Methods and Tools for Developing Ontology-Based Data Management Solutions

Source Analysis and Mapping Development

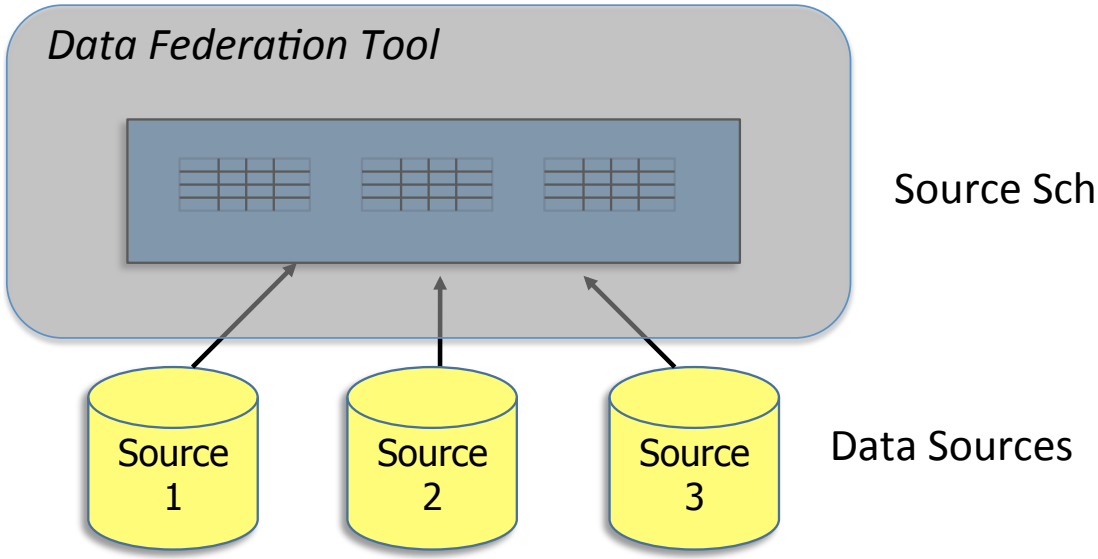
Domenico Lembo, Valerio Santarelli, Domenico Fabio Savo

SAPIENZA
UNIVERSITÀ DI ROMA

SEMANTICS 2018
Vienna, Austria, September 11, 2018

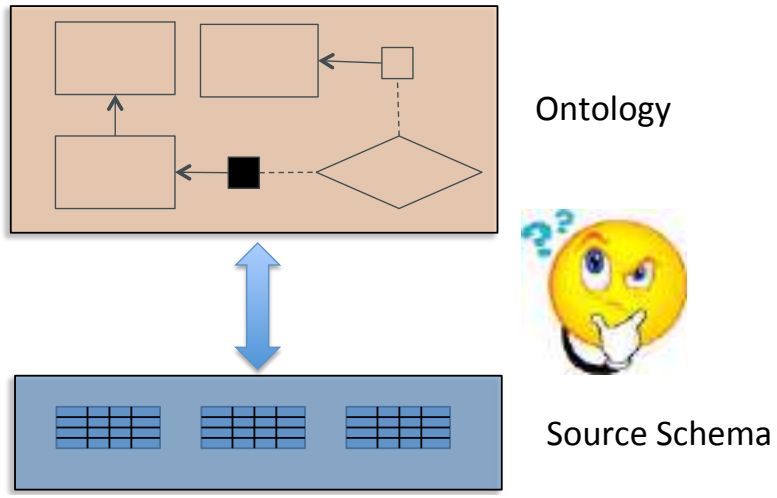
Source Schema

- In OBDA, data reside in autonomous data sources, typically pre-existing the ontology.
- Data sources are seen as a unique relational database that constitutes the Source Schema component of an OBDA specification.
- Off-the-shelf Data Federation/Virtualization tools can be used to wrap multiple, possibly non-relational, sources, and present them as they were structured according to a single relational schema.



The problem

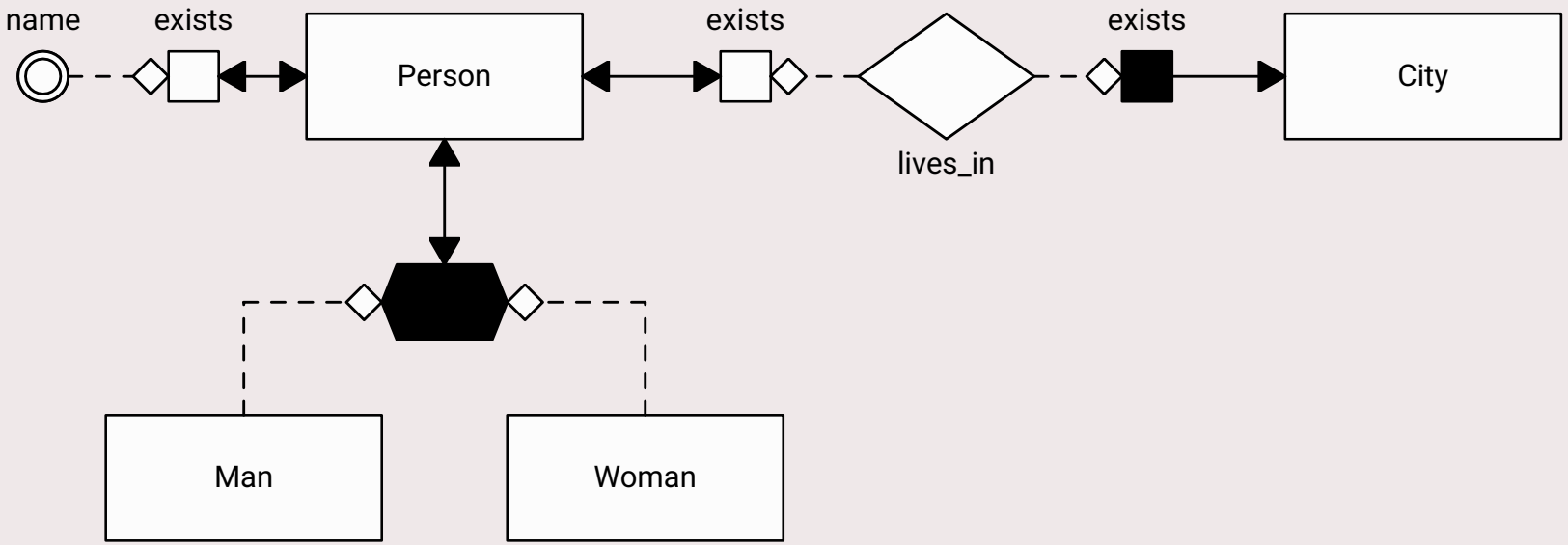
- Problem: How do we relate the ontology with the source schema?



- Main Design Challenges
  - Different representation languages, i.e., a DL TBox vs. a relational schema.
  - Different modeling: Data sources serve applications, and thus typically their structure does not directly reflect the abstract conceptualization given by the ontology,

Example

Ontology



Source Schema

RomanCitizens		
SSN	Name	Gender

Mapping relational schemas to ontologies: impedance mismatch

- In OBDA the correspondence between the ontology and the source schema is specified through GAV Mappings, a form of mapping commonly used in data integration [Len02].
- Differently from traditional Data Integration, in OBDA two different data models are used (relational databases vs. ontologies)
  - In relational databases, information is represented in forms of tuples of values.
  - In ontologies information is represented using both individuals (denoting objects of the domain) and values (as fillers of individuals's attributes) ...
- Solution: We need constructors to create individuals of the ontology out of tuples of values in the database.

Note: from a formal point of view, such constructors can be simply seen as Skolem functions!

Mapping in OBDA: formal definition

A Mapping in OBDA is a set of assertions having the following forms

$$\begin{aligned} \Phi(\vec{x}) &\rightsquigarrow C(\mathsf{f}(\vec{x})) \\ \Phi(\vec{x}) &\rightsquigarrow R(\mathsf{f}_1(\vec{x}_1), \mathsf{f}_2(\vec{x}_2)) \\ \Phi(\vec{x}) &\rightsquigarrow A(\mathsf{f}(\vec{x}_1), \vec{x}_2) \end{aligned}$$

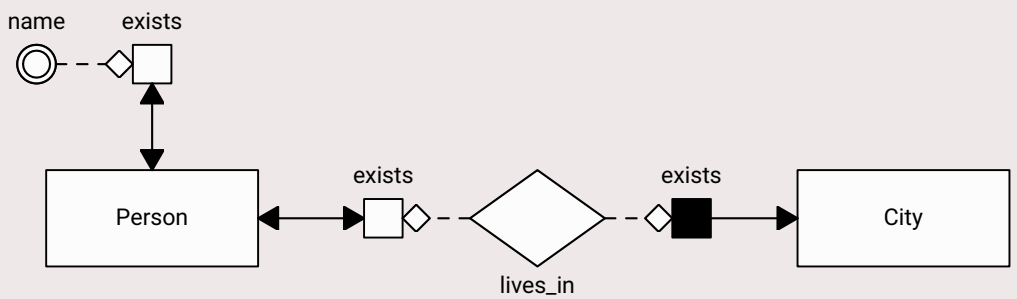
where:

- $\Phi(\vec{x})$ is an arbitrary SQL query over the source schema, returning attributes $\vec{x}$
- C is a named class (atomic concept), R is named object property (atomic role), and A is a named data property (attribute)
- f, f_1, f_2 are function symbols
- $\vec{x}_1$ and $\vec{x}_2$, possibly overlapping, contains only variables in $\vec{x}$

The left-hand side of a mapping assertion is called body, whereas the right-hand side is called head.

Example

Ontology



Source Data

RomanCitizens		
SSN	*Name*	*Gender*
1234	Marco	M
...	...	...

Mapping

```
SELECT SSN                          ⇝  Person(pers(SSN))
FROM RomanCitizens

SELECT SSN, 'Rome' AS City          ⇝  lives_in(pers(SSN),ct(City))
FROM RomanCitizens

SELECT SSN, Name                    ⇝  name(pers(SSN),Name)
FROM RomanCitizens
```

Semantics

Def.: Semantics of mapping

Given an OBDA specification $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$, and a DB instance D for $\mathcal{S}$.
We say that a FOL interpretation $\mathcal{I}$ satisfies $\Phi(\vec{x}) \rightsquigarrow C(\mathsf{f}(\vec{x}))$ wrt D if

$$\forall \vec{t} \in \mathsf{eval}(\Phi(\vec{x}), D),\ \mathcal{I} \models C(\mathsf{f}(\vec{t}))$$

Analogously for the other forms of mapping assertions.

$\mathcal{I}$ satisfies $\mathcal{M}$ wrt D if $\mathcal{I}$ satisfies all assertions in $\mathcal{M}$ wrt D.

Def.: Semantics of OBDA specification

$\mathcal{I}$ is a model of $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ wrt D if:

- $\mathcal{I}$ is a model of $\mathcal{O}$
- $\mathcal{I}$ satisfies $\mathcal{M}$ wrt D

R2RML

- R2RML is a W3C recommendation for expressing customized mappings from relational databases to RDF datasets [DSC12].
- Roughly, in R2RM, the mapping is a set of TripleMaps, each containing an SQL query and a set of RDF triples that share the same subject. Each triple assigns an object to a class or associates it with a property.
- In the triples, IRI templates using variables from the SQL query denote resources. They play the same role as constructors in OBDA mappings. For instance:

Query	*with constructor*	*with template*
SELECT SSN FROM RomanCitizens	Person(**pers**(SSN))	Person("http://www.ex.com/pers_{SSN}")

- R2RML is not specifically thought for OBDA, and indeed allows for more expressive forms of mappings (e.g., it is even possible to use IRI templates to construct ontology predicates).
- However, mappings used in ODBA are captured by R2RML.

R2RML: Example

Mapping

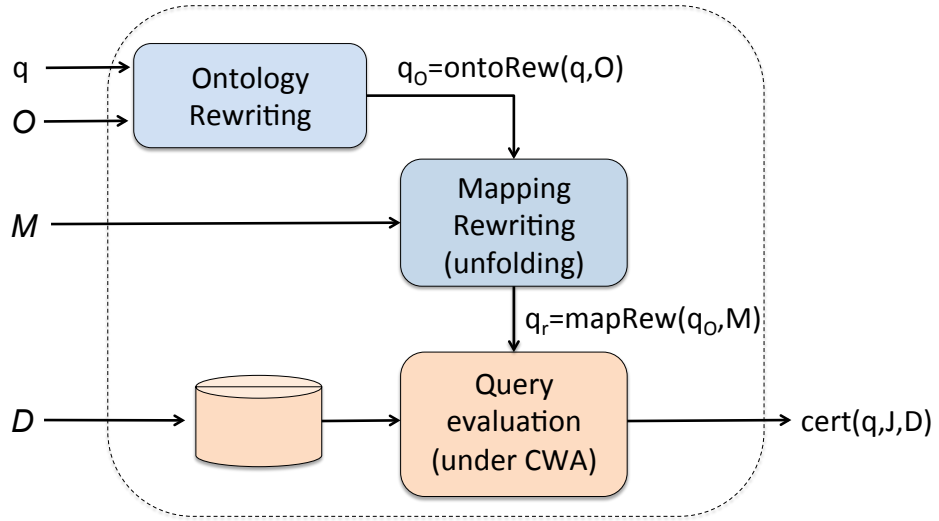
```
SELECT SSN                ⇝  Person(pers(SSN))
FROM RomanCitizens

SELECT SSN, Name          ⇝  name(pers(SSN),Name)
FROM RomanCitizens
```

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
<#TriplesMap1>
    a rr:TriplesMap;
    rr:logicalTable [rr:sqlQuery
                    """SELECT SSN,Name FROM RomanCitizens"""];
    rr:subjectMap [rr:template "http://www.ex.com/pers_{SSN}";
                   rr:class ex:Person
    ];
    rr:predicateObjectMap [rr:predicate ex:name;
                           rr:objectMap [rr:column "Name" ]
    ].
```

In the next slides we do not use R2RML, but we use templates instead of

Query Answering by Rewriting

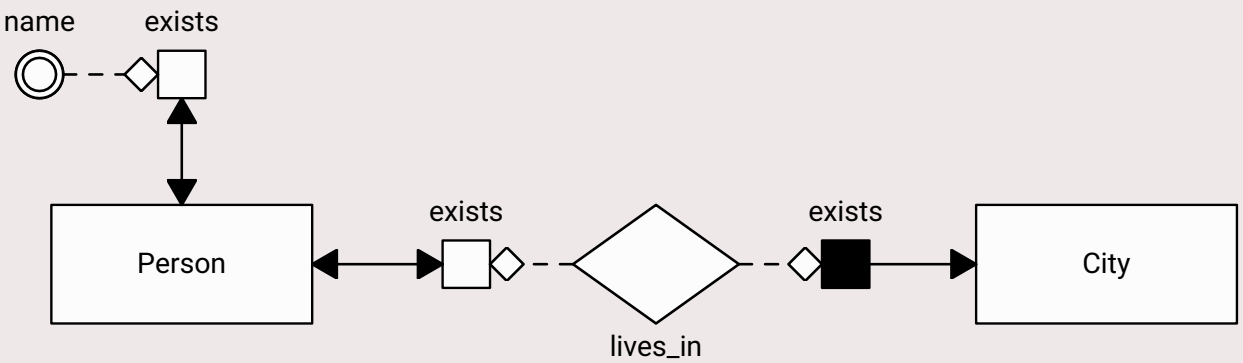
- Given an OBDA specification $\mathcal{J} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ and a user query $q(\vec{x})$ over $\mathcal{O}$, $q(\vec{x})$ is first rewritten according to the ontology $\mathcal{O}$ (ontology-rewriting), and then according to the mapping $\mathcal{M}$ (mapping-rewriting).



- The result of the entire rewriting process is a query $q_r(\vec{x})$ over $\mathcal{S}$, called the perfect rewriting, whose evaluation over any database D for $\mathcal{S}$ returns the certain answers to q over $\mathcal{J}$ wrt D, i.e., all those tuples $\vec{t}$ such that $\mathcal{I} \models q(\vec{t})$ for every model $\mathcal{I}$ of $\mathcal{J}$ wrt D.
- The language in which $q_r(\vec{x})$ has to be specified depends on the ontology language (and the form of the mapping). *DL-Lite* is essentially the only (family of) DL that allows perfect rewritings for Conjunctive Queries to be specified in SQL [CDGL$^+$07]. This property is crucial for the above query answering method to be feasible in the practice.

Example – Ontology Rewriting

Ontology



User Query

```
Select $x
Where { $x a :Person }
```

Ontology Rewriting

```
Select $x
Where {
  { $x a :Person }
  UNION
  { $x :lives_in $ndv1 }
  UNION
  { $x :name $ndv2 }
}
```

Example – Mapping Rewriting

Mapping

```
SELECT SSN,Name,'Rome' AS City ⇝ lives_in(ex:pers_{SSN},ex:ct_{City})
FROM RomanCitizens
```

Ontology Rewriting

```
Select $x
Where {
  { $x a :Person }
  UNION
  { $x :lives_in $ndv1 }
  UNION
  { $x :name $ndv2 }
}
```

Mapping Rewriting *(final rewriting)*

```
SELECT CONCAT('ex:pers_',V.SSN)
FROM (SELECT SSN, Name, 'Rome' AS City
      FROM RomanCitizens) AS V
```

Note: "ex:" is a prefix that corresponds to "http://www.ex.com/"

Data Source Analysis

- Before writing the mapping, an accurate investigation on the structure of the source databases has to be carried out.
- This is particularly complicated, since very often no adequate documentation on the source databases exist.
- Also, they commonly underwent various modifications during the years, to serve and optimize the access required by specific applications, and have loose connection with the original conceptual design (if it existed).
- This phase should not be underestimated, since it may become the real bottleneck in the design of an OBDA specification.
- Furthermore, even though the design of the ontology is initially carried out without looking at the source databases, from their analysis normally designers obtain additional details about the domain, which lead to (usually minor) changes in the ontology

Data Source Analysis

- Mappings are thus important also as a documentation means: They provide a declarative specification of the semantics of the data sources expressed in terms of the ontology, which within the organization represents that shared knowledge that did not exist before its design.
- For the mapping specification to be successful, experts of the database need to be involved. Notice that normally they are different from the domain experts, and their vision on it is driven by the way in which applications are written and data are structured in the database, rather than by the business requirements.
- Database experts participate to the mapping design process through interviews and the usage of forum/message-exchange facilities. In the ideal scenario, they are trained to take over the process of mapping design.

Data Source Analysis

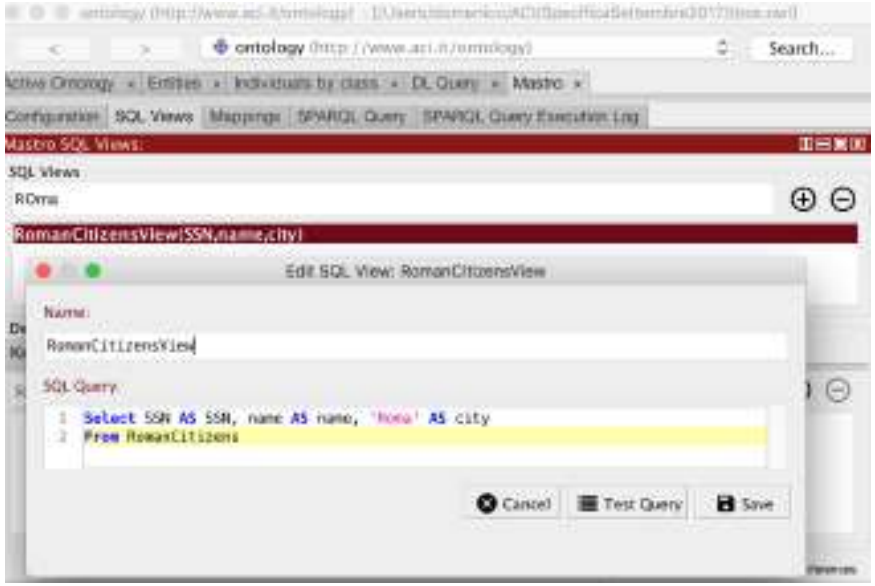
- UML or ER Diagrams of the source databases are helpful. If not available, re-engineering tools might be used to obtain them.
- Often the semantics of the sources is hidden in the applications. In this case, documentation on applications, if available, is crucial.
- Mapping designers definitely need to have a copy of the in-use source database (not just the schema), and possibly a set of prototypical queries on it, with a precise description of the output.
- In case the database is very large, not an exact copy but a representative summarized excerpt of it should be used. This will facilitate designers in testing queries that massively span on the database, often simply to understand its structure.
- If privacy is an issue, the copy of the database should be suitably anonymized.

Main Approaches to Mapping Specification

1. Ontology-driven: select the predicate to be mapped and write the query over the sources tailored to map such a predicate.
   - Pros: the focus is on a single mapping assertion at a time; source analysis done for the specific aims of selected mapping assertions.
   - Cons: designers might not be able to re-use in new assertions queries already specified in others; some relevant aspects of the domain not originally captured by the ontology but modelled in the database might be overlooked.
2. Source-driven: define a set of views (i.e., queries) that return relevant information from the sources, and use such views in mapping assertions.
   - Pros: separation of the concerns related to source analysis from those specifically connected to mapping specification; using the same view in different mapping assertions is more natural: by limiting the number of views, mapping are easier to manage.
   - Cons: requires a more general understanding of the sources; producing the first mapping assertions may take longer.

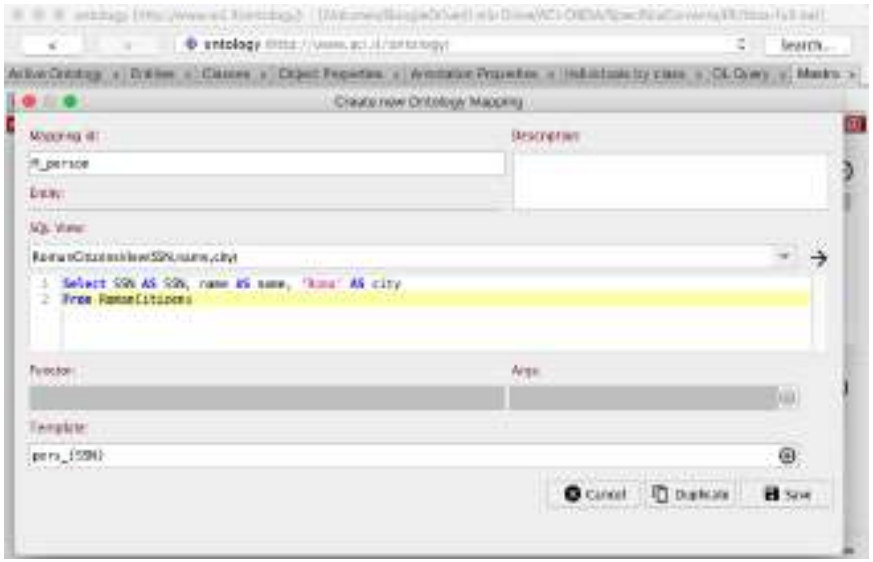
Specifying mapping using views in Mastro

In the Mastro plug-in for Protégé we indeed first define views over the source schema (e.g., we create the view RomanCitizensView(SSN,name,city)



Then we use the views to specify the mapping, e.g., we specify the mapping

RomanCitizensView(SSN,name,city)
$\rightsquigarrow$
Person(ex:pers_{SSN})



Some critical issues in mapping specification

We now discuss some main issues that a designer have to deal with when specifying mappings. The following list is definitely not complete but contains crucial aspects that necessarily need to be addressed.

- Define constructors, i.e., the functions used to construct individuals, which means deciding both the function symbols and their arguments (or, alternatively, the form of the IRI template).
- Select which ontology predicates to map.
- Manage the presence of `NULL` in the underlying database.

Mapping in the presence of NULL

- A NULL in a database instance is an unspecified value. It may have different interpretations:
  - The value exists but the database does not know it (e.g., a NULL data of birth for a person);
  - The property the NULL refers to is not applicable (e.g., a NULL for the salary of a student that is not also a worker);
  - It is unknown which of the two cases above applies (a NULL for the salary of a student that we do not know if it is also a worker).
- Of course, we cannot conclude that different occurrences of NULL denote the same value (unless this is implied by some constraints of the database).

Mapping in the presence of NULL

- In principle, a query in the body of a mapping assertion has not to return NULL.
- Indeed, according to the first-order semantics we use to interpret the ontology, a NULL occurring in the instance of a TBox constructed through the mapping (either as a value in an attribute or as an argument of a function) is simply considered as a constant denoting a value, and thus all its occurrences denote the same value, which is not the intended meaning of NULL in a relational database, as we have said before.
- Additionally, in the presence of the unique name assumption (UNA), which states that different constants must be interpreted by different objects or values, as in *DL-Lite*, a NULL denotes a value that is different from all the others, and thus it is interpreted with a value that is different from all the other values.
- For all the above reasons, it is recommended that arguments of constructors in the mapping never transmits NULL values from the sources.

Principles on how to construct individuals

- An aspect a designer should take care is to not specify the mapping in such a way that different domain objects are denoted by the same individual.

Example of wrong mapping

Let'us assume to have in the source schema two different tables representing persons coming from different databases:

TabPers1		
ID	*SSN*	*Gender*
1	123	M
...	...	...

TabPers2		
ID	*SSN*	*Gender*
1	456	F
...	...	...

and the following mapping assertions

`SELECT ID FROM TabPers1` $\rightsquigarrow$ Person(ex:pers_{ID})

`SELECT ID FROM TabPers2` $\rightsquigarrow$ Person(ex:pers_{ID})

Principles on how to construct individuals

Possible solution 1 - use different constructors

TabPers1		
ID	*SSN*	*Gender*
1	123	M

TabPers2		
ID	*SSN*	*Gender*
1	456	F

`SELECT ID FROM TabPers1` $\rightsquigarrow$ Person(ex:pers1_{ID})

`SELECT ID FROM TabPers2` $\rightsquigarrow$ Person(ex:pers2_{ID})

Possible solution 2 - use a business identifier

TabPers1		
ID	*SSN*	*Gender*
1	123	M

TabPers2		
ID	*SSN*	*Gender*
1	456	F

`SELECT SSN FROM TabPers1` $\rightsquigarrow$ Person(ex:pers_{SSN})

`SELECT SSN FROM TabPers2` $\rightsquigarrow$ Person(ex:pers_{SSN})

Principles on how to construct individuals

- The mapping should also guarantee that an object of the domain is always denoted by the same individual. This is crucial for DLs that adopt the UNA (as in *DL-Lite*), but also in DLs that do not allow to infer equalities between individuals (as in OWL 2 QL).

Example of wrong mapping

TabPers		
ID	*SSN*	*Occupation*
1	123	'stud'
...	...	...

```
SELECT ID FROM TabPers
WHERE Occupation='stud'
```
$\rightsquigarrow$ Student(ex:stud_{ID})

```
SELECT ID FROM TabPers
```
$\rightsquigarrow$ Person(ex:pers_{ID})

Principles on how to construct individuals

Possible solution - use the same constructor

TabPers		
ID	*SSN*	*Occupation*
1	123	'stud'

```
SELECT ID FROM TabPers
WHERE Occupation='stud'
```
$\rightsquigarrow$ Student(ex:pers_{ID})

```
SELECT ID FROM TabPers
```
$\rightsquigarrow$ Person(ex:pers_{ID})

Principles on how to construct individuals

- Generally speaking, constructing individuals from the values retrieved at the data sources is a very complex activity, for which no consolidated methods exists.
- Solving this task requires understanding how objects are identified in the domain, and finding out the identifier at the sources (e.g. for a person, her SSN).
- We have to guarantee that (i) different domain objects are not denoted with the same individual and, possibly (but it is crucial for *DL-Lite* and OWL 2 QL) also that (ii) a domain object is never denoted with different individuals.
- data matching methods need often to be adopted to find out different representations of the same object [Chi12].

Principles on how to construct individuals in *DL-Lite*

- Consider the simplified scenario in which for every object we can retrieve from the sources the values that identify it, and that such identification is uniform over all the source tables (e.g., a person is always identified by her SSN). We may proceed as follows:
  1. Let MGC (Most General Classes) be the set of all ontology classes that are subsumed only by owl:Thing (the Top class all individuals are instance of);
  2. For each class C in MGC, find out how instances of C are identified in the sources and define a constructor based on such identifier;
  3. Use the same constructor for all classes subsumed by C
- Comments: (a) if there are equivalent classes, put only a representative one in MGC (b) for objects that are instance of more than one class in MGC, select one constructor among the possible one (typically, classes in MGC are in fact all pair-wise disjoint) (c) define additional constructors for objects that are not instance of any class in MGC (typically, there are no such objects, since MGC is a partition of owl:Thing).

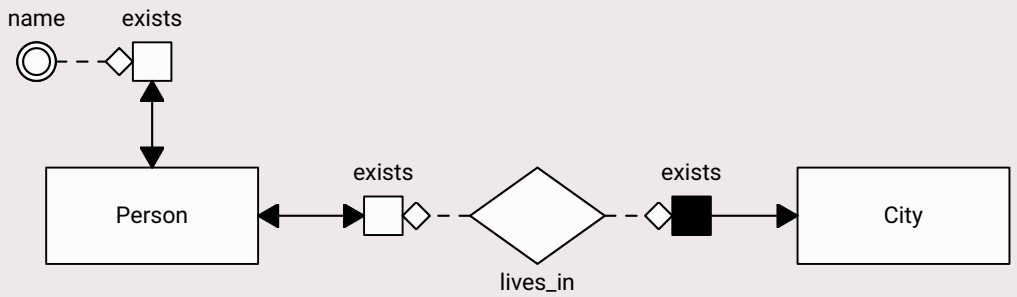
Select which ontology predicates to map

- Let us consider the (extreme) case where the ontology is empty, i.e., it has no axioms and thus it is simply a set of predicates.
- In this case we have to map every ontology predicate. Indeed, the ontology cannot infer new facts besides those directly constructed through the mapping.
- A most interesting case, however is the one of non-empty ontologies.
- In this case, we can exploit inclusions in the ontology to reduce the number of mapping assertions to write. Intuitively, we avoid to write assertions that are implied by the OBDA specification.
- Furthermore, we have to avoid writing mappings that are *intensionally inconsistent*, i.e., such there are no source instances for which the specification has a model (e.g., two disjoint classes mapped to the same query, using the same constructor) [LMR+15].

Example

A mapping assertion for an object/data property R and the typing of the domain of R over a class C imply a mapping assertion for C.

Ontology



Source Data

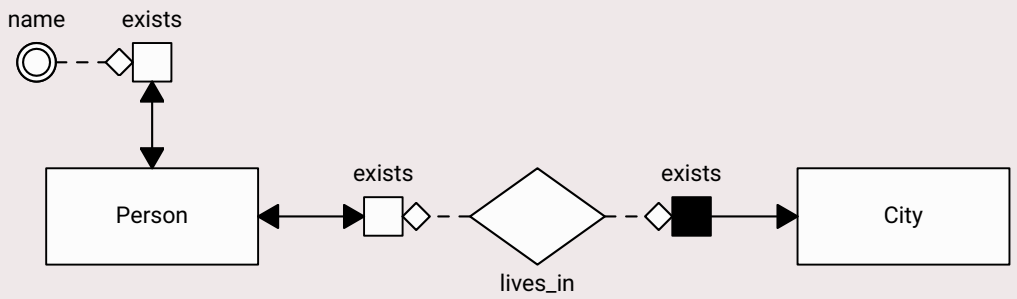
RomanCitizens		
SSN	*Name*	*Gender*
1234	Marco	M
...	...	...

Mapping

```
SELECT SSN                          ⇝  Person(ex:pers_{SSN})
FROM RomanCitizens

SELECT SSN, 'Rome' AS City          ⇝  lives_in(ex:pers_{SSN}, ex:ct_{City})
FROM RomanCitizens

SELECT SSN, Name                    ⇝  name(ex:pers_{SSN}, Name)
FROM RomanCitizens
```

Example

Ontology



Source Data

RomanCitizens		
SSN	*Name*	*Gender*
1234	Marco	M
...	...	...

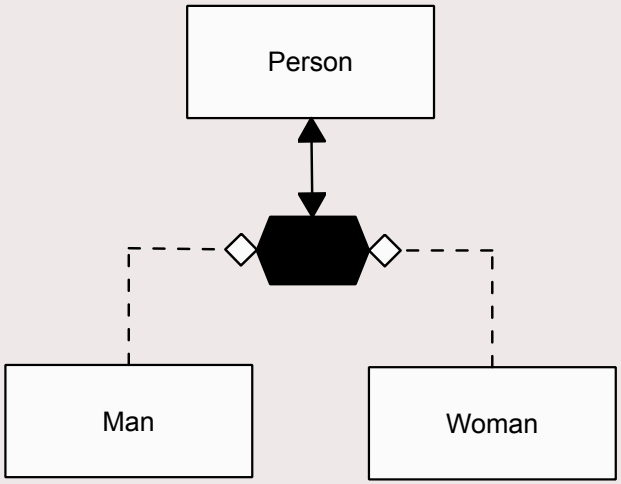
Mapping

```
SELECT SSN, 'Rome' AS City
FROM RomanCitizens
```
$\rightsquigarrow$ lives_in(ex:pers_{SSN}, ex:ct_{City})

```
SELECT SSN, Name
FROM RomanCitizens
```
$\rightsquigarrow$ name(ex:pers_{SSN}, Name)

Example

Ontology



Source Data

RomanCitizens		
SSN	*Name*	*Gender*
1234	Marco	M
...	...	...

If we know that 'F' and 'M' are the only allowed values for *Gender*, and that *Gender* is not not nullable, we can avoid to write a mapping for Person.

Mapping

```
SELECT SSN FROM RomanCitizens ⇝ Man(ex:pers_{SSN})
WHERE Gender='M'

SELECT SSN FROM RomanCitizens ⇝ Woman(ex:pers_{SSN})
WHERE Gender='F'
```

OBDA tools

- Mastro [DGLL+12, CDGL+11] is a tool for OBDA by Sapienza and OBDA Systems It implements various optimizations aimed at reducing the size of perfect rewritings, both w.r.t. the ontology [RA10, Ros12], and the mapping [DPLL+13]. Mastro is freely available for any non-commercial purpose, including academic and research.
- Ontop [CCK+17] is an open-source system for OBDA developed at the Free University of Bolzano. It has common origins with Mastro, but adopts different algorithms for both ontology and mapping rewriting [KKZ12]. It works also as a triplestore.
- Ultrawrap (`https://capsenta.com/ultrawrap/`) is a commercial tool by Capsenta for virtualising relational databases into RDF. It supports W3C R2RML and SPARQL 1.1 queries, and allows for some forms of reasoning over extensions of RDFS.
- StarDog (`www.stardog.com/`), a commercial tool (with also a free community edition) that is primarily a triple store, but also features the mapping of databases to OWL ontologies through R2RML, and query answering by rewriting. It supports SPARQL 1.1 and the OWL 2 Direct Semantics entailment regime.

OBDA tools

- DR2Q (`http://d2rq.org/`), developed at the Free University of Berlin and DERI, has been one of the first OBDA tools. It is partially compliant with R2RML, and has its own mapping language. Reasoning it is not supported and its development has been dismissed to date.
- Morph-RDB (`https://github.com/oeg-upm/morph-rdb`) is an open-source R2RML Engine. It supports query answering by rewriting but with no reasoning abilities.
- Virtuoso (`https://virtuoso.openlinksw.com/`) is a commercial multi-model database management system. As a triplestore, it provides limited forms of reasoning for SPARQL queries. As a data virtualization tool it partially supports R2RML, but it has no reasoning is allowed in this modality.
- Other tools allow for query answering over (fragments of) OWL 2, and provide various forms of reasoning, (see, e.g., RDFox (`http://www.cs.ox.ac.uk/isg/tools/RDFox`), GraphDB (`https://ontotext.com/products/graphdb`), or the Oracle Spatial and Graph tool (`https://www.oracle.com/database/spatial`)). Such tools materialize data and also (part of) the reasoning (no pure query rewriting is performed), thus do not support data virtualization through mappings.

Acknowledgement

- The present slides are co-authored by Giuseppe De Giacomo, Domenico Lembo, Valerio Santarelli and Domenico Fabio Savo. Antonella Poggi and Giuseppe De Giacomo were also instructors in a previous edition of this tutorial (IJCAI-16).
- We are also thankful to several colleagues and students who contributed to the material and research presented in this tutorial and to the development of the tools we use in the hand-on session. In particular,
  - Maurizio Lenzerini (Univ. Sapienza)
  - Riccardo Rosati (Univ. Sapienza)
  - Diego Calvanese (Univ. Bolzano)
  - Marco Ruzzi (OBDA Systems)
  - Giacomo Ronconi (OBDA Systems)
  - Federico Scafoglieri (Univ. Sapienza)
  - Daniele Pantaleone (Univ. Sapienza – ex student)

References I

[CCK+17] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao.

Ontop: Answering SPARQL queries over relational databases.

Semantic Web, 8(3):471–487, 2017.

[CDGL+07] Diego Calvanese, 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.

[CDGL+11] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo.

The Mastro system for ontology-based data access.

Semantic Web J., 2(1):43–53, 2011.

References II

[Chi12] Peter Chisten.

Data Matching: concepts and techniques for record linkage, entity resolution, and duplicate detection.

Springer, 2012.

[DGLL+12] Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo.

Mastro: A reasoner for effective ontology-based data access.

In *Proc. of the OWL Reasoner Evaluation Workshop (ORE 2012)*, volume 858 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2012.

[DPLL+13] Floriana Di Pinto, Domenico Lembo, Maurizio Lenzerini, Riccardo Mancini, Antonella Poggi, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo.

Optimizing query rewriting in ontology-based data access.

In *Proc. of the 16th Int. Conf. on Extending Database Technology (EDBT 2013)*, pages 561–572. ACM Press, 2013.

References III

[DSC12] Souripriya Das, Seema Sundara, and Richard Cyganiak.
R2RML: RDB to RDF mapping language.
W3C Recommendation, World Wide Web Consortium, September 2012.
Available at `http://www.w3.org/TR/r2rml/`.

[KKZ12] 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)*, 2012.

[Len02] Maurizio Lenzerini.
Data integration: A theoretical perspective.
In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.

References IV

[LMR+15] Domenico Lembo, José Mora, Riccardo Rosati, Domenico Fabio Savo, and Evgenij Thorstensen.

Mapping analysis in ontology-based data access: Algorithms and complexity.

In *Proc. of the 14th Int. Semantic Web Conf. (ISWC 2015)*, pages 217–234, 2015.

[RA10] 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.

[Ros12] Riccardo Rosati.

Prexto: Query rewriting under extensional constraints in *DL-Lite*.

In *Proc. of the 9th Extended Semantic Web Conf. (ESWC 2012)*, volume 7295 of *Lecture Notes in Computer Science*, pages 360–374. Springer, 2012.