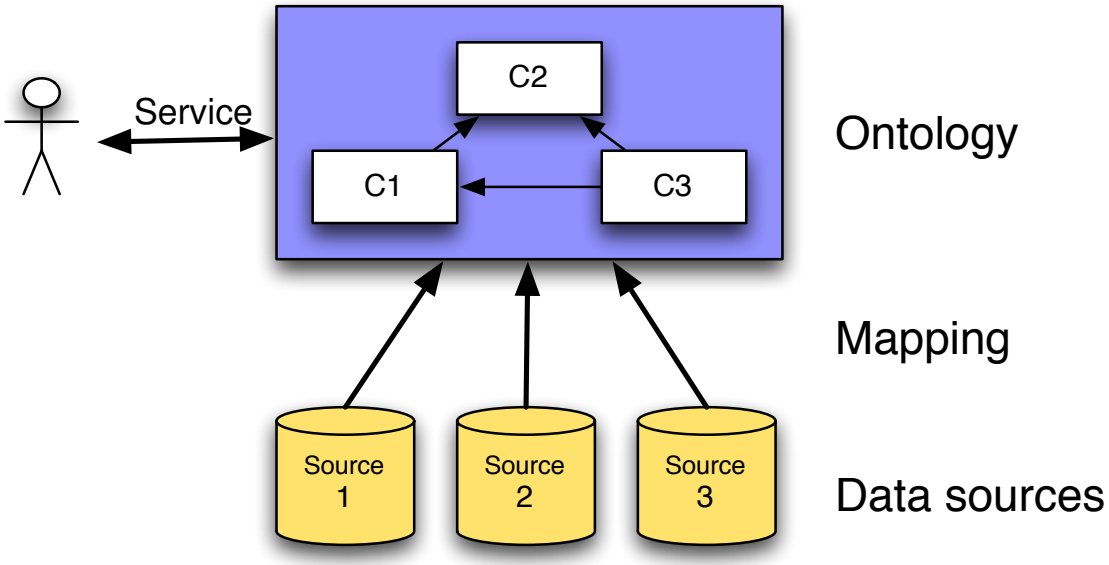
Use Knowledge Representation and Reasoning principles and techniques for a new way of managing data.

- Leave the data where they are
- Build a conceptual specification of the domain of interest, in terms of knowledge structures
- Map such knowledge structures to concrete data sources
- Express all services over the abstract representation
- Automatically translate knowledge services to data services

Experiment techniques in real-world settings

- Logistic (2007)
- Bank (2009)
- Public Administration (2010 – )
- Telecom (2011 – )
- The Optique project (2012 – )

# Ontology-based data management: architecture




Based on three main components:

- **Ontology**, a declarative, ogic-based specification of the domain of interest, used as a unified, conceptual view for clients.
- **Data sources**, representing external, independent, heterogeneous, storage (or, more generally, computational) structures.
- **Mappings**, used to semantically link data at the sources to the ontology.

# Outline

# Outline

# Formal framework of ontology-based data management

An **ontology-based data management system** is a triple $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$, where

- $\mathcal{O}$ is the ontology, expressed as TBox in a Description Logic
- $\mathcal{S}$ is a database with a fixed schema, representing the sources
- $\mathcal{M}$ is a set of GLAV 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 values for $\vec{x}$
- $\Psi(\vec{x})$ is a FOL query over $\mathcal{O}$, whose free variables are from $\vec{x}$.

Note that if $\Psi$ is a conjunctive query (as usually is the case, for instances, when $\mathcal{M}$ is of type "global-as-view"), and we "apply" mapping $\mathcal{M}$ to $\mathcal{S}$, we obtain an ABox (i.e., a set of ground facts in the alphabet of $\mathcal{O}$), denoted by $\mathcal{M}(\mathcal{S})$.

# Semantics

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation for the ontology $\mathcal{O}$.

**Def.: Semantics**

$\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a **model** of $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ if:

- $\mathcal{I}$ is a model of $\mathcal{O}$;
- $\mathcal{I}$ satisfies $\mathcal{M}$ wrt $\mathcal{S}$, i.e., satisfies every assertion in $\mathcal{M}$ wrt $\mathcal{S}$.

**Def.: Mapping satisfaction (sound mappings)**

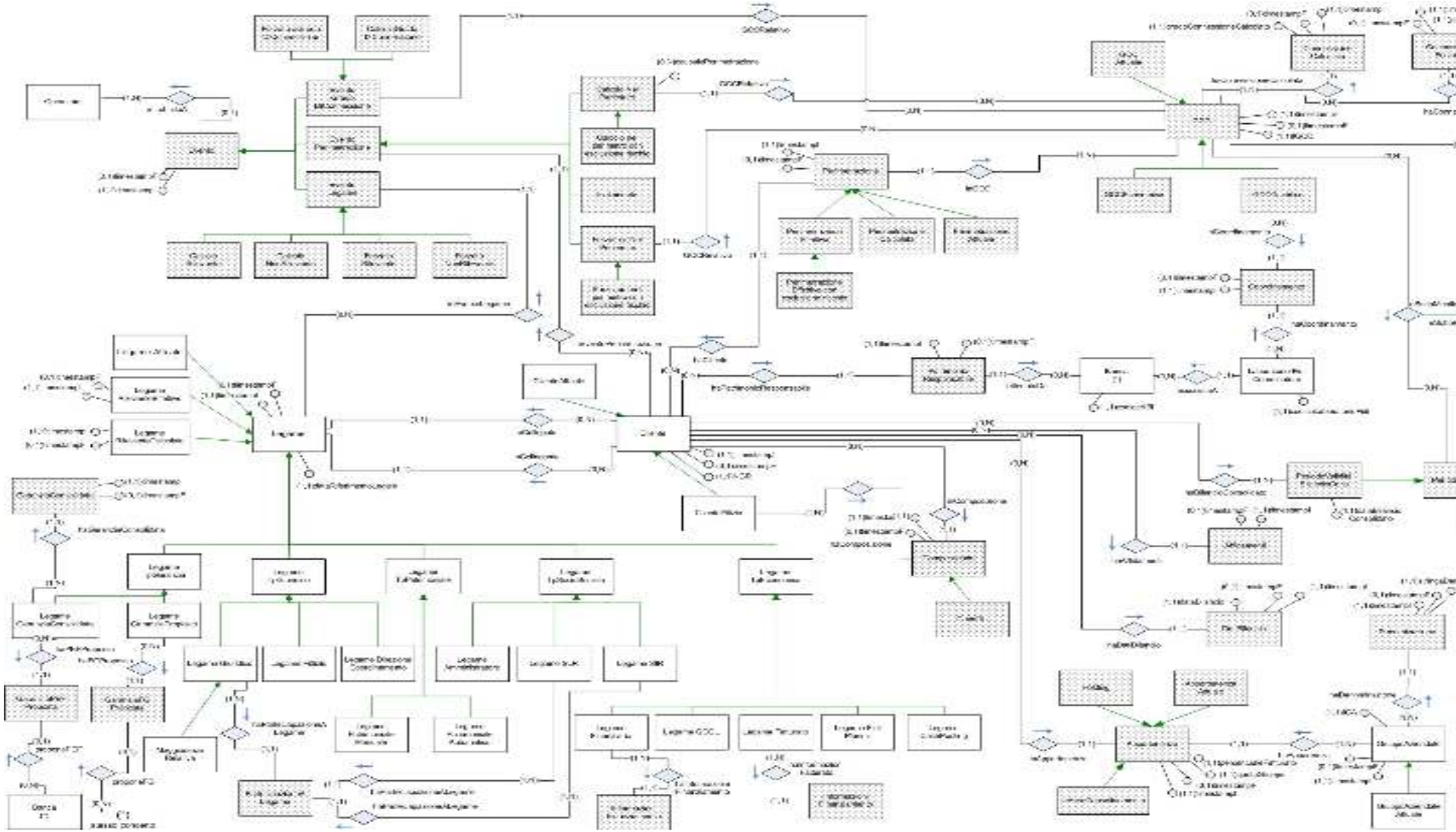
We say that $\mathcal{I}$ satisfies $\Phi(\vec{x}) \rightsquigarrow \Psi(\vec{x})$ wrt a database $\mathcal{S}$, if the sentence

$$\forall \vec{x} \, (\Psi(\vec{x}) \rightarrow \Psi(\vec{x}))$$

is true in $\mathcal{I} \cup \mathcal{S}$.

The set of models of $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ is denoted by $Mod(\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle)$

# Example of OBDM system



# Example of OBDM system (fragment)

Ontology $\mathcal{T}$:

$\mathsf{PublicOrg} \sqsubseteq \mathsf{Organization}$
$\exists \mathsf{worksWith} \sqsubseteq \mathsf{Organization}$
(**funct** name)

$\mathsf{PublicDep} \sqsubseteq \mathsf{PublicOrg}$
$\exists \mathsf{worksWith}^- \sqsubseteq \mathsf{Organization}$
(**funct** address)

Schema $\mathcal{S}$:

```
Dept_MinistryA(dep_id,dep_name)   Works_On(dep_id,proj_name)
Dept_MinistryB(dep_id,dep_addr)   Cooperate(dept1,dept2)
```

Mapping $\mathcal{M}$:

```
SELECT dep_id AS x, dep_name AS y FROM Dept_MinistryA
```
$\rightsquigarrow \{x, y \mid \mathsf{PublicDep}(x) \wedge \mathsf{name}(x, y)\}$

```
SELECT dep_id AS x, dep_addr AS y FROM Dept_MinistryB
```
$\rightsquigarrow \{x, y \mid \mathsf{PublicDep}(x) \wedge \mathsf{address}(x, y)\}$

```
SELECT w1.dep_id as x, w2.dep_id as y, w2.proj_name as z
FROM Works_On w1,Works_On w2,Dept_MinistryA d1,Dept_MinistryA d2
WHERE d1.dep_id=w1.dep_id AND d2.dep_id=w2.dep_id AND
      w1.proj=w2.proj AND w1.dep_id <> w2.dep_id
```
$\rightsquigarrow \{x, y, z \mid \mathsf{worksWith}(x, y) \wedge \mathsf{prjName}(x, z) \wedge \mathsf{prjName}(y, z)\}$

```
SELECT d1.dep_id as x, d2.dep_id as y
FROM Cooperate c, Dept_MinistryB d1, Dept_MinistryB d2
WHERE c.dept1=d1.dep_id AND c.dept2=d2.dep_id
```
$\rightsquigarrow \{x, y \mid \mathsf{worksWith}(x, y)\}$

# Ontology-based data management (OBDM): topics

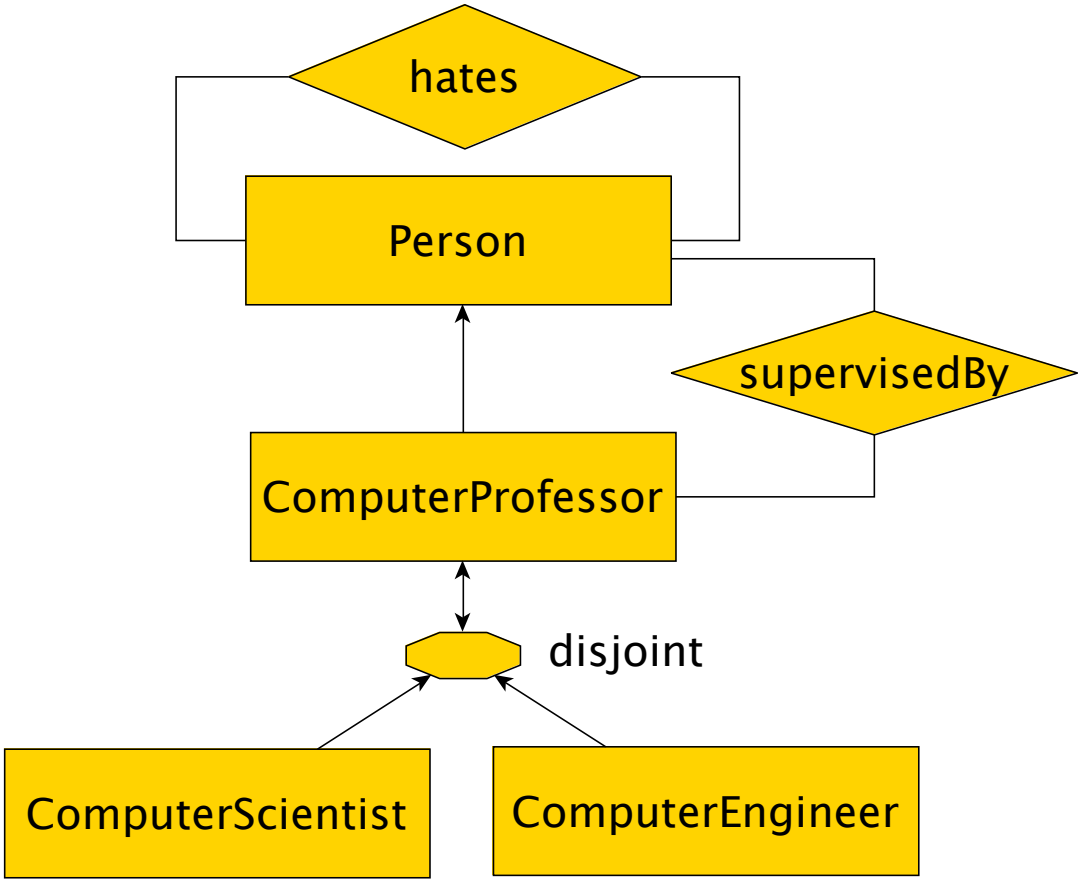
- *Ontology-based data access (OBDA, aka Ontology-based query answering (OBQA))*
- *Ontology-based data integration (OBDI)*
- *Ontology-based data quality assessment (OBDQ)*
- *Ontology-based data publishing/exchange (OBDP/OBDE)*
- *Ontology-based data governance (OBDG)*
- *Ontology-based business intelligence (OBBI)*
- *Ontology-based data design (OBDD)*
- *Ontology-based data update (OBDU)*

General requirements:

- large data collections
- efficiency with respect to size of data (data complexity)

# Outline

# Example of query




$$q(x) \leftarrow \text{supervisedBy}(x, y), \text{ComputerScientist}(y), \text{hates}(y, z), \text{ComputerEngineering}(z)$$

# Semantics of queries: certain answers

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation for the ontology $\mathcal{O}$.

**Def.: Semantics**

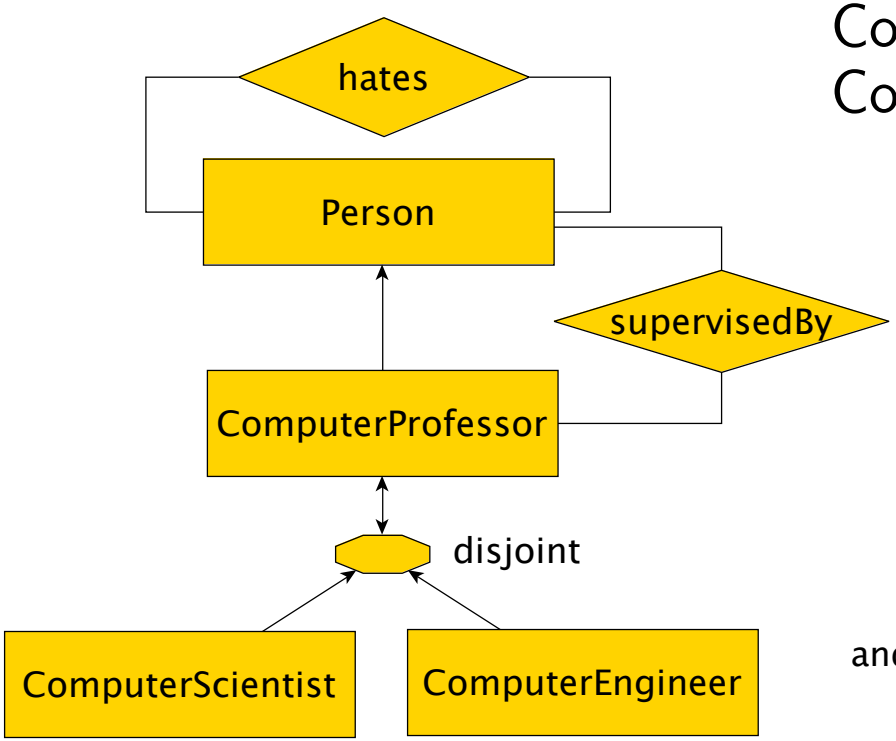
$\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a **model** of $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$, i.e., $\mathcal{I} \in \mathit{Mod}(\langle \mathcal{O}, \mathcal{S}\mathcal{M} \rangle)$ if:

- $\mathcal{I}$ is a model of $\mathcal{O}$;
- $\mathcal{I}$ satisfies $\mathcal{M}$ wrt $\mathcal{S}$, i.e., satisfies every assertion in $\mathcal{M}$ wrt $\mathcal{S}$.

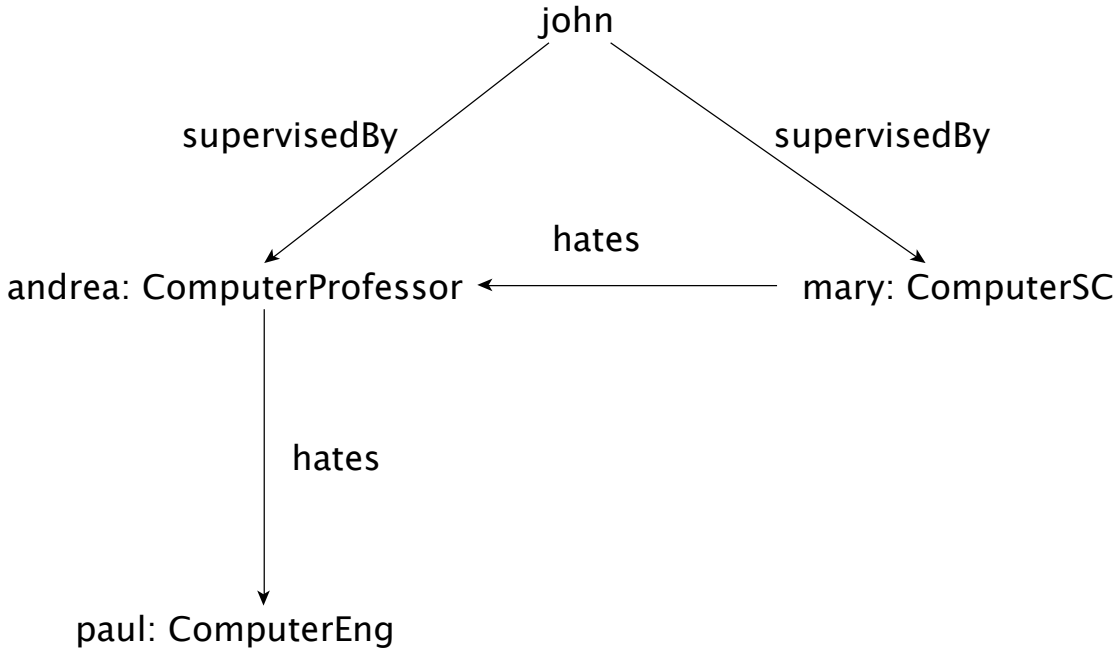
**Def.: The certain answers to a query $q(\vec{x})$ over $\mathcal{K} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$**

$$cert(q, \mathcal{K}) = \{ \vec{c}^{\mathcal{I}} \mid \vec{c}^{\mathcal{I}} \in q^{\mathcal{I}} \text{ for every model } \mathcal{I} \text{ of } \mathcal{K} \}$$

# QA in OBDA – Example(*)

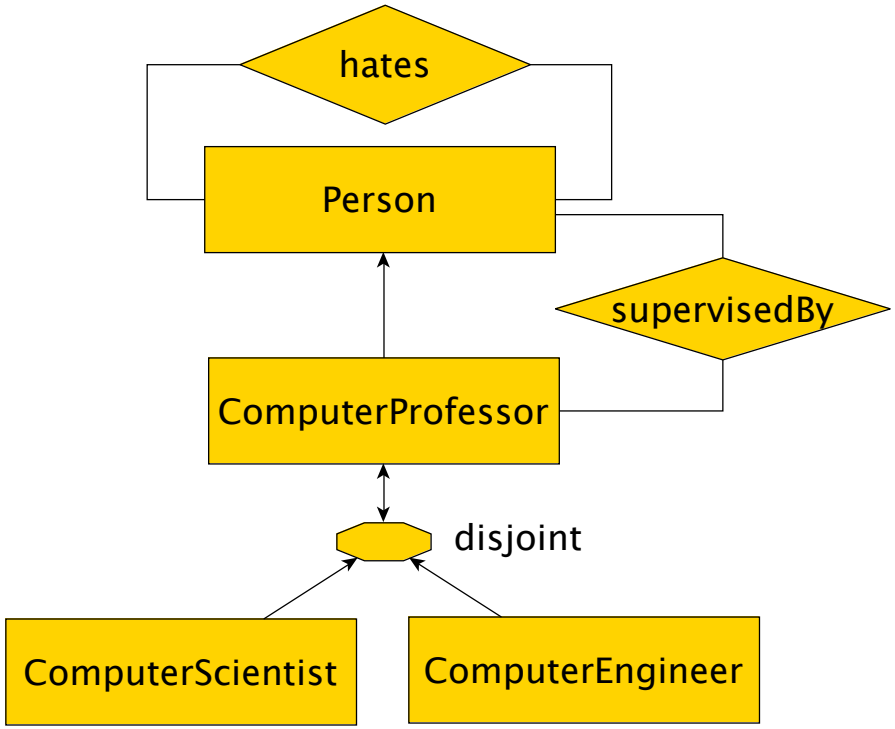


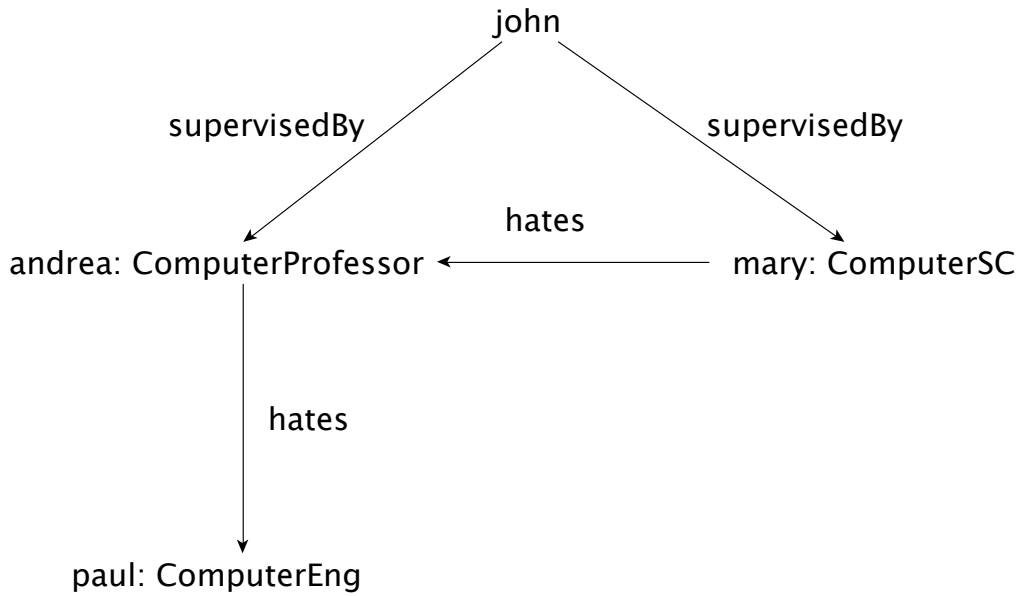

ComputerProfessor is **partitioned into** ComputerScientist and ComputerEngineer.




(*) [Andrea Schaerf 1993]

# QA in OBDA – Example (cont'd)

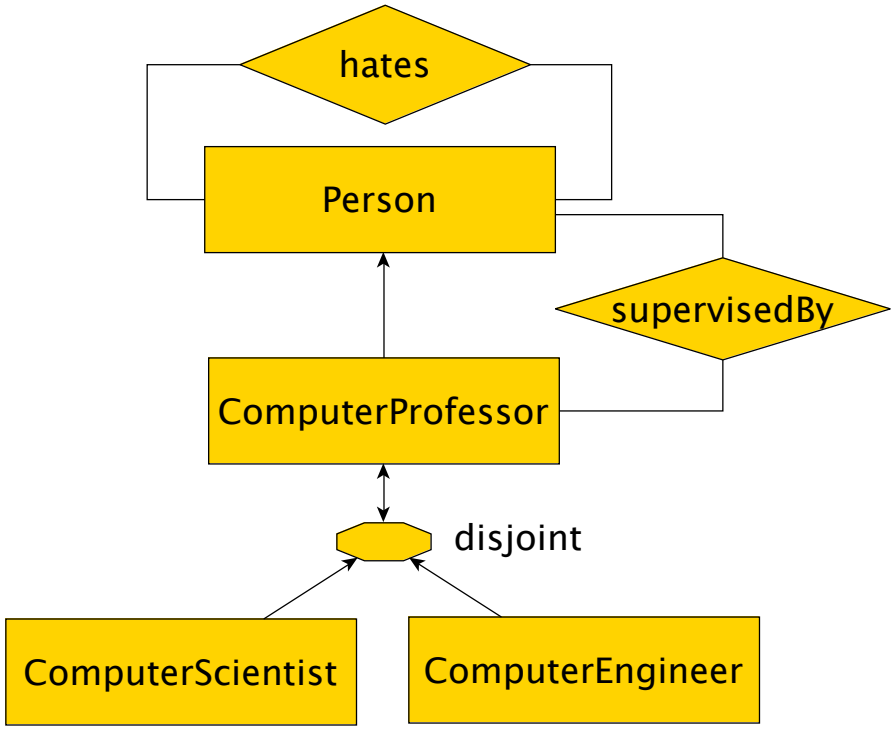


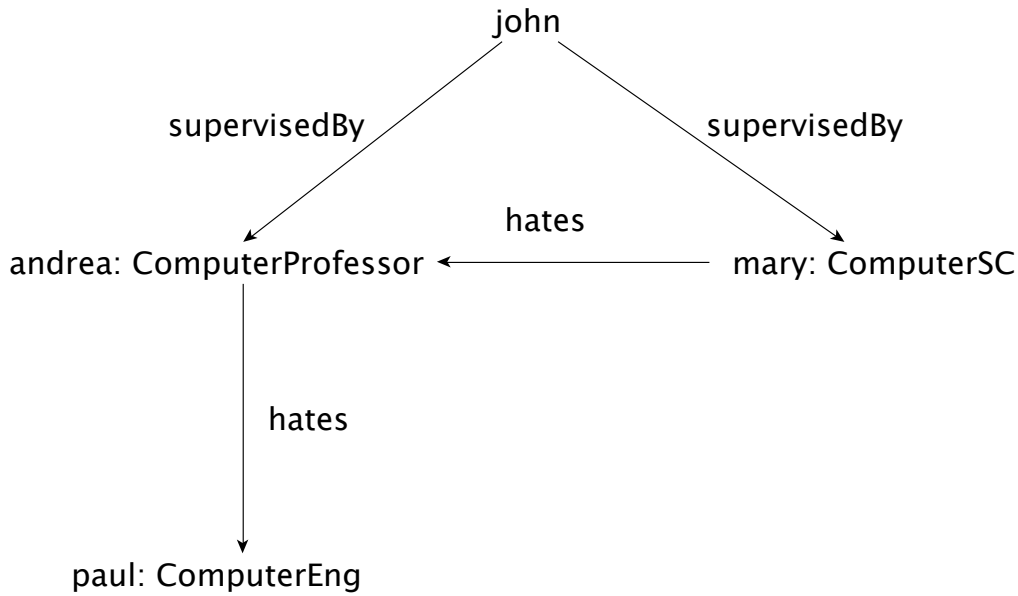




$q(x) \leftarrow \text{supervisedBy}(x, y), \text{ComputerScientist}(y),$
$\text{hates}(y, z), \text{ComputerEngineer}(z)$

Answer: ???

# QA in OBDA – Example (cont'd)




$q(x) \leftarrow \text{supervisedBy}(x, y), \text{ComputerScientist}(y),$
$\text{hates}(y, z), \text{ComputerEngineer}(z)$

Answer: { `john` }

To determine this answer, we need to resort to **reasoning by cases** on the instances.

# Complexity of conjunctive query answering in DLs

| | Combined complexity | Data complexity |
|---|---|---|
| Plain databases | NP-complete | in LogSpace [1] |
| OWL 2 (and less) | ? | coNP-hard [2] |

[1] Going beyond probably means not scaling with the data.
[2] Already for a TBox with a single disjunction (see example above).

**Questions**

- Can we find interesting DLs for which the query answering problem can be solved efficiently (in LogSpace wrt data complexity)?
- If yes, can we leverage relational database technology for query answering in OBDA?

# Complexity of conjunctive query answering in DLs

| | Combined complexity | Data complexity |
|---|---|---|
| Plain databases | NP-complete | in LogSpace (1) |
| OWL 2 (and less) | ? | coNP-hard (2) |

(1) Going beyond probably means not scaling with the data.

(2) Already for a TBox with a single disjunction (see example above).

**Questions**

- Can we find interesting DLs for which the query answering problem can be solved efficiently (in LogSpace wrt data complexity)?
- If yes, can we leverage relational database technology for query answering in OBDA?

# Complexity of conjunctive query answering in DLs

| | Combined complexity | Data complexity |
|---|---|---|
| Plain databases | NP-complete | in LogSpace (1) |
| OWL 2 (and less) | ? | coNP-hard (2) |

(1) Going beyond probably means not scaling with the data.
(2) Already for a TBox with a single disjunction (see example above).

**Questions**

- Can we find interesting DLs for which the query answering problem can be solved efficiently (in LogSpace wrt data complexity)?
- If yes, can we leverage relational database technology for query answering in OBDA?

# Complexity of conjunctive query answering in DLs

| | Combined complexity | Data complexity |
|---|---|---|
| Plain databases | NP-complete | in LogSpace (1) |
| OWL 2 (and less) | ? | coNP-hard (2) |

(1) Going beyond probably means not scaling with the data.
(2) Already for a TBox with a single disjunction (see example above).

**Questions**

- Can we find interesting DLs for which the query answering problem can be solved efficiently (in LogSpace wrt data complexity)?
- If yes, can we leverage relational database technology for query answering in OBDA?

# Semantics of DL-Lite$_{A,id}$

| Construct | Syntax | Example | Semantics |
|---|---|---|---|
| atomic conc. | $A$ | Doctor | $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ |
| exist. restr. | $\exists Q$ | $\exists$child$^-$ | $\{d \mid \exists e.\, (d,e) \in Q^{\mathcal{I}}\}$ |
| at. conc. neg. | $\neg A$ | $\neg$Doctor | $\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$ |
| conc. neg. | $\neg\exists Q$ | $\neg\exists$child | $\Delta^{\mathcal{I}} \setminus (\exists Q)^{\mathcal{I}}$ |
| atomic role | $P$ | child | $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ |
| inverse role | $P^-$ | child$^-$ | $\{(o,o') \mid (o',o) \in P^{\mathcal{I}}\}$ |
| role negation | $\neg Q$ | $\neg$manages | $(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus Q^{\mathcal{I}}$ |
| conc. incl. | $B \sqsubseteq C$ | Father $\sqsubseteq \exists$child | $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ |
| role incl. | $Q \sqsubseteq R$ | hasFather $\sqsubseteq$ child$^-$ | $Q^{\mathcal{I}} \subseteq R^{\mathcal{I}}$ |
| funct. asser. | (**funct** $Q$) | (**funct** succ) | $\forall d,e,e'.(d,e) \in Q^{\mathcal{I}} \wedge (d,e') \in Q^{\mathcal{I}} \rightarrow e = e'$ |
| mem. asser. | $A(c)$ | Father(bob) | $c^{\mathcal{I}} \in A^{\mathcal{I}}$ |
| mem. asser. | $P(c_1, c_2)$ | child(bob, ann) | $(c_1^{\mathcal{I}}, c_2^{\mathcal{I}}) \in P^{\mathcal{I}}$ |

*DL-Lite*$_{A,id}$ (as all DLs of the *DL-Lite* family) adopts the Unique Name Assumption (UNA), i.e., different individuals denote different objects.

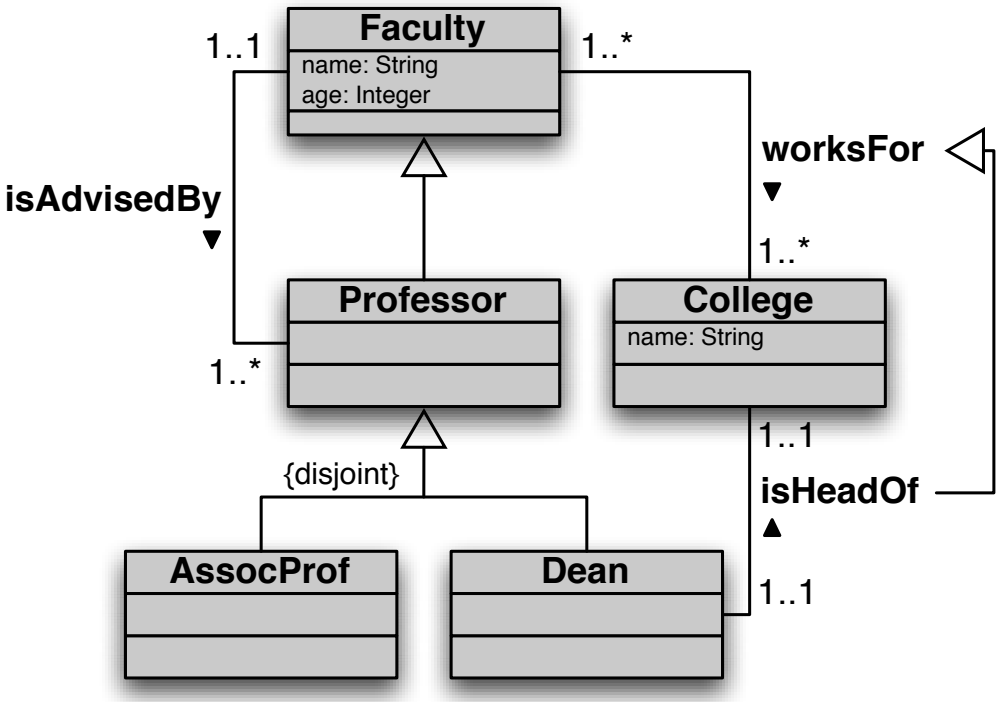
# Capturing basic ontology constructs in DL-Lite$_{A,id}$

| | |
|---|---|
| ISA between classes | $A_1 \sqsubseteq A_2$ |
| Disjointness between classes | $A_1 \sqsubseteq \neg A_2$ |
| Domain and range of properties | $\exists P \sqsubseteq A_1 \quad \exists P^- \sqsubseteq A_2$ |
| Mandatory participation *(min card = 1)* | $A_1 \sqsubseteq \exists P \quad A_2 \sqsubseteq \exists P^-$ |
| Functionality of relations *(max card = 1)* | (**funct** $P$) (**funct** $P^-$) |
| ISA between properties | $Q_1 \sqsubseteq Q_2$ |
| Disjointness between properties | $Q_1 \sqsubseteq \neg Q_2$ |

*Note 1:* *DL-Lite*$_{A,id}$ cannot capture completeness of a hierarchy. This would require **disjunction** (i.e., **OR**).

*Note 2:* *DL-Lite*$_{A,id}$ can be extended to capture also **min cardinality constraints** ($A \sqsubseteq \leq\ n\ Q$), **max cardinality constraints** ($A \sqsubseteq \geq\ n\ Q$) [Artale et al, JAIR 2009], $n$-**ary** relations, **identification assertions**, and **denial assertions** (not considered here for simplicity).

# Example of DL-Lite$_{A,id}$ ontology




$$
\begin{aligned}
\text{Professor} &\sqsubseteq \text{Faculty}\\
\text{AssocProf} &\sqsubseteq \text{Professor}\\
\text{Dean} &\sqsubseteq \text{Professor}\\
\text{AssocProf} &\sqsubseteq \neg\text{Dean}
\end{aligned}
$$

$$
\begin{aligned}
\text{Faculty} &\sqsubseteq \exists \mathit{age}\\
\exists \mathit{age}^- &\sqsubseteq \texttt{xsd:integer}\\
&(\textbf{funct}\ \mathit{age})
\end{aligned}
$$

$$
\begin{aligned}
\exists\text{worksFor} &\sqsubseteq \text{Faculty}\\
\exists\text{worksFor}^- &\sqsubseteq \text{College}\\
\text{Faculty} &\sqsubseteq \exists\text{worksFor}\\
\text{College} &\sqsubseteq \exists\text{worksFor}^-
\end{aligned}
$$

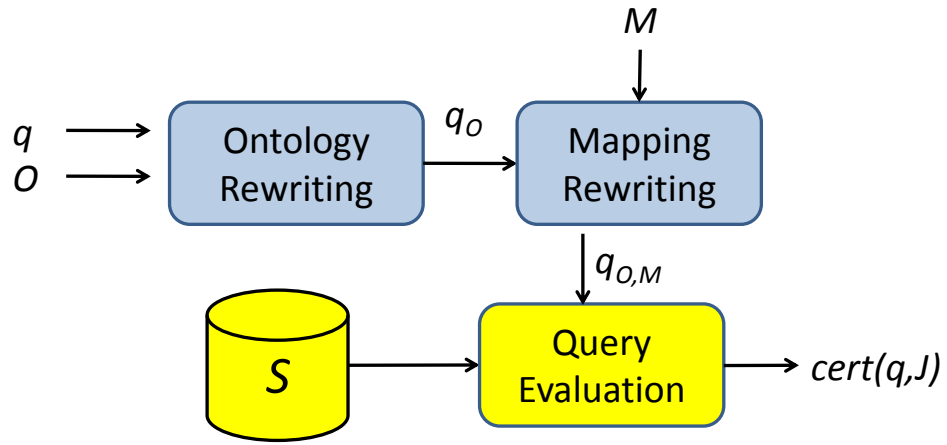
$$
\begin{aligned}
\exists\text{isHeadOf} &\sqsubseteq \text{Dean}\\
\exists\text{isHeadOf}^- &\sqsubseteq \text{College}\\
\text{Dean} &\sqsubseteq \exists\text{isHeadOf}\\
\text{College} &\sqsubseteq \exists\text{isHeadOf}^-\\
\text{isHeadOf} &\sqsubseteq \text{worksFor}\\
&(\textbf{funct}\ \text{isHeadOf})\\
&(\textbf{funct}\ \text{isHeadOf}^-)
\end{aligned}
$$

.

# Query answering by rewriting in OBDA

Given (U)CQ $q$, $\mathcal{J} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{M}$ is of type "global-as-view":

1. **Ontology rewriting**: rewrite $q$ into the perfect ontology rewriting $q_{\mathcal{O}}$ w.r.t. $\mathcal{O}$, which is a query (a UCQ, under our assumptions) over $\mathcal{O}$ such that
$$cert(q, \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle) = cert(q_{\mathcal{O}}, \langle \emptyset, \mathcal{S}, \mathcal{M} \rangle)$$
2. **Mapping rewriting**: rewrite $q_{\mathcal{O}}$ into the perfect mapping rewriting $q_{\mathcal{O},\mathcal{M}}$ w.r.t. $\mathcal{M}$, which is a query over $\mathcal{S}$ such that
$$cert(q_{\mathcal{O}}, \langle \emptyset, \mathcal{S}, \mathcal{M} \rangle) = cert(q_{\mathcal{O}}, \langle \emptyset, \mathcal{S}, \emptyset \rangle) = q^{\mathcal{S}}_{\mathcal{O},\mathcal{M}}$$
3. **Evaluation**: compute $q^{\mathcal{S}}_{\mathcal{O},\mathcal{M}}$ (globally, $q_{\mathcal{O},\mathcal{M}}$ is called the perfect rewriting of $q$ under $\mathcal{J}$)



# Query answering in DL-Lite$_{A,id}$: Example

**TBox:** $\mathsf{Professor} \sqsubseteq \exists \mathsf{teaches}$
$\exists \mathsf{teaches}^- \sqsubseteq \mathsf{Course}$

**Query:** $\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y), \mathsf{Course}(y)$

**Perfect Rewriting:** $\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y), \mathsf{Course}(y)$
$\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y), \mathsf{teaches}(z, y)$
$\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, z)$
$\mathsf{q}(x) \leftarrow \mathsf{Professor}(x)$

$\mathcal{M}(\mathcal{S})$: $\mathsf{teaches}(\mathtt{John}, \mathtt{databases})$
$\mathsf{Professor}(\mathtt{Mary})$

It is easy to see that the evaluation of $r_{q,\mathcal{O}}$ over $\mathcal{M}(\mathcal{S})$ in this case produces the set $\{\mathtt{John}, \mathtt{Mary}\}$.

# Outline

# Example: an inconsistent DL-Lite ontology

## $\mathcal{O}$

$\text{RedWine} \sqsubseteq \text{Wine}$ $\quad$ $\text{WhiteWine} \sqsubseteq \text{Wine}$

$\text{RedWine} \sqsubseteq \neg\ \text{WhiteWIne}$ $\quad$ $\text{Wine} \sqsubseteq \neg\ \text{Beer}$

$\text{Wine} \sqsubseteq \exists \text{producedBy}$ $\quad$ $\exists \text{producedBy} \sqsubseteq \text{Wine}$

$\text{Wine} \sqsubseteq \neg\ \text{Winery}$ $\quad$ $\text{Beer} \sqsubseteq \neg\ \text{Winery}$

$\exists \text{producedBy}^{-} \sqsubseteq \text{Winery}$ $\quad$ $(\mathit{funct}\ \text{producedBy})$

## $\mathcal{M}$

$\text{R1(x,y,'white')} \rightsquigarrow \text{WhiteWine(x)}$ $\quad$ $\text{R1(x,y,'red')} \rightsquigarrow \text{RedWine(x)}$

$\text{R2(x,y)} \rightsquigarrow \text{Beer(x)}$ $\quad$ $\text{R1(x,y,z)} \vee \text{R2(x,y)} \rightsquigarrow \text{producedBy(x,y)}$

## $\mathcal{S}$

R1(grechetto,p1,'white') $\quad$ R1(grechetto,p1,'red')

R2(guinnes,p2) $\quad$ R1(falanghina,p1,'white')

# The problem

One popular approach to dealing with inconsistency in data management is data cleaning

However, data cleaning is impossible in virtual data integration, and, even with data cleaning, inconsistencies may remain, and we would like our system to provide meaningful answers to queries.

The problem is that query answering based on classical logic becomes meaningless in the presence of inconsistency (ex falso quodlibet)

**Question**

How to handle classically-inconsistent OBDM systems in a more meaningful way?

## Inconsistent-tolerant semantics

The semantics we propose [Lembo et al, RR 2010] for querying inconsistent OBDM systems is based on the following principles:

- We assume that $\mathcal{O}$ and $\mathcal{M}$ are always consistent (this is true if $\mathcal{O}$ is expressed in *DL-Lite*$_{\mathcal{A},id}$)
- Inconsistencies are caused by the interaction between the data at $\mathcal{S}$ and the other components of the system, i.e., between $\mathcal{M}(\mathcal{S})$ and $\mathcal{O}$
- We resort to the notion of *repair* [Arenas, Bertossi, Chomicki, PODS 1999]. Intuitively, a repair for $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ is an ontology $\langle \mathcal{O}, \mathcal{A} \rangle$ that is consistent, and "minimally" differs from $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$.

  See [Leopoldo Bertossi, "Database Repairing and Consistent Query Answering", *Synthesis Lectures on Data Management*, Vol. 3, No. 5, Morgan and Claypool].

# Inconsistent-tolerant semantics

What does it mean for $\mathcal{A}$ to be "minimally different" from $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$? We base this concept on the notion of symmetric difference.

We write $S_1 \oplus S_2$ to denote the symmetric difference between $S_1$ and $S_2$, i.e.,

$$S_1 \oplus S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$$

## Definition (Repair)

Let $\mathcal{K} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ be an OBDM system. A **repair** of $\mathcal{K}$ is an ABox $\mathcal{A}$ such that:

1. $Mod(\langle \mathcal{O}, \mathcal{A} \rangle) \neq \emptyset$,
2. no set of facts $\mathcal{A}'$ exists such that
   - $Mod(\langle \mathcal{O}, \mathcal{A}' \rangle) \neq \emptyset$,
   - $\mathcal{A}' \oplus \mathcal{M}(\mathcal{S}) \subset \mathcal{A} \oplus \mathcal{M}(\mathcal{S})$

# Example: Repairs

$Rep_1$

{WhiteWine(grechetto), Beer(guinnes), WhiteWine(falanghina)}

$Rep_2$

{RedWine(grechetto), Beer(guinnes), WhiteWine(falanghina)}

$Rep_3$

{WhiteWine(grechetto), producedBy(guinnes, p2),
WhiteWine(falanghina)}

$Rep_4$

{RedWine(grechetto), producedBy(guinnes, p2),
WhiteWine(falanghina)}

# Reasoning wih all repairs: the AR semantics

Problems:

- Many repairs in general
- What is the complexity of reasoning about all such repairs?

**Theorem**

Let $\mathcal{K} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ be an OBDM system, and let $\alpha$ be a ground atom. Deciding whether $\alpha$ is logically implied by every repair of $\mathcal{K}$ is coNP-complete with respect to data complexity.

## When in doubt, throw it out: the IAR semantics

Other intractability results of the AR semantics, even for simpler languages (e.g., [Bienvenu, DL 2012])

**Idea: The IAR semantics**

Consider the "intersection of all repairs", and consider the set of models of such intersection as the semantics of the system (When in Doubt, Throw It Out).

Note that the IAR semantics is an approximation of the AR semantics

# Inconsistent-tolerant query answering

Two possible methods for answering queries posed to $\mathcal{K} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ according to the inconsistency-tolerant semantics:

- Compute the intersection $\mathcal{A}$ of all repairs of $\mathcal{K}$, and then compute $\vec{t}$ such that $\langle \mathcal{O}, \mathcal{A} \rangle \models q(\vec{t})$
- Rewrite the query $q$ into $q_1$ in such a way that, for all $\vec{t}$, we have that $\mathcal{K} \models_{IAR} q(\vec{t})$ is equivalent to $\vec{t} \in q_1^{\mathcal{S}}$. Then, evaluate $q_1$ over $\mathcal{S}$.

We have devised a rewriting technique which encodes a UCQ $q$ into a FOL query $q_1$ which, evaluated against the original $\mathcal{S}$ retrieves only the certain answers of $q$ w.r.t the IAR semantics [Lembo et al, DL 2012].

# Example

Let us consider the CQ

$$q = \exists x.\mathsf{RedWine}(x)$$

We have that the rewriting is

$$\exists x.\mathsf{RedWine}(x) \land \neg\mathsf{WhiteWine}(x) \land \neg\mathsf{Beer}(x) \land \neg\mathsf{Winery}(x) \land \neg(\exists y.\mathsf{producedBy}(x, y) \land x \neq y)$$

# Complexity

**Theorem**

Let $Q$ be a UCQ over $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$. Deciding whether $\vec{t} \in cert_{IAR}(Q, \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle)$ is in $AC^0$ in data complexity.

| problem | $AR$-semantics | $IAR$-semantics |
|---|---|---|
| instance checking | coNP-complete | in $AC_0$ |
| UCQ answering | coNP-complete | in $AC_0$ |

# Outline

# Ontology-based data integration

- We have to deal with heterogeneous and distributed sources
- Data federation may help, but it is open whether it scales up
- Even more challenges with Big Data
- Semantic heterogeneity is also a problem (see next slides)

# Dealing with semantic heterogeneity: mapping intensional knowledge

Source $\mathcal{S}$:

`T-CarTypes`

| Code | Name |
|---|---|
| T1 | Coupé |
| T2 | SUV |
| T3 | Sedan |
| T4 | Estate |

`T-Cars`

| CarCode | CarType | EngineSize | BreakPower | Color | TopSpeed |
|---|---|---|---|---|---|
| AB111 | T1 | 2000 | 200 | Silver | 260 |
| AF333 | T2 | 3000 | 300 | Black | 200 |
| BR444 | T2 | 4000 | 400 | Grey | 220 |
| AC222 | T4 | 2000 | 125 | Dark Blue | 180 |
| BN555 | T3 | 1000 | 75 | Light Blue | 180 |
| BP666 | T1 | 3000 | 600 | Red | 240 |

# Example

Ontology $\mathcal{O}$: $\texttt{Car} \sqsubseteq \texttt{Vehicle}$

Source $\mathcal{S}$:

T-CarTypes

| Code | Name |
|---|---|
| T1 | Coupé |
| T2 | SUV |
| T3 | Sedan |
| T4 | Estate |

T-Cars

| CarCode | CarType | EngineSize | BreakPower | Color | TopSpeed |
|---|---|---|---|---|---|
| AB111 | T1 | 2000 | 200 | Silver | 260 |
| AF333 | T2 | 3000 | 300 | Black | 200 |
| BR444 | T2 | 4000 | 400 | Grey | 220 |
| AC222 | T4 | 2000 | 125 | Dark Blue | 180 |
| BN555 | T3 | 1000 | 75 | Light Blue | 180 |
| BP666 | T1 | 3000 | 600 | Red | 240 |

Mapping $\mathcal{M}$:

- $\{y \mid \texttt{T-CarTypes}(x, y)\} \rightsquigarrow y \sqsubseteq \texttt{Car}$
- $\{(x, v, z) \mid \texttt{T-Cars}(x, y, t, u, v, q) \wedge \texttt{T-CarTypes}(y, z)\} \rightsquigarrow z(x)$
- $\{(x, y) \mid \texttt{T-CarTypes}(z_1, x) \wedge \texttt{T-CarTypes}(z_2, y) \wedge x \neq y\} \rightsquigarrow x \sqsubseteq \neg y$

The ontology $\mathcal{O}$ is enriched through $\mathcal{M}$ and $\mathcal{S}$.

# Higher-order Description Logics

Technically, we need higher-order logic (e.g., $Hi(\textbf{DL-Lite}_{\mathcal{R}})$ [De Giacomo et al, AAAI 2011, Di Pinto et al, AAAI 2012])

Consequently, Higher-order queries become natural, e.g.:

**Example**

Interesting queries that can be posed to $\langle \mathcal{S}, \mathcal{M} \rangle$ exploit the higher-order nature of the system:

- Return all the instances of *Car*, each one with its own type:
  $q(x, y) \leftarrow y(x), \texttt{Car}(x)$
- Return all the concepts which car *AB111* is an instance of:
  $q(x) \leftarrow x(AB111)$

# Ontology-based data quality assessment

- Static analysis techniques
  Quality of schema: how well the data sources are suited to store data concerning the instances of the ontology?
- Run-time techniques
  Quality of data: how much tha data conform to the ontology?

In both cases, the ontology provides the yardstick to define "quality" parameters.

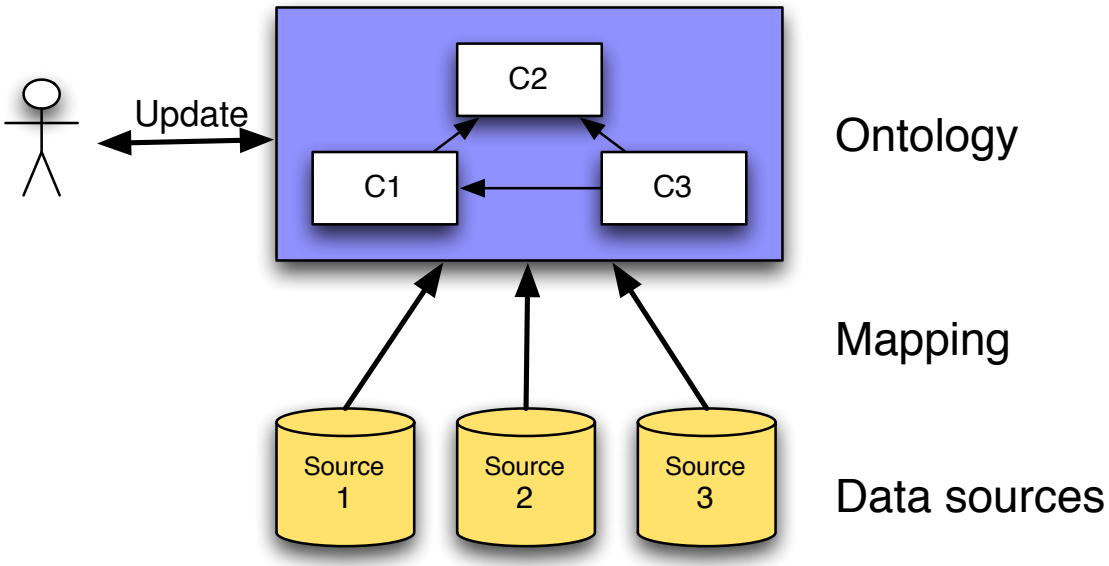
# Ontology-based data publishing/exchange

- Which data to open?
- How to structure the data to publish?
- Ontology-based privacy-aware access and publishing
  based on the specification of positive and negative views associated to the users, the system can answer queries and publishe data by making sure that no private data are disclosed (neither explicitely, nor implicitely)
- Crucial notion: **views over the ontology**

# Ontology-based data design

- **Inverse process** wrt the one described so far: from the ontology to the data sources
- Need of new methodologies
- Mappings are also a product of the design process

# Ontology-based update: challenges




- Which is a reasonable semantics for updates expressed over an ontology?
- How to "push" updates espressed over the ontology to updates over the sources?

# The problem of multiple results

## Example

$\mathcal{O}: \quad \exists R.C \sqsubseteq B, \quad B \sqsubseteq \neg D, \quad B \sqsubseteq E$

$\mathcal{A}: \quad \{R(a_1, a_2), C(a_2)\}, \quad$ with

$\mathsf{cl}_{\mathcal{O}}(\mathcal{A}) = \{R(a_1, a_2), C(a_2), B(a_1), E(a_1)\}$

insert $F = \{D(a_1)\}$

# The problem of multiple results

## Example

$\mathcal{O}: \quad \exists R.C \sqsubseteq B, \quad B \sqsubseteq \neg D, \quad B \sqsubseteq E$

$\mathcal{A}: \quad \{R(a_1, a_2), C(a_2)\}, \quad$ with

$\mathsf{cl}_{\mathcal{O}}(\mathcal{A}) = \{R(a_1, a_2), C(a_2), B(a_1), E(a_1)\}$

insert $F = \{D(a_1)\}$

# The problem of multiple results

## Example

$\mathcal{O}: \quad \exists R.C \sqsubseteq B, \quad B \sqsubseteq \neg D, \quad B \sqsubseteq E$

$\mathcal{A}: \quad \{R(a_1, a_2), C(a_2)\}, \quad$ with

$\mathsf{cl}_{\mathcal{O}}(\mathcal{A}) = \{R(a_1, a_2), C(a_2), B(a_1), E(a_1)\}$

insert $F = \{D(a_1)\}$

$\mathcal{A}_1 = \{R(a_1, a_2), D(a_1), E(a_1)\}, \quad$ with $\mathsf{cl}_{\mathcal{O}}(\mathcal{A}_1) = \mathcal{A}_1$

$\mathcal{A}_2 = \{C(a_2), D(a_1), E(a_1)\}, \quad$ with $\mathsf{cl}_{\mathcal{O}}(\mathcal{A}_2) = \mathcal{A}_2$

# The problem of multiple results

## Example

$\mathcal{O}: \quad \exists R.C \sqsubseteq B, \quad B \sqsubseteq \neg D, \quad B \sqsubseteq E$

$\mathcal{A}: \quad \{R(a_1, a_2), C(a_2)\}, \quad$ with

$\mathsf{cl}_{\mathcal{O}}(\mathcal{A}) = \{R(a_1, a_2), C(a_2), B(a_1), E(a_1)\}$

insert $F = \{D(a_1)\}$

$\mathcal{A}_1 = \{R(a_1, a_2), D(a_1), E(a_1)\}, \quad$ with $\mathsf{cl}_{\mathcal{O}}(\mathcal{A}_1) = \mathcal{A}_1$

$\mathcal{A}_2 = \{C(a_2), D(a_1), E(a_1)\}, \quad$ with $\mathsf{cl}_{\mathcal{O}}(\mathcal{A}_2) = \mathcal{A}_2$

Several approaches to deal with this problem are possible, including:

- Keep all of them, so that the result is a set of ABoxes [Fagin, Ullman, Vardi 1983]
- Choose one ABox nondeterministically [Calvanese, Kharlamov, Nutt, Zheleznyakov, 2010]
- Adopt a "When In Doubt Throw It Out" (WIDTIO) approach

# The result of inserting and deleting [L. and Savo, DL 2011]

**Definition**

Let $\mathcal{U}$ be the set of all ABoxes accomplishing the insertion (deletion) of $F$ into (from) $\langle \mathcal{O}, \mathcal{A} \rangle$ minimally, and let $\mathcal{A}'$ be an ABox. Then, $\langle \mathcal{O}, \mathcal{A}' \rangle$ is the result of changing $\langle \mathcal{O}, \mathcal{A} \rangle$ with the insertion (deletion) of $F$ if

- $\mathcal{U}$ is empty, and $\langle \mathcal{O}, \mathsf{cl}_{\mathcal{O}}(\mathcal{A}') \rangle = \langle \mathcal{O}, \mathsf{cl}_{\mathcal{O}}(\mathcal{A}) \rangle$, or
- $\mathcal{U}$ is nonempty, and $\langle \mathcal{O}, \mathsf{cl}_{\mathcal{O}}(\mathcal{A}') \rangle = \langle \mathcal{O}, \bigcap \{ \mathsf{cl}_{\mathcal{O}}(\mathcal{A}_i) \mid \mathcal{A}_i \in U \} \rangle$.

- Up to logical equivalence, the result of changing $\langle \mathcal{O}, \mathcal{A} \rangle$ with the insertion or the deletion of $F$ is unique.

# Outline

# Many challenges

- Many challenges
  - Still a lot to do for improving efficiency of query answering (hot research topic)
  - Synergy with data federation
  - Pushing the updates to the data sources
  - Natural language interface for querying
  - Desperate need of effective tools for modeling both the ontology and the mapping, and for supporting their evolution
  - Add processes/services to the picture
- On-going work
  - Three big industrial experimentations
  - Optique:European project on OBDA
  - ACM SIGMOD blog: **wp.sigmod.org**
    this month hosts a post of mine on OBDA, where other on-going experiences are mentioned architec