When Conceptual Modeling meets Databases

Enrico Franconi

KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy

http://krdb.eu/franconi

EROSS 2020

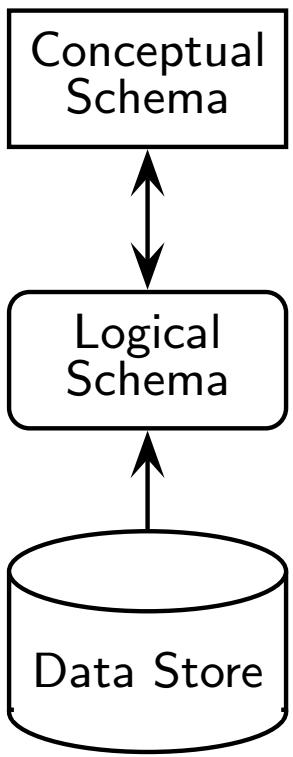
Summary of this talk

- What is a Conceptual Schema
- The role of a Conceptual Schema in Databases
- Querying via a Conceptual Schema
  - The classical Database case
  - Lossless transformations
  - Lossy transformations
    - DBoxes
    - ABoxes
  - Bad news and fixes

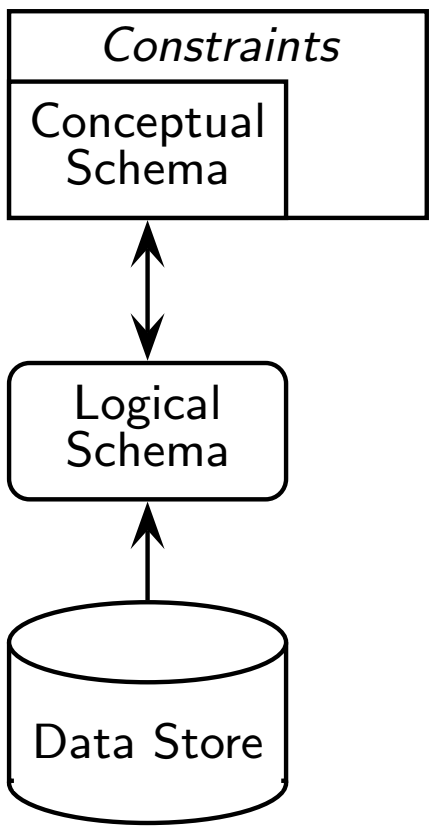
What is a Conceptual Schema

- A semantic layer gives an abstract perspective to a system, different from its original representation.
- Its purpose is to introduce a vocabulary understandable by the agents that need to interact with the system.
- A semantic layer is a formal conceptualisation of the agents' perspective: a conceptual schema (ontology, logical theory).
- A conceptual schema includes a set of constraints (the *semantics*) over the given vocabulary, which specifies what should hold in any possible configuration of the system.
- Any possible configuration of the system (an instance) conforms to the constraints expressed by the conceptual schema.
- Given a conceptual schema, a *legal instance* is a finite possible configuration of the system satisfying the constraints.
- We focus on database systems, their conceptual schemas, and their legal database instances.

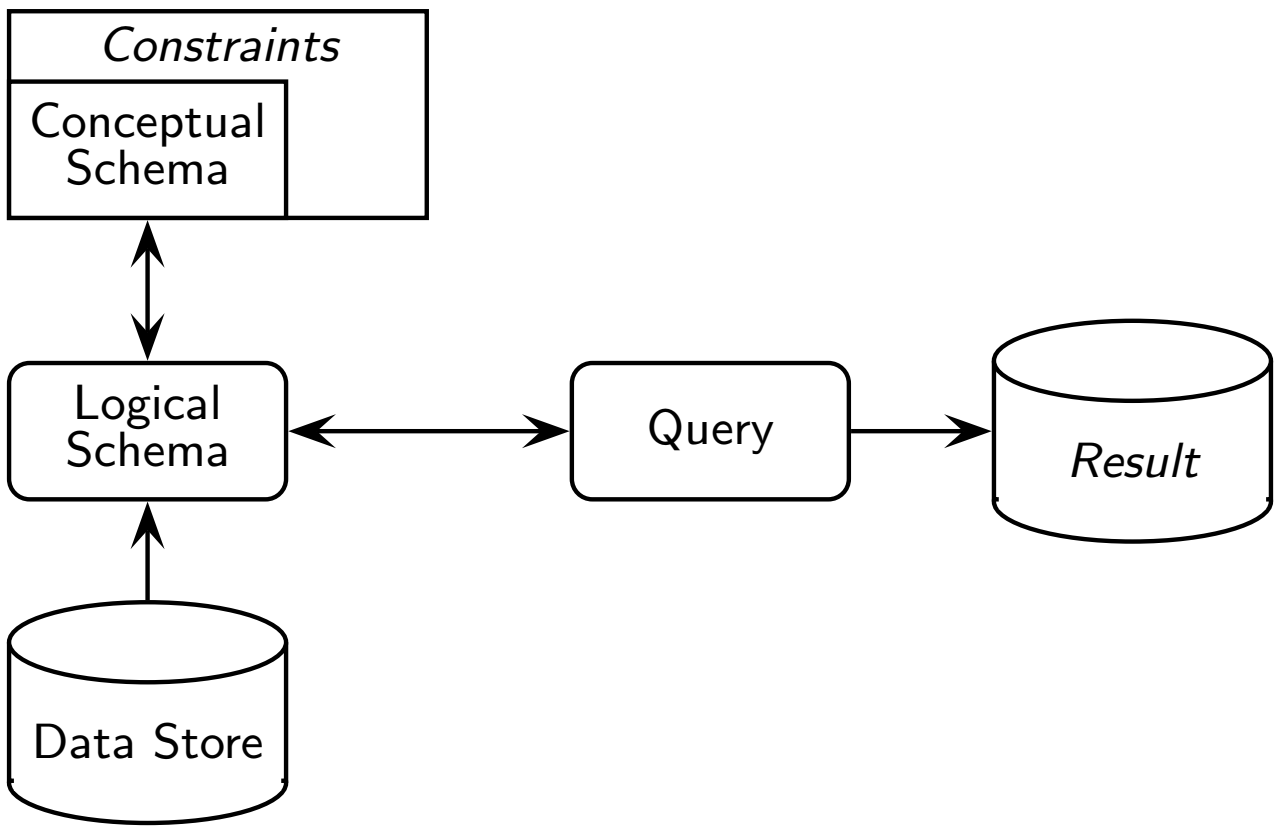
The role of a Conceptual Schema in Databases



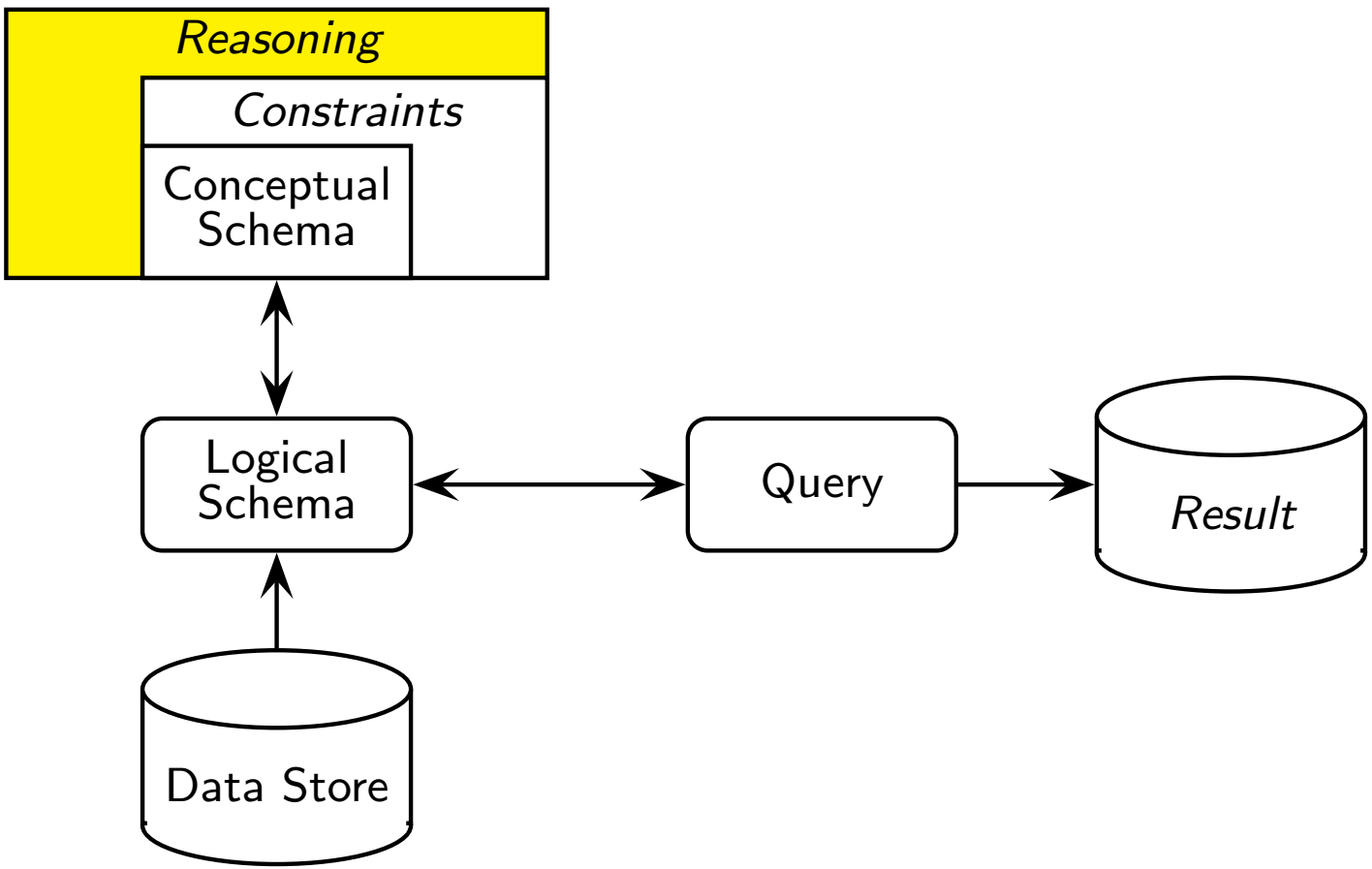
The role of a Conceptual Schema in Databases



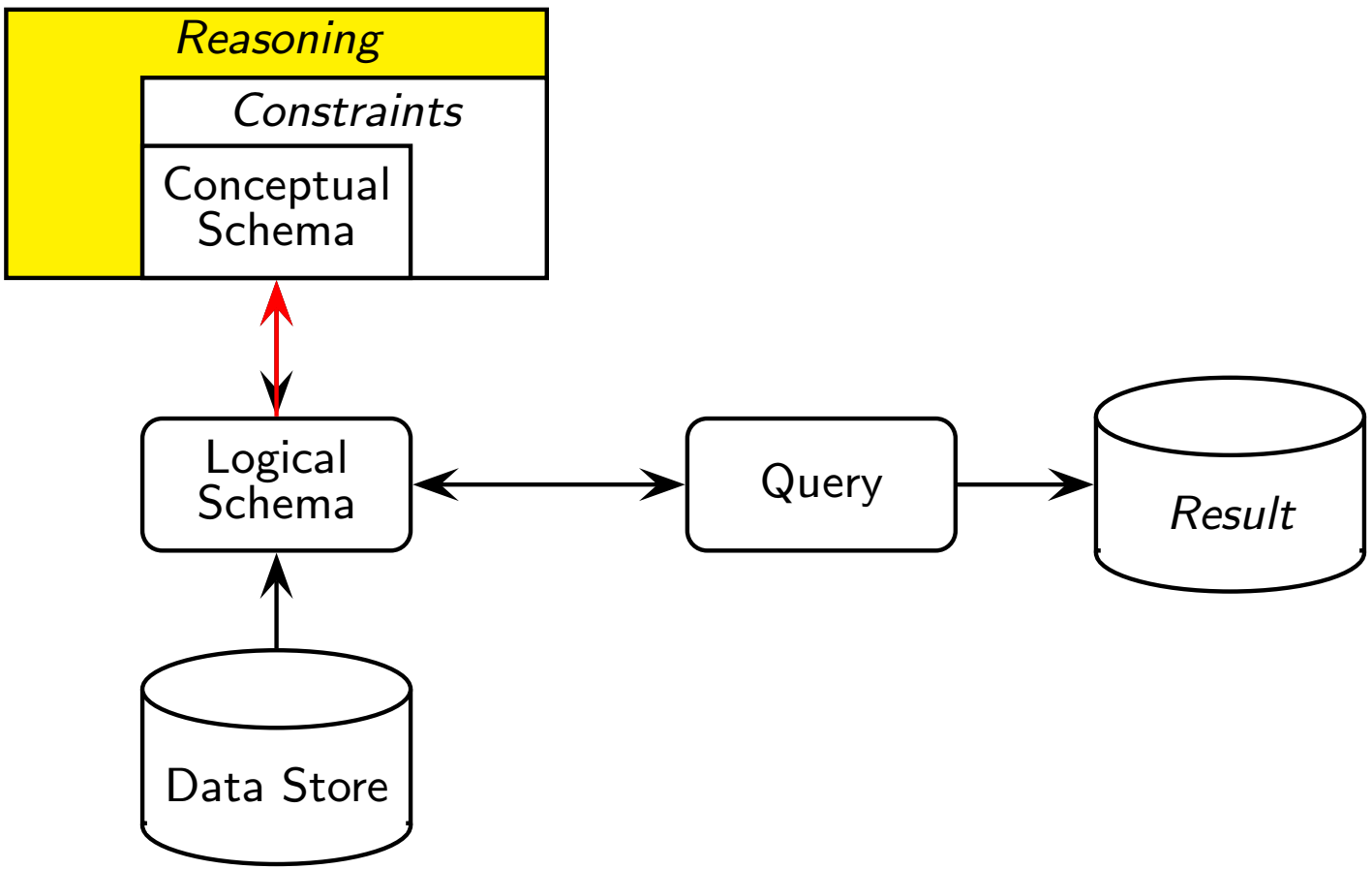
The role of a Conceptual Schema in Databases



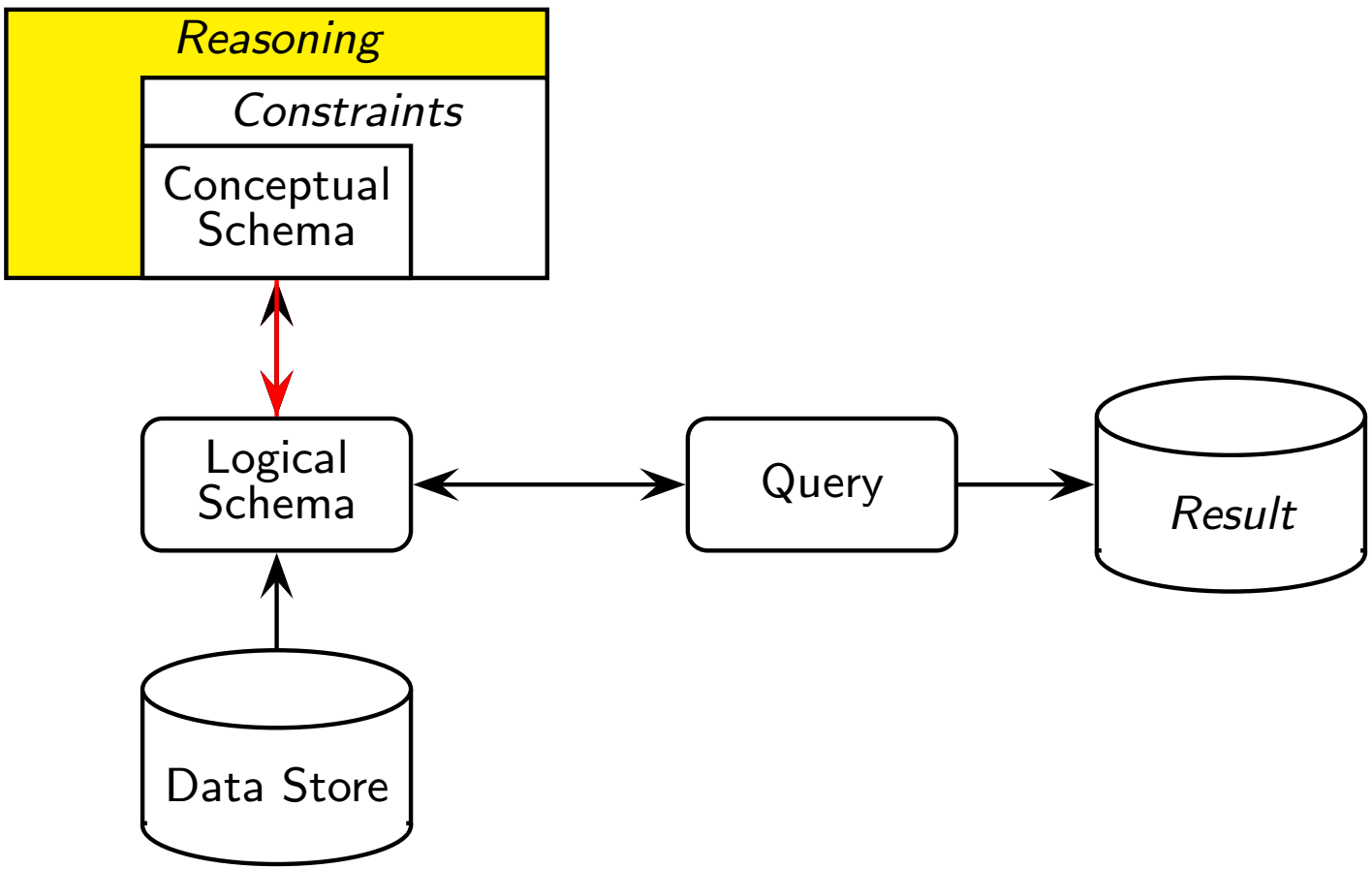
The role of a Conceptual Schema in Databases



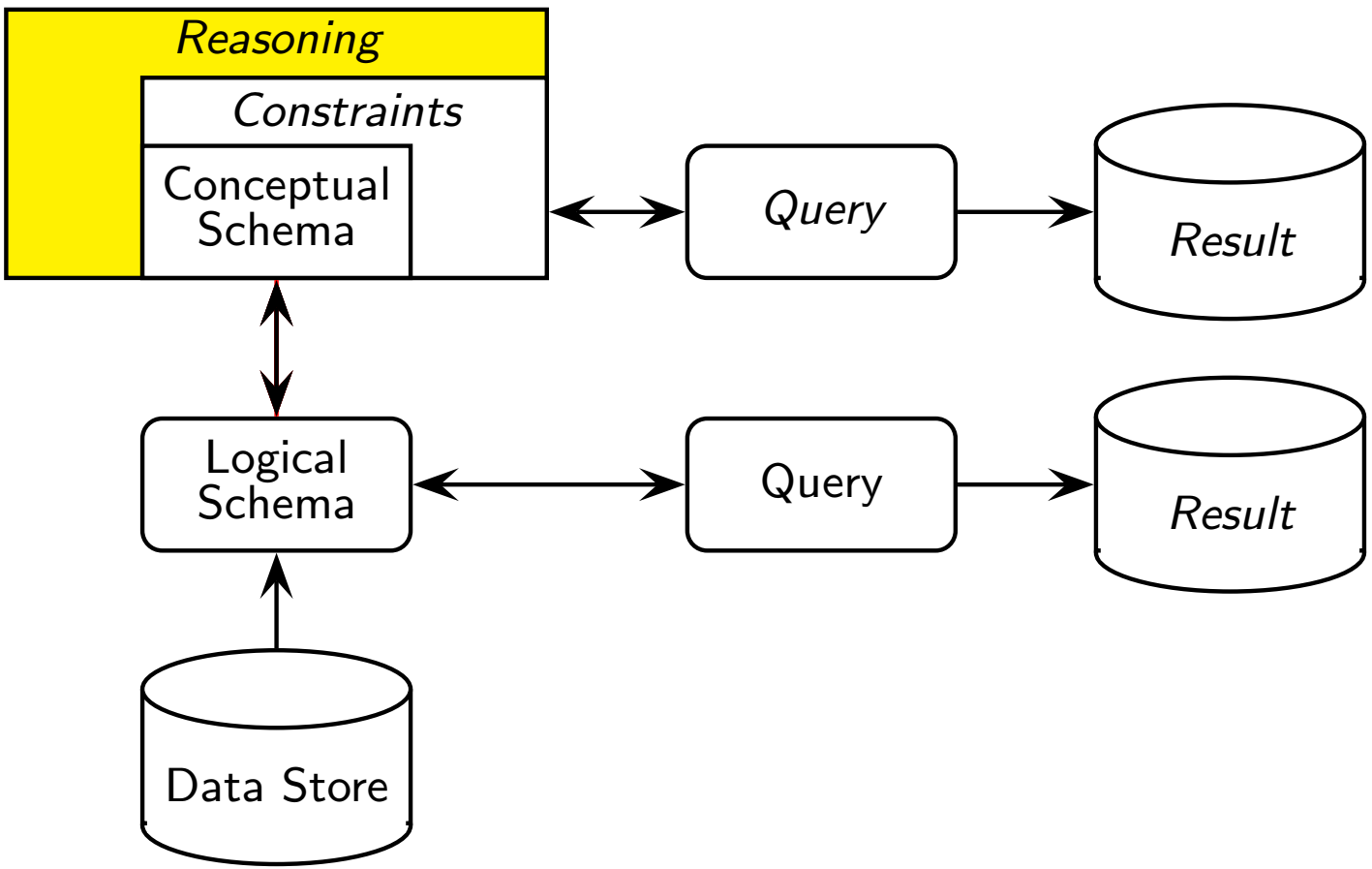
The role of a Conceptual Schema in Databases



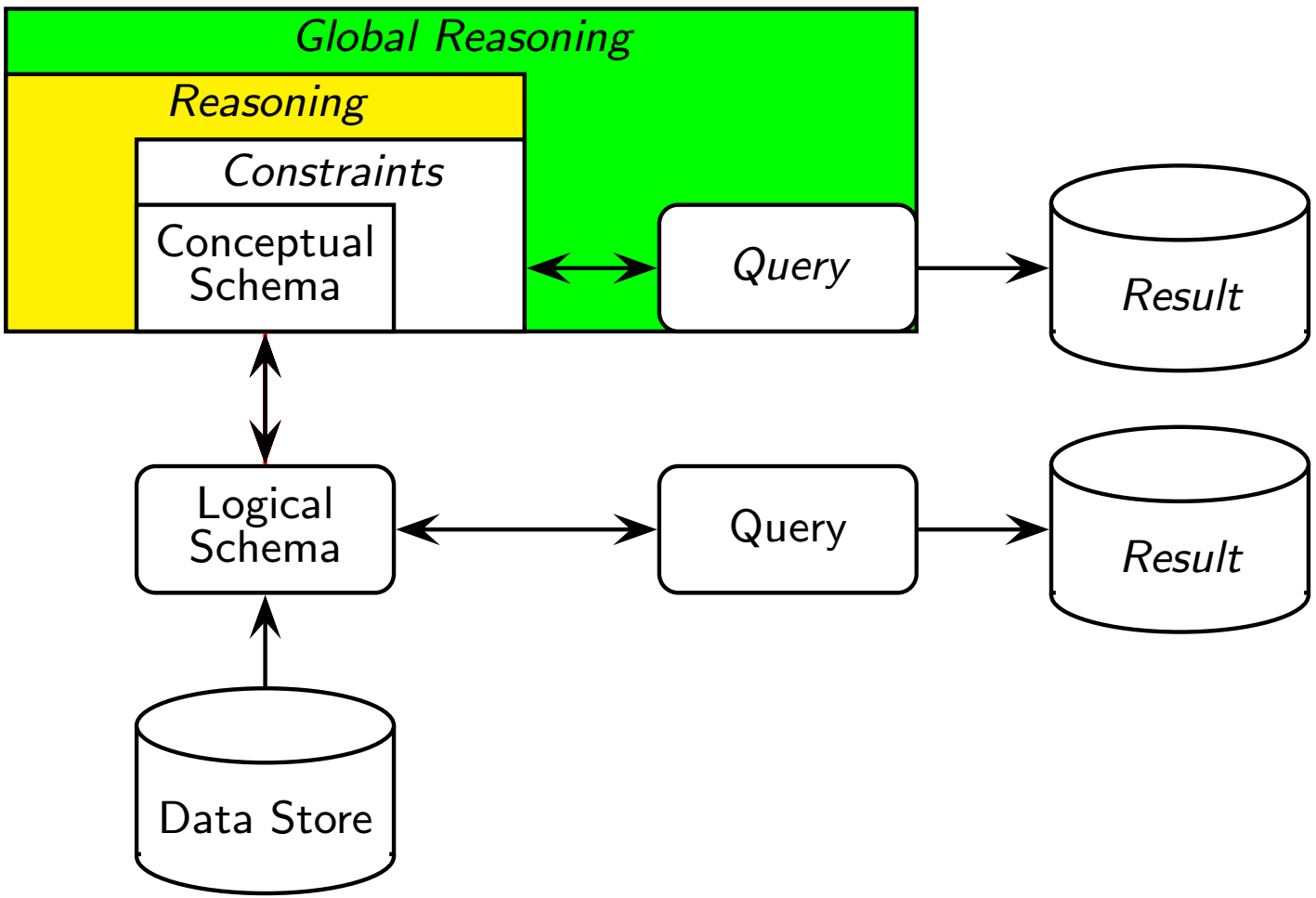
The role of a Conceptual Schema in Databases



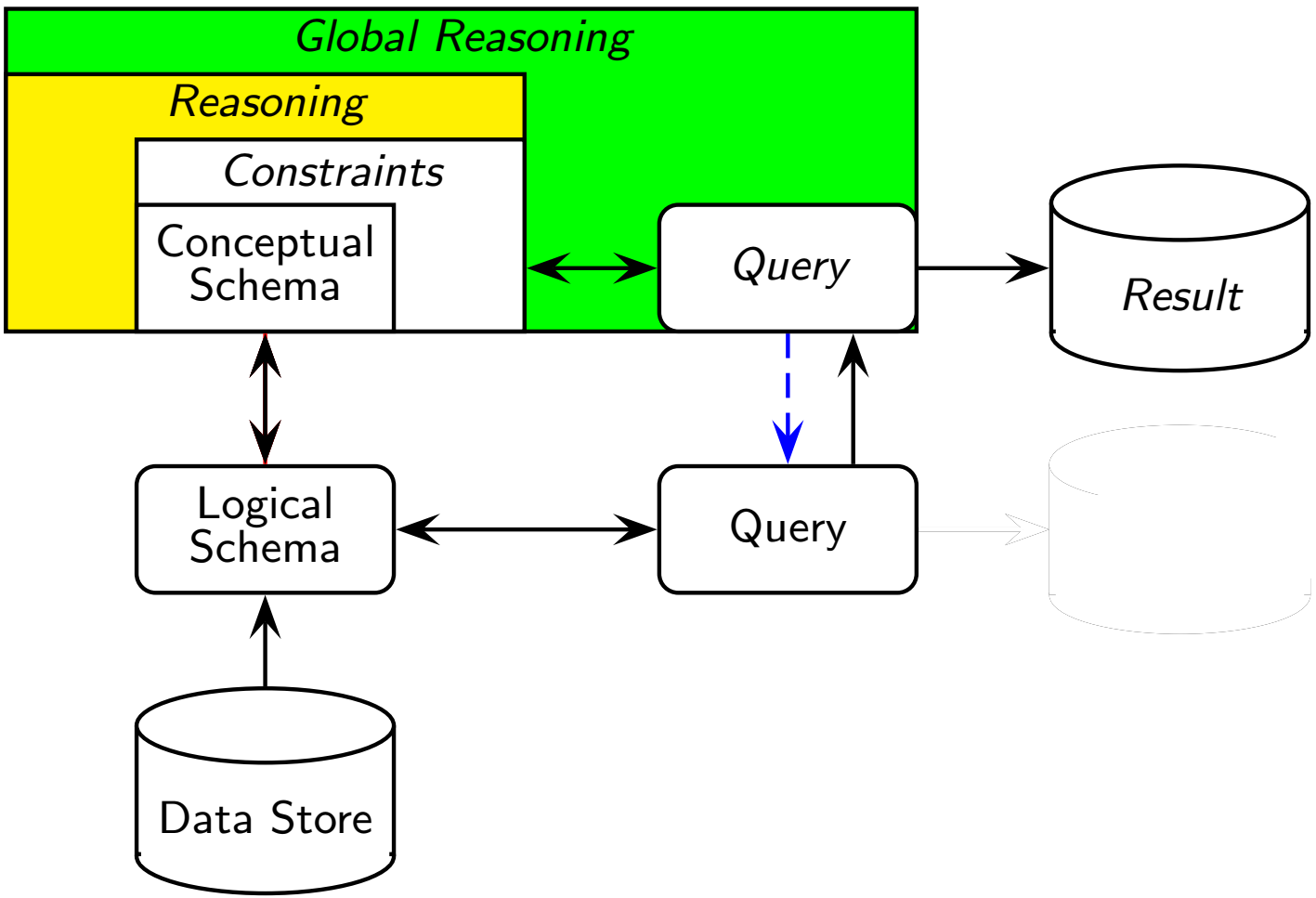
The role of a Conceptual Schema in Databases



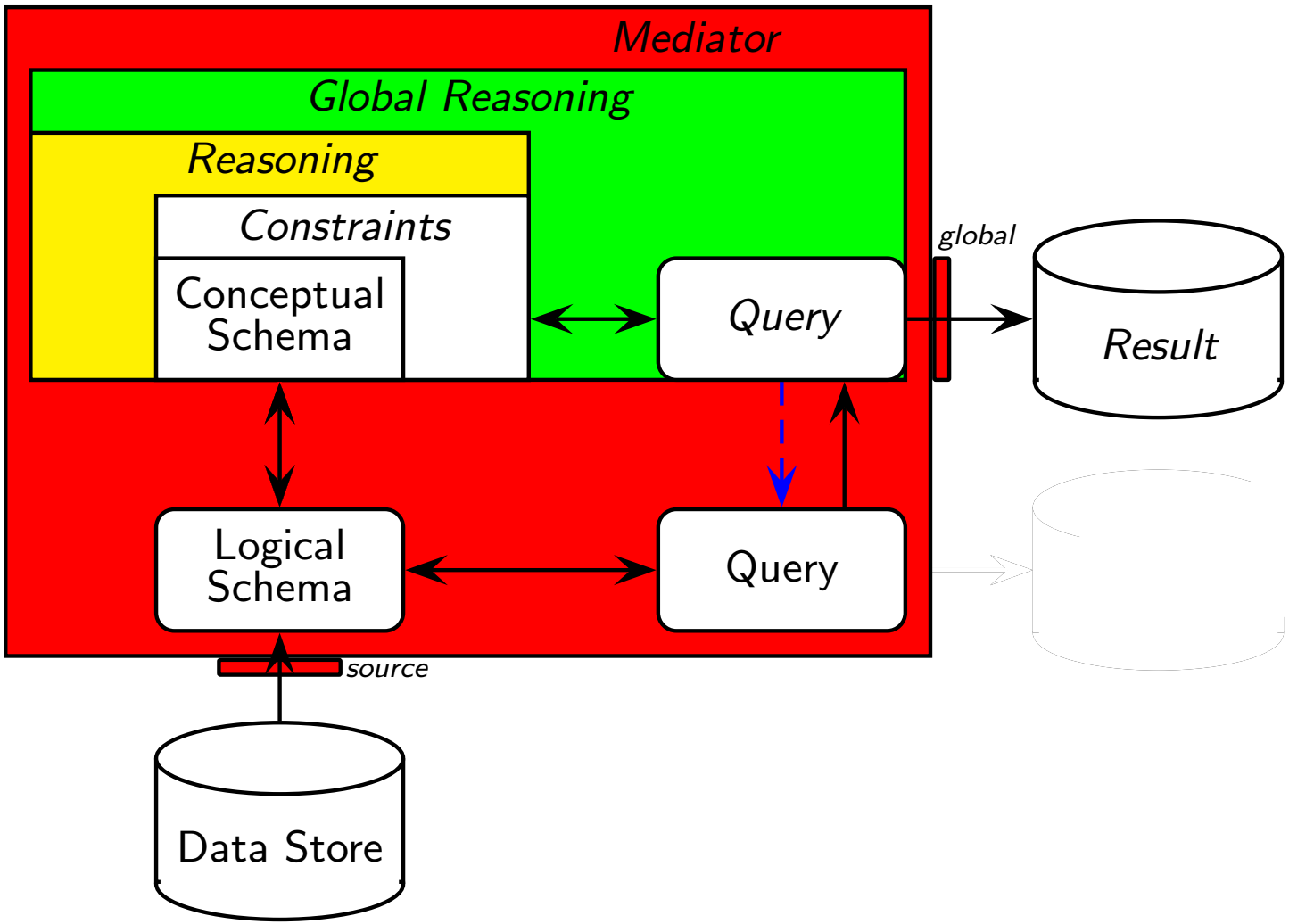
The role of a Conceptual Schema in Databases



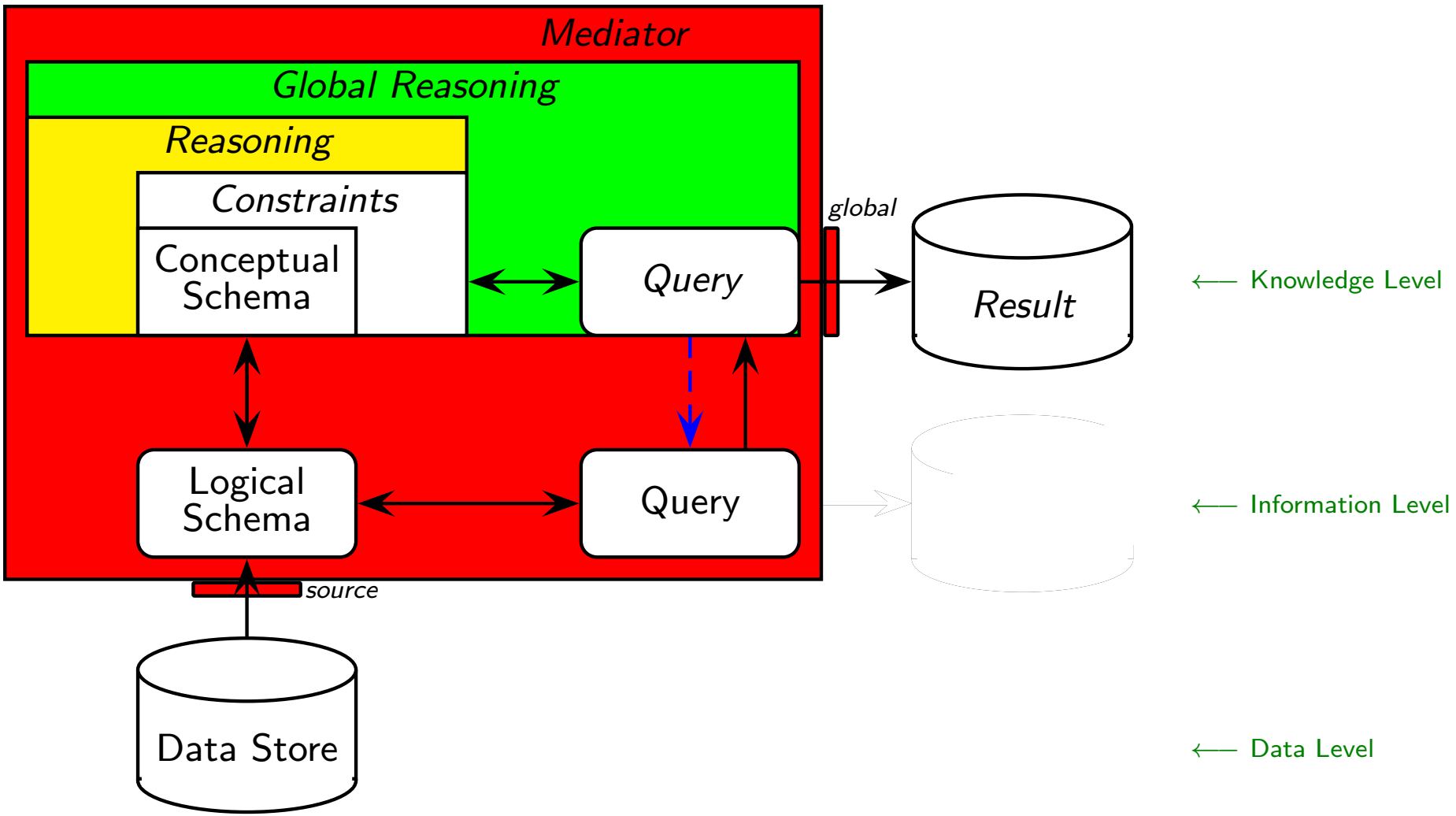
The role of a Conceptual Schema in Databases



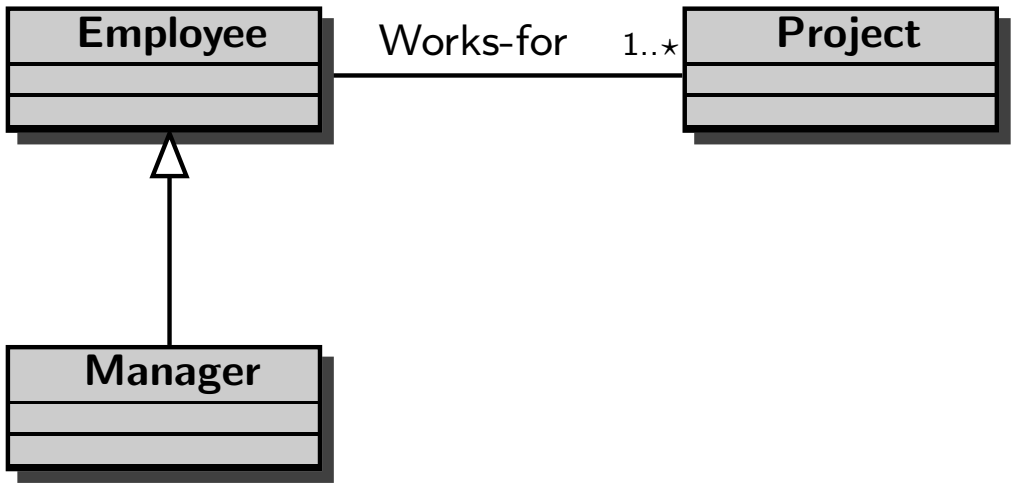
The role of a Conceptual Schema in Databases



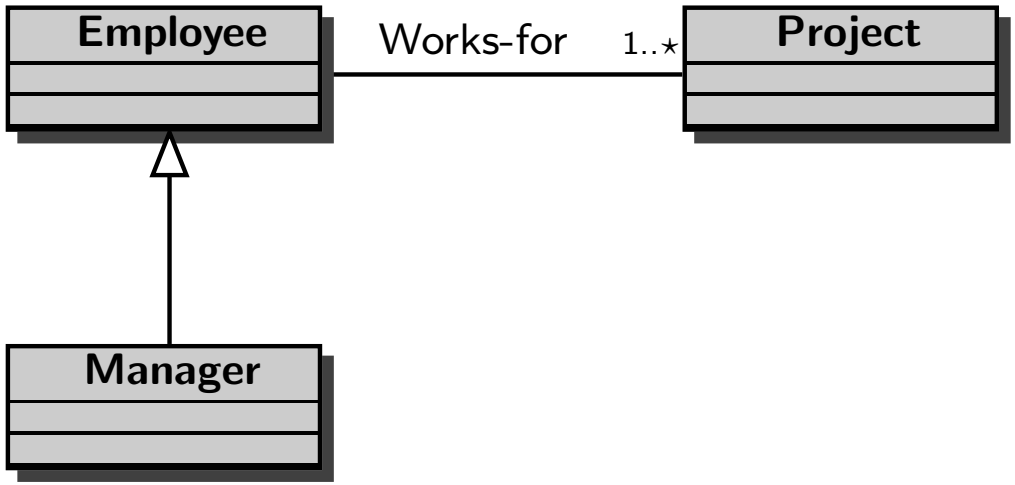
The role of a Conceptual Schema in Databases



Queries via Conceptual Schemas: the DB case

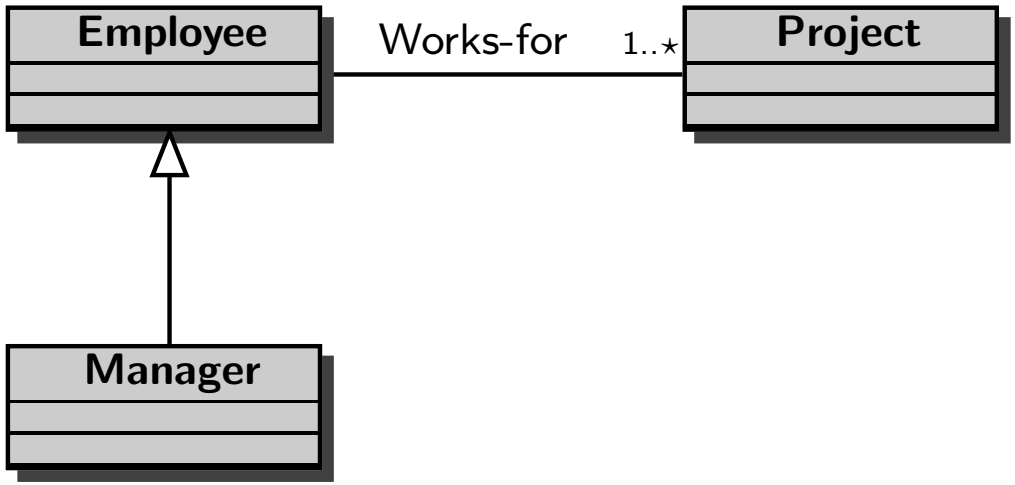


Queries via Conceptual Schemas: the DB case



Employee = { John, Mary, Paul }
Manager = { John, Paul }
Works-for = { ⟨John,Prj-A⟩, ⟨Mary,Prj-B⟩ }
Project = { Prj-A, Prj-B }

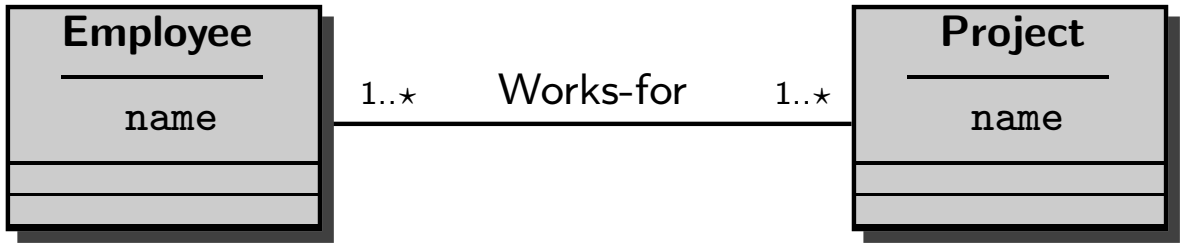
Queries via Conceptual Schemas: the DB case



Employee = { John, Mary, Paul }
Manager = { John, Paul }
Works-for = { ⟨John,Prj-A⟩, ⟨Mary,Prj-B⟩ }
Project = { Prj-A, Prj-B }

Q(x) :- ∃y. Manager(x) ∧ Works-for(x,y) ∧ Project(y)
⟹ { John }

(Conceptual) Schemas with different vocabularies



Employee = { John, Mary}
EmployeeName = { ⟨John,"John"⟩, ⟨Mary,"Mary"⟩ }
Project = { Prj-A, Prj-B }
ProjectName = { ⟨Prj-A,"Prj-A"⟩, ⟨Prj-B,"Prj-B"⟩ }
Works-for = { ⟨John,Prj-A⟩, ⟨John,Prj-B⟩, ⟨Mary,Prj-A⟩ }

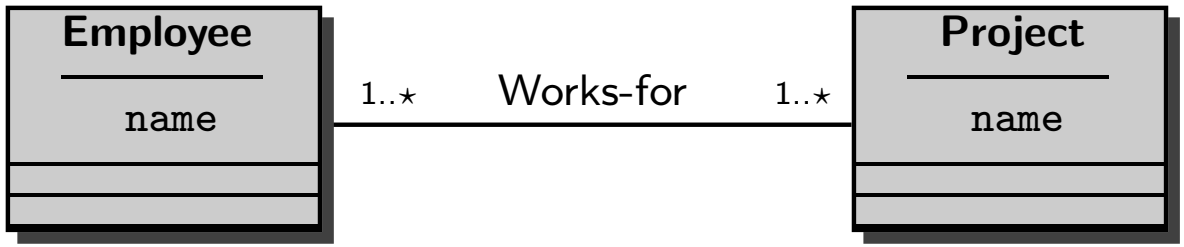
Employee-table:(EmpOID, Ename, ProjOID, Pname)
FD: ProjOID → Pname

Employee-table = {⟨John,"John", Prj-A,"Prj-A"⟩,
⟨John,"John", Prj-B,"Prj-B"⟩,
⟨mary,"Mary", Prj-A,"Prj-A"⟩}

Lossless Transformations (aka Equivalence)

- Two (conceptual) schemas with different vocabularies are lossless transformations of one another (i.e., they are equivalent) if there exist two total injective mappings from legal database instances in one schema to legal database instances in the other schema such that their composition is the identity.
- Query answering across equivalent schemas involves expanding those mappings as views.
- This notion is the formal foundation of Database Design, of Database Normalisation, and of Database Reverse Engineering.

Lossless Transformations (aka Equivalence)



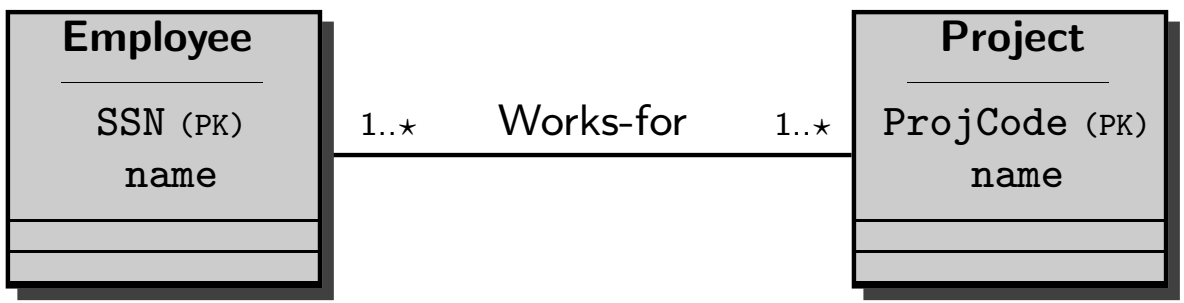
$\texttt{Employee} = \pi_{\texttt{EmpOID}}\ \texttt{Employee-table}$
$\texttt{EmployeeName} = \pi_{\texttt{EmpOID,Ename}}\ \texttt{Employee-table}$
$\texttt{Project} = \pi_{\texttt{ProjOID}}\ \texttt{Employee-table}$
$\texttt{ProjectName} = \pi_{\texttt{ProjOID,Pname}}\ \texttt{Employee-table}$
$\texttt{Works-for} = \pi_{\texttt{EmpOID,ProjOID}}\ \texttt{Employee-table}$

$\texttt{Employee-table} =$
$\quad \bowtie\ (\texttt{EmployeeName}, \texttt{ProjectName}, \texttt{Works-for})$

Employee-table:(<u>EmpOID</u>, Ename, <u>ProjOID</u>, Pname)
FD: ProjID $\rightarrow$ Pname

Object Identifiers

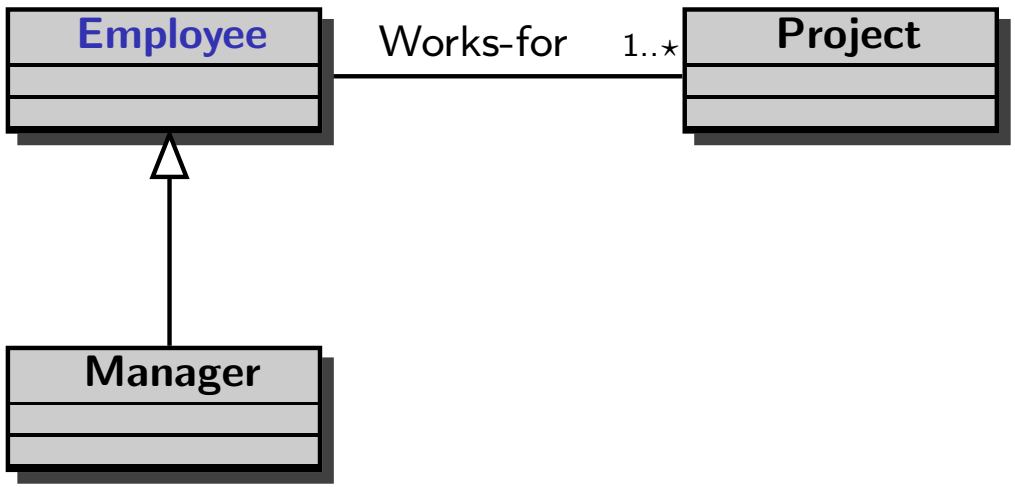
- A special lossless transformation: elimination of the Object Identifiers using Primary Keys or (better) Reference Scheme Modelling (see Terry Halpin's talk in this EROSS series).



Employee-table:(SSN, Ename, ProjCode, Pname)
FD: ProjCode $\rightarrow$ Pname

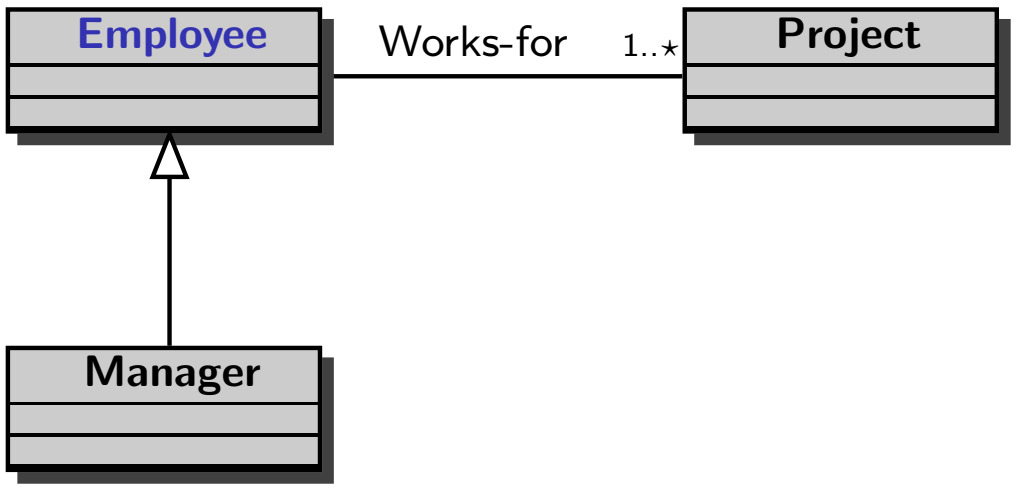
Remarkable Lossy Transformations

Queries via Conceptual Schemas: the DBox case



Manager = { John, Paul }
Works-for = { ⟨John,Prj-A⟩, ⟨Mary,Prj-B⟩ }
Project = { Prj-A, Prj-B }

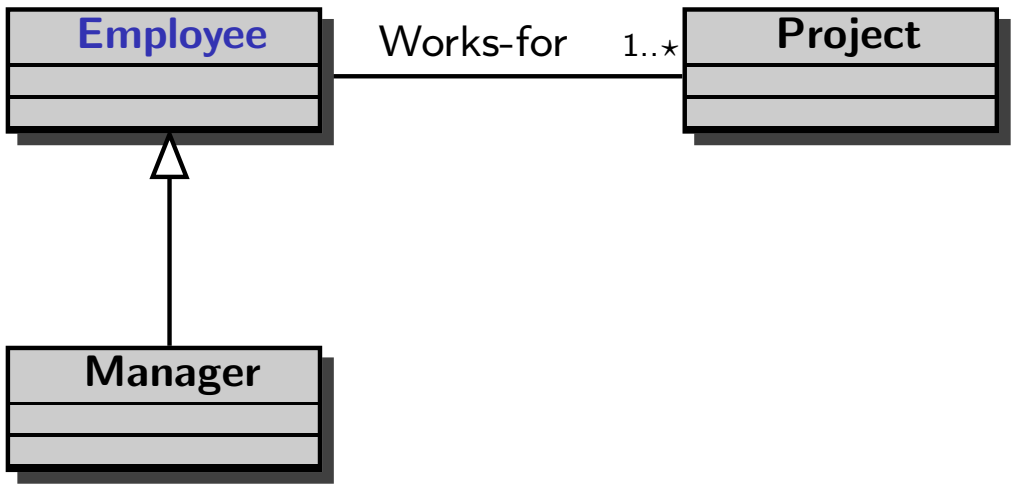
Queries via Conceptual Schemas: the DBox case



Manager = { John, Paul }
Works-for = { ⟨John,Prj-A⟩, ⟨Mary,Prj-B⟩ }
Project = { Prj-A, Prj-B }

Q(x) :- Employee(x)

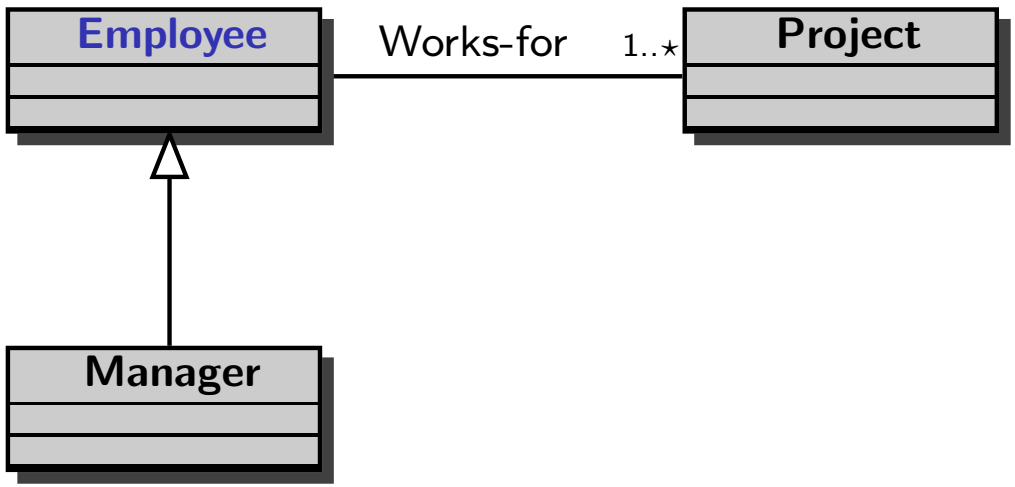
Queries via Conceptual Schemas: the DBox case



Manager = { John, Paul }
Works-for = { ⟨John,Prj-A⟩, ⟨Mary,Prj-B⟩ }
Project = { Prj-A, Prj-B }

Q(x) :- Employee(x)
$\Longrightarrow$ { John, Paul, Mary } *certain answer*

Queries via Conceptual Schemas: the DBox case

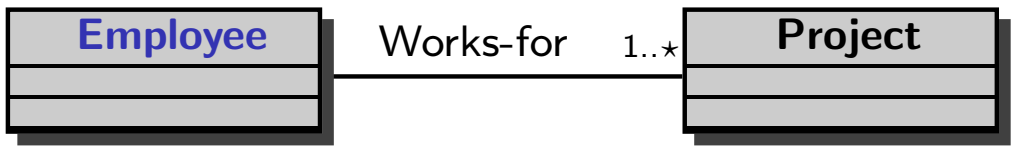


Manager = { John, Paul }
Works-for = { ⟨John,Prj-A⟩, ⟨Mary,Prj-B⟩ }
Project = { Prj-A, Prj-B }

Q(x) :- Employee(x)
$\Longrightarrow$ { John, Paul, Mary } *certain answer*

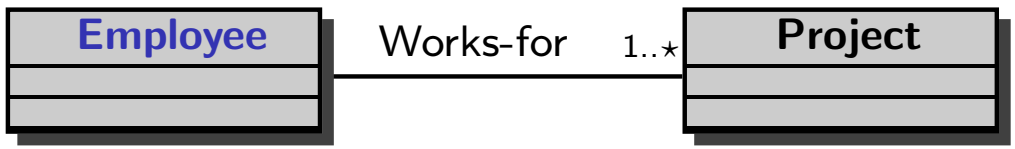
$\Longrightarrow$ Q'(x) :- Manager(x) $\vee$ $\exists$y. Works-for(x,y)

Queries via Conceptual Schemas: the ABox case



$$\texttt{Works-for} = \{\ \langle \texttt{John}, \texttt{Prj-A} \rangle,\ \langle \texttt{Mary}, \texttt{Prj-A} \rangle,\ \ldots\ \}$$
$$\texttt{Project} = \{\ \texttt{Prj-A},\ \texttt{Prj-B},\ \ldots\ \}$$

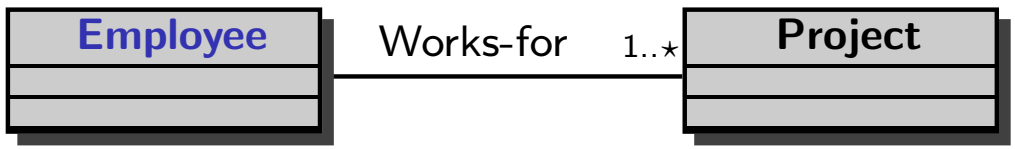
Queries via Conceptual Schemas: the ABox case



$$\texttt{Works-for} = \{\ \langle \texttt{John}, \texttt{Prj-A}\rangle,\ \langle \texttt{Mary}, \texttt{Prj-A}\rangle,\ \ldots\ \}$$
$$\texttt{Project} = \{\ \texttt{Prj-A},\ \texttt{Prj-B},\ \ldots\ \}$$

$\texttt{Q(y)} \texttt{ :- } \exists \texttt{x. Works-for(x,y)}$ *certain answer*

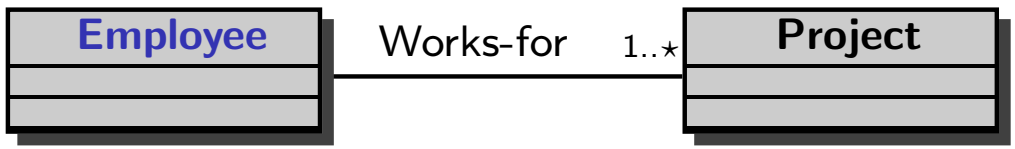
Queries via Conceptual Schemas: the ABox case



Works-for = { ⟨John,Prj-A⟩, ⟨Mary,Prj-A⟩, ... }
Project = { Prj-A, Prj-B, ... }

Q(y) :- ∃x. Works-for(x,y) *certain answer*
⟹ { Prj-A, Prj-B }

Queries via Conceptual Schemas: the ABox case



Works-for = { ⟨John,Prj-A⟩, ⟨Mary,Prj-A⟩, ... }
Project = { Prj-A, Prj-B, ... }

Q(y) :- ∃x. Works-for(x,y) *certain answer*
⟹ { Prj-A, Prj-B }

⟹ Q'(y) :- Project(y) ∨ ∃x. Works-for(x,y)

Bad news

Query answering with certain answer semantics
with DBoxes or ABoxes
with expressive conceptual modelling languages
is coNP-hard in data complexity.

Bad news

Query answering with certain answer semantics
with DBoxes or ABoxes
with expressive conceptual modelling languages
is coNP-hard in data complexity.

Fixes:

1. reduce expressivity of conceptual modelling languages and queries (OBDA approach);
2. use exact answer semantics allowing only determined queries (reduction to lossless transformations).

Bad news

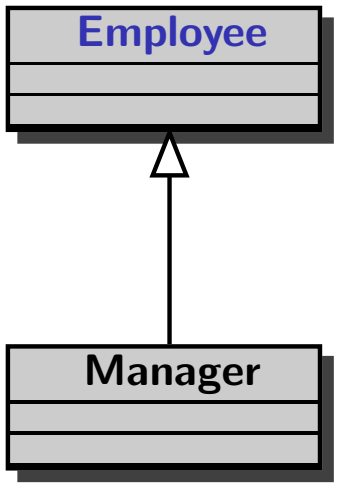
Query answering with certain answer semantics
with DBoxes or ABoxes
with expressive conceptual modelling languages
is coNP-hard in data complexity.

Fixes:

1. reduce expressivity of conceptual modelling languages and queries (OBDA approach);
2. use exact answer semantics allowing only determined queries (reduction to lossless transformations).

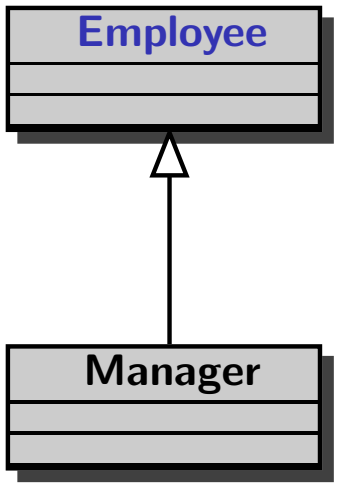
More bad news follow.

Queries with certain answer semantics



Manager = { John, Paul }

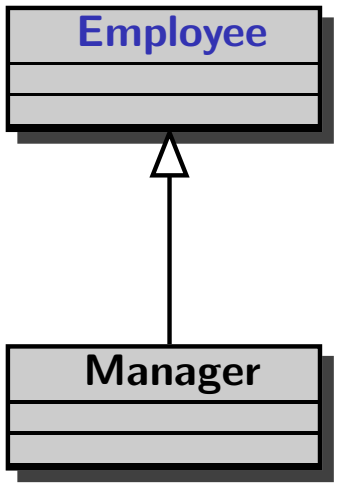
Queries with certain answer semantics




```
Manager = { John, Paul }

Q(x) :- Employee(x)
```

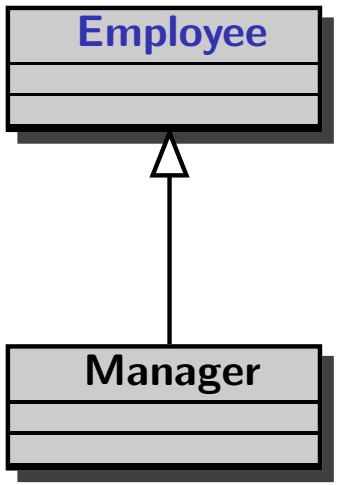
Queries with certain answer semantics



`Manager = { John, Paul }`

`Q(x) :- Employee(x)` $\Longrightarrow$ `{ John, Paul }` *certain answer*

Queries with certain answer semantics

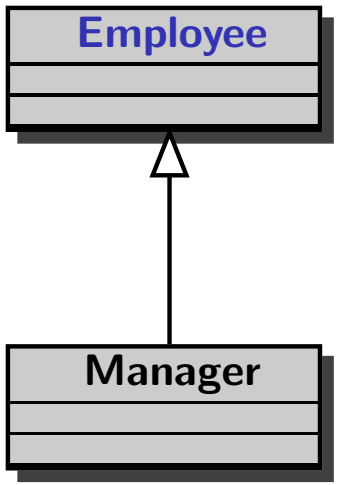


`Manager` = { `John`, `Paul` }

`Q(x) :- Employee(x)` $\Longrightarrow$ { `John`, `Paul` } *certain answer*

So: are `Managers` and `Employees` the same?

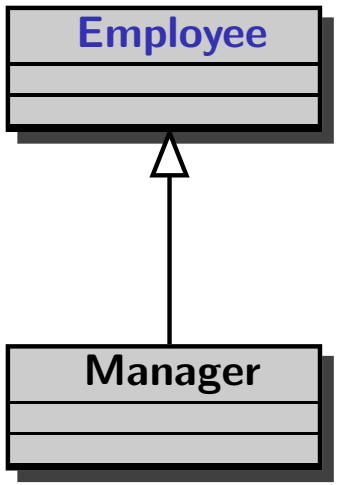
Queries with certain answer semantics



$\texttt{Manager} = \{\ \texttt{John}, \texttt{Paul}\ \}$

$\texttt{Q(x)} \texttt{ :- Employee(x)} \implies \{\ \texttt{John}, \texttt{Paul}\ \}$ *certain answer*

Queries with certain answer semantics



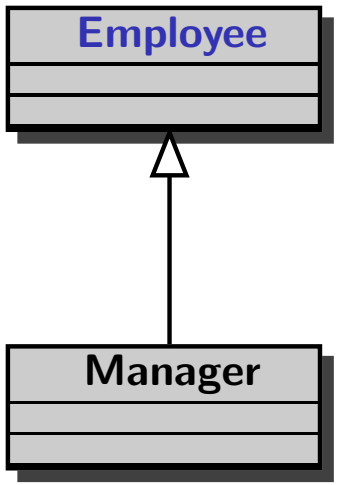
Manager = { John, Paul }

Q(x) :- Employee(x) $\Longrightarrow$ { John, Paul } *certain answer*

Q^1 :- Manager(George)

Q^2 :- Employee(George)

Queries with certain answer semantics



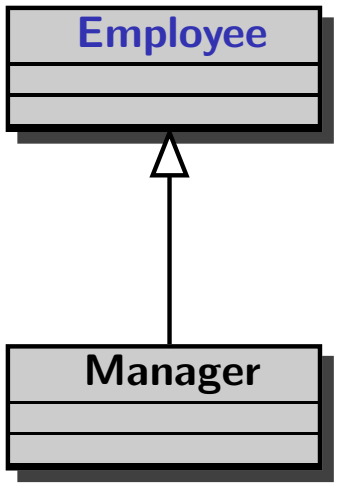
Manager = { John, Paul }

Q(x) :- Employee(x) $\Longrightarrow$ { John, Paul } *certain answer*

Q^1 :- Manager(George) $\Longrightarrow$ FALSE

Q^2 :- Employee(George)

Queries with certain answer semantics



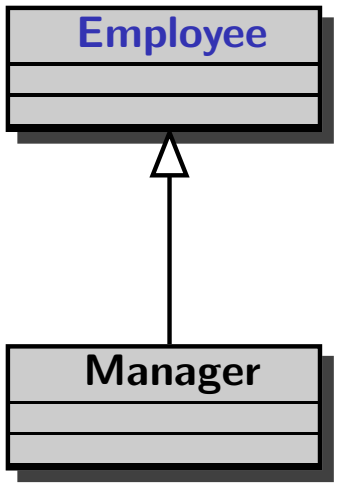
Manager = { John, Paul }

Q(x) :- Employee(x) $\Longrightarrow$ { John, Paul } *certain answer*

Q^1 :- Manager(George) $\Longrightarrow$ FALSE

Q^2 :- Employee(George) $\Longrightarrow$ DON'T KNOW

Queries with certain answer semantics



`Manager = { John, Paul }`

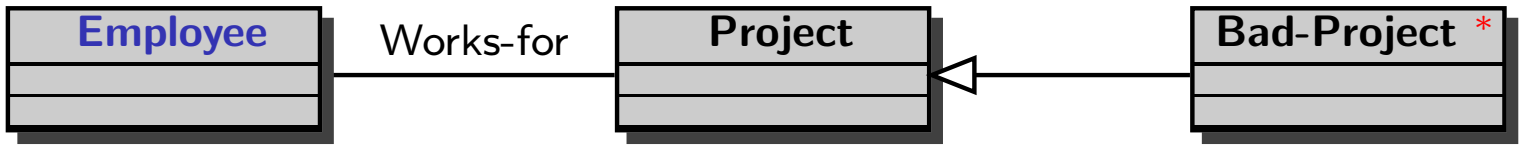
`Q(x) :- Employee(x)` $\Longrightarrow$ `{ John, Paul }` *certain answer*

Q^1 `:- Manager(George)` $\Longrightarrow$ FALSE

Q^2 `:- Employee(George)` $\Longrightarrow$ DON'T KNOW

The result of the query can not be materialised (as a view)

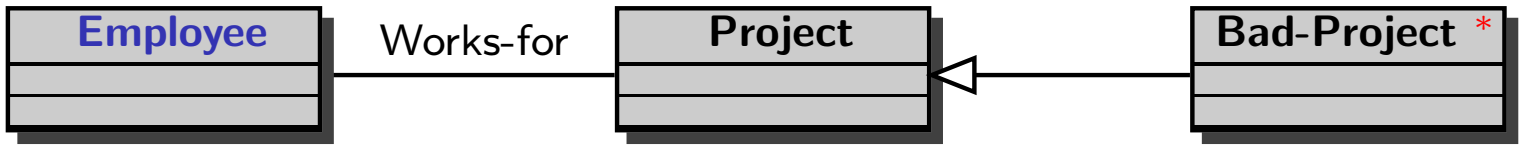
An ABox changes the DB semantics



`Bad-Project` defined with a rule/view:

* $\texttt{Bad-Project(x)} \leftrightarrow \texttt{Project(x)} \wedge \neg \exists \texttt{y}.\ \texttt{Works-for(y,x)}$

An ABox changes the DB semantics



Bad-Project defined with a rule/view:

* Bad-Project(x) $\leftrightarrow$ Project(x)$\wedge$ $\neg\exists$y. Works-for(y,x)

- DBox:

  Works-for $=$ { $\langle$John,Prj-A$\rangle$, $\langle$Mary,Prj-A$\rangle$ }

  Project $=$ { Prj-A, Prj-B }

- Q(X) :- Bad-Project(X)

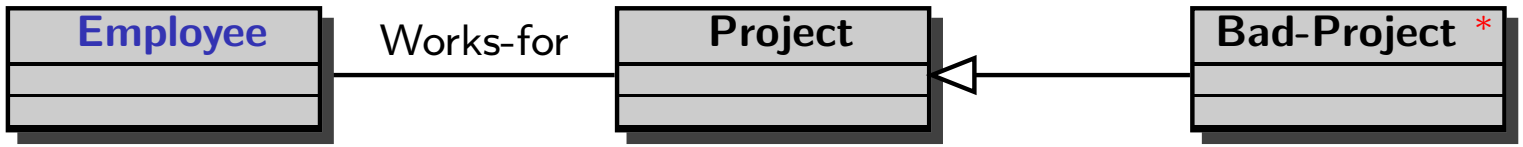
- ABox:

  Works-for = { $\langle$John,Prj-A$\rangle$, $\langle$Mary,Prj-A$\rangle$, ... }

  Project = { Prj-A, Prj-B, ... }

- Q(X) :- Bad-Project(X)

An ABox changes the DB semantics



Bad-Project defined with a rule/view:

* Bad-Project(x) $\leftrightarrow$ Project(x)$\wedge$ $\neg\exists$y. Works-for(y,x)

- DBox:

  Works-for $=$ { $\langle$John,Prj-A$\rangle$, $\langle$Mary,Prj-A$\rangle$ }

  Project $=$ { Prj-A, Prj-B }

- Q(X) :- Bad-Project(X)

  $\Longrightarrow$ { Prj-B }

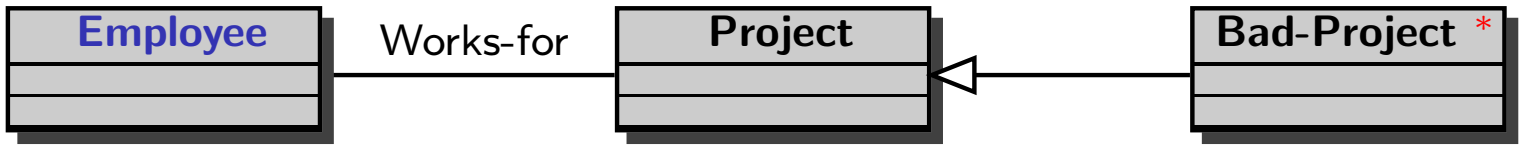
- ABox:

  Works-for = { $\langle$John,Prj-A$\rangle$, $\langle$Mary,Prj-A$\rangle$, ... }

  Project = { Prj-A, Prj-B, ... }

- Q(X) :- Bad-Project(X)

An ABox changes the DB semantics



Bad-Project defined with a rule/view:

* Bad-Project(x) $\leftrightarrow$ Project(x)$\wedge$ $\neg\exists$y. Works-for(y,x)

- DBox:

  Works-for $=$ { $\langle$John,Prj-A$\rangle$, $\langle$Mary,Prj-A$\rangle$ }

  Project $=$ { Prj-A, Prj-B }

- Q(X) :- Bad-Project(X)

  $\Longrightarrow$ { Prj-B }

- ABox:

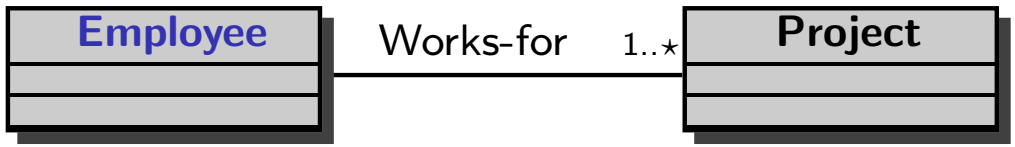
  Works-for = { $\langle$John,Prj-A$\rangle$, $\langle$Mary,Prj-A$\rangle$, ... }

  Project = { Prj-A, Prj-B, ... }

- Q(X) :- Bad-Project(X)

  $\Longrightarrow$ { }

Compositionality of queries

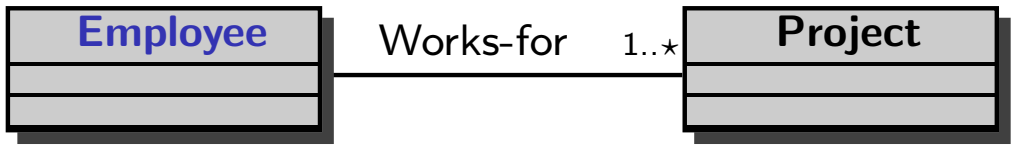


- ABox:

  Works-for = { ⟨John,Prj-A⟩, ... }

  Project = { Prj-A, Prj-B, ... }

Compositionality of queries



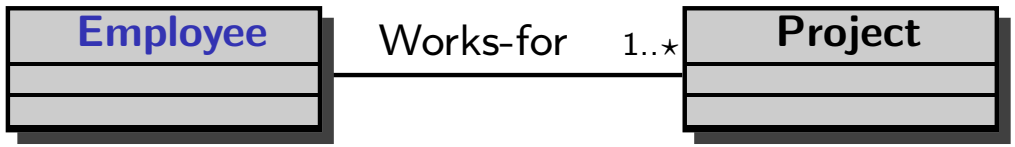
- ABox:

  $\texttt{Works-for} = \{\ \langle \texttt{John}, \texttt{Prj-A} \rangle, \ \ldots\ \}$

  $\texttt{Project} = \{\ \texttt{Prj-A}, \ \texttt{Prj-B}, \ \ldots\ \}$

- $\texttt{Q(x)} \ \texttt{:-} \ \exists \texttt{y}.\, \texttt{Works-for(y,x)} \qquad \texttt{Q} = \pi_2\, \texttt{Works-for}$

Compositionality of queries



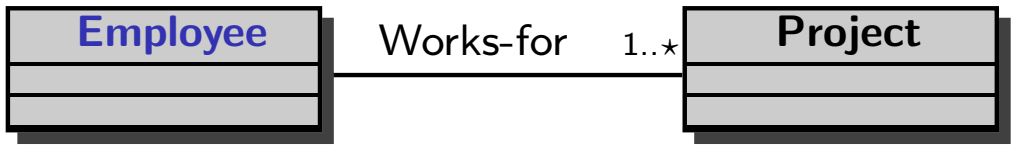
- ABox:

  $\texttt{Works-for} = \{ \langle \texttt{John}, \texttt{Prj-A} \rangle, \ldots \}$

  $\texttt{Project} = \{ \texttt{Prj-A}, \texttt{Prj-B}, \ldots \}$

- $\texttt{Q(x)} \texttt{ :- } \exists \texttt{y}.\, \texttt{Works-for(y,x)} \qquad \texttt{Q} = \pi_2\, \texttt{Works-for}$
  - $\texttt{Q} = \text{EVAL}(\pi_2\, \texttt{Works-for})$
  - $\texttt{Q} = \pi_2\, (\text{EVAL}(\texttt{Works-for}))$

Compositionality of queries



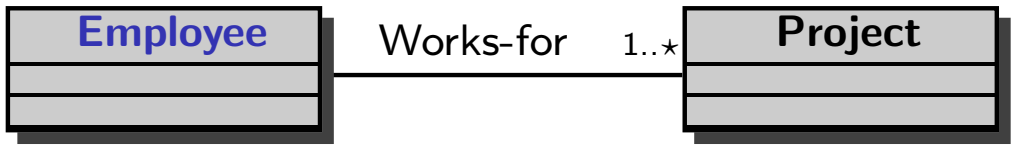
- ABox:

  $\texttt{Works-for} = \{ \langle \texttt{John}, \texttt{Prj-A} \rangle, \ldots \}$

  $\texttt{Project} = \{ \texttt{Prj-A}, \texttt{Prj-B}, \ldots \}$

- $\texttt{Q(x)} \texttt{ :- } \exists \texttt{y}.\, \texttt{Works-for(y,x)} \qquad \texttt{Q} = \pi_2\, \texttt{Works-for}$
  - $\texttt{Q} = \text{EVAL}(\pi_2\, \texttt{Works-for})$
    $\Longrightarrow \{ \texttt{Prj-A}, \texttt{Prj-B} \}$
  - $\texttt{Q} = \pi_2\, (\text{EVAL}(\texttt{Works-for}))$

Compositionality of queries



- ABox:

  $\texttt{Works-for} = \{ \langle \texttt{John}, \texttt{Prj-A} \rangle, \ldots \}$

  $\texttt{Project} = \{ \texttt{Prj-A}, \texttt{Prj-B}, \ldots \}$

- $\texttt{Q(x)} \texttt{ :- } \exists \texttt{y}.\, \texttt{Works-for(y,x)} \qquad \texttt{Q} = \pi_2\, \texttt{Works-for}$
  - $\texttt{Q} = \text{EVAL}(\pi_2\, \texttt{Works-for})$
    $\Longrightarrow \{ \texttt{Prj-A}, \texttt{Prj-B} \}$
  - $\texttt{Q} = \pi_2\, (\text{EVAL}(\texttt{Works-for}))$
    $\Longrightarrow \{ \texttt{Prj-A} \}$

Conclusions

Conclusions

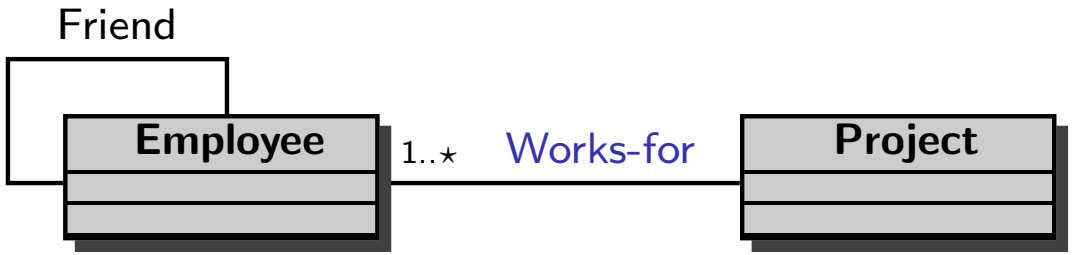
Do you want to exploit conceptual schema knowledge
(i.e., constraints or an ontology)
in your data intensive application?

Conclusions

Do you want to exploit conceptual schema knowledge
(i.e., constraints or an ontology)
in your data intensive application?

Pay attention!

Data Complexity of Query Answering with DBoxes



$\texttt{Friend} = \{\langle \texttt{John}, \texttt{Mary} \rangle, \texttt{etc}\}$

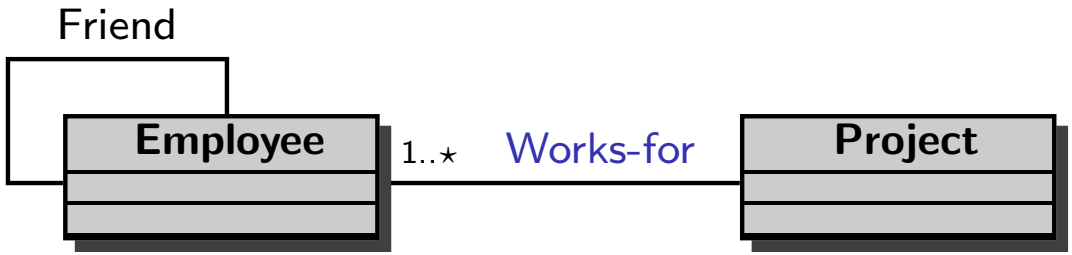
$\texttt{Employee} = \{\texttt{John}, \texttt{Mary}, \texttt{etc}\}$

$\texttt{Project} = \{\ \texttt{Prj-A}, \texttt{Prj-B}, \texttt{Prj-C}\ \}$

$\texttt{Q} \texttt{ :- } \texttt{Works-for}(e_1, \texttt{p}) \wedge \texttt{Works-for}(e_2, \texttt{p}) \wedge \texttt{Friend}(e_1, e_2)$

Are there two friends working for the same project?

Data Complexity of Query Answering with DBoxes



$\texttt{Friend} = \{\langle \texttt{John}, \texttt{Mary} \rangle, \texttt{etc}\}$

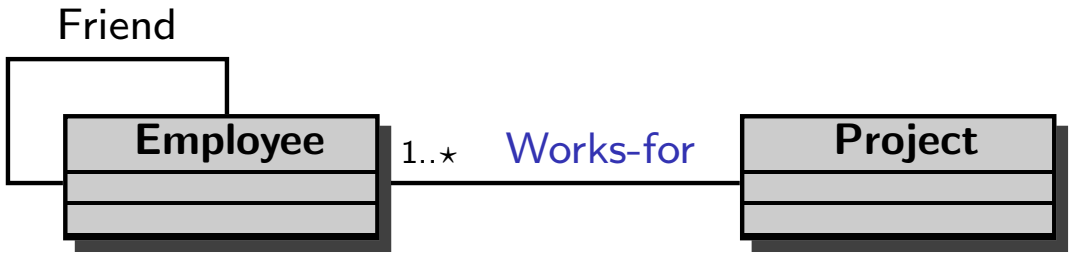
$\texttt{Employee} = \{\texttt{John}, \texttt{Mary}, \texttt{etc}\}$

$\texttt{Project} = \{\ \texttt{Prj-A}, \texttt{Prj-B}, \texttt{Prj-C}\ \}$

$\texttt{Q} \texttt{ :- } \texttt{Works-for}(e_1, \texttt{p}) \wedge \texttt{Works-for}(e_2, \texttt{p}) \wedge \texttt{Friend}(e_1, e_2)$

Are there two friends working for the same project?

Data Complexity of Query Answering with DBoxes



$\texttt{Friend} = \{\langle \texttt{John}, \texttt{Mary} \rangle, \texttt{etc}\}$

$\texttt{Employee} = \{\texttt{John}, \texttt{Mary}, \texttt{etc}\}$

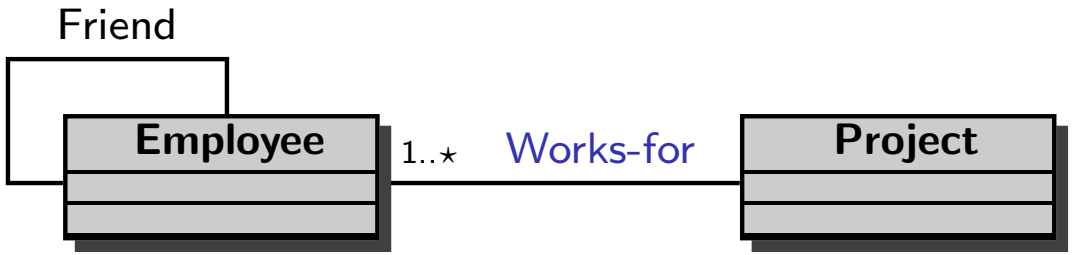
$\texttt{Project} = \{\ \texttt{Prj-A}, \texttt{Prj-B}, \texttt{Prj-C}\ \}$

$\texttt{Q} \texttt{ :- } \texttt{Works-for}(e_1, \texttt{p}) \wedge \texttt{Works-for}(e_2, \texttt{p}) \wedge \texttt{Friend}(e_1, e_2)$

Are there two friends working for the same project?

- YES: in any legal database instance, there are at least two friends working for the same project.

Data Complexity of Query Answering with DBoxes



$\texttt{Friend} = \{\langle \texttt{John}, \texttt{Mary} \rangle, \texttt{etc}\}$

$\texttt{Employee} = \{\texttt{John}, \texttt{Mary}, \texttt{etc}\}$

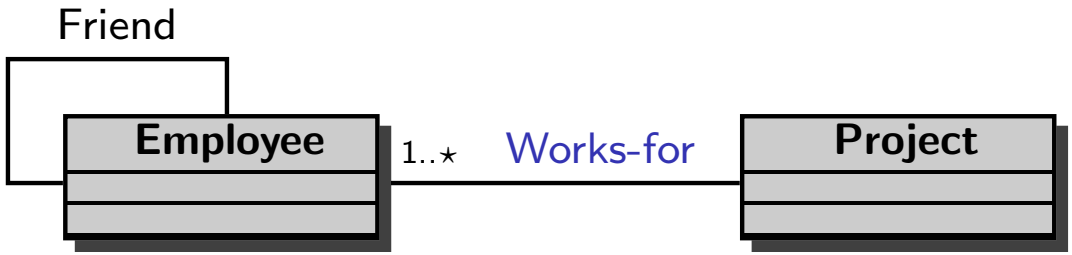
$\texttt{Project} = \{\ \texttt{Prj-A}, \texttt{Prj-B}, \texttt{Prj-C}\ \}$

$\texttt{Q} \texttt{ :- } \texttt{Works-for}(e_1, \texttt{p}) \wedge \texttt{Works-for}(e_2, \texttt{p}) \wedge \texttt{Friend}(e_1, e_2)$

Are there two friends working for the same project?

- YES: in any legal database instance, there are at least two friends working for the same project.
- NO: there is at least a legal database instance in which no two friends work for the same project.

Data Complexity of Query Answering with DBoxes



$\texttt{Friend} = \{\langle \texttt{John}, \texttt{Mary} \rangle, \texttt{etc}\}$

$\texttt{Employee} = \{\texttt{John}, \texttt{Mary}, \texttt{etc}\}$

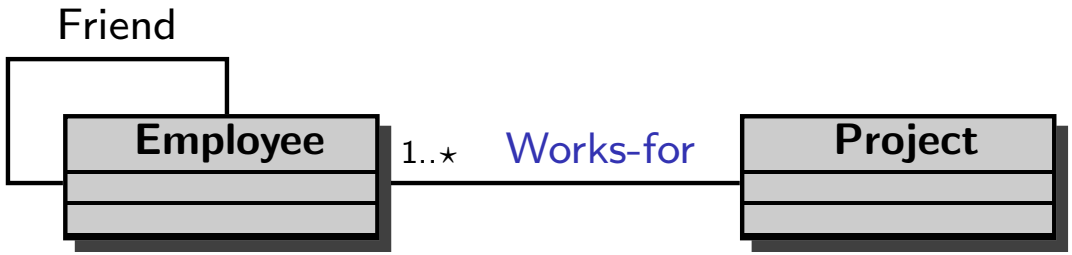
$\texttt{Project} = \{\ \texttt{Prj-A}, \texttt{Prj-B}, \texttt{Prj-C}\ \}$

$\texttt{Q} \texttt{ :- } \texttt{Works-for}(e_1, \texttt{p}) \wedge \texttt{Works-for}(e_2, \texttt{p}) \wedge \texttt{Friend}(e_1, e_2)$

Are there two friends working for the same project?

- YES: in any legal database instance, there are at least two friends working for the same project.
- NO: there is at least a legal database instance in which no two friends work for the same project.
- NOTE: with *ABox semantics* the answer is always NO, since there is at least a legal database instance with *enough* distinct projects so that no two friends work for the same project.

Data Complexity of Query Answering with DBoxes



$\texttt{Friend} = \{\langle \texttt{John}, \texttt{Mary} \rangle, \texttt{etc}\}$

$\texttt{Employee} = \{\texttt{John}, \texttt{Mary}, \texttt{etc}\}$

$\texttt{Project} = \{\ \texttt{Prj-A}, \texttt{Prj-B}, \texttt{Prj-C}\ \}$

$\texttt{Q} \texttt{ :- } \texttt{Works-for}(e_1, \texttt{p}) \wedge \texttt{Works-for}(e_2, \texttt{p}) \wedge \texttt{Friend}(e_1, e_2)$

Are there two friends working for the same project?

- YES: in any legal database instance, there are at least two friends working for the same project.
- NO: there is at least a legal database instance in which no two friends work for the same project.
- NOTE: with *ABox semantics* the answer is always NO, since there is at least a legal database instance with *enough* distinct projects so that no two friends work for the same project.

This encodes 3-colorability of maps $\rightarrow$ co-NP-hard