MAI4CAREU | Master programmes in Artificial Intelligence 4 Careers in Europe

University of Cyprus

MAI649: PRINCIPLES OF ONTOLOGICAL DATABASES

Ontological Databases

Andreas Pieris

Spring 2022-2023

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Learning Outcomes

- Syntax and semantics of existential rules
- Ontological query answering and universal models
- Ontology-based data access

Querying Relational Databases

List the codes of teaching staff

Lecturer	id	Name
	1	Alice
	2	Bob
	3	Tom
	4	Mary

Course	code	organiser
	CS100	2
	CS200	1
	CS300	5

Q(x) :- TeachingStaff(x,y)

Querying Relational Databases

List the codes of teaching staff

Lecturer	id	Name
	1	Alice
	2	Bob
	3	Tom
	4	Mary

Course	code	organiser
	CS100	2
	CS200	1
	CS300	5

Lecturers are teaching staff

Course organisers are teaching staff

Q(x) :- TeachingStaff(x,y)

Querying Relational Databases

List the codes of teaching staff

Lecturer	id	Name
	1	Alice
	2	Bob
	3	Tom
	4	Mary

Course	code	organiser
	CS100	2
	CS200	1
	CS300	5

$\forall x \forall y$ (Lecturer(x,y) $\rightarrow$ TeachingStaff(x,y))

$\forall x \forall y$ (Course(x,y) $\rightarrow \exists z$ TeachingStaff(y,z))

Q(x) :- TeachingStaff(x,y)

Querying Relational Databases

List the codes of teaching staff

Lecturer	id	Name
	1	Alice
	2	Bob
	3	Tom
	4	Mary

Course	code	organiser
	CS100	2
	CS200	1
	CS300	5

{1, 2, 3, 4, 5}

$\forall x \forall y\ (\text{Lecturer}(x,y) \rightarrow \text{TeachingStaff}(x,y))$

$\forall x \forall y\ (\text{Course}(x,y) \rightarrow \exists z\ \text{TeachingStaff}(y,z))$

Q(x) :- TeachingStaff(x,y)

Some Terminology

- Our basic vocabulary:
  - A countable set **Const** of constants - domain of a database
  - A countable set **Nulls** of marked nulls - globally ∃-quantified variables
  - A countable set **Vars** of variables - used in rules and queries

- A term is a constant, marked null, or variable

- An atom has the form $R(t_1,\ldots,t_n)$ - R is an n-ary relation and t_i's are terms

- An instance is a (*possibly infinite*) set of atoms with constants and nulls

- A database is a finite instance with only constants

Syntax of Existential Rules

An existential rule is an expression

$$\forall \mathbf{x} \forall \mathbf{y}\, (\underbrace{\varphi(\mathbf{x},\mathbf{y})}_{\text{body}} \rightarrow \exists \mathbf{z}\, \underbrace{\psi(\mathbf{x},\mathbf{z})}_{\text{head}})$$

- **x**,**y** and **z** are tuples of variables of **Vars**
- $\varphi(\mathbf{x},\mathbf{y})$ and $\psi(\mathbf{x},\mathbf{z})$ are (constant-free) conjunctions of atoms

…also known as tuple-generating dependencies

Semantics of Existential Rules

- An instance J is a model of the rule

$$\sigma = \forall \mathbf{x} \forall \mathbf{y} \, (\varphi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z} \, \psi(\mathbf{x},\mathbf{z}))$$

written as $J \models \sigma$, if the following holds:

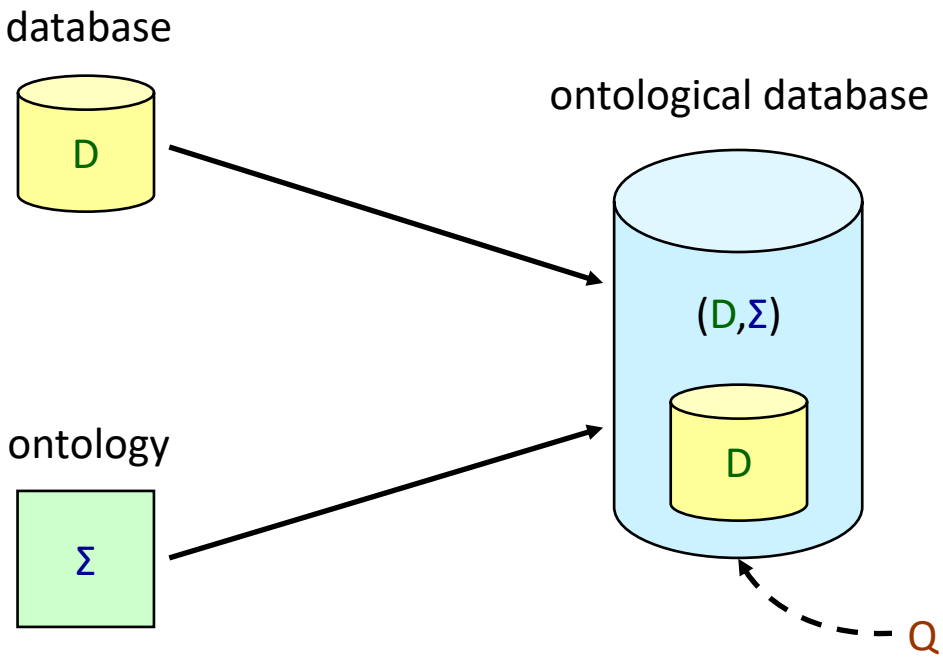
whenever there exists a homomorphism h such that $h(\varphi(\mathbf{x},\mathbf{y})) \subseteq J$,

then there exists $g \supseteq h_{|\mathbf{x}}$ such that $g(\psi(\mathbf{x},\mathbf{z})) \subseteq J$

$\{t \mapsto h(t) \mid t \in \mathbf{x}\}$ - the restriction of h to **x**

- Given a set Σ of existential rules, J is a model of Σ, written as $J \models \Sigma$, if, for each $\sigma \in \Sigma$, $J \models \sigma$

Ontological Query Answering (OQA)



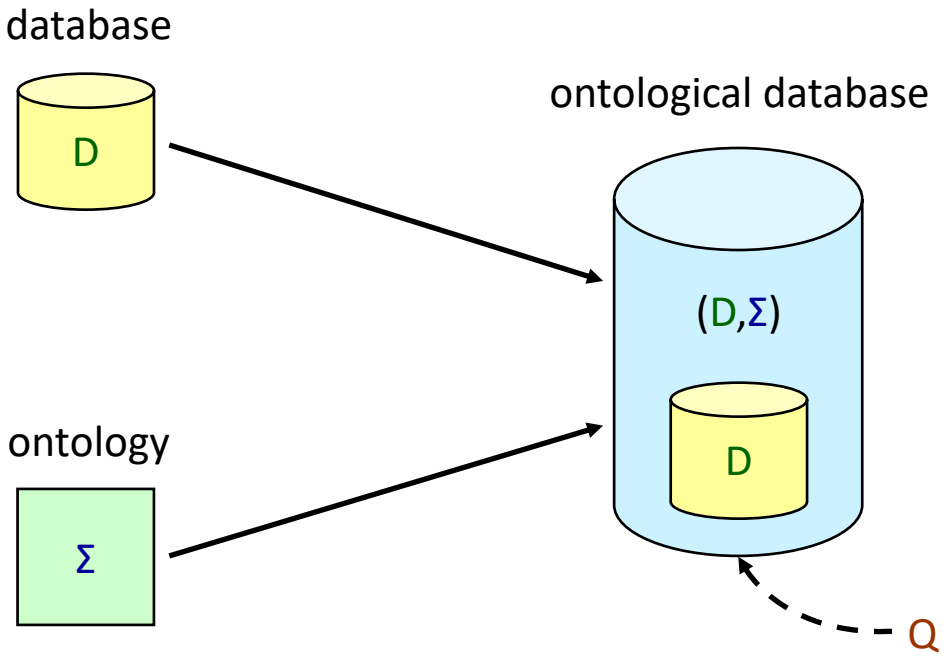
existential rules

$\forall \mathbf{x} \forall \mathbf{y}\ (\varphi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z}))$

conjunctive query

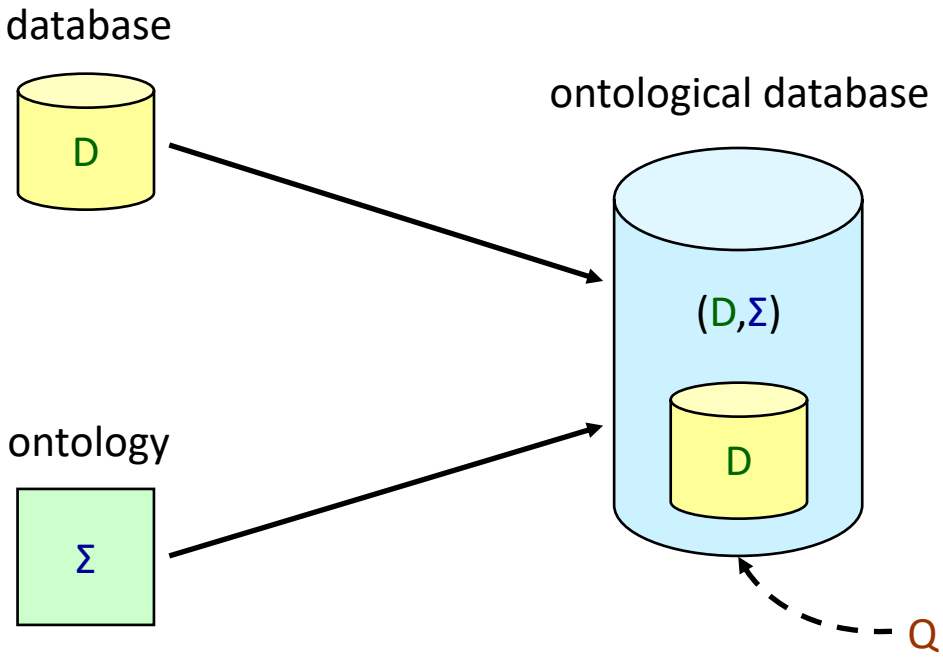
$Q(\mathbf{x})\ :-\ R_1(\mathbf{v_1}),\ldots,R_m(\mathbf{v_m})$

Ontological Query Answering (OQA)



$$models(D,\Sigma) = \{J \mid J \supseteq D \text{ and } J \models \Sigma\}$$

Ontological Query Answering (OQA)



$$\text{Answer}(Q,D,\Sigma) = \bigcap_{J \in \text{models}(D,\Sigma)} Q(J)$$

Exercise: Compute the Certain Answers

D = {Person(john), Person(bob), Person(tom),
hasFather(john,bob), hasFather(bob,tom)}

Σ = {$\forall x$ (Person(x) $\rightarrow$ $\exists y$ hasFather(x,y)),
$\forall x \forall y$ (hasFather(x,y) $\rightarrow$ Person(x) $\wedge$ Person(y))}

$Q_1(x,y)$:- hasFather(x,y)

$Q_2(x)$:- hasFather(x,y)

$Q_3(x)$:- hasFather(x,y), hasFather(y,z), hasFather(z,w)

$Q_4(x,w)$:- hasFather(x,y), hasFather(y,z), hasFather(z,w)

Exercise: Compute the Certain Answers

D = {Person(john), Person(bob), Person(tom),
hasFather(john,bob), hasFather(bob,tom)}

Σ = {$\forall x$ (Person(x) $\rightarrow$ $\exists y$ hasFather(x,y)),
$\forall x \forall y$ (hasFather(x,y) $\rightarrow$ Person(x) $\wedge$ Person(y))}

$Q_1(x,y)$:- hasFather(x,y)

{(john,bob), (bob,tom)}

Exercise: Compute the Certain Answers

D = {Person(john), Person(bob), Person(tom),
hasFather(john,bob), hasFather(bob,tom)}

Σ = {$\forall x$ (Person(x) $\rightarrow$ $\exists y$ hasFather(x,y)),
$\forall x \forall y$ (hasFather(x,y) $\rightarrow$ Person(x) $\wedge$ Person(y))}

$Q_2(x)$:- hasFather(x,y)

{(john), (bob), (tom)}

Exercise: Compute the Certain Answers

D = {Person(john), Person(bob), Person(tom),
hasFather(john,bob), hasFather(bob,tom)}

Σ = {$\forall x$ (Person(x) $\rightarrow$ $\exists y$ hasFather(x,y)),
$\forall x \forall y$ (hasFather(x,y) $\rightarrow$ Person(x) $\wedge$ Person(y))}

$Q_3(x)$:- hasFather(x,y), hasFather(y,z), hasFather(z,w)

{(john), (bob), (tom)}

Exercise: Compute the Certain Answers

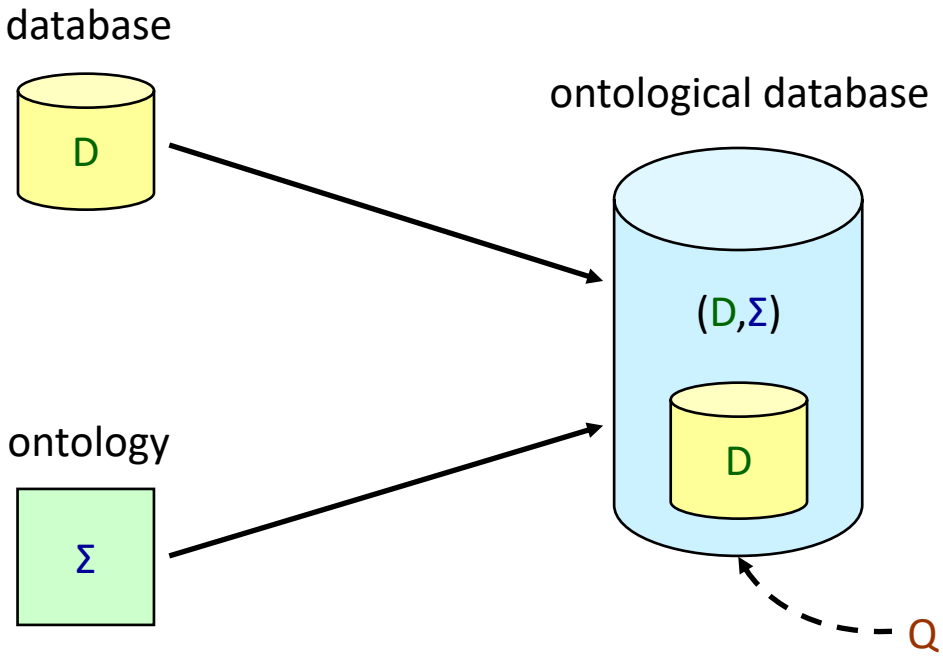
D = {Person(john), Person(bob), Person(tom),
hasFather(john,bob), hasFather(bob,tom)}

Σ = {$\forall x$ (Person(x) $\rightarrow$ $\exists y$ hasFather(x,y)),
$\forall x \forall y$ (hasFather(x,y) $\rightarrow$ Person(x) $\wedge$ Person(y))}

Q_4(x,w) :- hasFather(x,y), hasFather(y,z), hasFather(z,w)

{ }

Ontological Query Answering (OQA)



$$\text{Answer}(Q,D,\Sigma) = \bigcap_{J \in \text{models}(D,\Sigma)} Q(J)$$

keep only tuples with constants

Ontological Query Answering (OQA)

ontology language based on existential rules

OQA(**L**)

Input: a database D, a set of existential rules $\Sigma \in \mathbf{L}$, a CQ Q/k, a tuple of constants $\mathbf{t} \in \mathbf{adom}(D)^k$

Question: $\mathbf{t} \in$ Answer(Q,D,Σ)?

BOQA(**L**)

Input: a database D, a set of existential rules $\Sigma \in \mathbf{L}$, a Boolean query Q

Question: is Answer(Q,D,Σ) non-empty?

Theorem: OQA(**L**) $\equiv_L$ BOQA(**L**) for every language **L**

($\equiv_L$ means logspace-equivalent)

Data Complexity of BOQA

input D, fixed Σ and Q

BOQA[Σ,Q](**L**)

Input: a database D

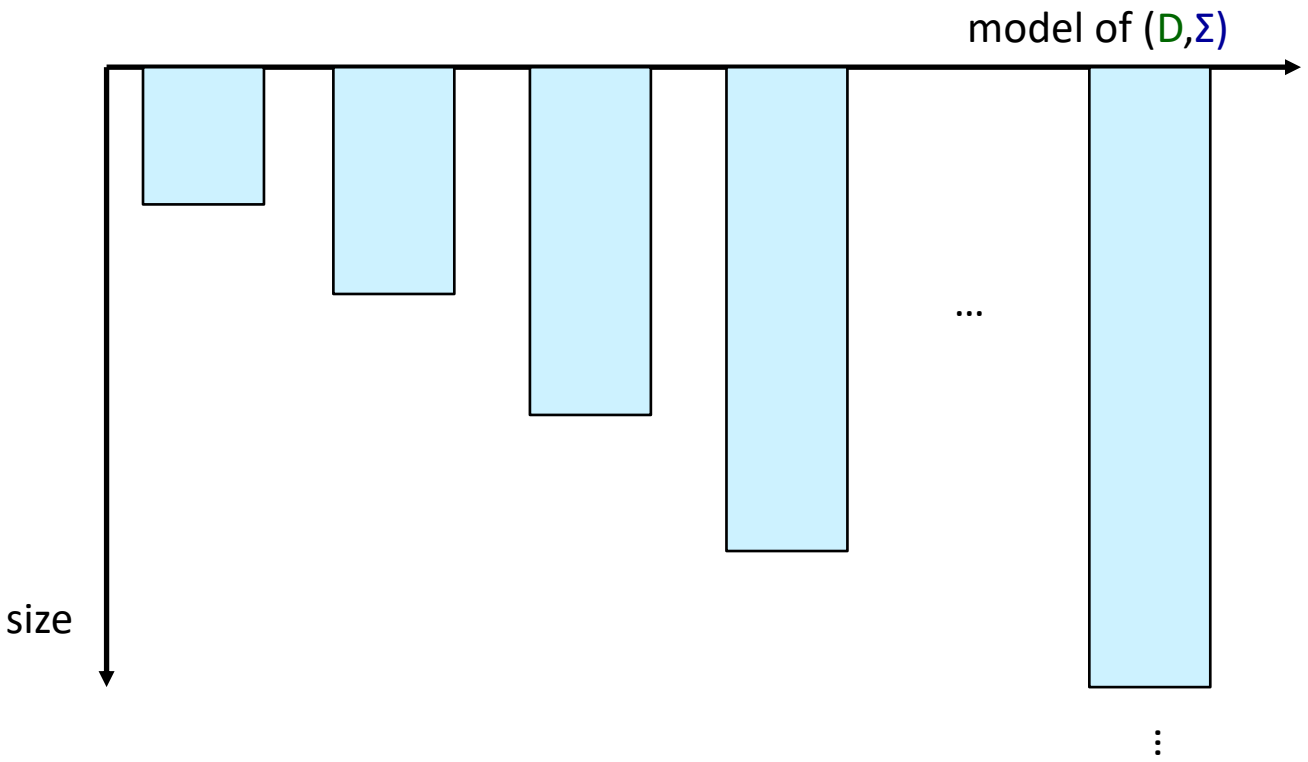
Question: is Answer(Q,D,Σ) non-empty?

Why is OQA technically challenging?

What is the right tool for tackling this problem?

The Two Dimensions of Infinity

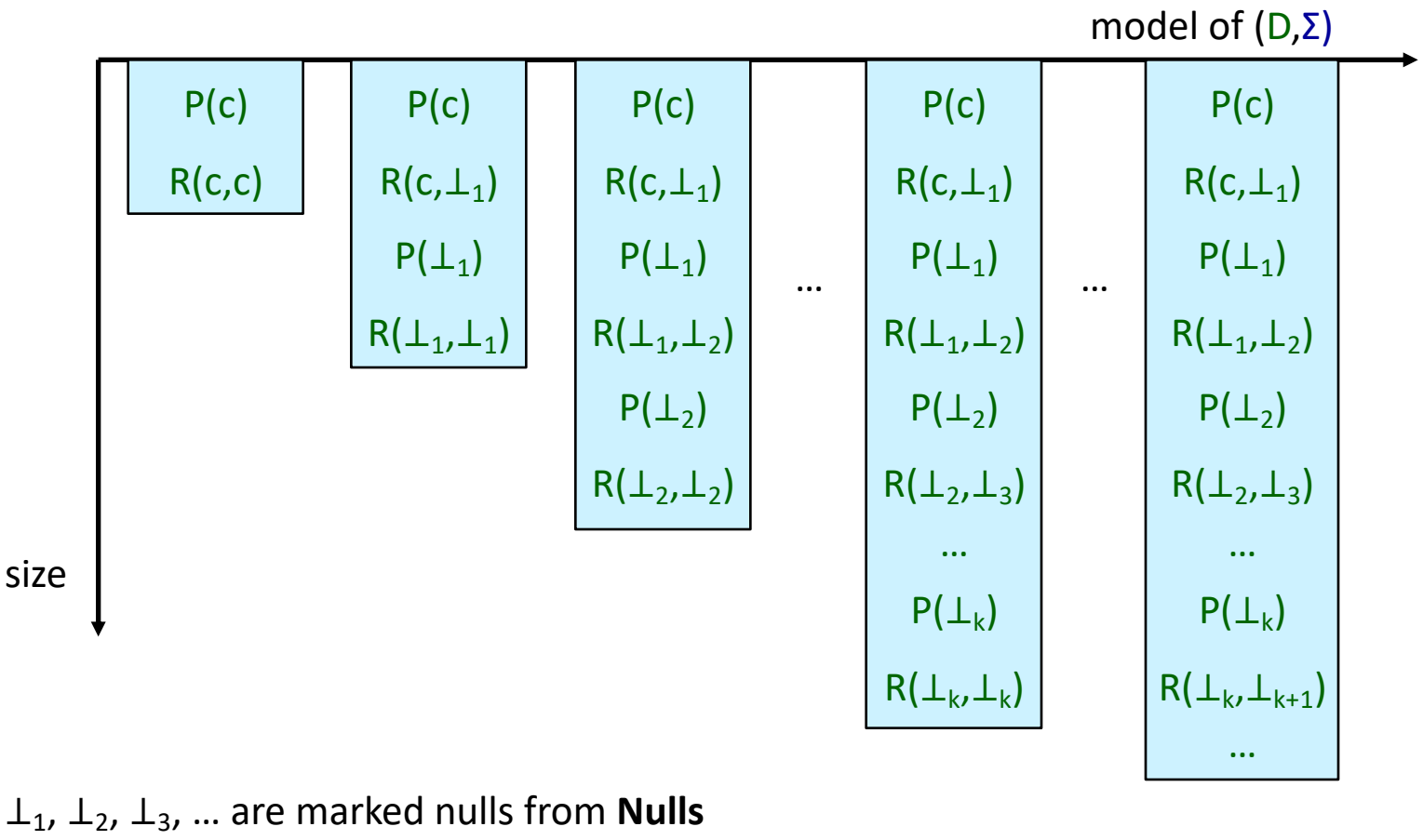
Consider a database D, and a set of existential rules Σ



(D,Σ) admits infinitely many models, of possibly infinite size

The Two Dimensions of Infinity

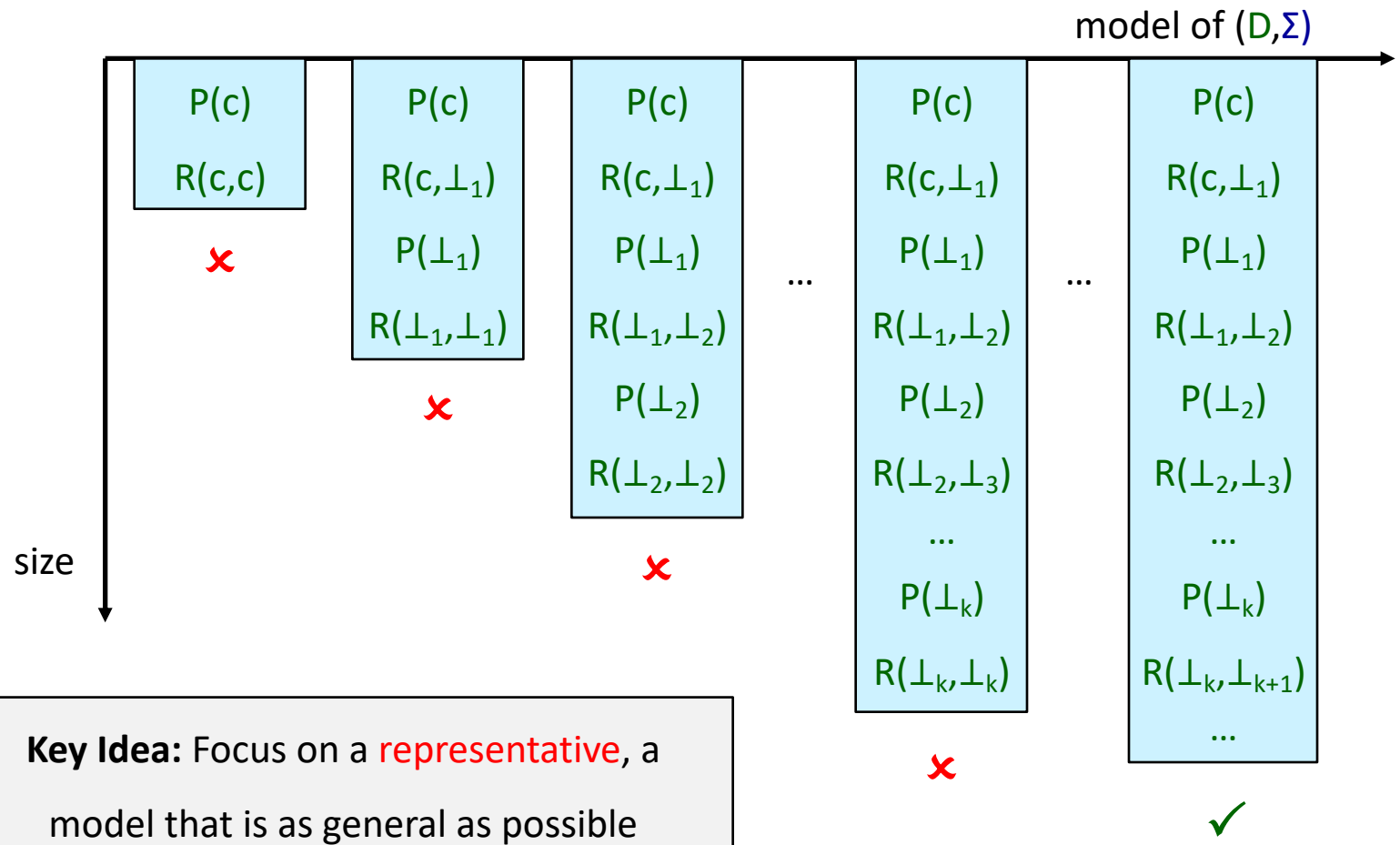
$D = \{P(c)\}$ $\Sigma = \{\forall x\,(P(x) \rightarrow \exists y\,(R(x,y) \wedge P(y)))\}$



$\perp_1, \perp_2, \perp_3, \ldots$ are marked nulls from **Nulls**

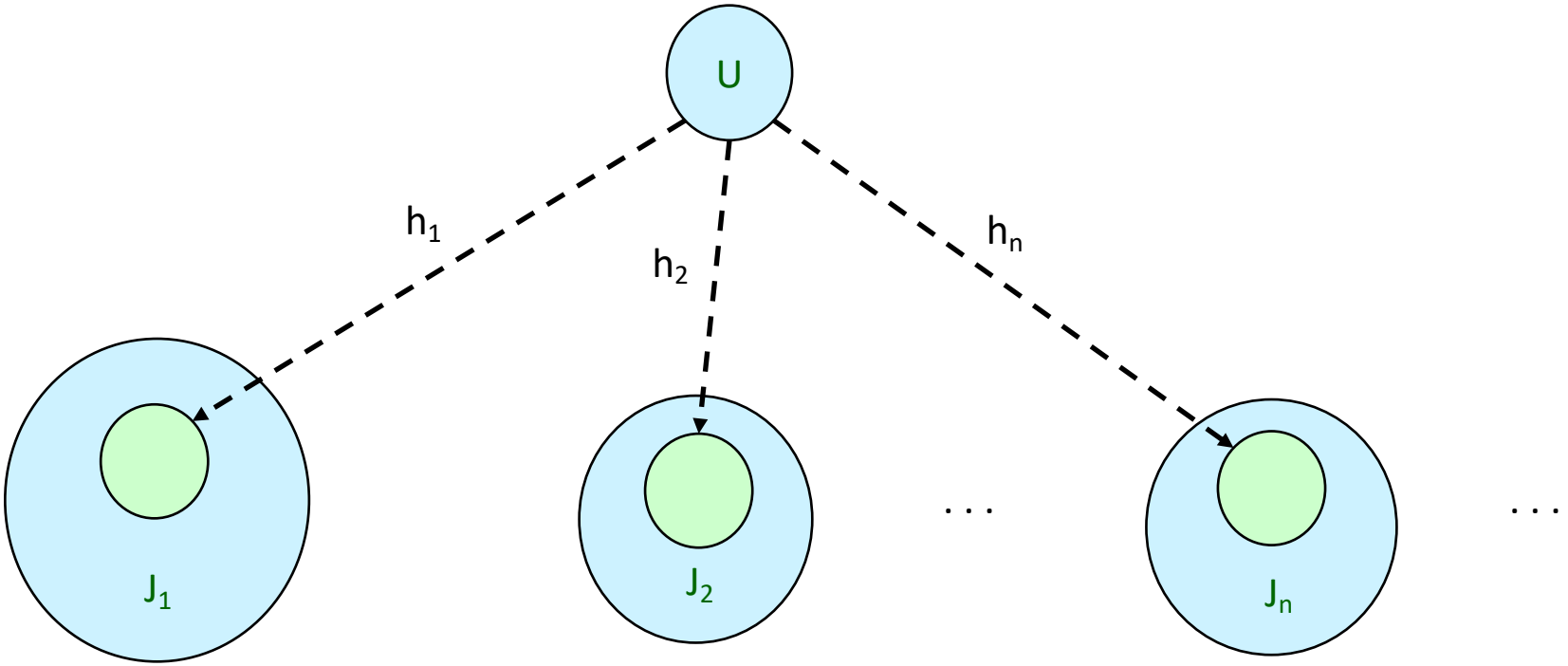
The Two Dimensions of Infinity

$D = \{P(c)\}$ $\Sigma = \{\forall x\,(P(x) \rightarrow \exists y\,(R(x,y) \wedge P(y)))\}$



Key Idea: Focus on a representative, a model that is as general as possible

Universal Models (a.k.a. Canonical Models)



An instance U is a universal model of (D,Σ) if the following holds:

1. U is a model of (D,Σ)

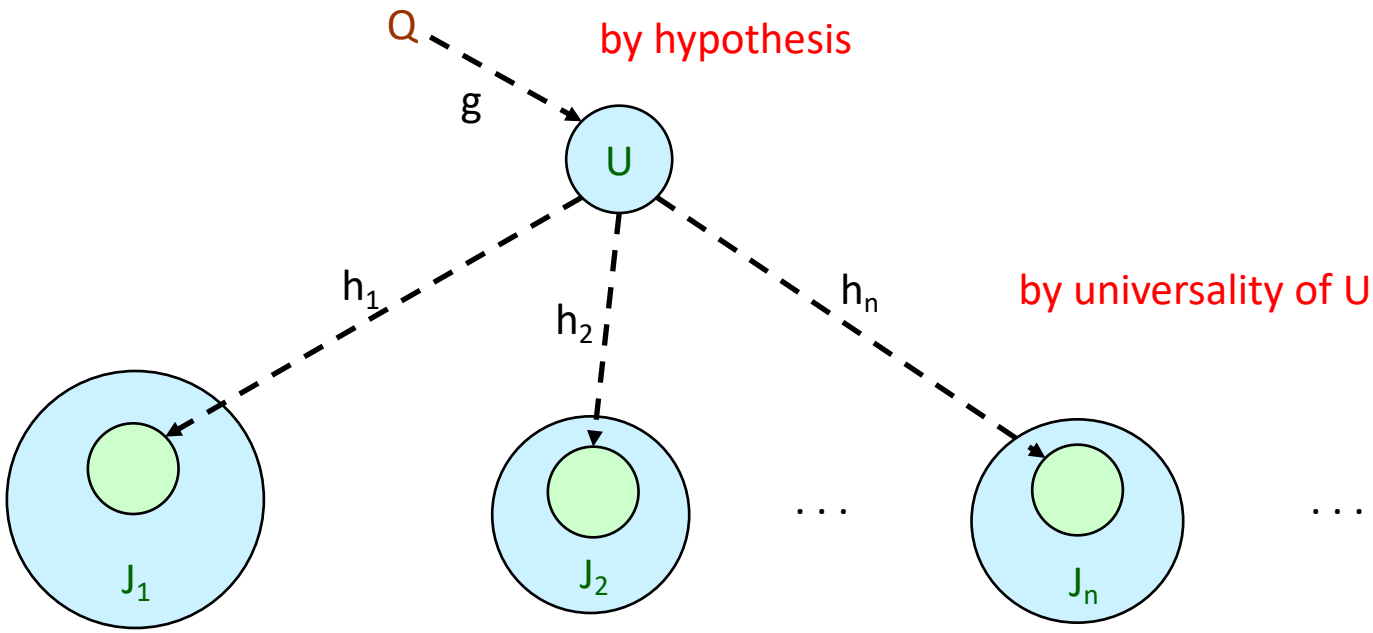
2. for each J ∈ models(D,Σ), there exists a homomorphism h such that h(U) ⊆ J

Query Answering via Universal Models

Theorem: Answer(Q,D,Σ) is non-empty iff Q(U) is non-empty, where U a universal model of (D,Σ)

Proof: (⇒) Trivial since, for every J ∈ models(D,Σ), Q(J) is non-empty

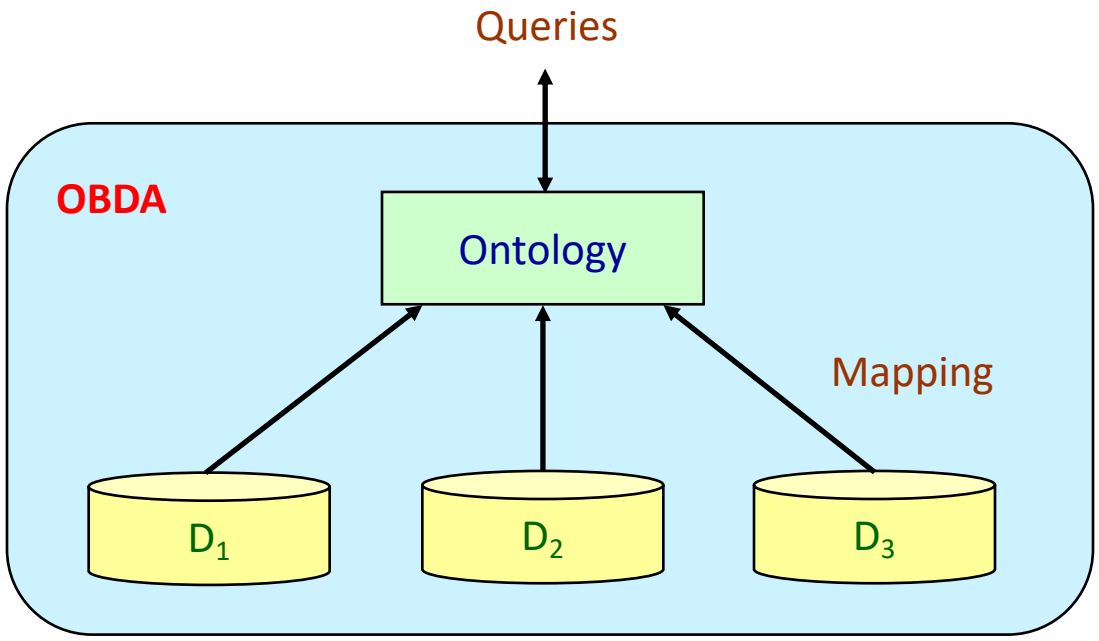
(⇐) By exploiting the universality of U



∀J ∈ models(D,Σ), ∃h such that h(g(Q)) ⊆ J ⇒ ∀J ∈ models(D,Σ), Q(J) is non-empty

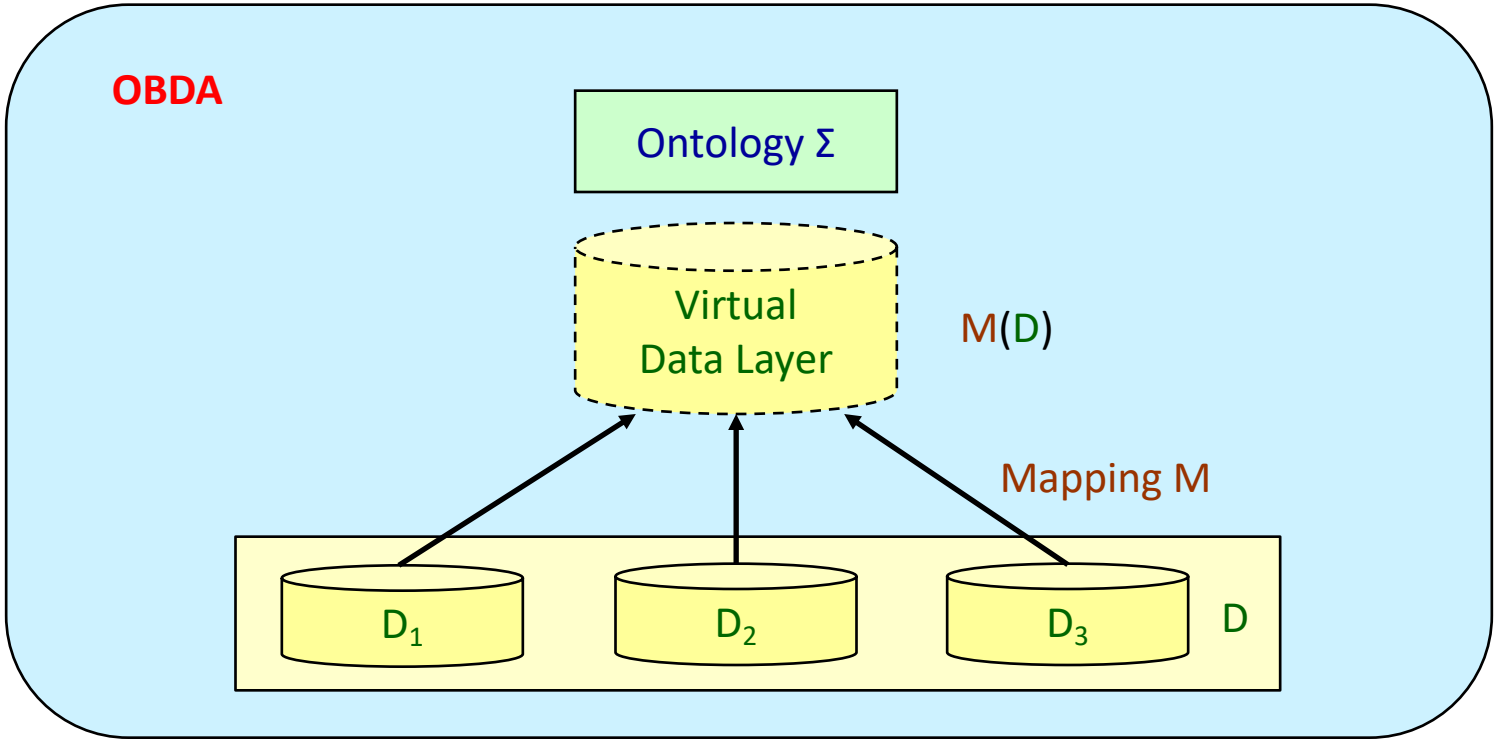
⇒ Answer(Q,D,Σ) is non-empty

Ontology-based Data Access: Architecture



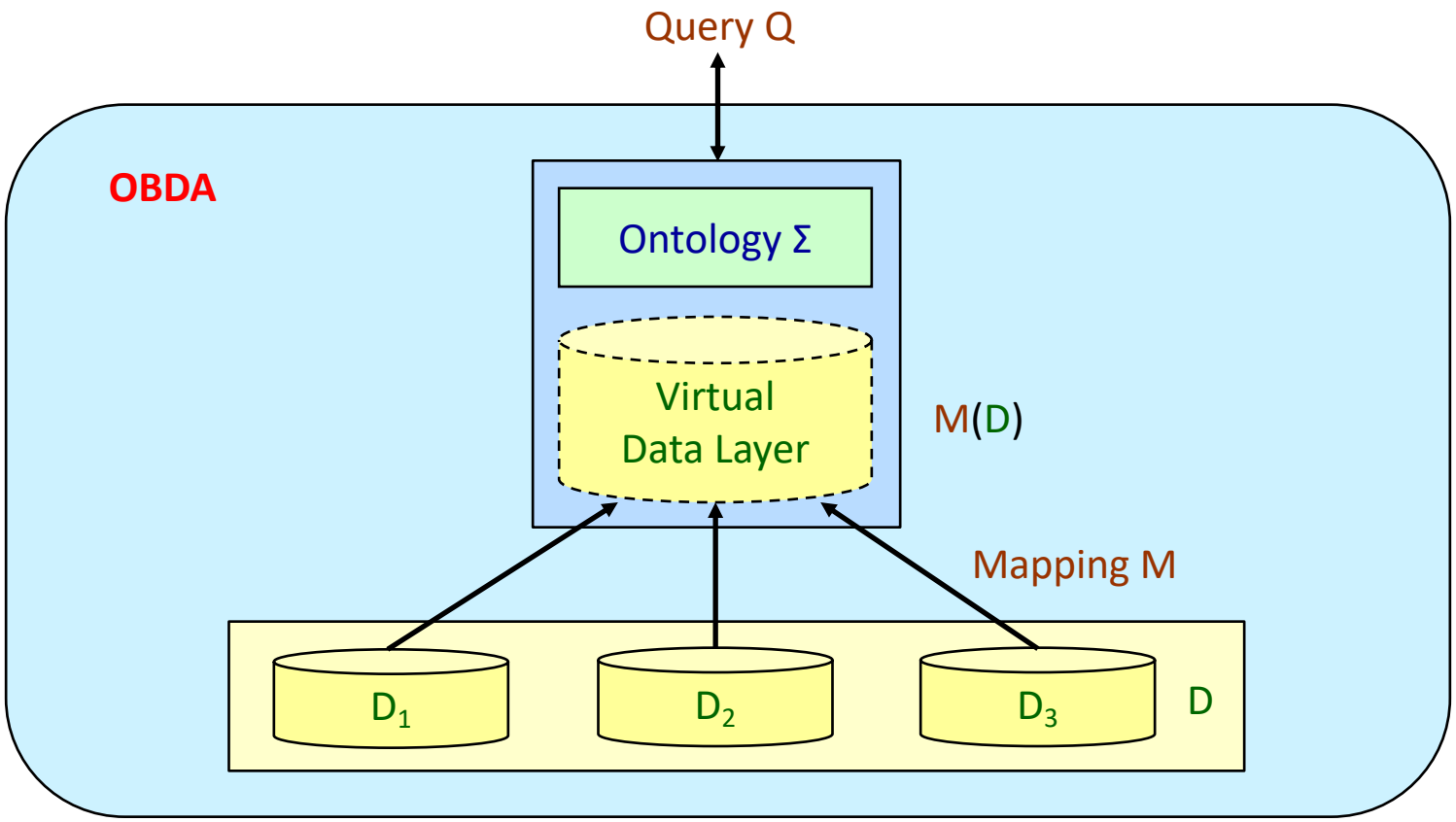
- Ontology: provides a unified conceptual “global view” of the data
- Data Sources: external and independent (possibly multiple and heterogeneous)
- Mapping: semantically link data at the sources with the ontology

Query Answering in OBDA



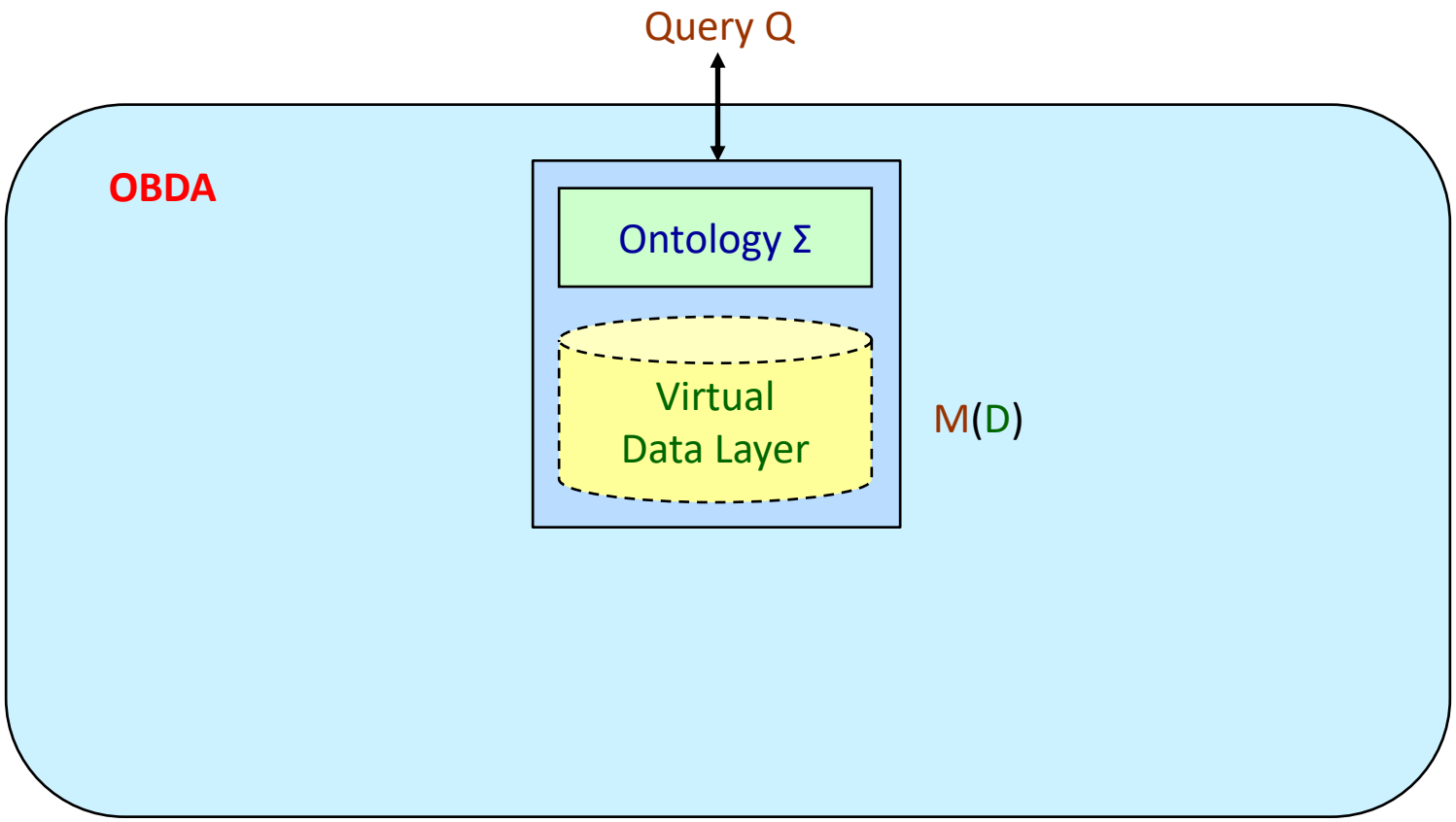
- The sources and the mapping define a virtual data layer M(D)

Query Answering in OBDA



- The sources and the mapping define a virtual data layer M(D)
- Queries are answered against the ontological database (M(D), Σ)

Query Answering in OBDA



Ontological Query Answering

MAI4CAREU

Master programmes in Artificial Intelligence 4 Careers in Europe

University of Cyprus

MAI649: PRINCIPLES OF ONTOLOGICAL DATABASES

Thank You!

Andreas Pieris

Spring 2022-2023

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423