Description Logics for Conceptual Design, Information Access, and Ontology Integration

(ISWC-2002)

Enrico Franconi

`franconi@cs.man.ac.uk`

`http://www.cs.man.ac.uk/~franconi`

Department of Computer Science, University of Manchester

Summary

- Logic and Conceptual Modelling
- Description Logics for Conceptual Modelling
- Queries with an Ontology
- Ontology Integration

Summary

- Logic and Conceptual Modelling
- Description Logics for Conceptual Modelling
- Queries with an Ontology
- Ontology Integration

What is an Ontology

- An ontology is a formal conceptualisation of the world: a conceptual schema.

What is an Ontology

- An ontology is a formal conceptualisation of the world: a conceptual schema.
- An ontology specifies a set of constraints, which declare what should necessarily hold in any possible world.

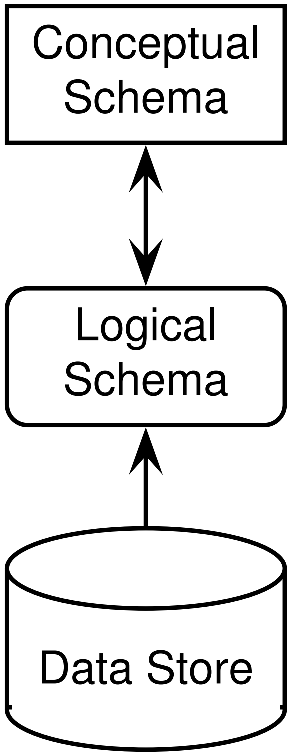
What is an Ontology

- An ontology is a formal conceptualisation of the world: a conceptual schema.
- An ontology specifies a set of constraints, which declare what should necessarily hold in any possible world.
- Any possible world should conform to the constraints expressed by the ontology.

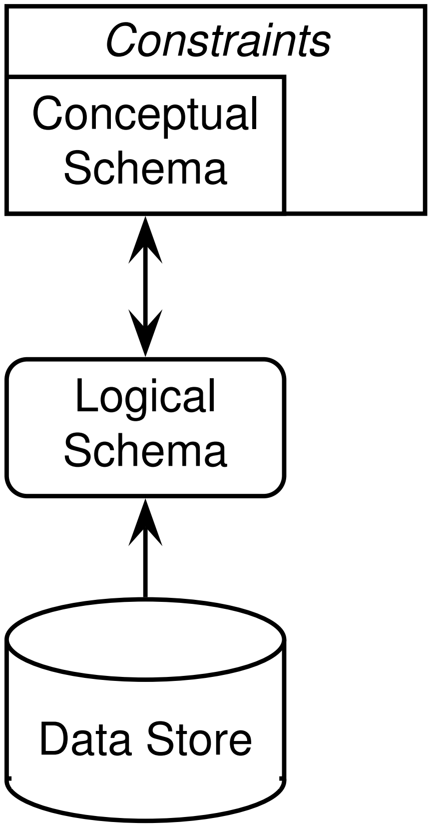
What is an Ontology

- An ontology is a formal conceptualisation of the world: a conceptual schema.
- An ontology specifies a set of constraints, which declare what should necessarily hold in any possible world.
- Any possible world should conform to the constraints expressed by the ontology.
- Given an ontology, a *legal world description* is a possible world satisfying the constraints.

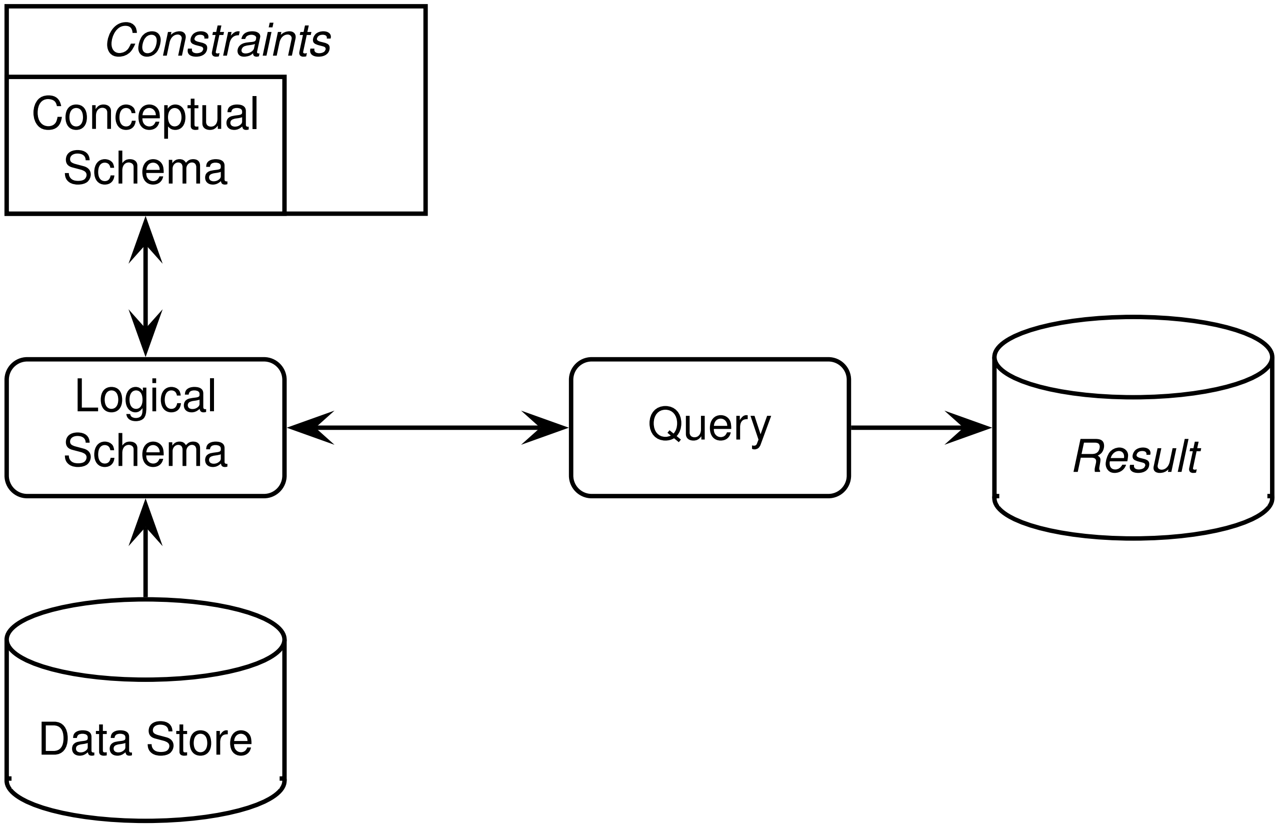
The role of a Conceptual Schema



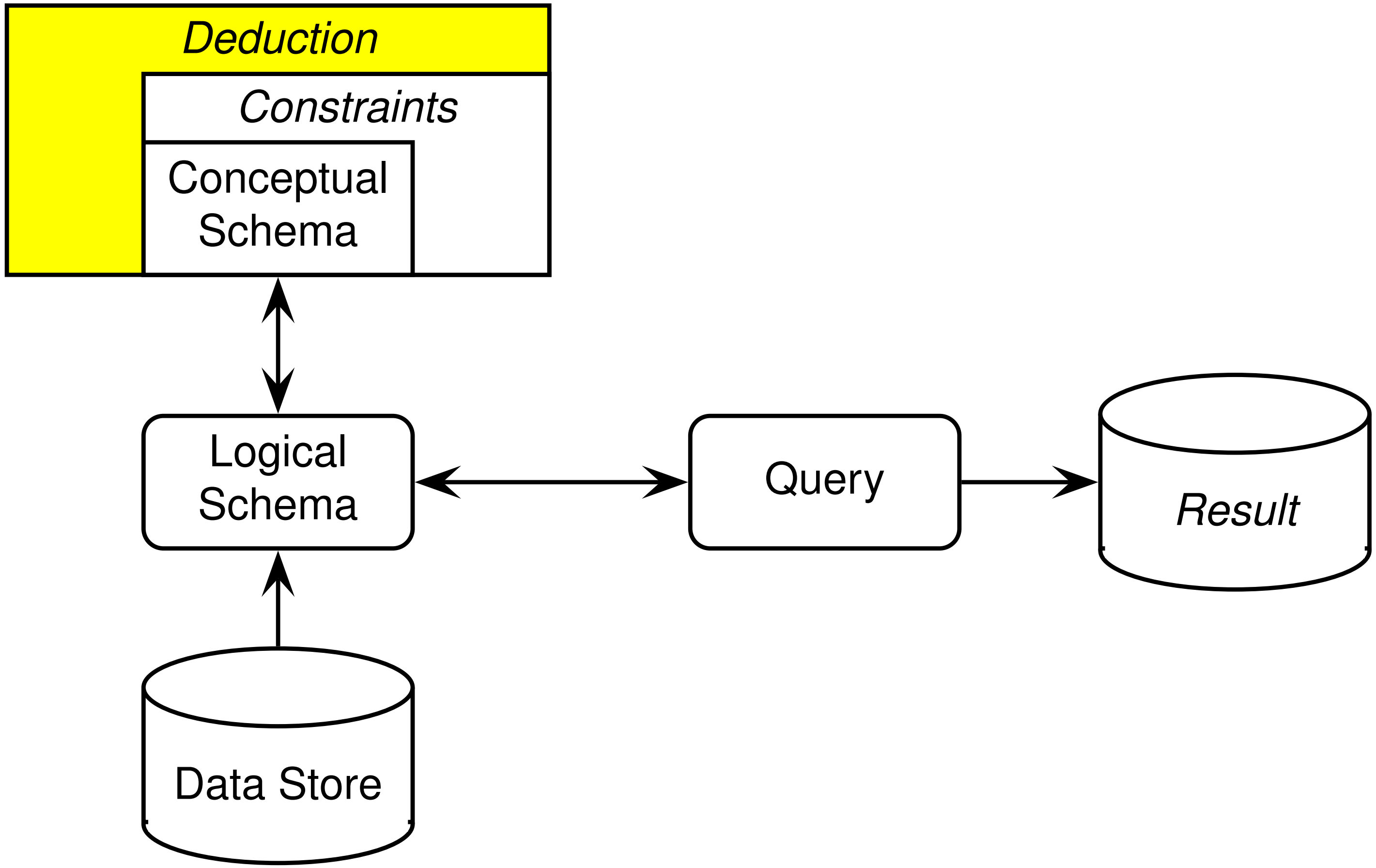
The role of a Conceptual Schema



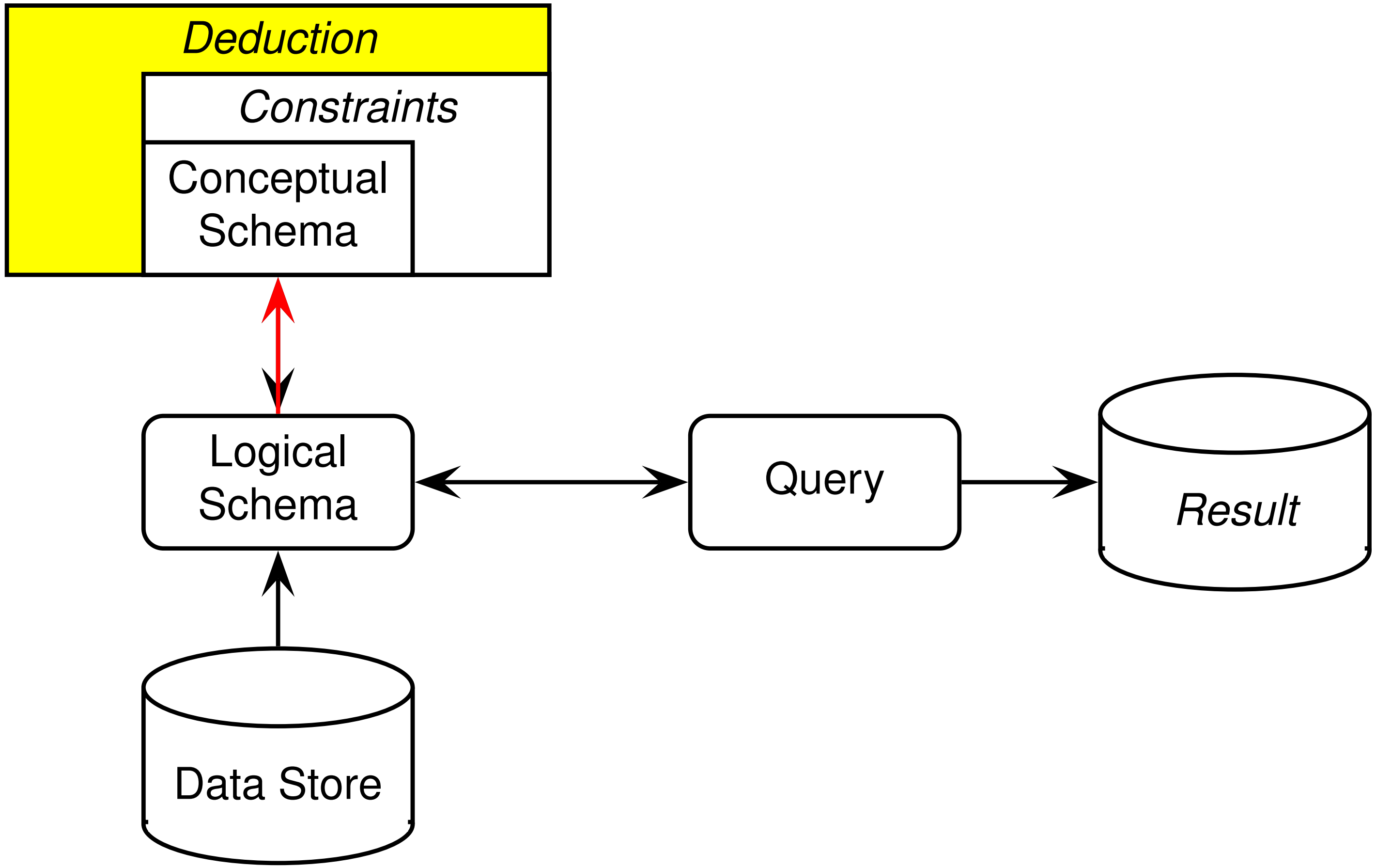
The role of a Conceptual Schema



The role of a Conceptual Schema



The role of a Conceptual Schema



Ontology languages and Conceptual Data Models

- An ontology language usually introduces concepts (aka classes, entities), properties of concepts (aka slots, attributes, roles), relationships between concepts (aka associations), and additional constraints.

Ontology languages and Conceptual Data Models

- An ontology language usually introduces concepts (aka classes, entities), properties of concepts (aka slots, attributes, roles), relationships between concepts (aka associations), and additional constraints.
- Ontology languages may be simple (e.g., having only concepts and taxonomies), frame-based (having only concepts and properties), or logic-based (e.g. Ontolingua and DAML+OIL).

Ontology languages and Conceptual Data Models

- An ontology language usually introduces concepts (aka classes, entities), properties of concepts (aka slots, attributes, roles), relationships between concepts (aka associations), and additional constraints.

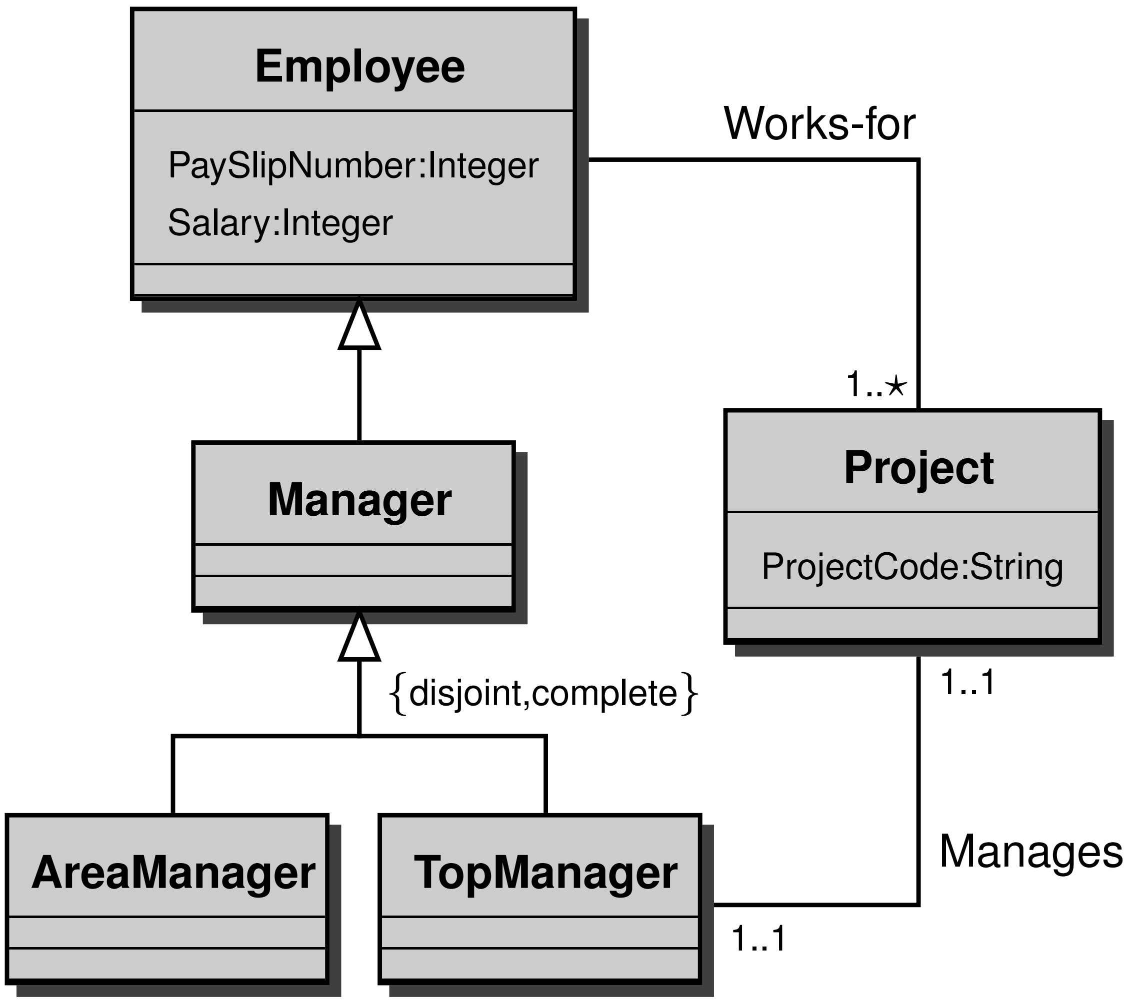
- Ontology languages may be simple (e.g., having only concepts and taxonomies), frame-based (having only concepts and properties), or logic-based (e.g. Ontolingua and DAML+OIL).

- Ontology languages are typically expressed by means of diagrams.

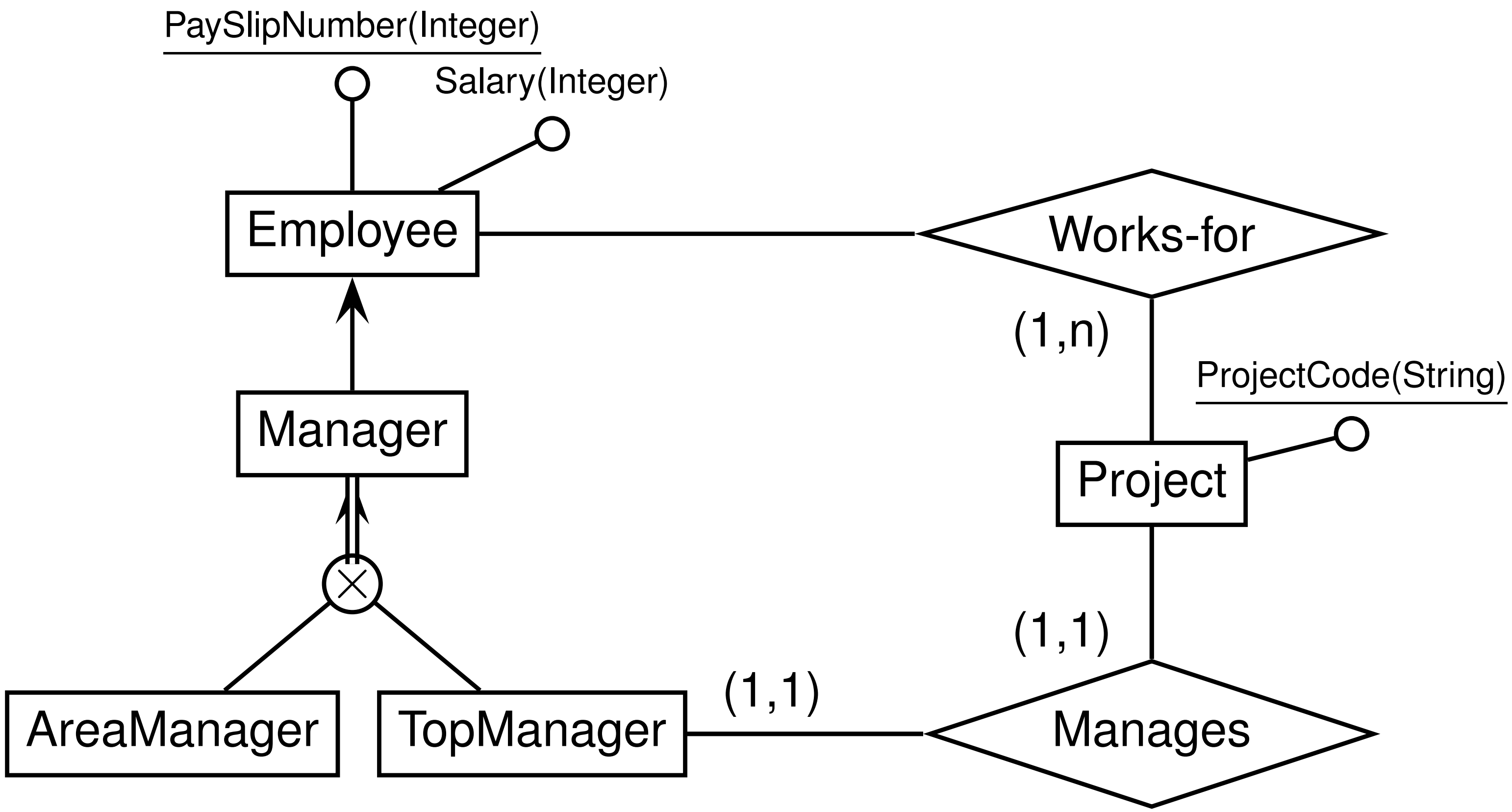
Ontology languages and Conceptual Data Models

- An ontology language usually introduces concepts (aka classes, entities), properties of concepts (aka slots, attributes, roles), relationships between concepts (aka associations), and additional constraints.
- Ontology languages may be simple (e.g., having only concepts and taxonomies), frame-based (having only concepts and properties), or logic-based (e.g. Ontolingua and DAML+OIL).
- Ontology languages are typically expressed by means of diagrams.
- **Entity-Relationship** schemas and **UML** class diagrams can be considered as ontologies.

UML Class Diagram



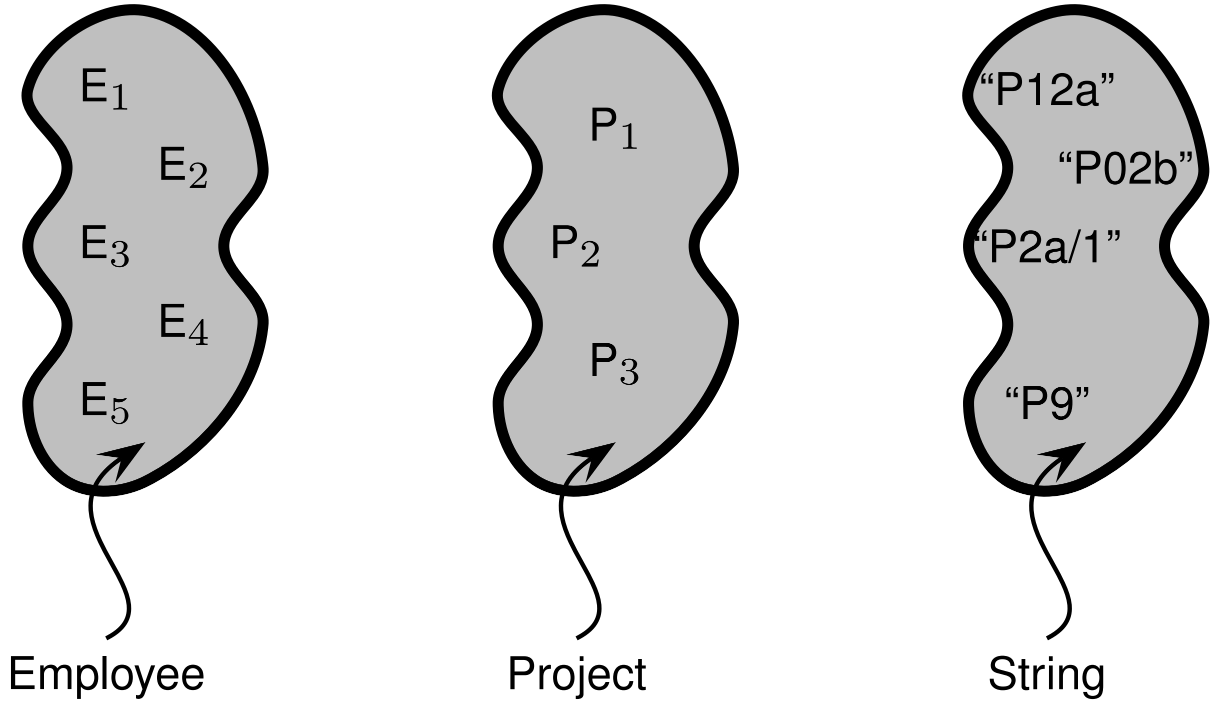
Entity-Relationship Schema



Semantics

In a specific world:

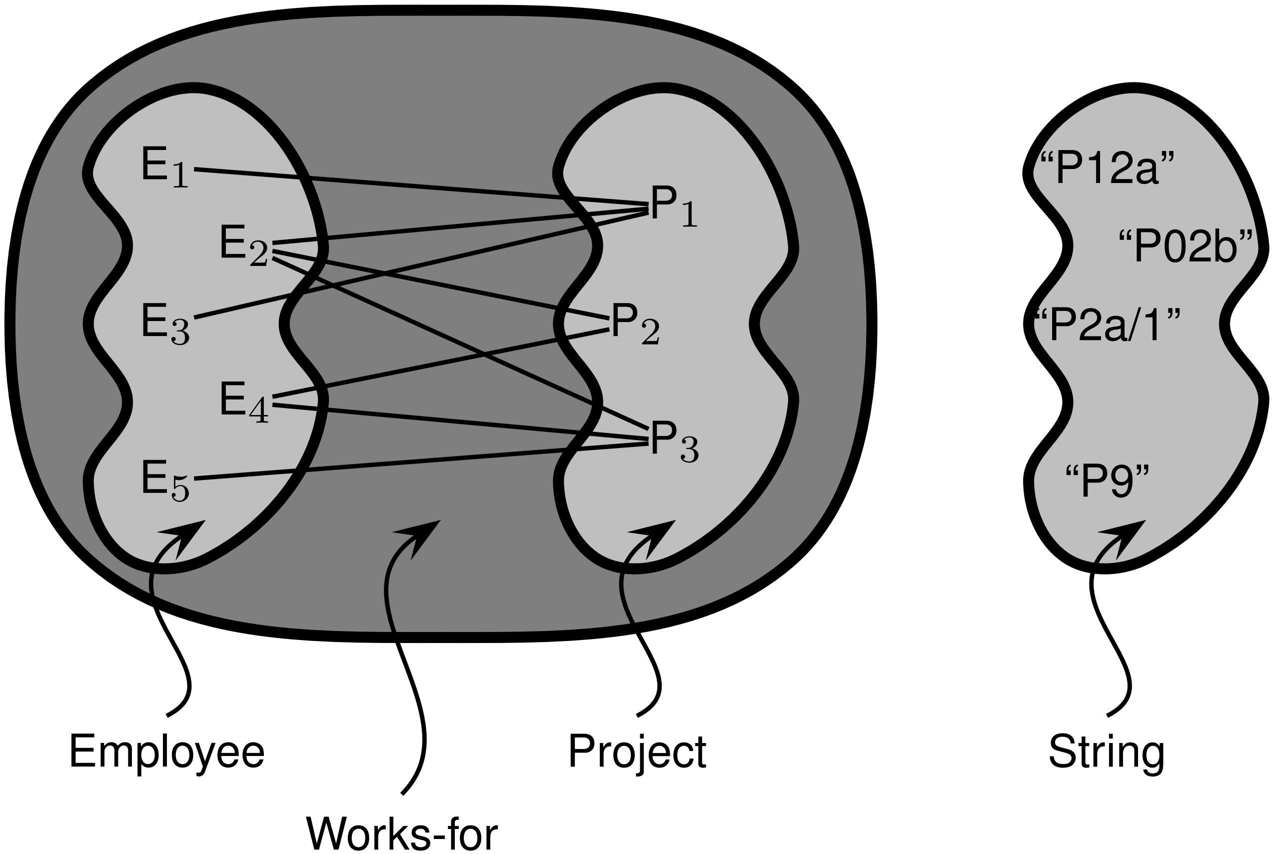
- A class is a ***set of instances***;
- a n-ary relationship is a ***set of n-tuples of instances***;
- an attribute is a ***set of pairs of an instance and a domain element***.



Semantics

In a specific world:

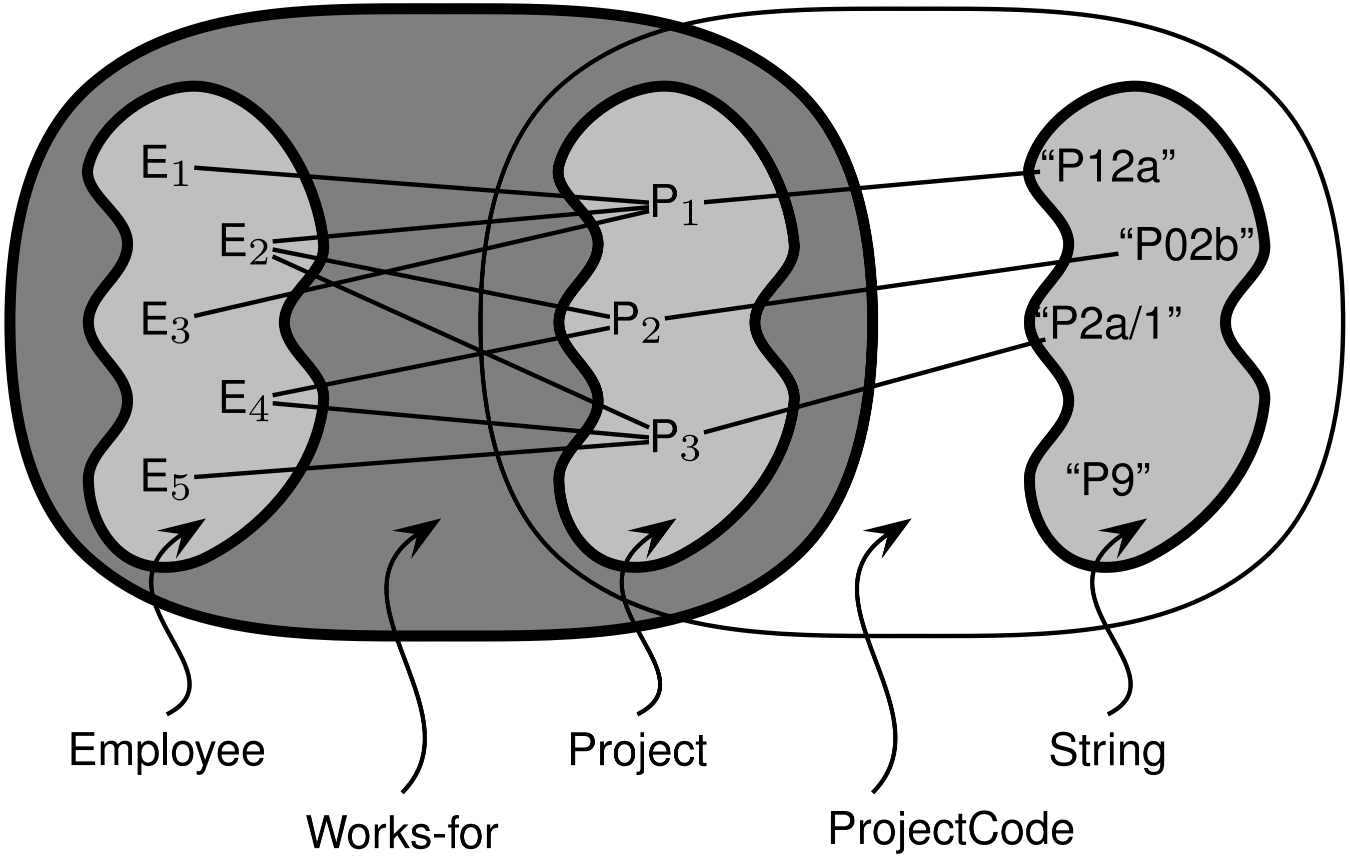
- A class is a ***set of instances***;
- a n-ary relationship is a ***set of n-tuples of instances***;
- an attribute is a ***set of pairs of an instance and a domain element***.



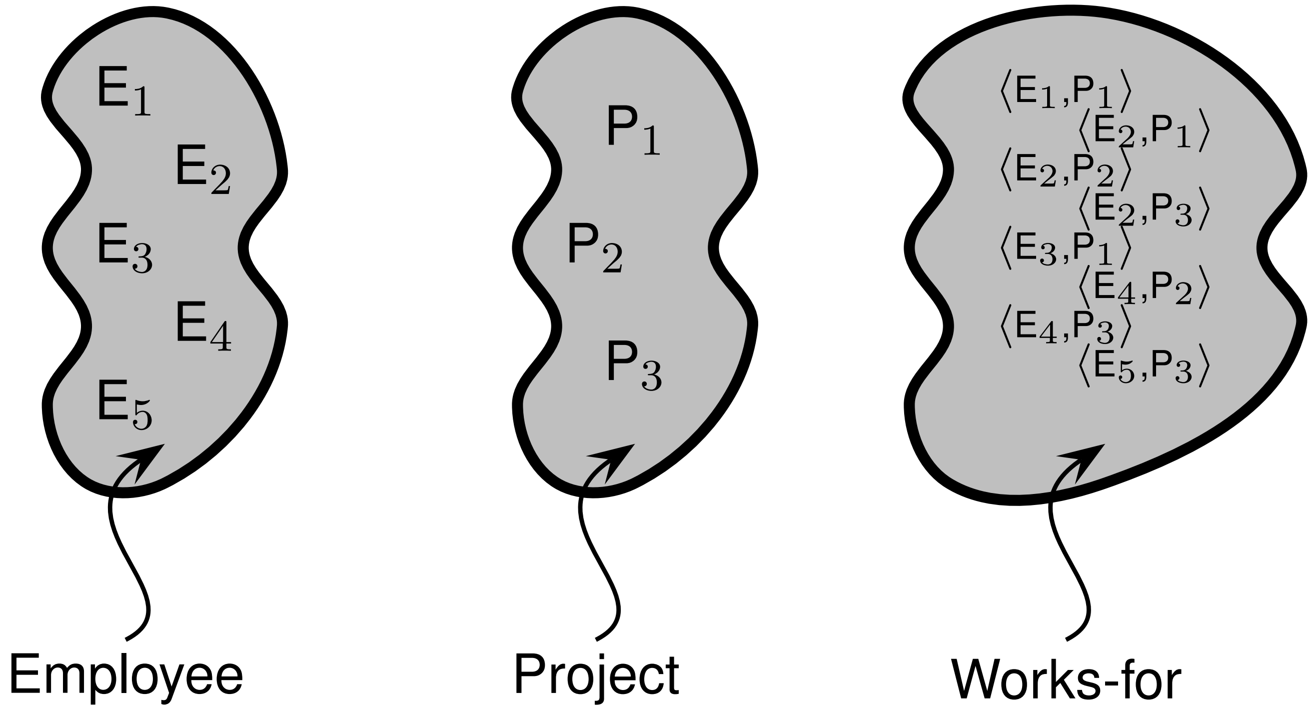
Semantics

In a specific world:

- A class is a ***set of instances***;
- a n-ary relationship is a ***set of n-tuples of instances***;
- an attribute is a ***set of pairs of an instance and a domain element***.



A world is described by sets of instances



The relational representation of a world

Employee

employeeId
E_1
E_2
E_3
E_4
E_5

Project

projectId
P_1
P_2
P_3

String

anystring
“P12a”
“P02b”
“P2a/1”
“P9”
...

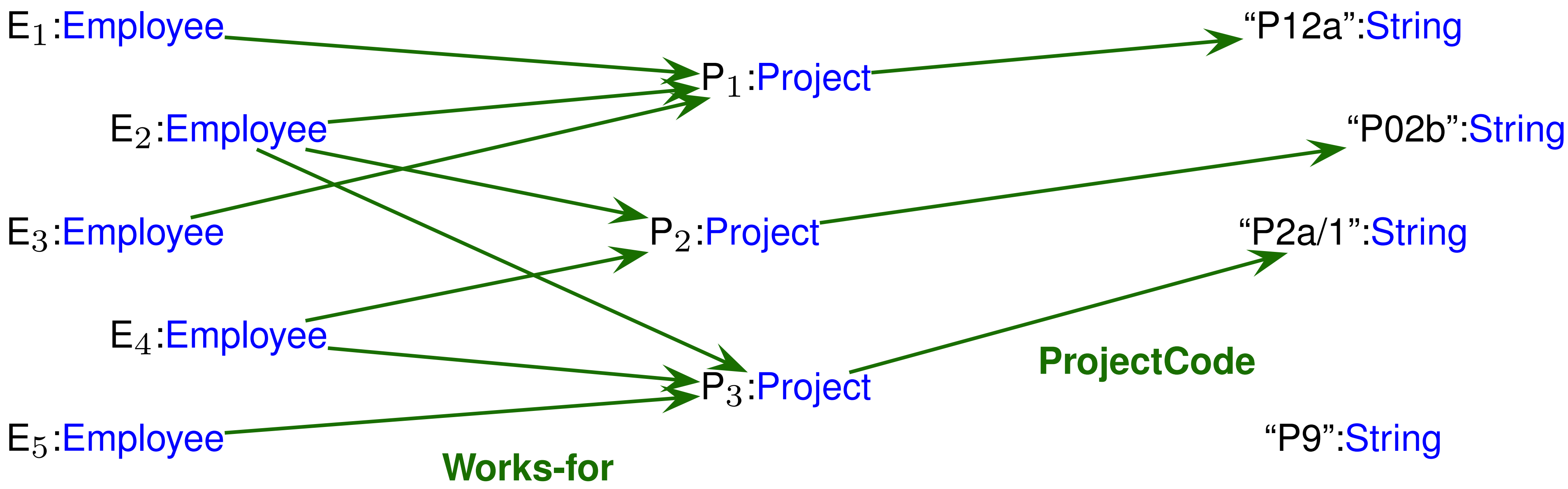
Works-for

employeeId	*projectId*
E_1	P_1
E_2	P_1
E_2	P_2
E_2	P_3
E_3	P_1
E_4	P_2
E_4	P_3
E_5	P_3

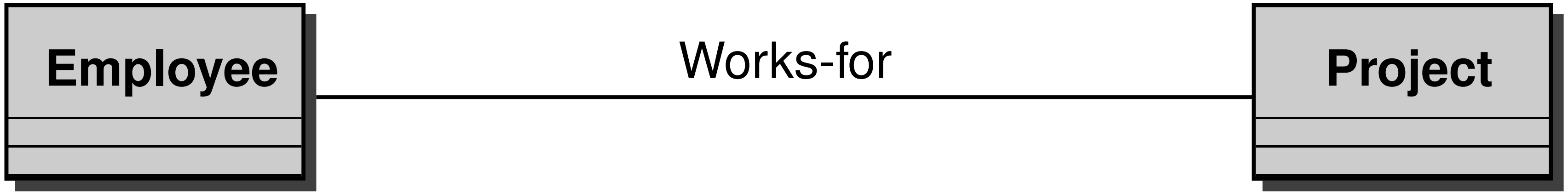
ProjectCode

projectId	*pcode*
P_1	“P12a”
P_2	“P02b”
P_3	“P2a/1”

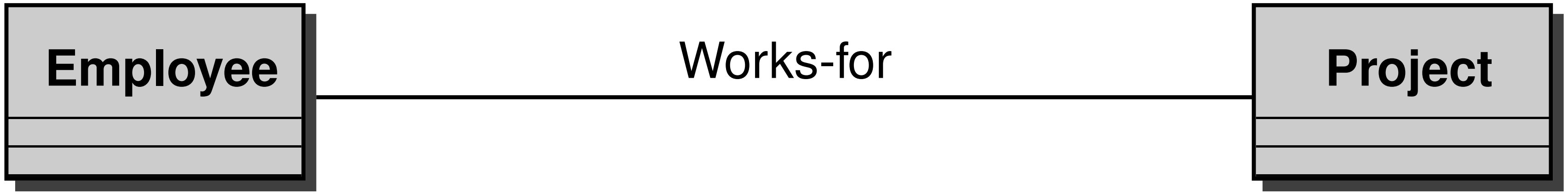
The graph representation of a world – e.g. RDF triples



Constraints introduced by Relationships

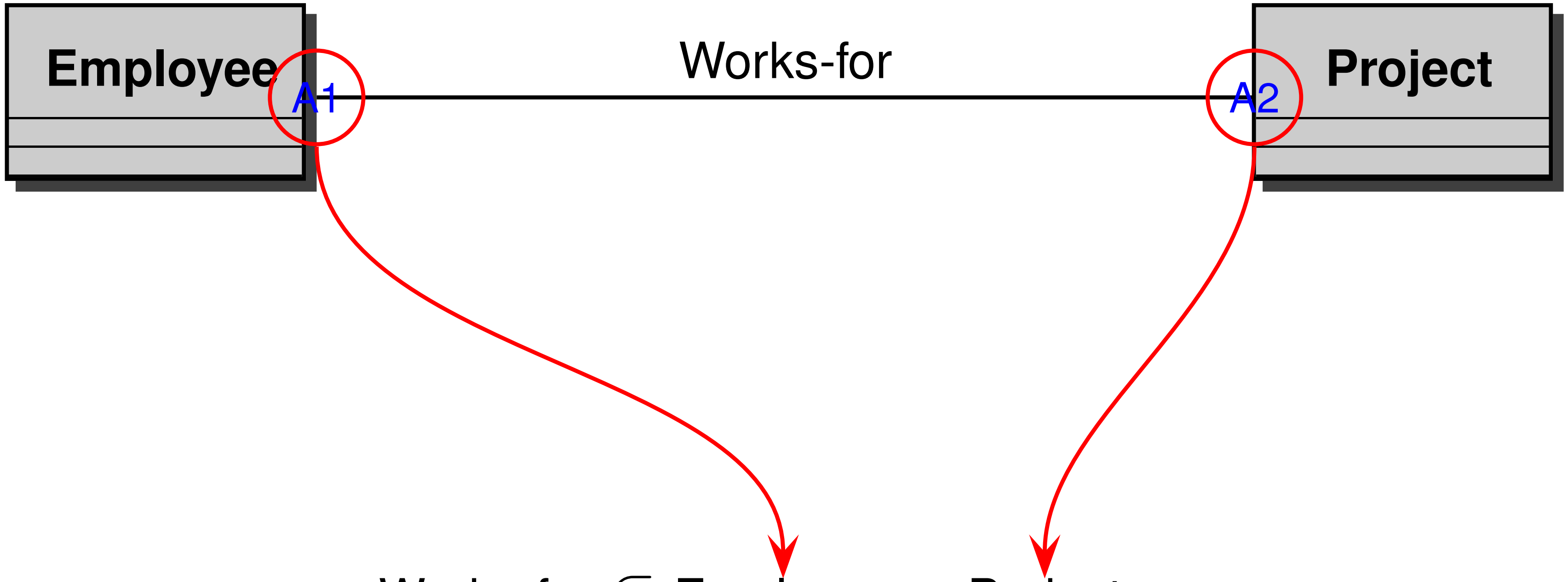


Constraints introduced by Relationships



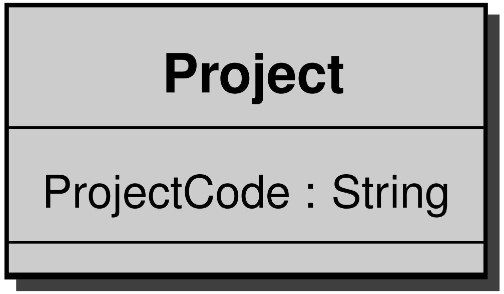
$$\text{Works-for} \subseteq \text{Employee} \times \text{Project}$$

Constraints introduced by Relationships

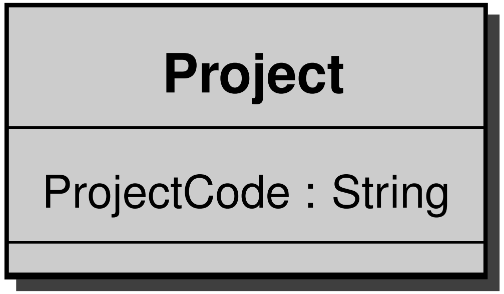


Works-for $\subseteq$ Employee $\times$ Project

Constraints introduced by Attributes

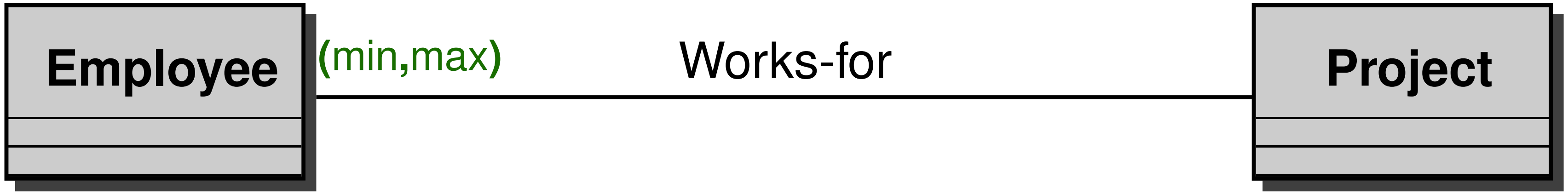


Constraints introduced by Attributes

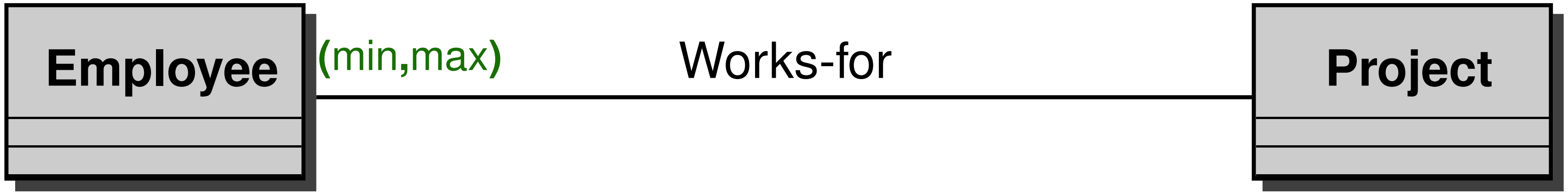


$$\mathsf{Project} \subseteq \{p \mid \sharp(\mathsf{ProjectCode} \cap (\{p\} \times \mathtt{String})) \geq 1\}$$

Constraints introduced by Cardinality Constraints



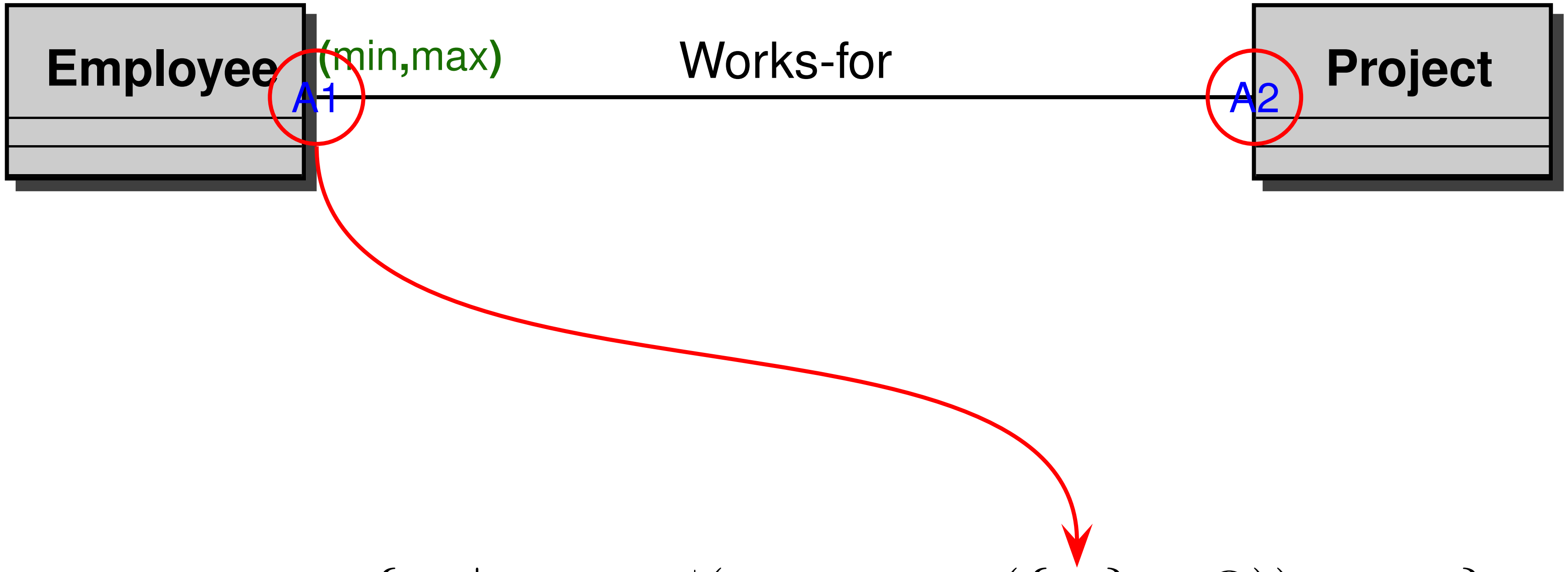
Constraints introduced by Cardinality Constraints



$$\text{TopManager} \subseteq \{m \mid \text{max} \geq \sharp(\text{Manages} \cap (\{m\} \times \Omega)) \geq \text{min}\}$$

(where Ω is the set of all instances)

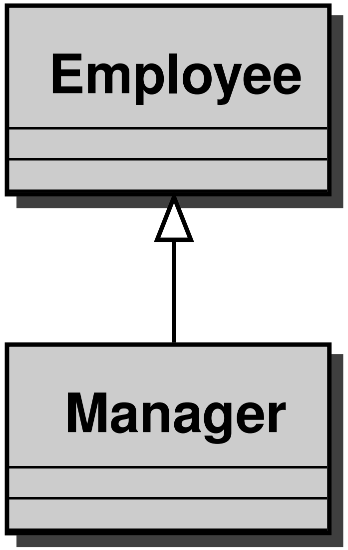
Constraints introduced by Cardinality Constraints



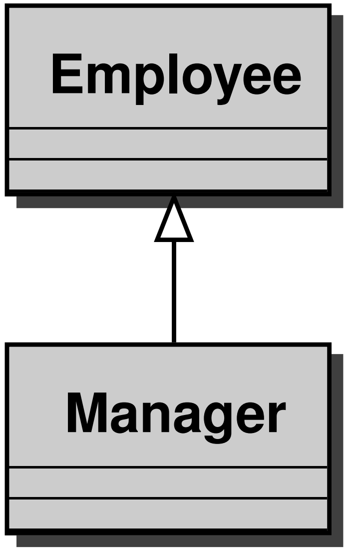
$$\text{TopManager} \subseteq \{m \mid \text{max} \geq \sharp(\text{Manages} \cap (\{m\} \times \Omega)) \geq \text{min}\}$$

(where Ω is the set of all instances)

Constraints introduced by ISA

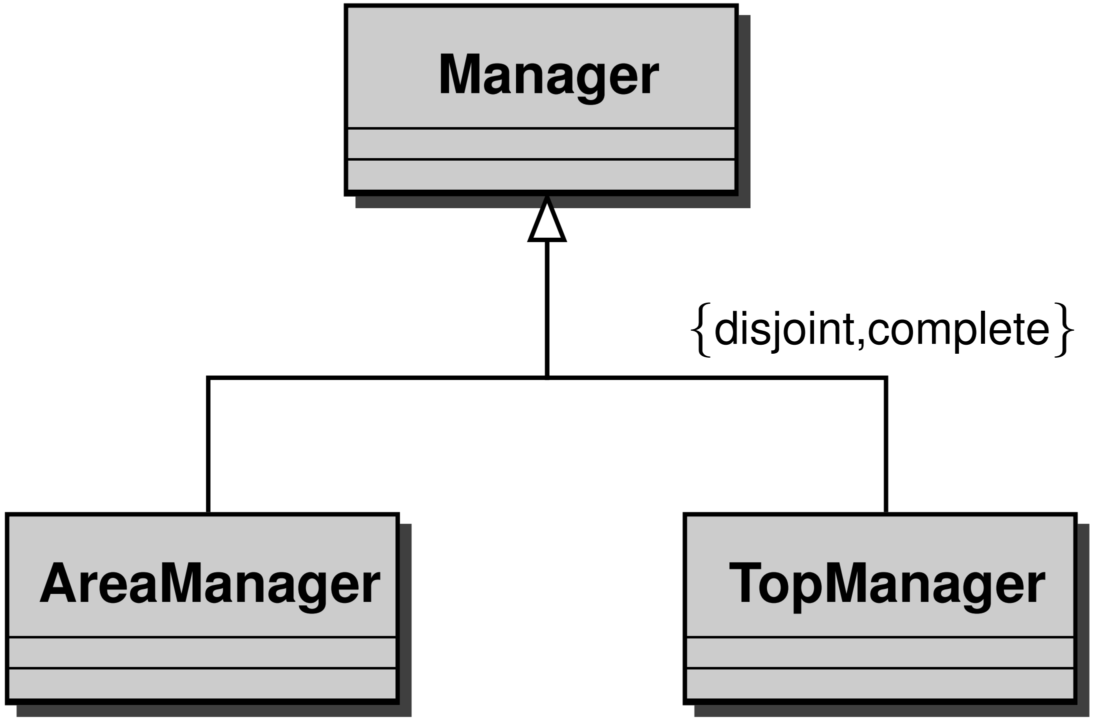


Constraints introduced by ISA

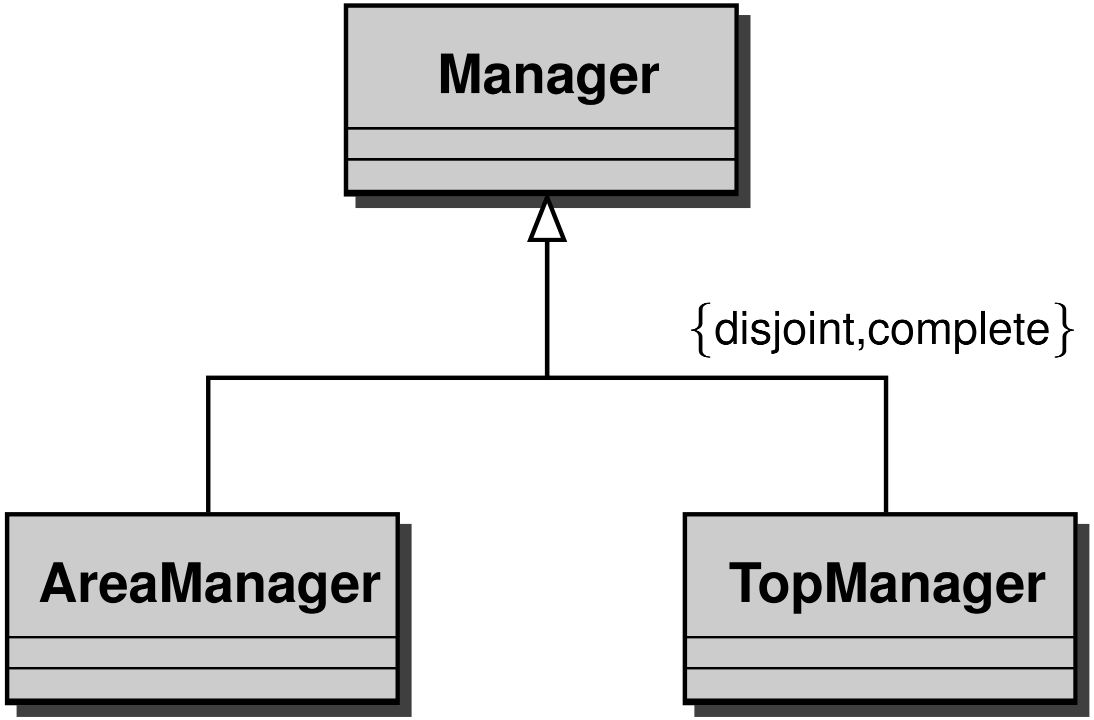


$$\text{Manager} \subseteq \text{Employee}$$

Disjoint and *Total* constraints



Disjoint and *Total* constraints



- *ISA:* AreaManager $\subseteq$ Manager
- *ISA:* TopManager $\subseteq$ Manager
- *disjoint:* AreaManager $\cap$ TopManager $= \emptyset$
- *total:* Manager $\subseteq$ AreaManager $\cup$ TopManager

Constraints introduced by the initial diagram

$\text{Works-for} \subseteq \text{Employee} \times \text{Project}$

$\text{Manages} \subseteq \text{TopManager} \times \text{Project}$

$\text{Employee} \subseteq \{e \mid \sharp(\text{PaySlipNumber} \cap (\{e\} \times \texttt{Integer})) \geq 1\}$

$\text{Employee} \subseteq \{e \mid \sharp(\text{Salary} \cap (\{e\} \times \texttt{Integer})) \geq 1\}$

$\text{Project} \subseteq \{p \mid \sharp(\text{ProjectCode} \cap (\{p\} \times \texttt{String})) \geq 1\}$

$\text{TopManager} \subseteq \{m \mid 1 \geq \sharp(\text{Manages} \cap (\{m\} \times \Omega)) \geq 1\}$

$\text{Project} \subseteq \{p \mid 1 \geq \sharp(\text{Manages} \cap (\Omega \times \{p\})) \geq 1\}$

$\text{Project} \subseteq \{p \mid \sharp(\text{Works-for} \cap (\Omega \times \{p\})) \geq 1\}$

$\text{Manager} \subseteq \text{Employee}$

$\text{AreaManager} \subseteq \text{Manager}$

$\text{TopManager} \subseteq \text{Manager}$

$\text{AreaManager} \cap \text{TopManager} = \emptyset$

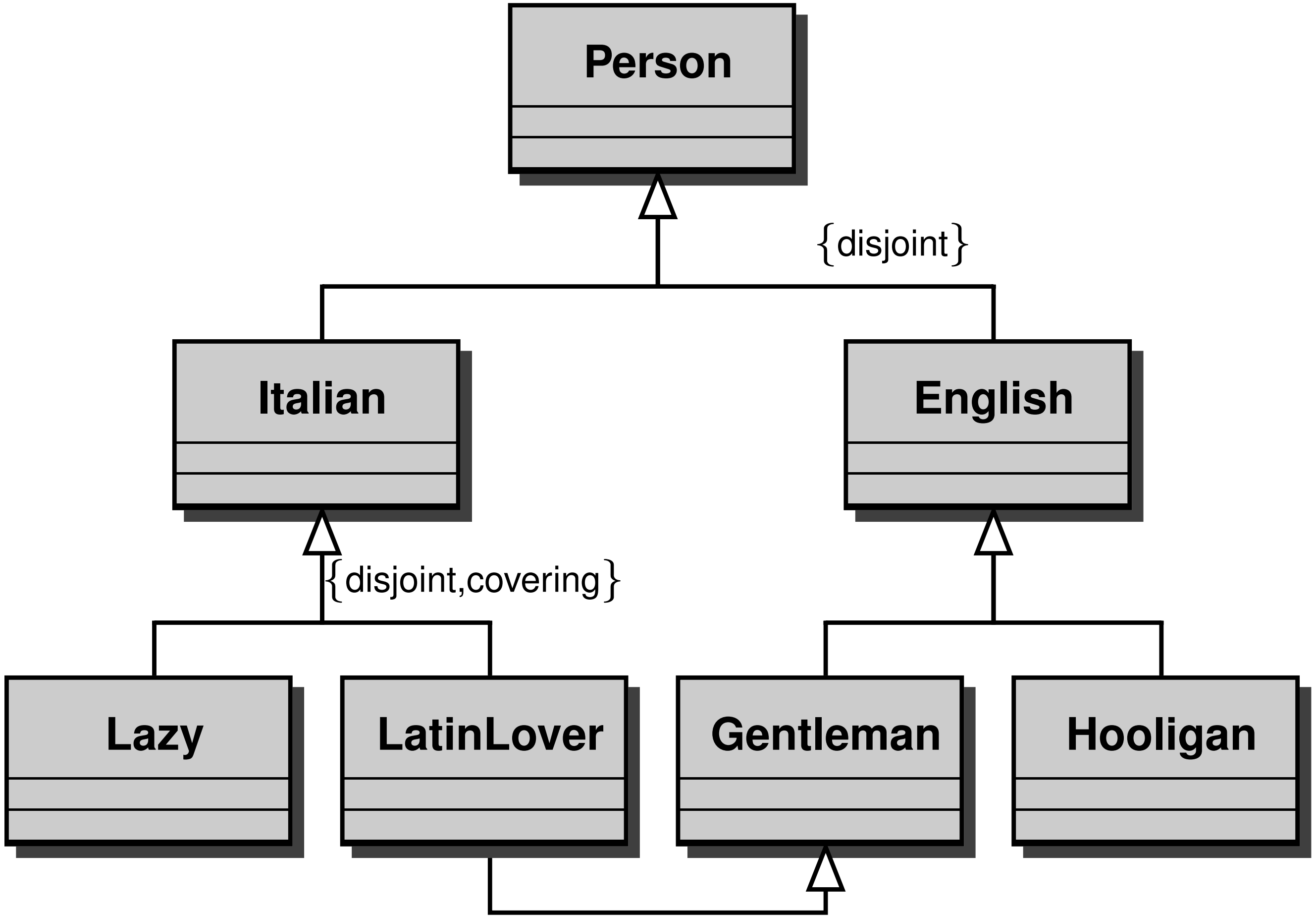
$\text{Manager} \subseteq \text{AreaManager} \cup \text{TopManager}$

Reasoning

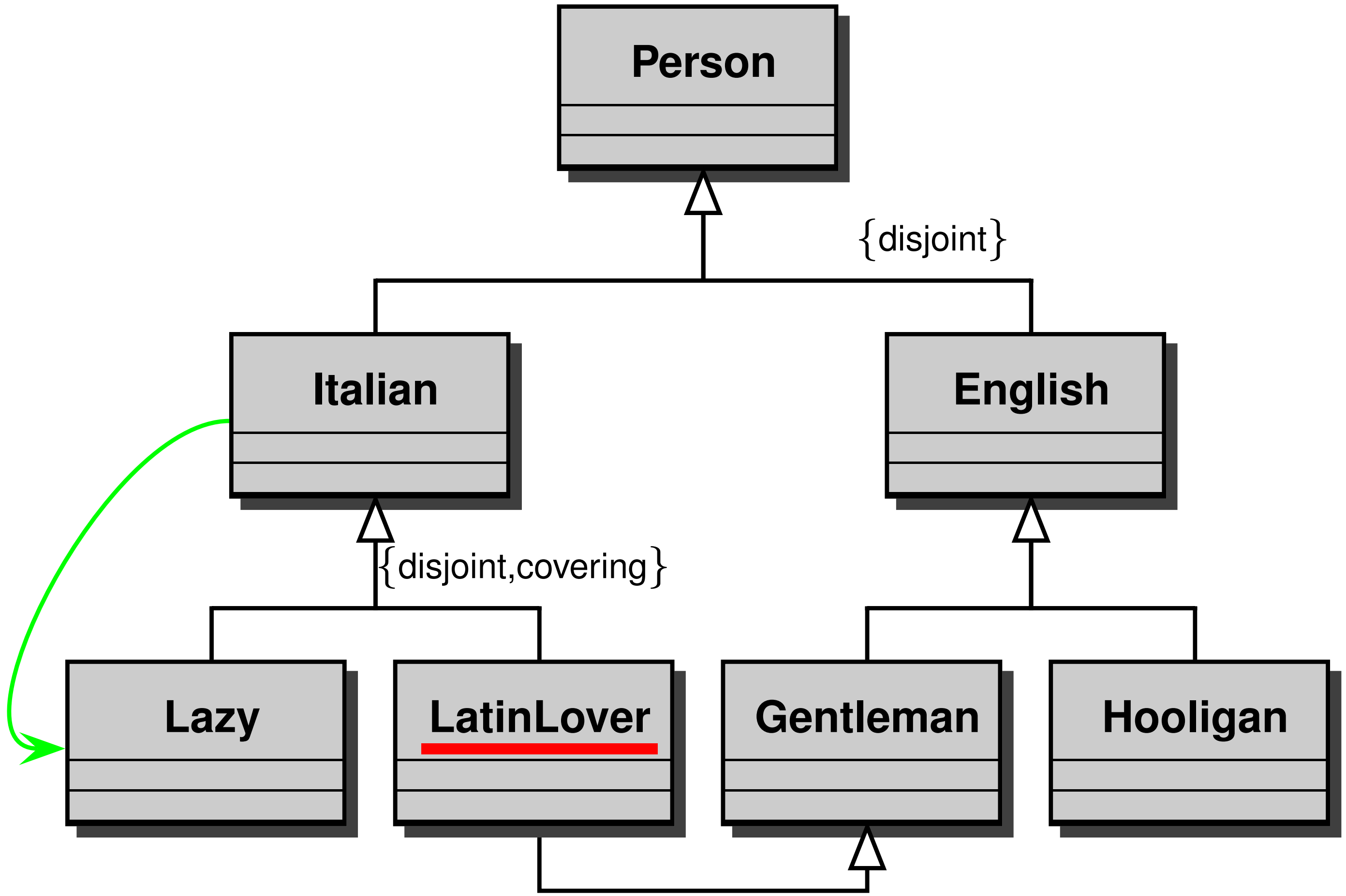
Given an ontology – seen as a collection of constraints – it is possible that additional constraints can be inferred.

- A class is inconsistent if it denotes the empty set in any legal world description.
- A class is a subclass of another class if the former denotes a subset of the set denoted by the latter in any legal world description.
- Two classes are equivalent if they denote the same set in any legal world description.
- A stricter contraint is inferred – e.g., a **cardinality** contraint – if it holds in in any legal world description.
- . . .

Simple reasoning example



Simple reasoning example



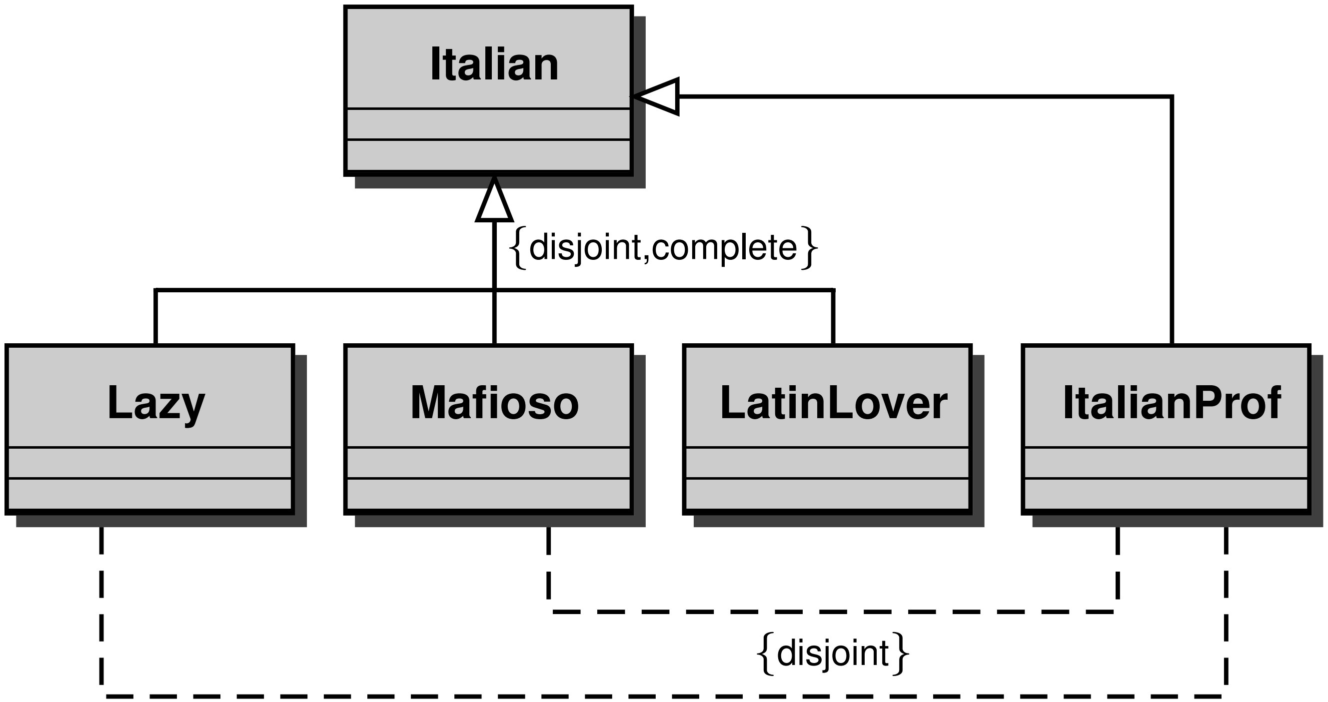
implies

LatinLover $= \emptyset$

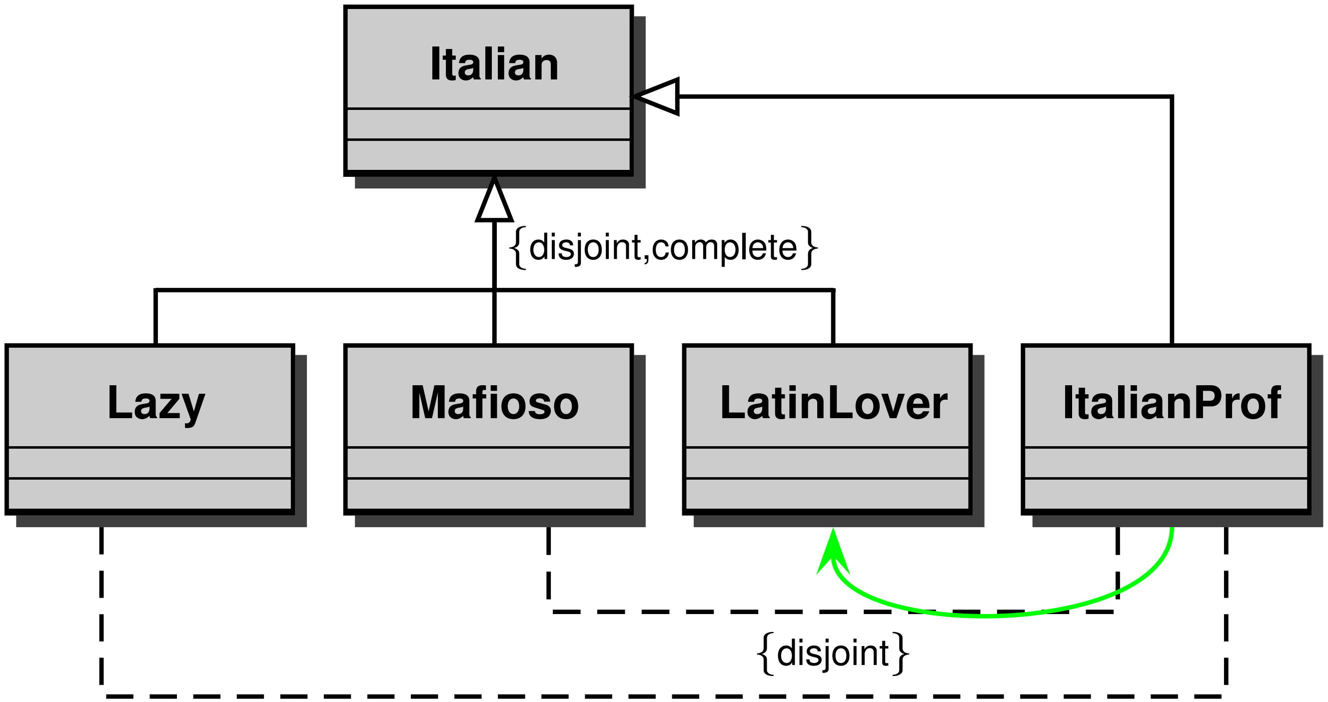
Italian $\subseteq$ Lazy

Italian $\equiv$ Lazy

Reasoning by cases



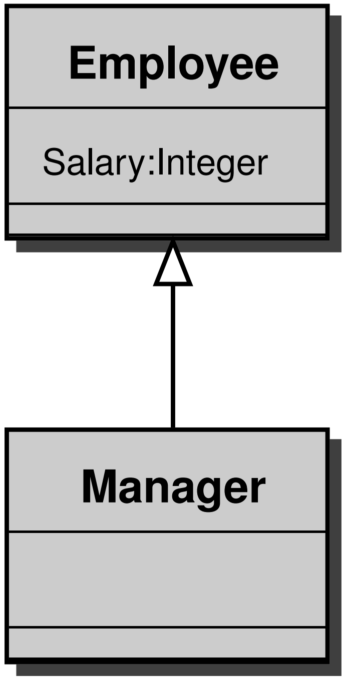
Reasoning by cases



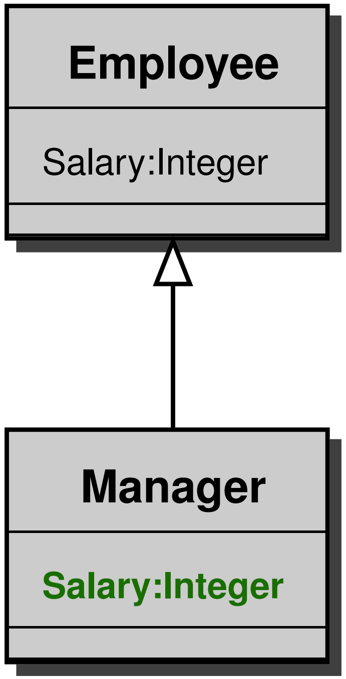
implies

ItalianProf $\subseteq$ LatinLover

ISA and Inheritance



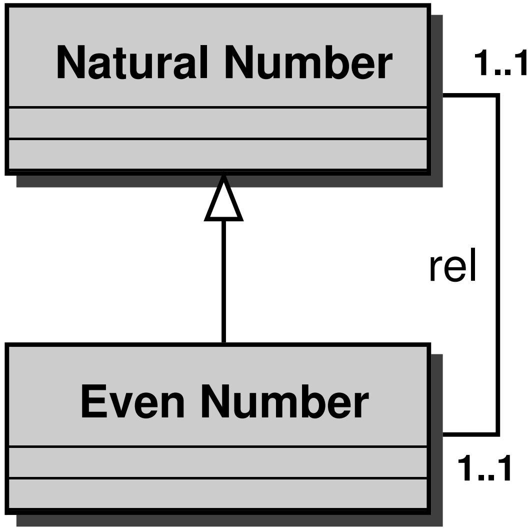
ISA and Inheritance



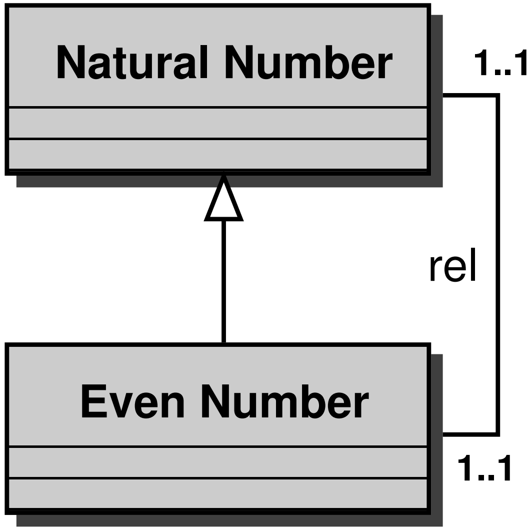
implies

$$\text{Manager} \subseteq \{m \mid \sharp(\text{Salary} \cap (\{m\} \times \texttt{Integer})) \geq 1\}$$

Bijection bewteen Classes



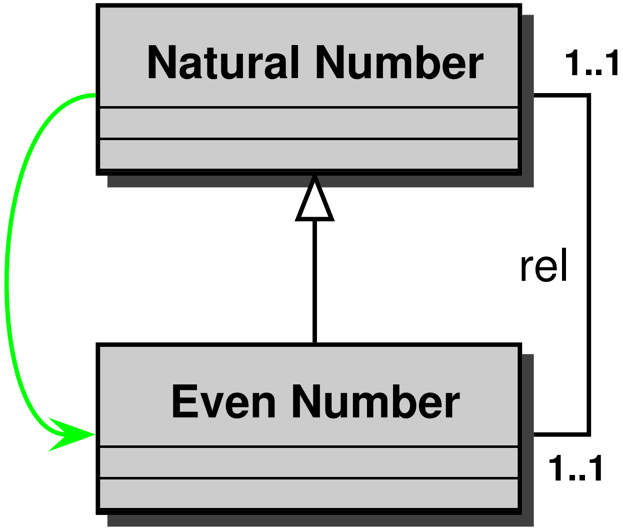
Bijection bewteen Classes



implies

"the classes *'Natural Number'* and *'Even Number'* contain the same number of instances".

Bijection bewteen Classes

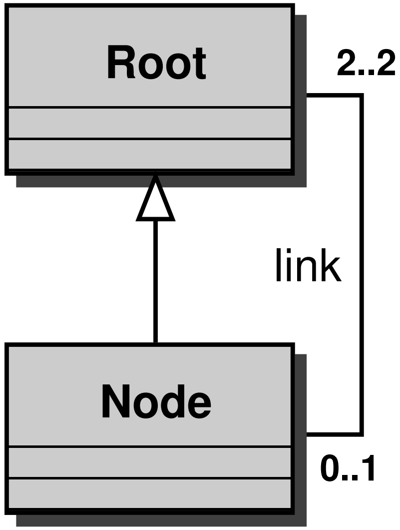


implies

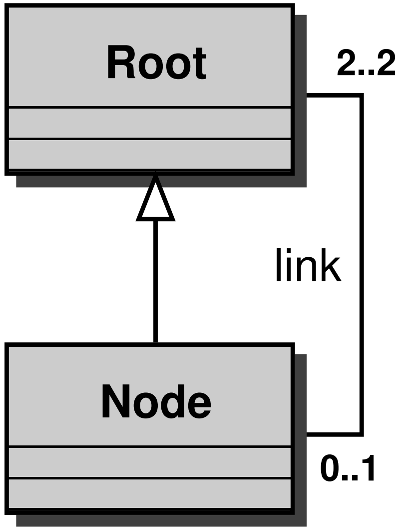
"the classes *'Natural Number'* and *'Even Number'* contain the same number of instances".

If the domain is finite: Natural Number $\equiv$ Even Number

Infinite worlds



Infinite worlds



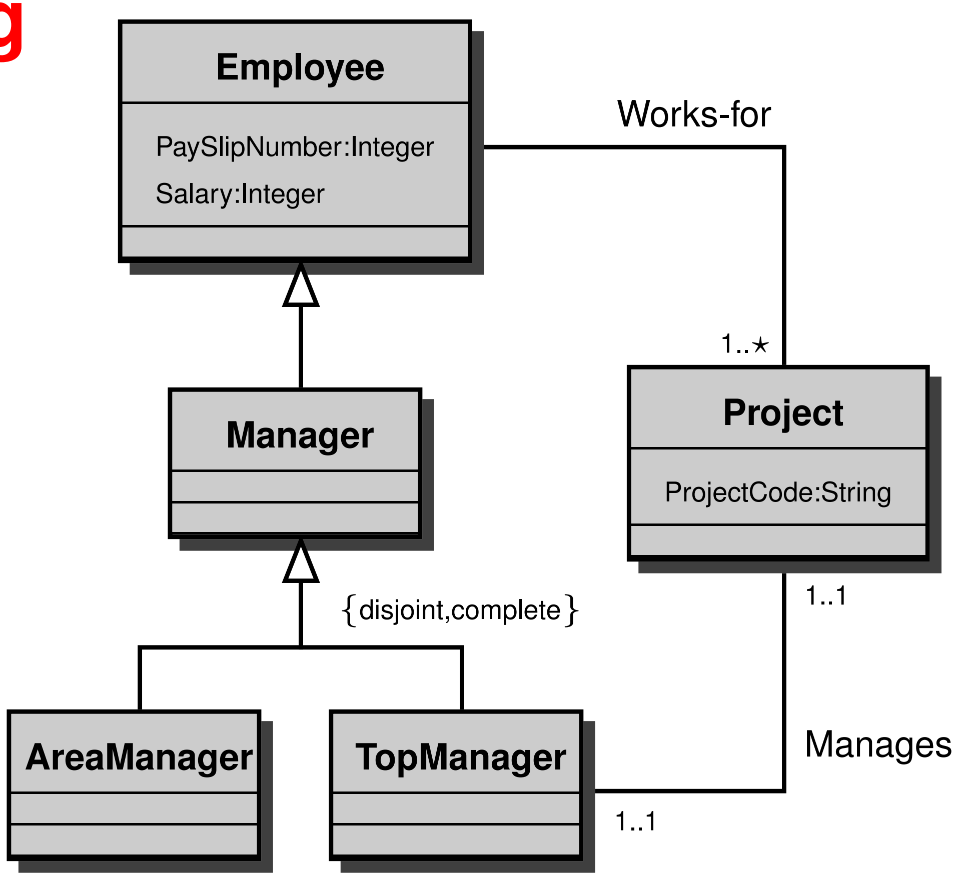
implies

“the classes Root and Node contain an infinite number of instances”.

Ontologies in First Order Logic

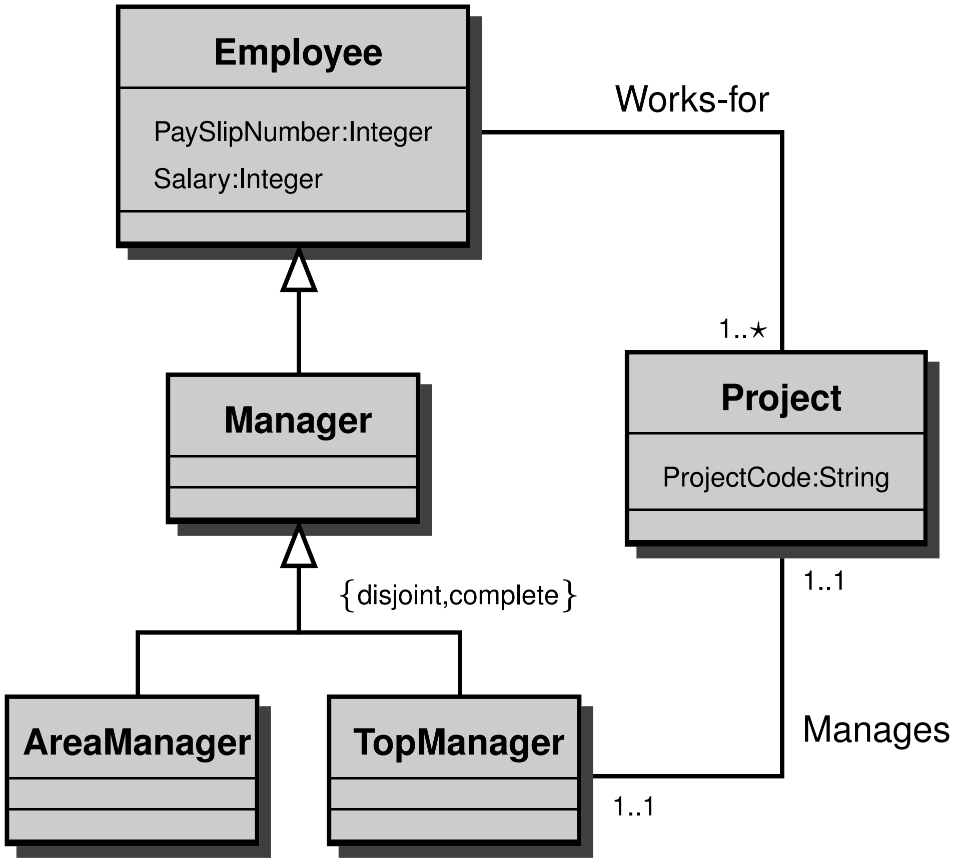
- We have introduced ontology languages that specify a set of constraints that should be satisfied by the world of interest.
- The *interpretation* of an ontology is therefore defined as the collection of all the *legal world descriptions* – i.e., all the (finite) relational structures which conform to the constraints imposed by the ontology.
- An alternative way to define the interpretation: an ontology is mapped into a set of First Order Logic (FOL) formulas.
- The legal world descriptions (i.e., the interpretation) of an ontology are all the models of the FOL theory associated to it.

FOL encoding



$$
\begin{array}{lcl}
\forall x, y.\, \texttt{Works-for}(x, y) & \rightarrow & \texttt{Employee}(x) \wedge \texttt{Project}(y) \\
\forall x, y.\, \texttt{Manages}(x, y) & \rightarrow & \texttt{Top-Manager}(x) \wedge \texttt{Project}(y) \\
\forall y.\, \texttt{Project}(y) & \rightarrow & \exists x.\, \texttt{Works-for}(x, y) \\
\forall y.\, \texttt{Project}(y) & \rightarrow & \exists^{=1} x.\, \texttt{Manages}(x, y) \\
\forall x.\, \texttt{Top-Manager}(x) & \rightarrow & \exists^{=1} y.\, \texttt{Manages}(x, y) \\
\forall x.\, \texttt{Manager}(x) & \rightarrow & \texttt{Employee}(x) \\
\forall x.\, \texttt{Manager}(x) & \rightarrow & \texttt{Area-Manager}(x) \vee \texttt{Top-Manager}(x) \\
\forall x.\, \texttt{Area-Manager}(x) & \rightarrow & \texttt{Manager}(x) \wedge \neg \texttt{Top-Manager}(x) \\
\forall x.\, \texttt{Top-Manager}(x) & \rightarrow & \texttt{Manager}(x)
\end{array}
$$

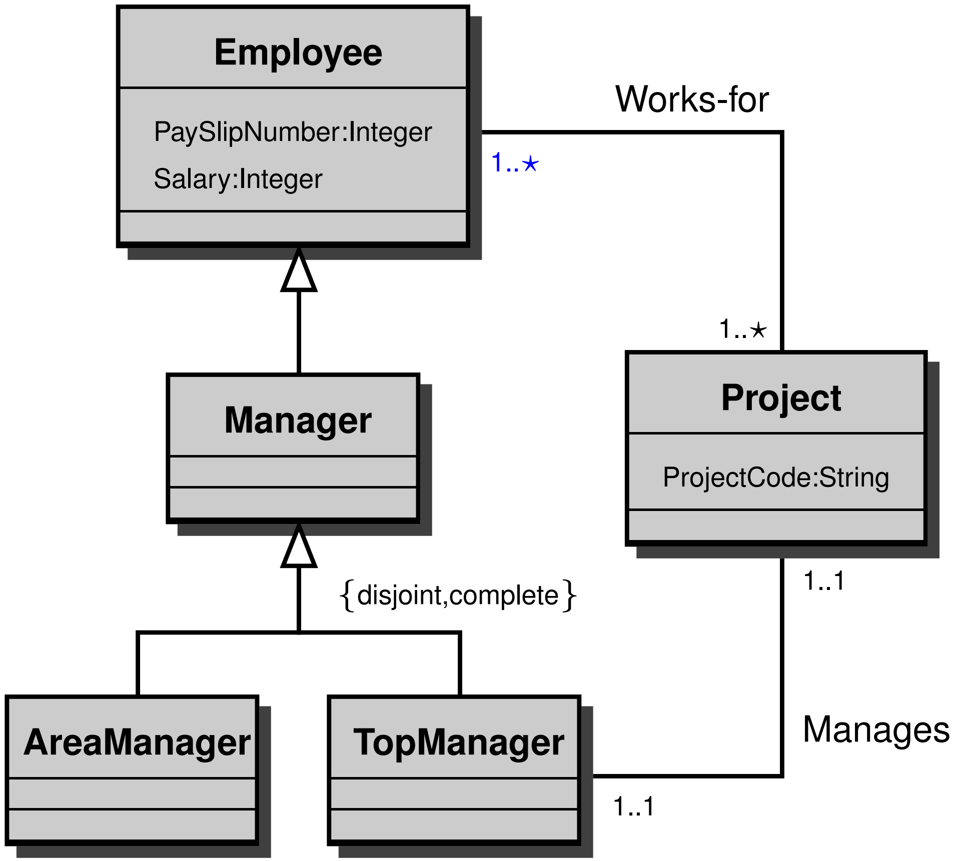
Additional constraints



- Managers do not work for a project (she/he just manages it):

  $\forall x.\, \texttt{Manager}(x) \rightarrow \forall y.\, \neg\texttt{WORKS-FOR}(x, y)$

Additional constraints



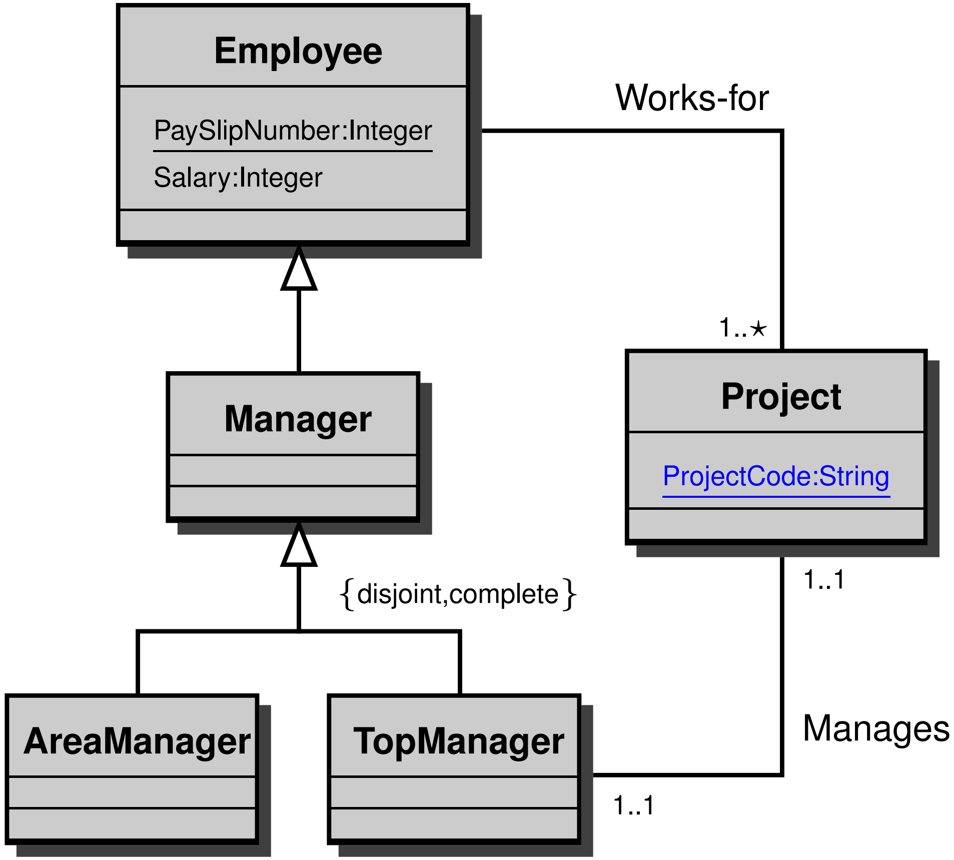
- Managers do not work for a project (she/he just manages it):

  $\forall x. \texttt{Manager}(x) \rightarrow \forall y. \neg \texttt{WORKS-FOR}(x, y)$

- If the minimum cardinality for the participation of employees to the *works-for* relationship is increased, then . . .

Key constraints

A key is a set of attributes of a class whose value uniquely identify elements of the class itself.



$$\forall x.\ \mathtt{Project}(x) \rightarrow \exists^{=1} y.\ \mathtt{ProjectCode}(x,y) \wedge \mathtt{String}(y)$$

$$\forall y.\ \exists x.\ \mathtt{ProjectCode}(x,y) \rightarrow \exists^{=1} x.\ \mathtt{ProjectCode}(x,y) \wedge \mathtt{Project}(x)$$

Summary

- Logic and Conceptual Modelling
- Description Logics for Conceptual Modelling
- Queries with an Ontology
- Ontology Integration

The $\mathcal{DLR}$ Description Logic – a fragment of FOL

- **relationships:** interpreted as **sets of tuples** of a given arity

$$R \to \top_n \mid RN \mid \neg R \mid R_1 \sqcap R_2 \mid R_1 \sqcup R_2 \mid i/n : C$$

- **classes:** interpreted as **sets of objects**

$$C \to \top \mid CN \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists^{\leqslant k}[i]R$$

- **conceptual schema**: $R \sqsubseteq R' \mid C \sqsubseteq C' \mid R \not\sqsubseteq R' \mid C \not\sqsubseteq C'$

$$\texttt{Works-for} \sqsubseteq \texttt{subj}/2 : \texttt{Employee} \sqcap \texttt{obj}/2 : \texttt{Project}$$

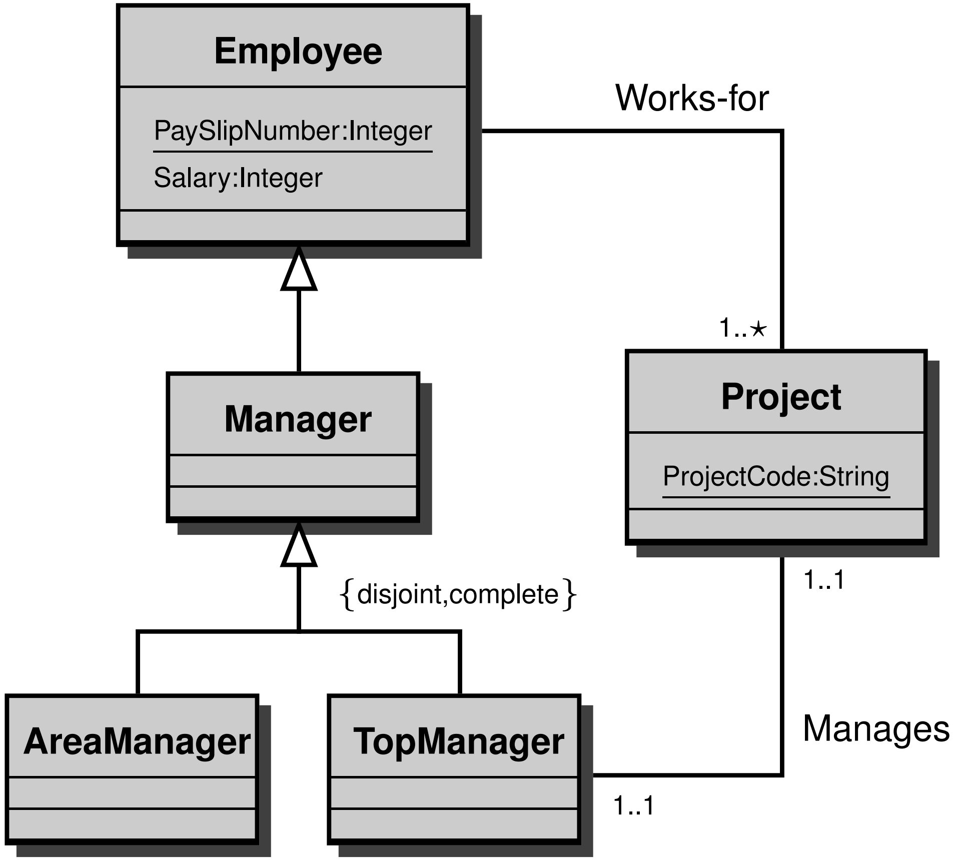
$$\texttt{TopManager} \sqsubseteq \texttt{Manager} \sqcap \exists^{=1}[\texttt{man}]\texttt{Manages}$$

Encoding conceptual data models in $\mathcal{DLR}$

- Object-oriented data models (e.g., UML and ODMG)
- Semantic data models (e.g., EER and ORM)
- Frame-based ontology languages (e.g., DAML+OIL)

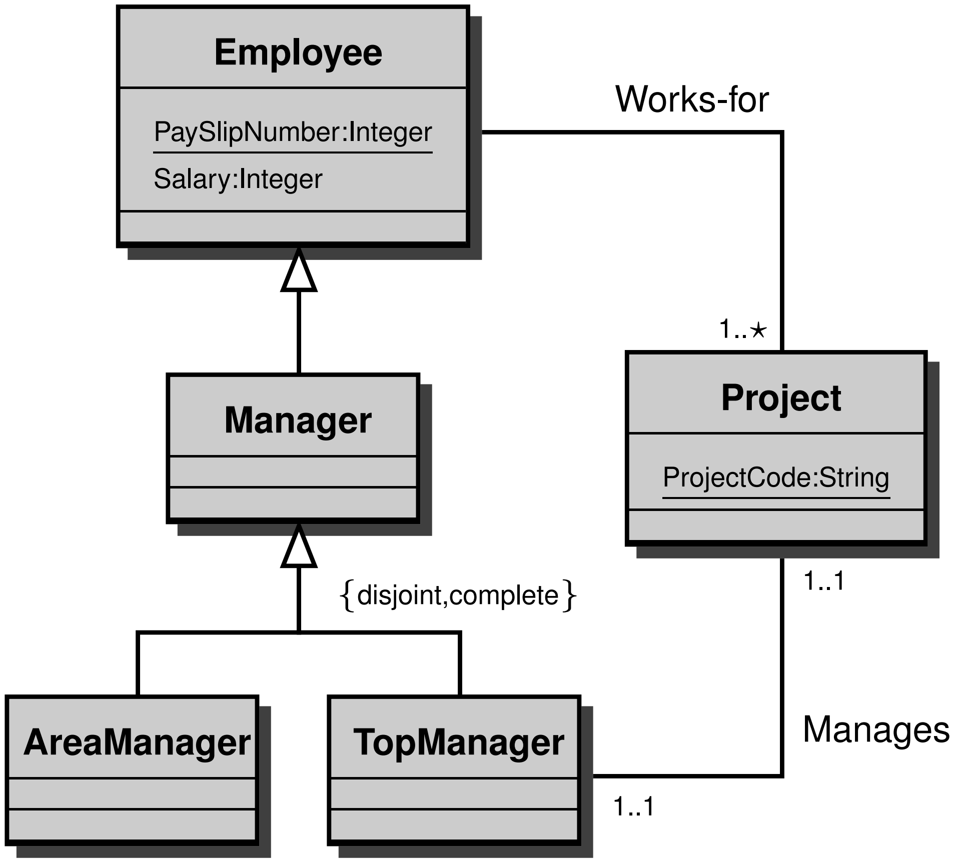
Encoding conceptual data models in $\mathcal{DLR}$

- Object-oriented data models (e.g., UML and ODMG)
- Semantic data models (e.g., EER and ORM)
- Frame-based ontology languages (e.g., DAML+OIL)
- Theorems **prove** that an ontology and its encoding as $\mathcal{DLR}$ knowledge bases constrain every world description in the same way – i.e., the models of the $\mathcal{DLR}$ theory correspond to the legal world descriptions of the ontology, and vice-versa.



$$
\begin{array}{lcl}
\mathtt{Works\text{-}for} & \sqsubseteq & \mathtt{emp}/2 : \mathtt{Employee} \sqcap \mathtt{act}/2 : \mathtt{Project} \\
\mathtt{Manages} & \sqsubseteq & \mathtt{man}/2 : \mathtt{TopManager} \sqcap \mathtt{prj}/2 : \mathtt{Project} \\
\mathtt{Employee} & \sqsubseteq & \exists^{=1}[\mathtt{worker}](\mathtt{PaySlipNumber} \sqcap \mathtt{num}/2 : \mathtt{Integer}) \sqcap \\
 & & \exists^{=1}[\mathtt{payee}](\mathtt{Salary} \sqcap \mathtt{amount}/2 : \mathtt{Integer}) \\
\top & \sqsubseteq & \exists^{\leq 1}[\mathtt{num}](\mathtt{PaySlipNumber} \sqcap \mathtt{worker}/2 : \mathtt{Employee}) \\
\mathtt{Manager} & \sqsubseteq & \mathtt{Employee} \sqcap (\mathtt{AreaManager} \sqcup \mathtt{TopManager}) \\
\mathtt{AreaManager} & \sqsubseteq & \mathtt{Manager} \sqcap \neg\mathtt{TopManager} \\
\mathtt{TopManager} & \sqsubseteq & \mathtt{Manager} \sqcap \exists^{=1}[\mathtt{man}]\mathtt{Manages} \\
\mathtt{Project} & \sqsubseteq & \exists^{\geq 1}[\mathtt{act}]\mathtt{Works\text{-}for} \sqcap \exists^{=1}[\mathtt{prj}]\mathtt{Manages} \\
\ldots & &
\end{array}
$$

Reasoning with constraints

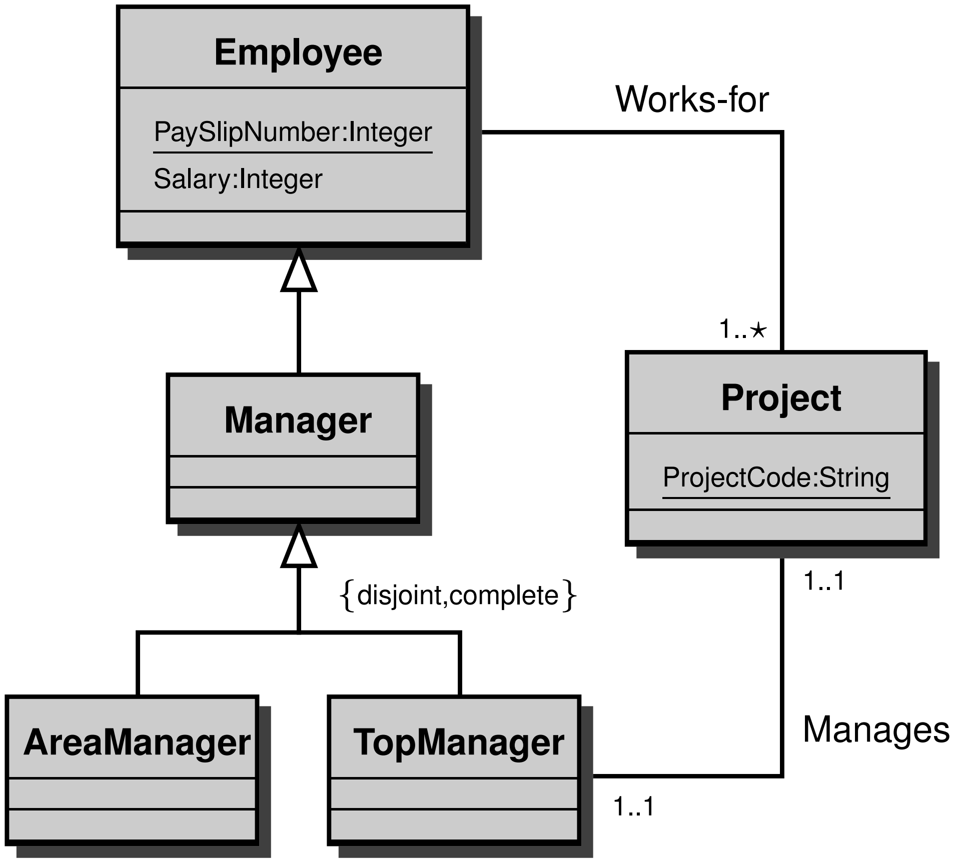


Managers are employees who do not work for a project (she/he just manages it):

$$\texttt{Employee} \sqcap \neg(\exists^{\geq 1}[\texttt{emp}]\texttt{Works-for}) \sqsubseteq \texttt{Manager}$$

$$\texttt{Manager} \sqsubseteq \neg(\exists^{\geq 1}[\texttt{emp}]\texttt{Works-for})$$

Reasoning with constraints



Managers are employees who do not work for a project (she/he just manages it):

$$\mathtt{Employee} \sqcap \neg(\exists^{\geq 1}[\mathtt{emp}]\mathtt{Works\text{-}for}) \sqsubseteq \mathtt{Manager}$$

$$\mathtt{Manager} \sqsubseteq \neg(\exists^{\geq 1}[\mathtt{emp}]\mathtt{Works\text{-}for})$$

$\Longrightarrow$ For every project, there is at least one employee who is not a manager:

$$\mathtt{Project} \sqsubseteq \exists^{\geq 1}[\mathtt{act}](\mathtt{Works\text{-}for} \sqcap \mathtt{emp} : \neg\mathtt{Manager})$$

Extensions of $\mathcal{DLR}$

- $\mathcal{DLR}_{reg}$: regular expressions and recursive views (beyond FOL)
- $\mathcal{DLR}_{\mathcal{US}}$: temporal constructs to model temporal databases (temporal logic)
- $\mathcal{DLR}_{key}$: general key constraints

Reasoning with Ontologies

- Exploit the $\mathcal{DLR}$ reasoning procedures for solving reasoning problems in the ontology enriched with constraints.
- Logical implication and consistency for $\mathcal{DLR}$ knowledge bases is decidable and EXPTIME-complete, and practical, proved correct and complete algorithms exist in implemented systems.

Reasoning with Ontologies

- Exploit the $\mathcal{DLR}$ reasoning procedures for solving reasoning problems in the ontology enriched with constraints.

- Logical implication and consistency for $\mathcal{DLR}$ knowledge bases is decidable and EXPTIME-complete, and practical, proved correct and complete algorithms exist in implemented systems.

- $\leadsto$ Ontology consistency checking with constraints and logical implication of constraints in ontologies are all decidable EXPTIME-complete problems.

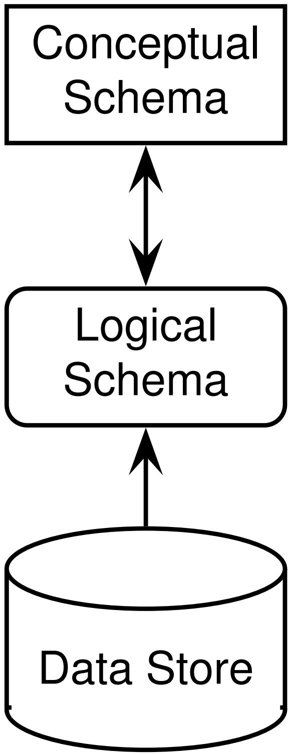
Reasoning with Ontologies

- Exploit the $\mathcal{DLR}$ reasoning procedures for solving reasoning problems in the ontology enriched with constraints.
- Logical implication and consistency for $\mathcal{DLR}$ knowledge bases is decidable and EXPTIME-complete, and practical, proved correct and complete algorithms exist in implemented systems.
- $\leadsto$ Ontology consistency checking with constraints and logical implication of constraints in ontologies are all decidable EXPTIME-complete problems.
- i●com is an implemented conceptual modelling tool using in the background a $\mathcal{DLR}$ ontology server supporting the ontology design.

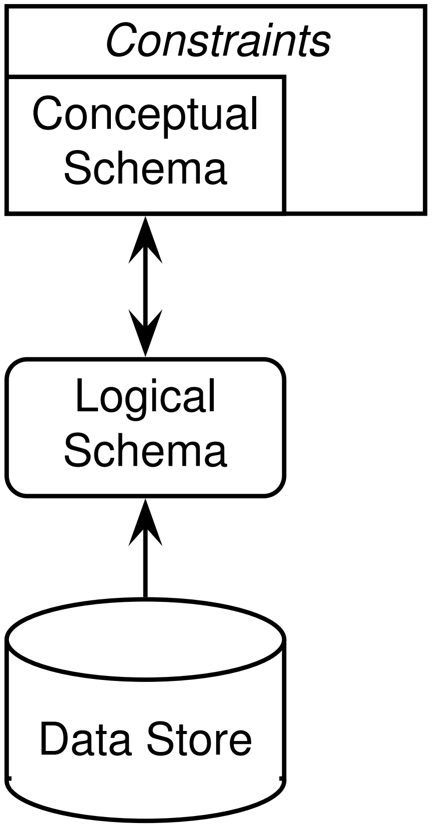
Summary

- Logic and Conceptual Modelling
- Description Logics for Conceptual Modelling
- Queries with an Ontology
- Ontology Integration

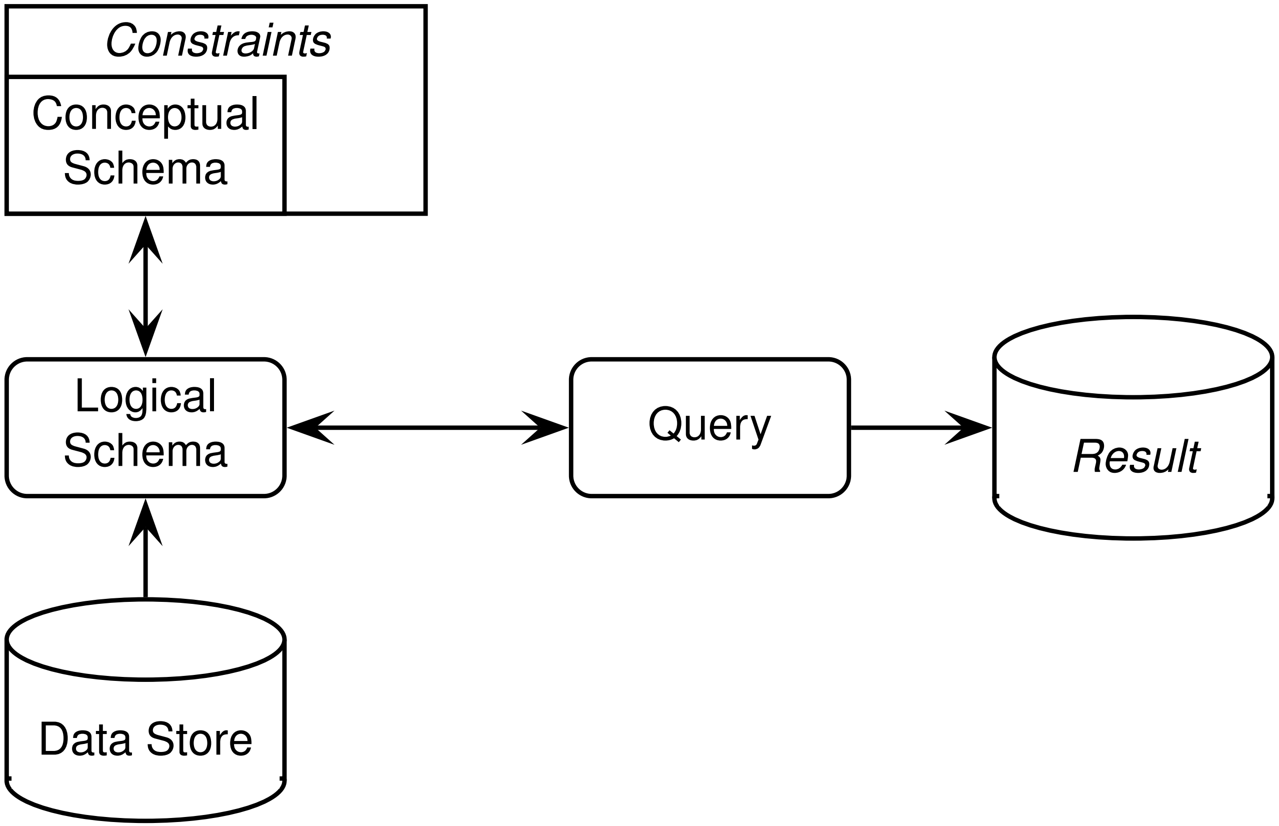
The role of a Conceptual Schema – revisited



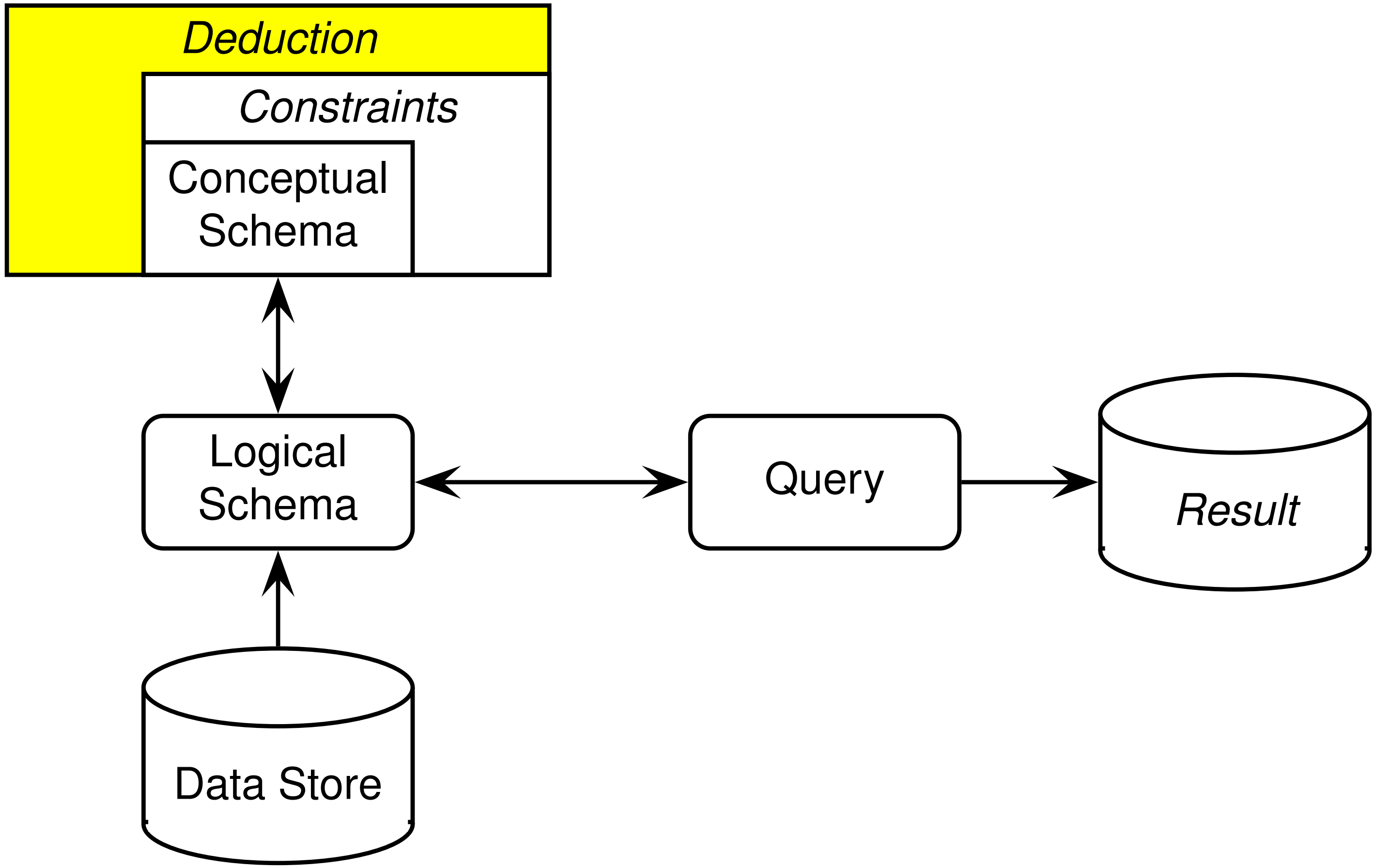
The role of a Conceptual Schema – revisited



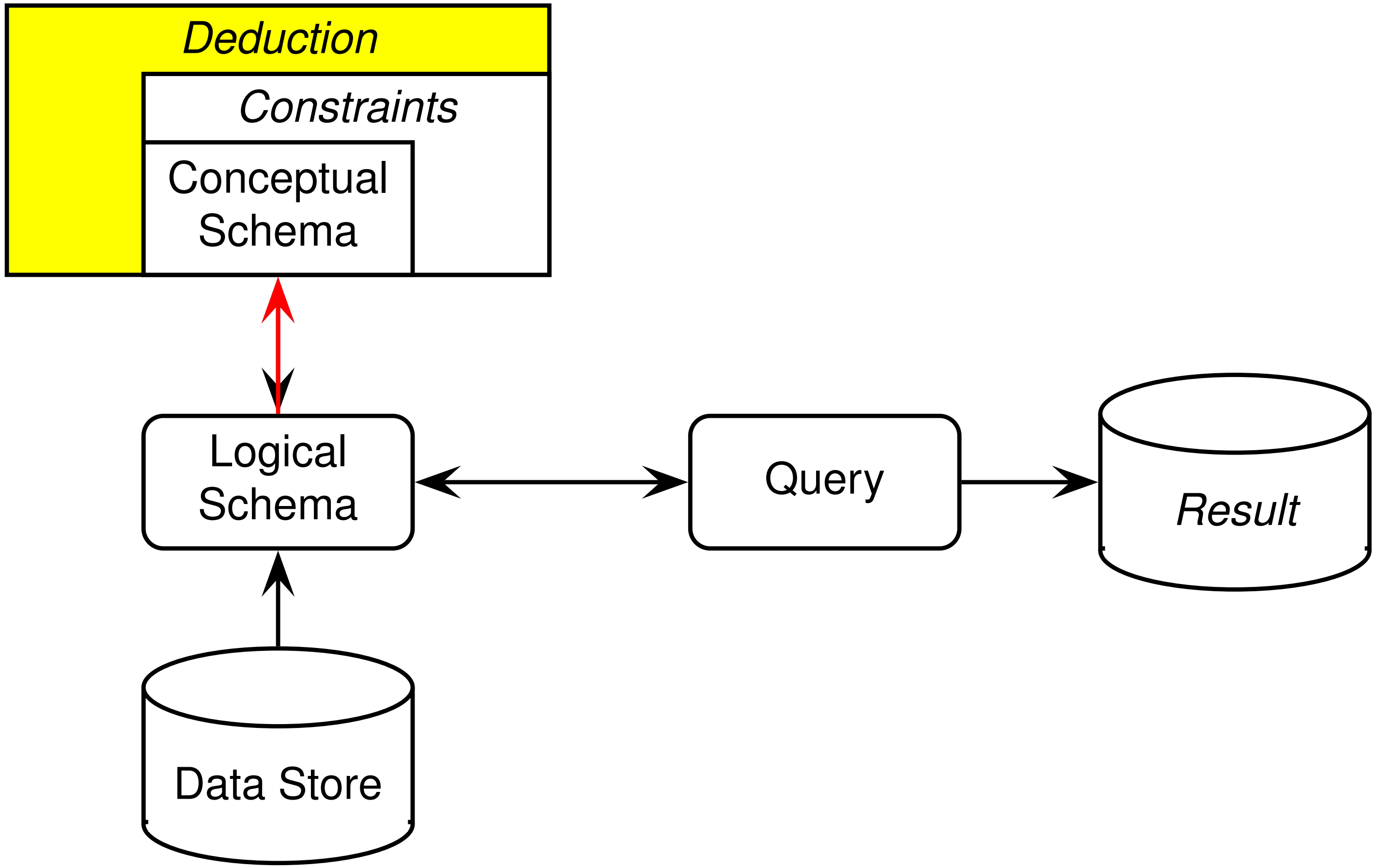
The role of a Conceptual Schema – revisited



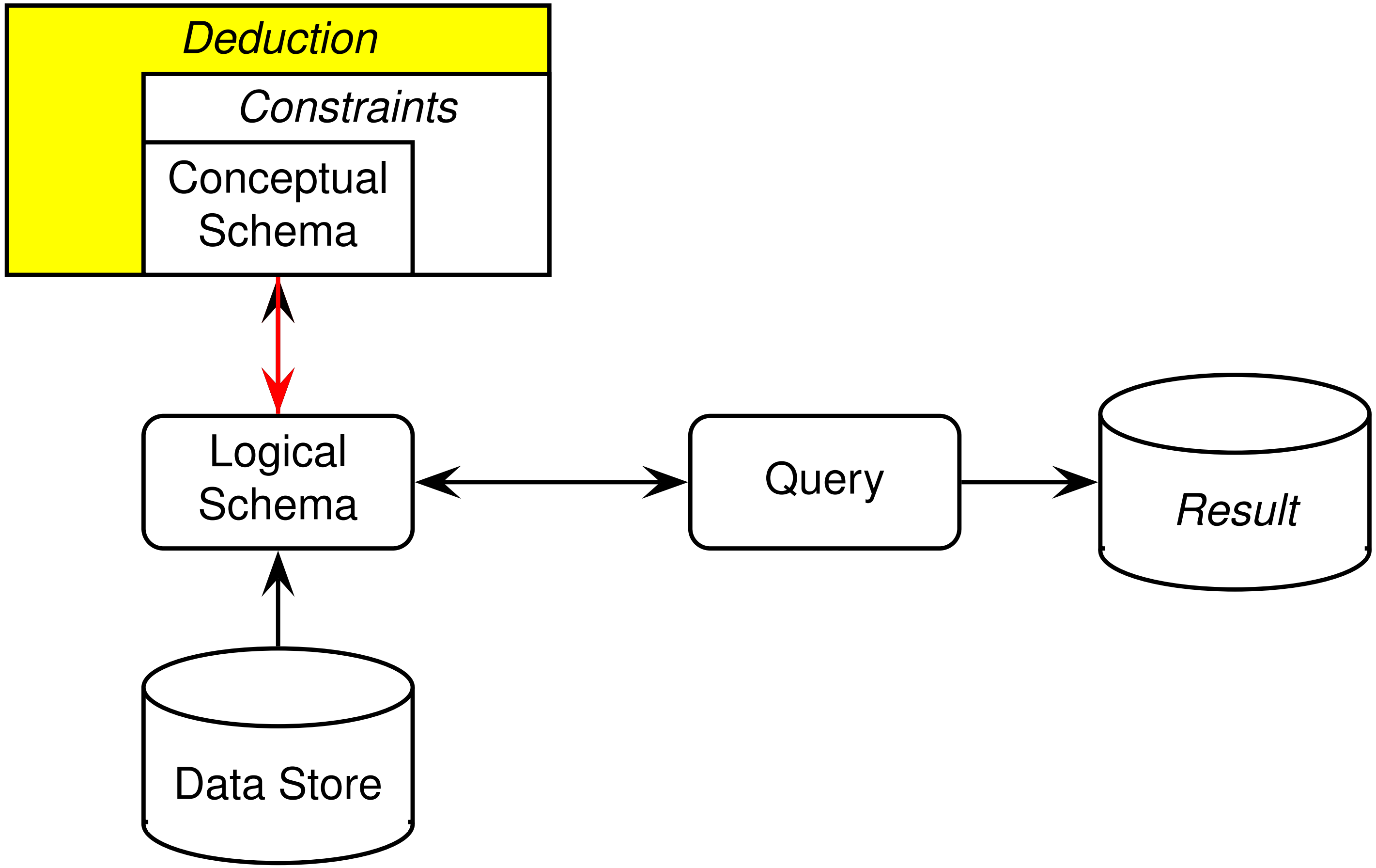
The role of a Conceptual Schema – revisited



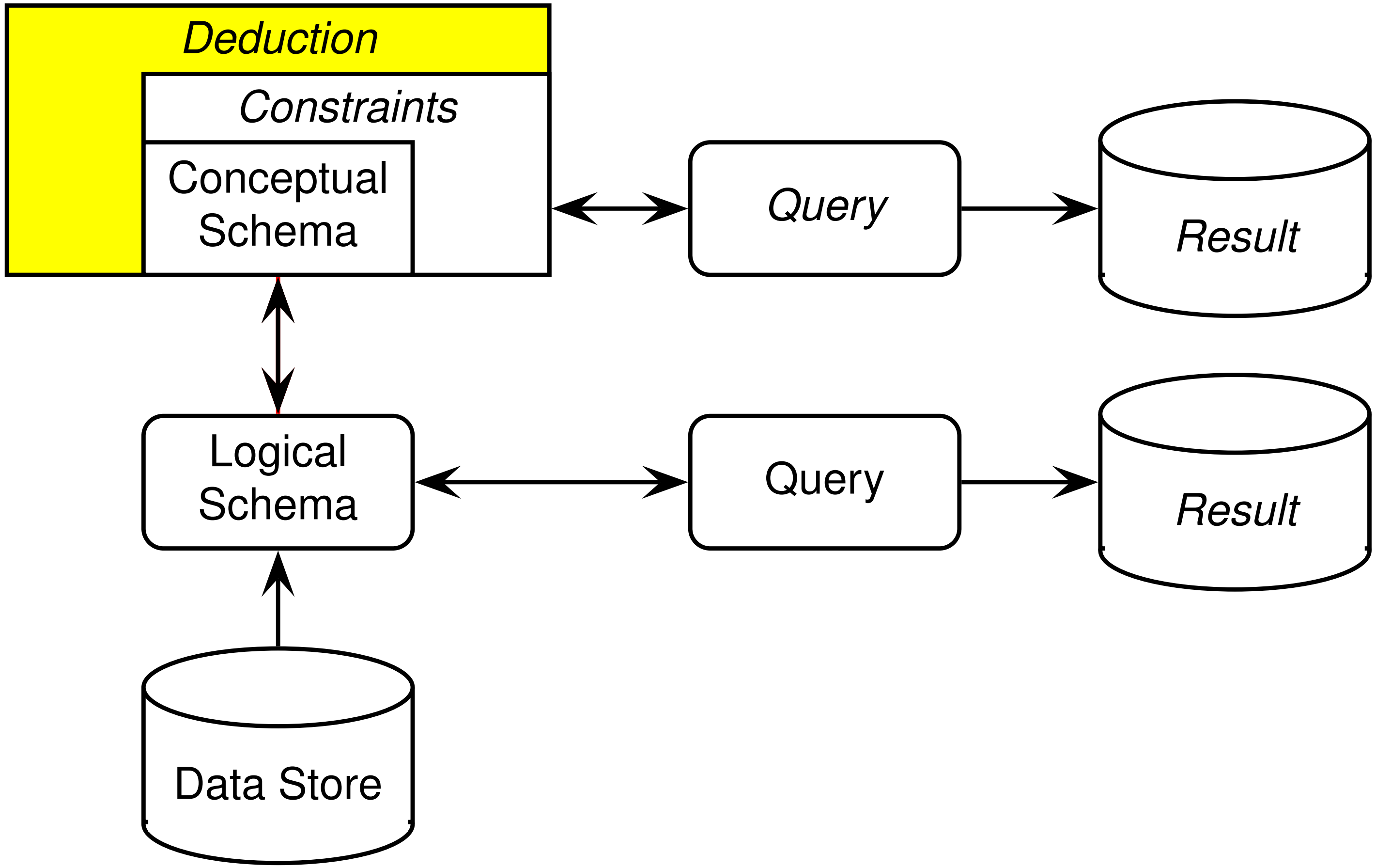
The role of a Conceptual Schema – revisited



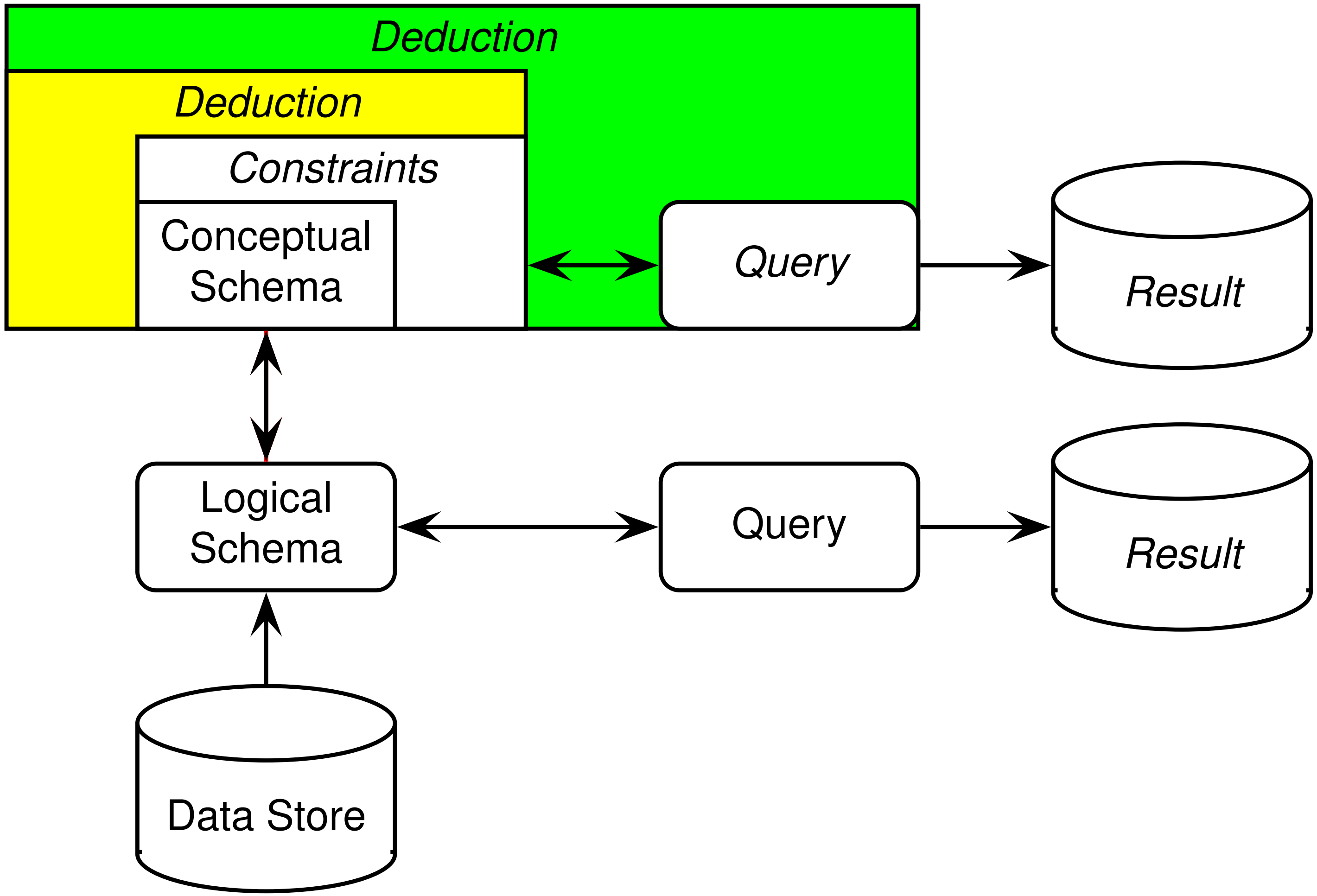
The role of a Conceptual Schema – revisited



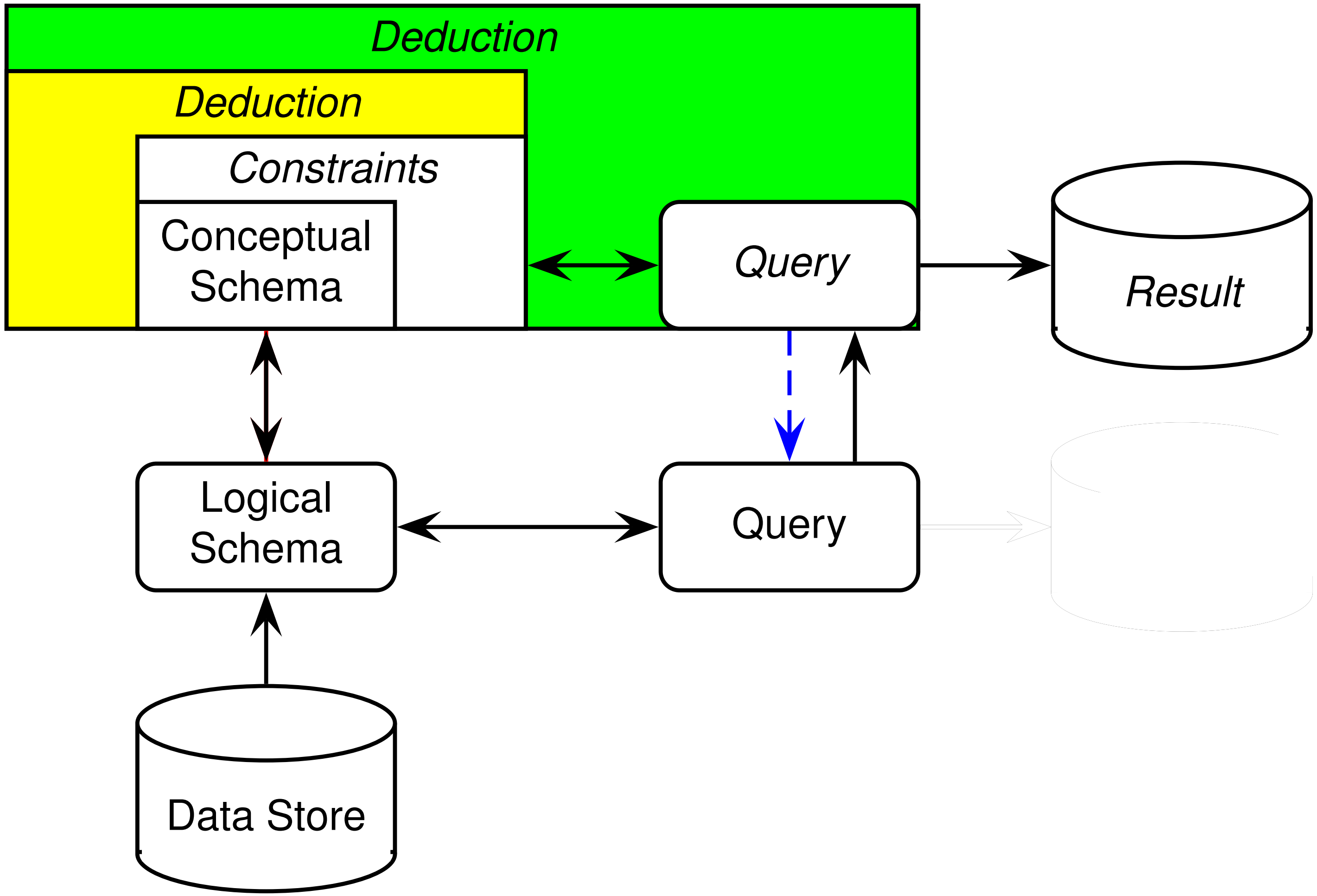
The role of a Conceptual Schema – revisited



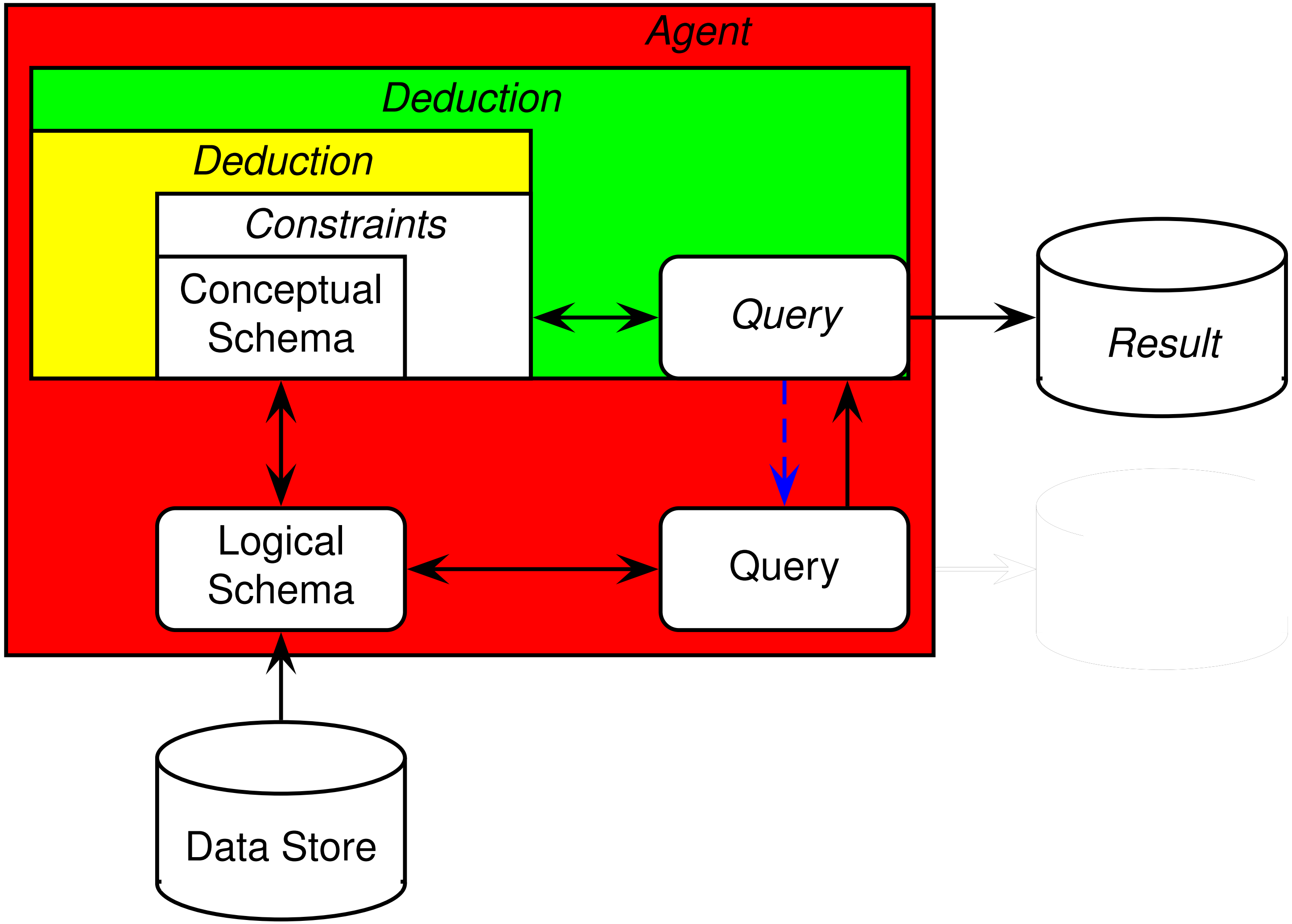
The role of a Conceptual Schema – revisited



The role of a Conceptual Schema – revisited



The role of a Conceptual Schema – revisited



Adapting standard DB query technology

- Assumption 1: complete information **about each term** appearing in the ontology
- Assumption 2: consistent information with respect to the constraints introduced by the ontology
- Problem: answer a query over the ontology vocabulary

Adapting standard DB query technology

- Assumption 1: complete information **about each term** appearing in the ontology
- Assumption 2: consistent information with respect to the constraints introduced by the ontology
- Problem: answer a query over the ontology vocabulary
- Solution: use a standard DB technology (e.g., SQL, datalog, etc)

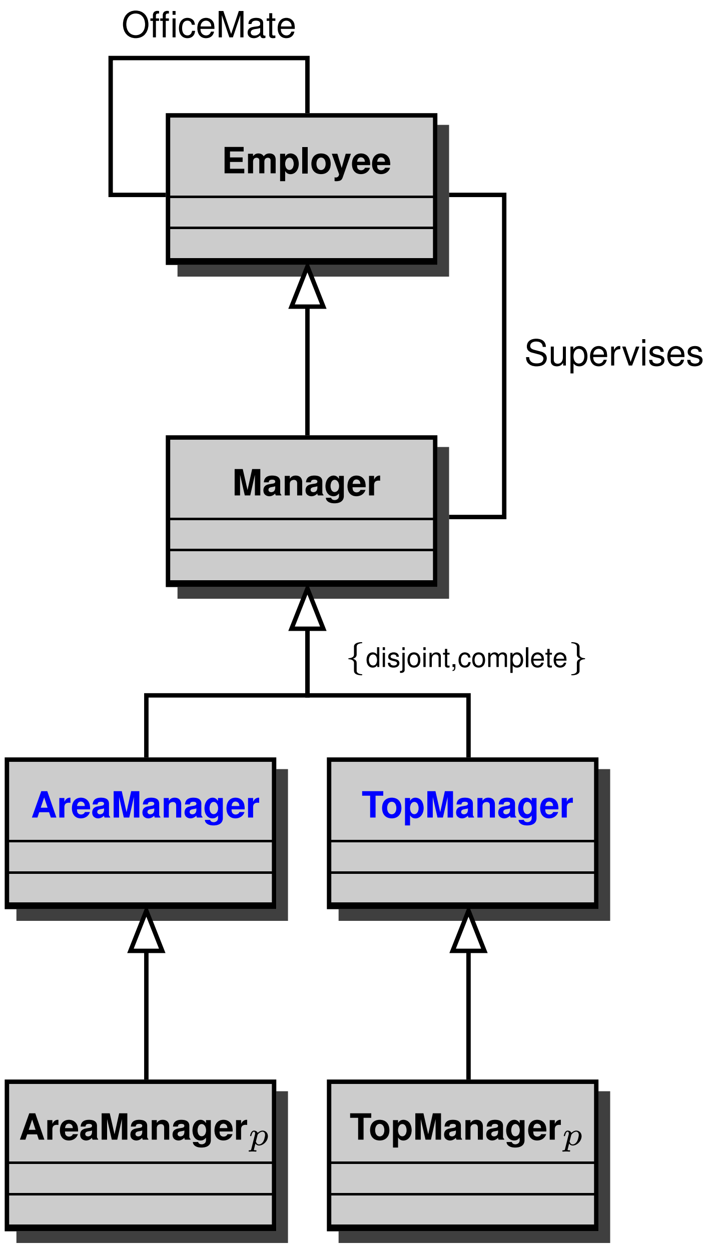
Adapting standard DB query technology

- Assumption 1: complete information **about each term** appearing in the ontology
- Assumption 2: consistent information with respect to the constraints introduced by the ontology
- Problem: answer a query over the ontology vocabulary
- Solution: use a standard DB technology (e.g., SQL, datalog, etc)
- Assumption 1 is against the principle that an ontology presents a richer vocabulary than the data stores.

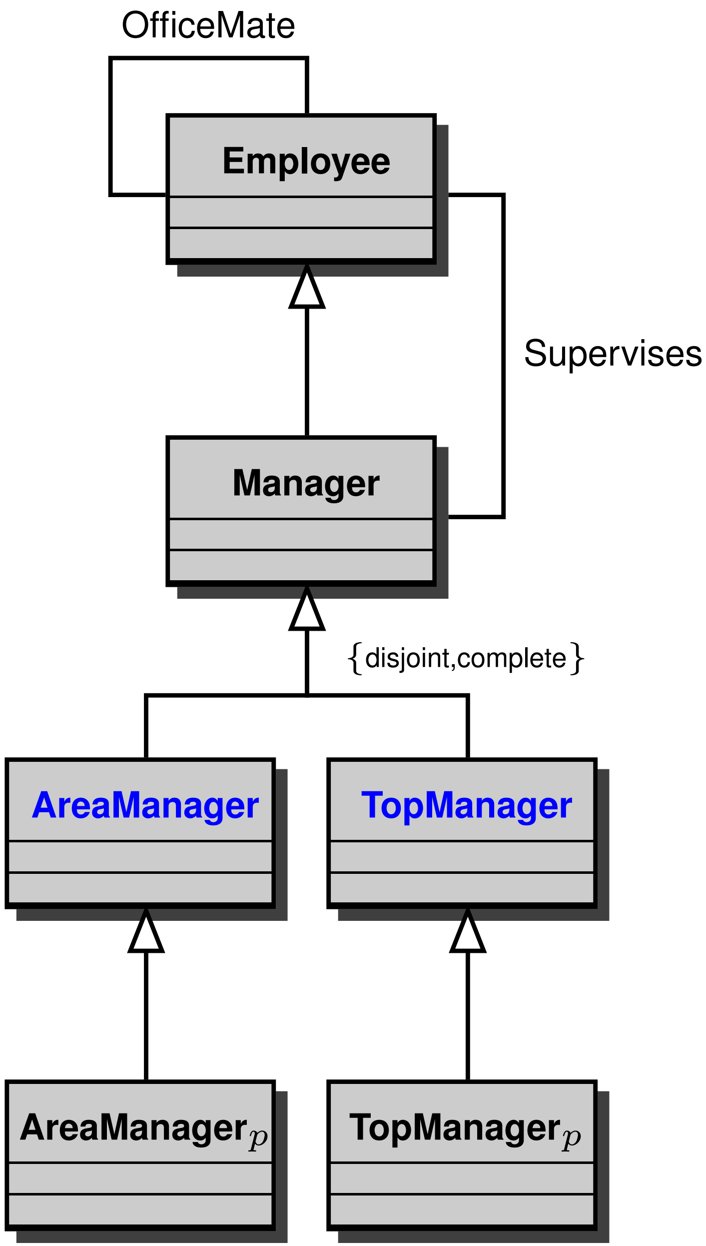
Weakening the assumptions

- Assumption 1_{weak}: complete information **about *some* term** appearing in the ontology
- Standard DB technologies do not apply
- The query answering problem in this context is inherently complex

Example


OfficeMate
Employee
Supervises
Manager
{disjoint,complete}
AreaManager
TopManager
AreaManager$_p$
TopManager$_p$

Example



$\texttt{Employee} = \{$ **John**, **Andrea**, **Mary**, **Paul** $\}$

$\texttt{Manager} = \{$ **John**, **Andrea** $\}$

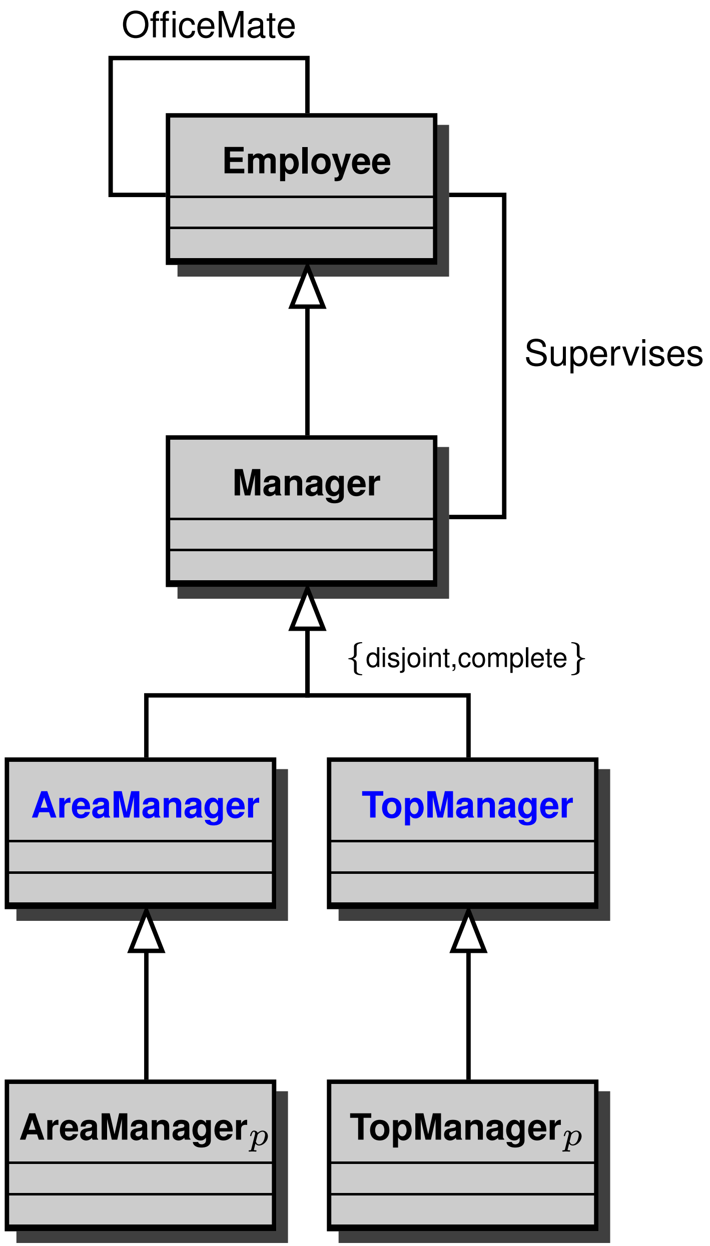
$\texttt{AreaManager}_p = \{$ **Paul** $\}$

$\texttt{TopManager}_p = \{$ **Mary** $\}$

$\texttt{Supervises} = \{$ (**John**,**Andrea**), (**John**,**Mary**) $\}$

$\texttt{OfficeMate} = \{$ (**Mary**,**Andrea**), (**Andrea**,**Paul**) $\}$

Example



Employee = { **John**, **Andrea**, **Mary**, **Paul** }

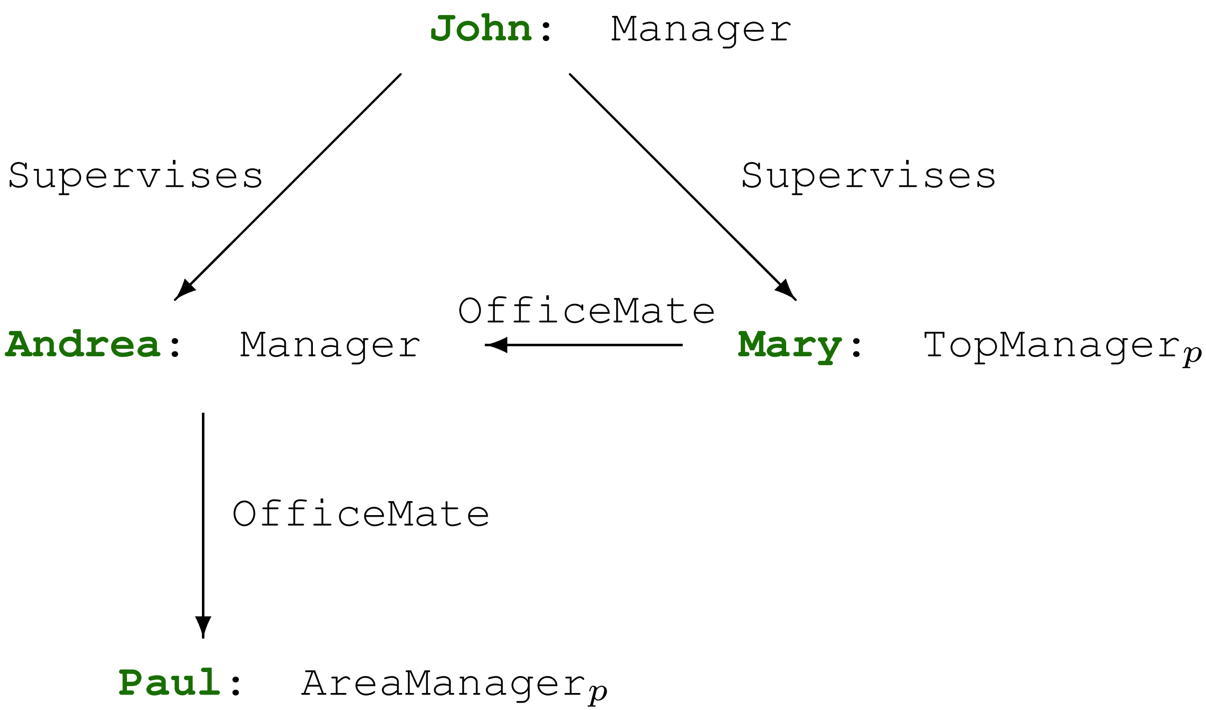
Manager = { **John**, **Andrea** }

AreaManager$_p$ = { **Paul** }

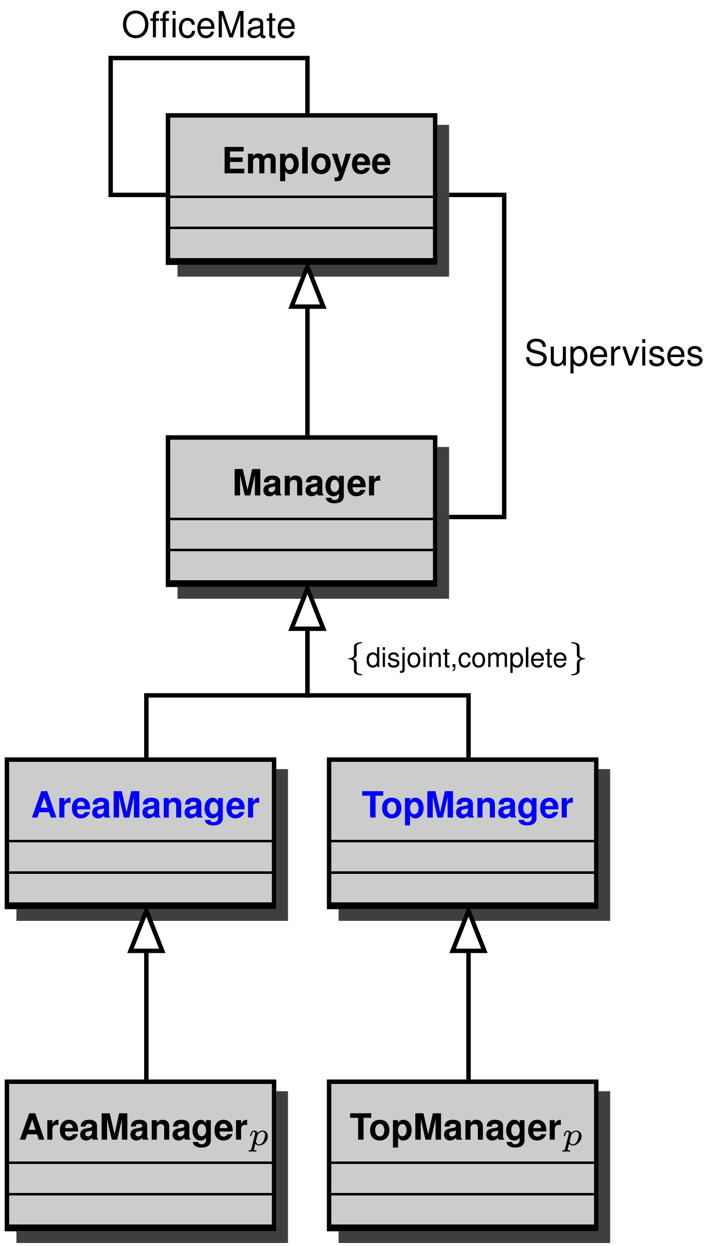
TopManager$_p$ = { **Mary** }

Supervises = { (**John**,**Andrea**), (**John**,**Mary**) }

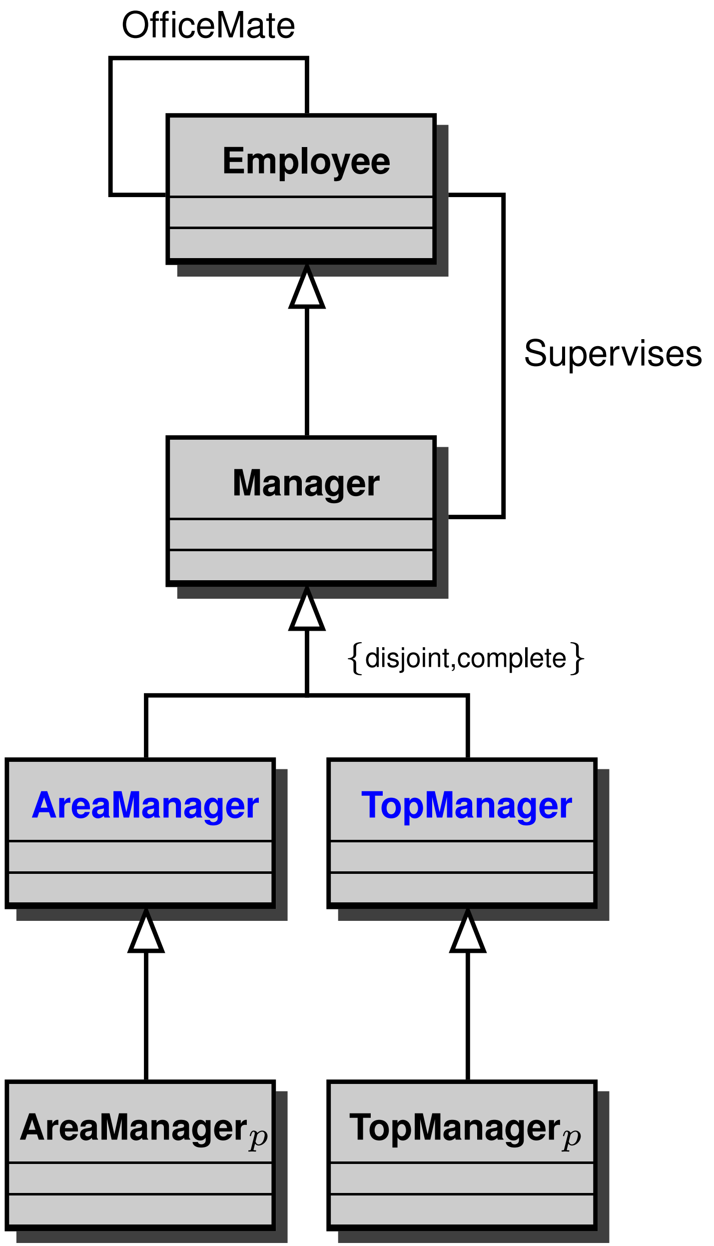
OfficeMate = { (**Mary**,**Andrea**), (**Andrea**,**Paul**) }

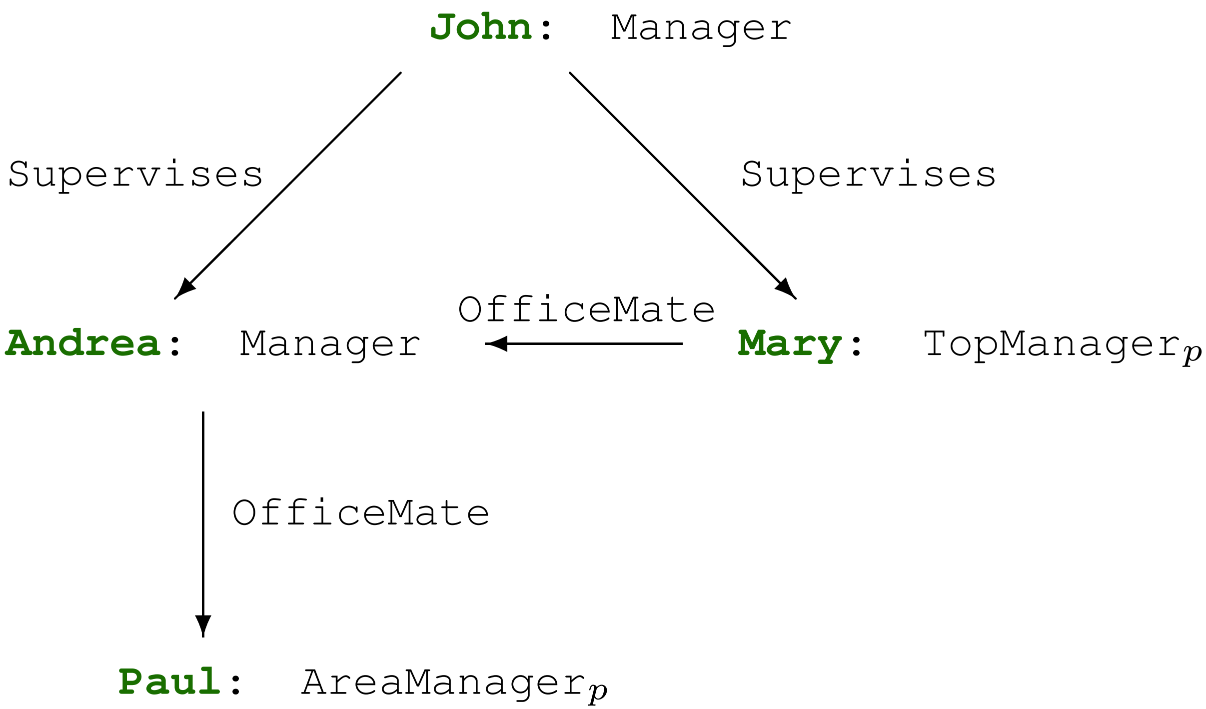


Example (cont.)

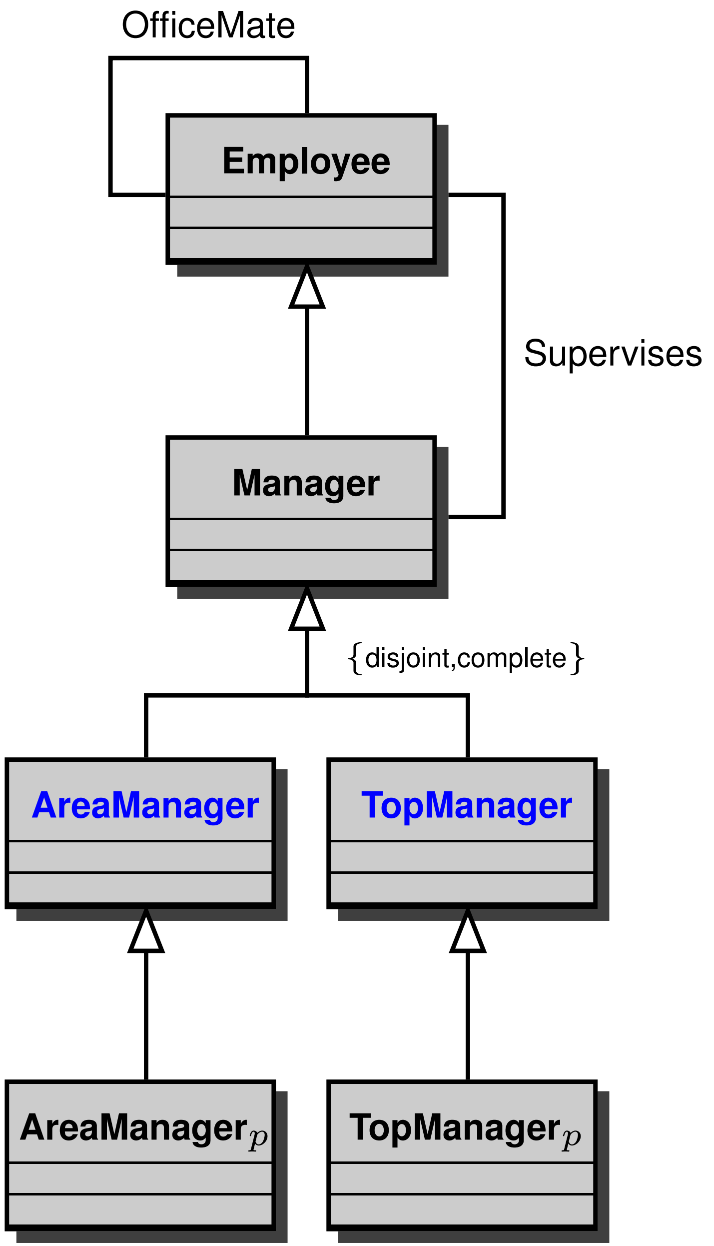


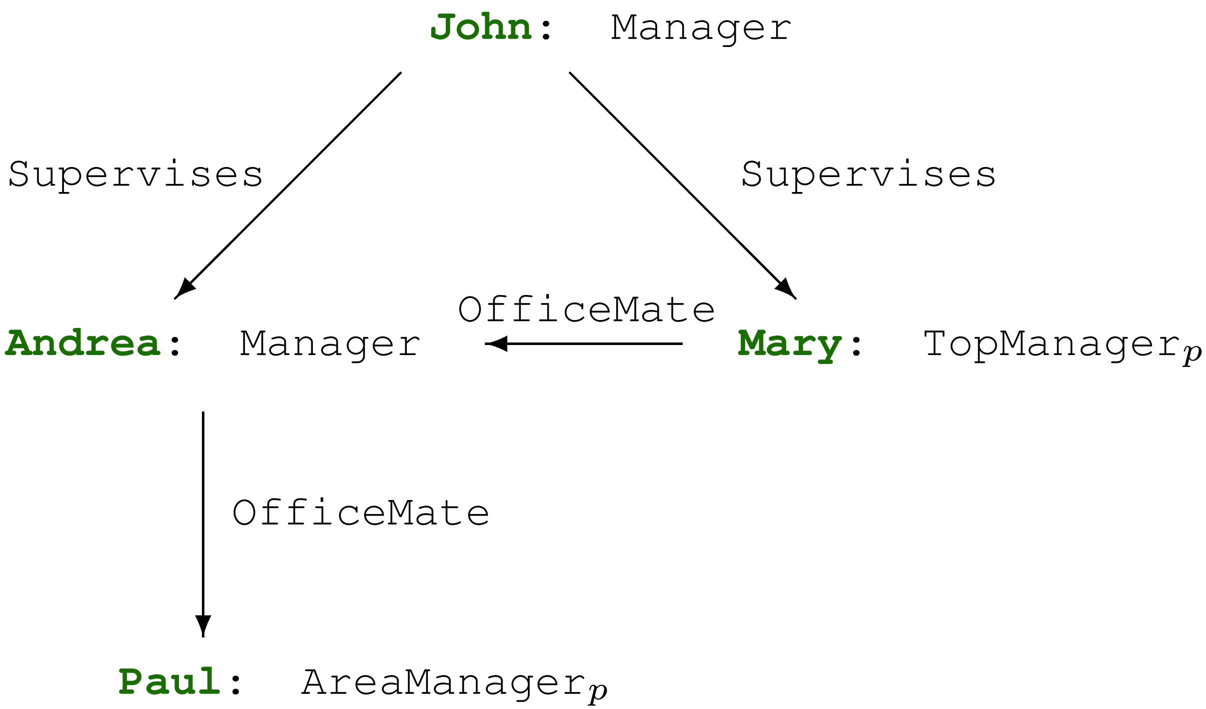
Example (cont.)





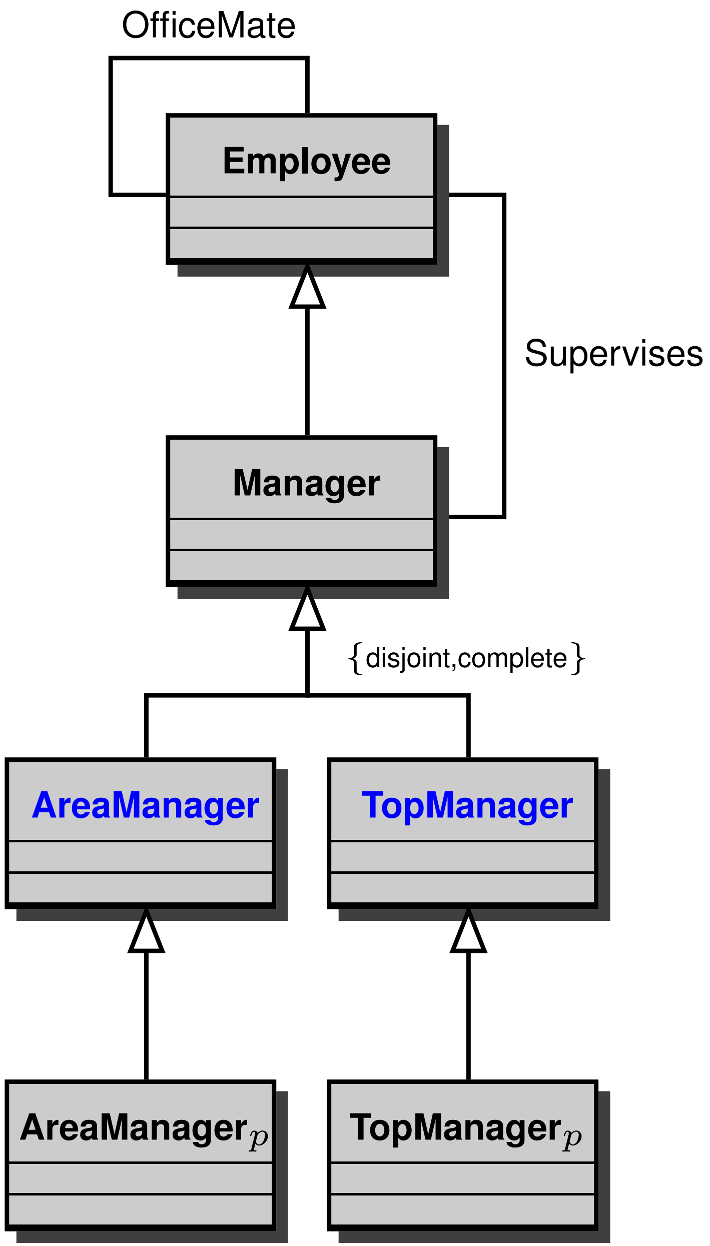
Example (cont.)

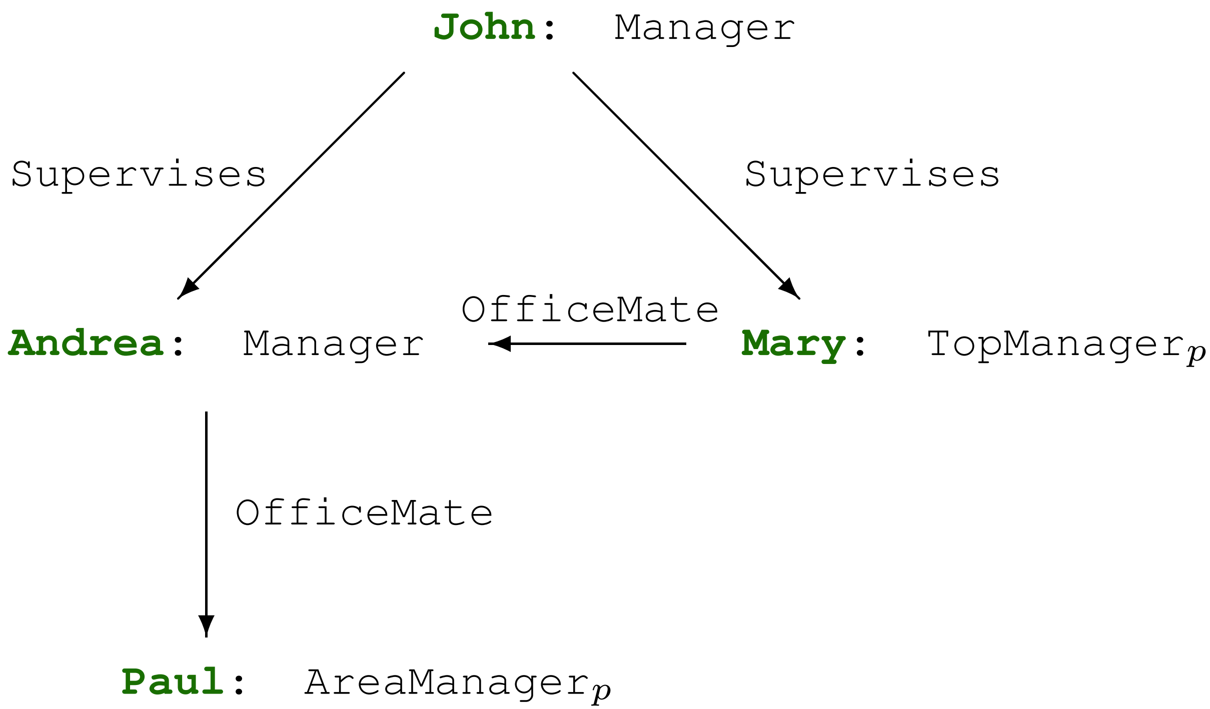




Q :- Supervises(**John**, **X**), TopManager(**X**),
Officemate(**X**, **Y**), AreaManager(**Y**)

Example (cont.)






```
Q :- Supervises(John, X), TopManager(X),
     Officemate(X, Y), AreaManager(Y)
```

YES!

Weakening the assumptions, II

In general, the link of the ontology with the information source (called mapping) can be given in terms of a set of views:

- **GAV** (global-as-view): for each ontology term one view over the information source is given;
- **LAV** (local-as-view): for each information source term one view over the ontology terms is given;

An Information Source Example

CompanyEmployee/2; CompanyProject/3

CompanyEmployee	
name	project
john	esprit-dwq
...	...

CompanyProject		
project	manager	department
esprit-dwq	enrico	cs-uman
...	...	...

An Information Source Example

$\texttt{CompanyEmployee}/2; \texttt{CompanyProject}/3$

CompanyEmployee	
name	project
john	esprit-dwq
...	...

CompanyProject		
project	manager	department
esprit-dwq	enrico	cs-uman
...	...	...

Q = “Tell me the projects in which John works, and their managers and departments.”

An Information Source Example

CompanyEmployee/2; CompanyProject/3

CompanyEmployee	
name	project
john	esprit-dwq
...	...

CompanyProject		
project	manager	department
esprit-dwq	enrico	cs-uman
...	...	...

Q = "Tell me the projects in which John works, and their managers and departments."

```
SELECT project, manager, department
FROM CompanyEmployee, CompanyProject
WHERE CompanyEmployee.name = "john" AND
      CompanyEmployee.project = CompanyProject.project
```

An Information Source Example

$\texttt{CompanyEmployee}/2; \texttt{CompanyProject}/3$

CompanyEmployee	
name	project
john	esprit-dwq
...	...

CompanyProject		
project	manager	department
esprit-dwq	enrico	cs-uman
...	...	...

Q = "Tell me the projects in which John works, and their managers and departments."

$\texttt{SELECT project, manager, department}$

$\texttt{FROM CompanyEmployee, CompanyProject}$

$\texttt{WHERE CompanyEmployee.name} = \text{“}\texttt{john}\text{”} \texttt{ AND}$

$\qquad \texttt{CompanyEmployee.project} = \texttt{CompanyProject.project}$

$$\texttt{Q} \equiv \pi_{\texttt{proj.,manager,dept.}} \sigma_{\texttt{name=john}} (\texttt{CompanyEmployee} \bowtie_{\texttt{project}} \texttt{CompanyProject})$$

An Information Source Example

$\texttt{CompanyEmployee}/2; \texttt{CompanyProject}/3$

CompanyEmployee	
name	project
john	esprit-dwq
...	...

CompanyProject		
project	manager	department
esprit-dwq	enrico	cs-uman
...	...	...

Q = "Tell me the projects in which John works, and their managers and departments."

$\texttt{SELECT}\ \texttt{project}, \texttt{manager}, \texttt{department}$

$\texttt{FROM}\ \texttt{CompanyEmployee}, \texttt{CompanyProject}$

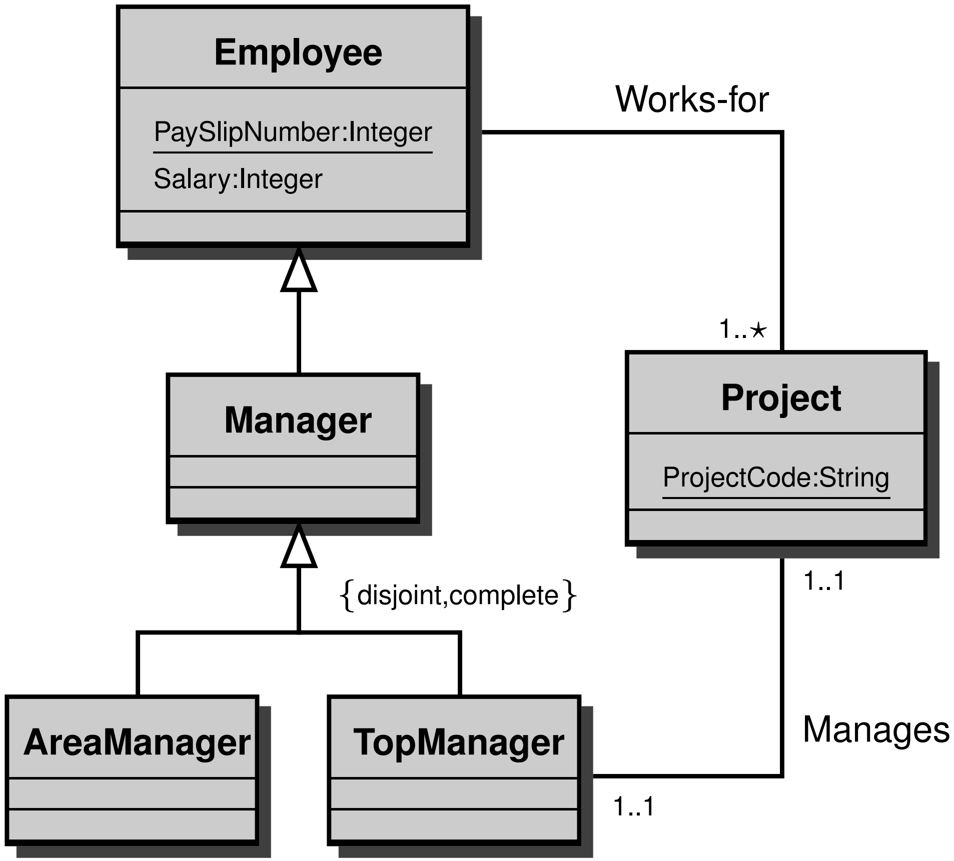
$\texttt{WHERE}\ \texttt{CompanyEmployee.name} = \text{``}\texttt{john}\text{''}\ \texttt{AND}$

$\qquad \texttt{CompanyEmployee.project} = \texttt{CompanyProject.project}$

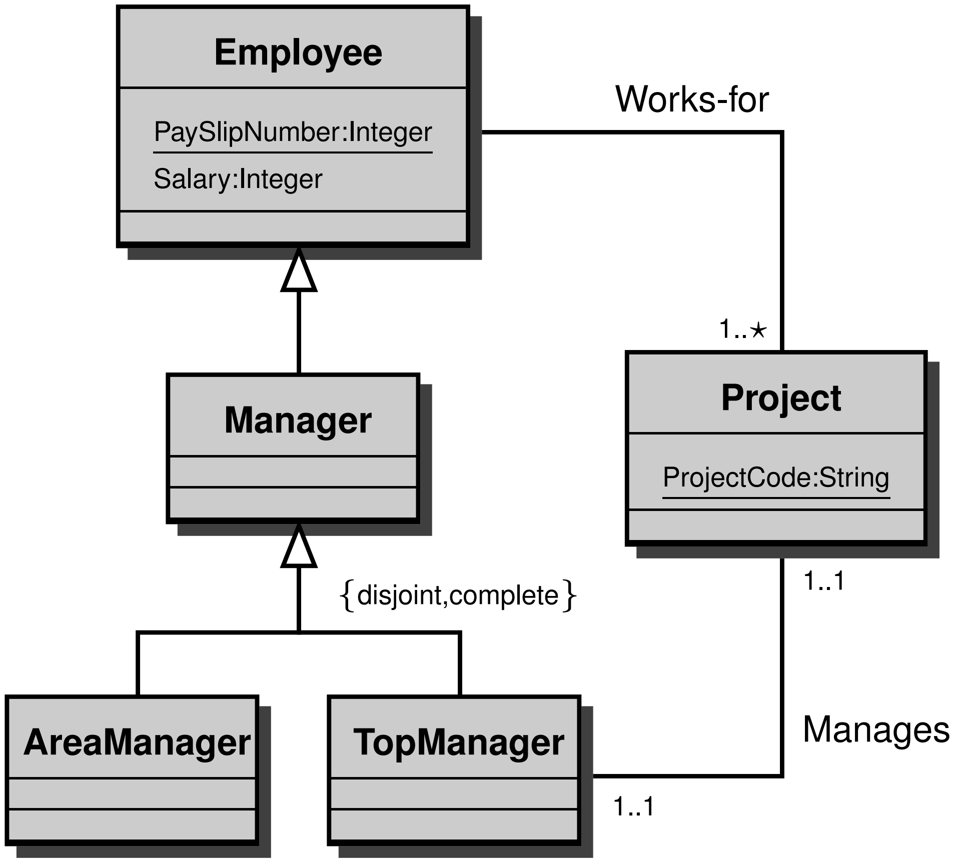
$$\texttt{Q} \equiv \pi_{\texttt{proj.,manager,dept.}} \sigma_{\texttt{name=john}} (\texttt{CompanyEmployee} \bowtie_{\texttt{project}} \texttt{CompanyProject})$$

$$\texttt{Q}(\texttt{x}, \texttt{y}, \texttt{z}) \Leftarrow \texttt{CompanyEmployee}(\texttt{john}, \texttt{x}) \wedge \texttt{CompanyProject}(\texttt{x}, \texttt{y}, \texttt{z})$$

LAV: local-as-view



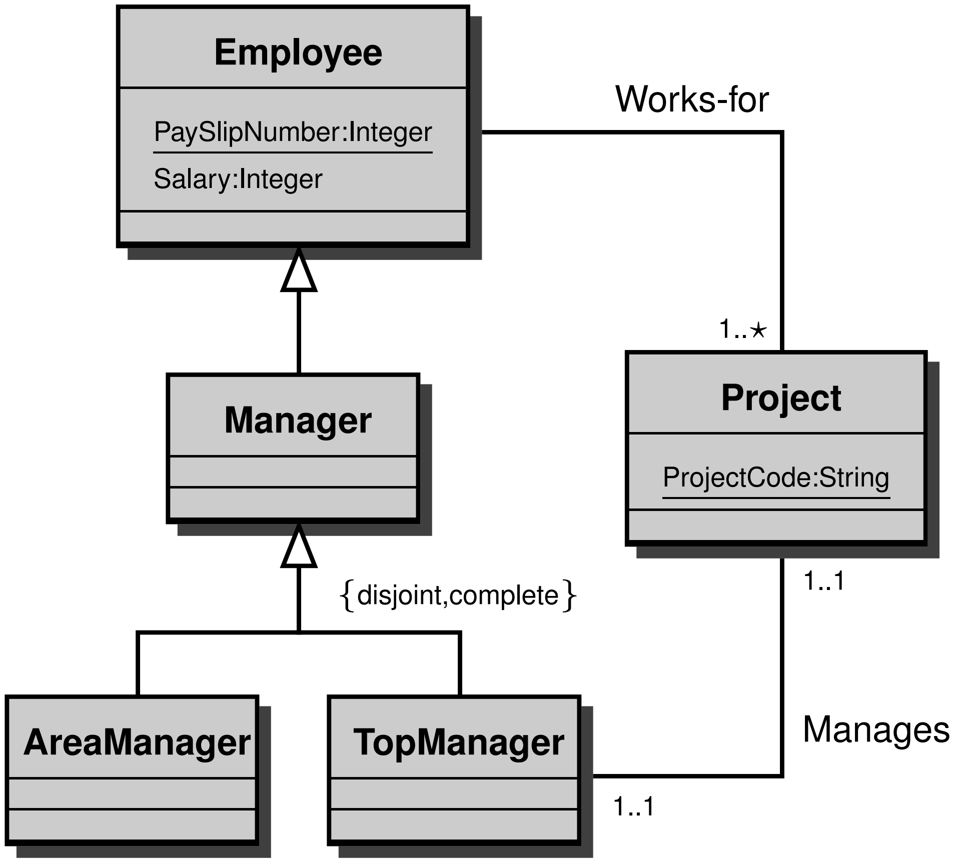
LAV: local-as-view



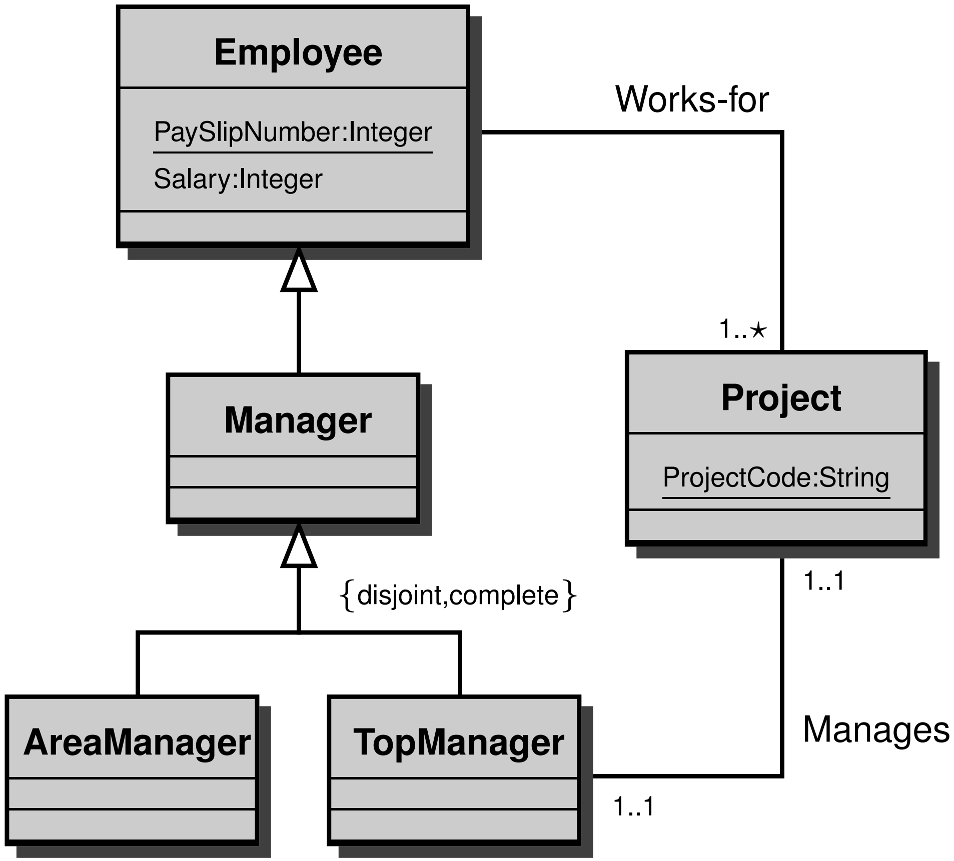
$$\texttt{CompanyEmployee}(x, y) \Leftarrow \texttt{Employee}(x) \wedge \texttt{Project}(y) \wedge \texttt{Works-for}(x, y).$$

$$\texttt{CompanyProject}(x, y, z) \Leftarrow \texttt{Project}(x) \wedge \texttt{Manager}(y) \wedge \texttt{Department}(z) \wedge \texttt{Manages}(y, x) \wedge \texttt{Resp-for}(z, x).$$

GAV: global-as-view



GAV: global-as-view



$$\texttt{Project}(x) \Leftarrow \texttt{CompanyEmployee}(y, x) \cup \texttt{CompanyProject}(x, y, z)$$

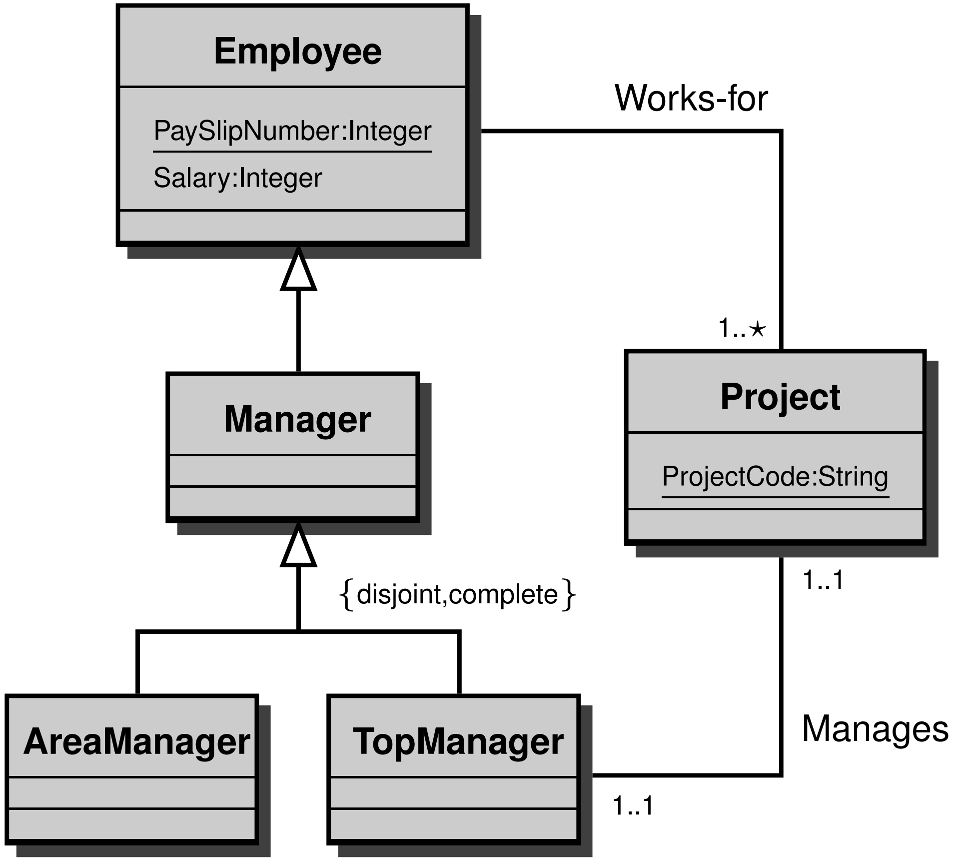
$$\texttt{Works-for}(x, y) \Leftarrow \texttt{CompanyEmployee}(x, y)$$

$$\texttt{TopManager}(x) \Leftarrow \texttt{CompanyProject}(y, x, z)$$

$$\texttt{Manages}(x, y) \Leftarrow \texttt{CompanyProject}(y, x, z)$$

Querying via the Ontology (local-as-view)

$$\mathtt{Q}(x, y, z) \Leftarrow \mathtt{Project}(x) \wedge \mathtt{Works\text{-}for}(john, x) \wedge \mathtt{TopManager}(y) \wedge \mathtt{Manages}(y, x) \wedge \neg\mathtt{InterestGroup}(z) \wedge \mathtt{Resp\text{-}for}(z, x).$$

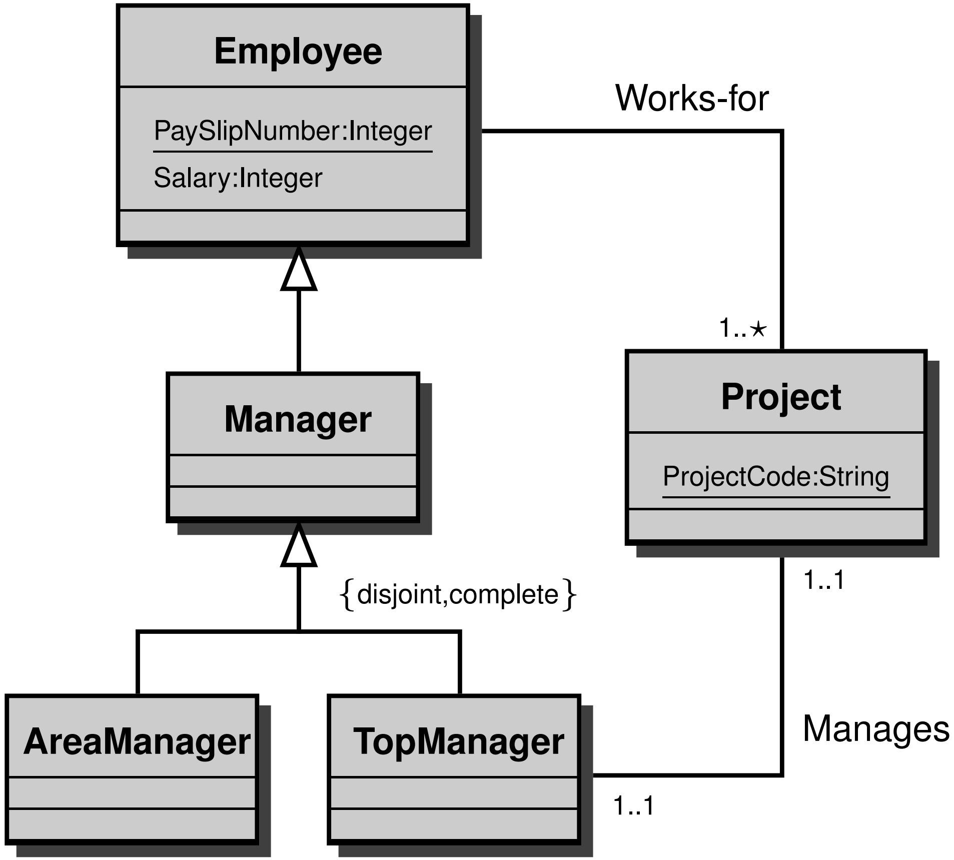


$$\mathtt{CompanyEmployee}(x, y) \Leftarrow \mathtt{Employee}(x) \wedge \mathtt{Project}(y) \wedge \mathtt{Works\text{-}for}(x, y).$$

$$\mathtt{CompanyProject}(x, y, z) \Leftarrow \mathtt{Project}(x) \wedge \mathtt{Manager}(y) \wedge \mathtt{Department}(z) \wedge \mathtt{Manages}(y, x) \wedge \mathtt{Resp\text{-}for}(z, x).$$

Querying via the Ontology (local-as-view)

$$Q(x,y,z) \Leftarrow \texttt{Project}(x) \wedge \texttt{Works-for}(john,x) \wedge \texttt{TopManager}(y) \wedge \texttt{Manages}(y,x) \wedge \neg\texttt{InterestGroup}(z) \wedge \texttt{Resp-for}(z,x).$$



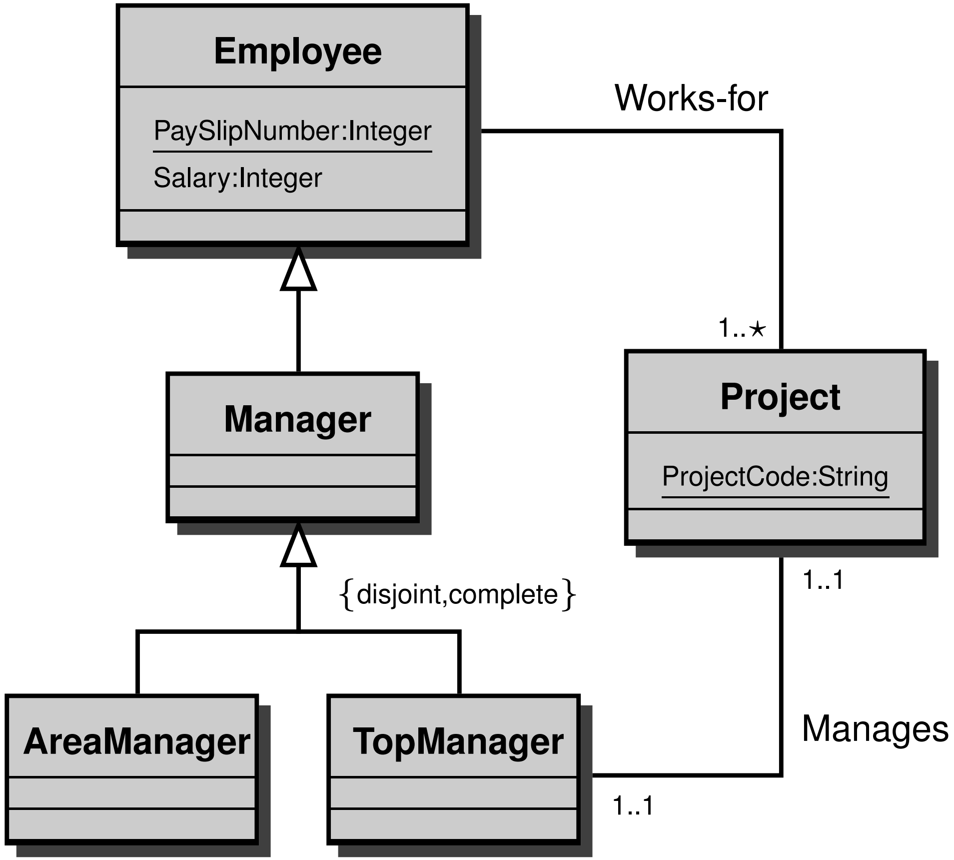
$$\texttt{CompanyEmployee}(x,y) \Leftarrow \texttt{Employee}(x) \wedge \texttt{Project}(y) \wedge \texttt{Works-for}(x,y).$$

$$\texttt{CompanyProject}(x,y,z) \Leftarrow \texttt{Project}(x) \wedge \texttt{Manager}(y) \wedge \texttt{Department}(z) \wedge \texttt{Manages}(y,x) \wedge \texttt{Resp-for}(z,x).$$

$$\rightsquigarrow Q(x,y,z) \Leftarrow \texttt{CompanyEmployee}(john,x) \wedge \texttt{CompanyProject}(x,y,z)$$

Querying via the Ontology (global-as-view)

$$\mathtt{Q}(x,y,z) \Leftarrow \mathtt{Project}(x) \wedge \mathtt{Works\text{-}for}(john,x) \wedge \mathtt{TopManager}(y) \wedge \mathtt{Manages}(y,x) \wedge \neg\mathtt{InterestGroup}(z) \wedge \mathtt{Resp\text{-}for}(z,x).$$



$$\mathtt{Project}(\mathbf{x}) \Leftarrow \mathtt{CompanyEmployee}(y,\mathbf{x}) \cup \mathtt{CompanyProject}(\mathbf{x},y,z)$$

$$\mathtt{Works\text{-}for}(\mathbf{x},\mathbf{y}) \Leftarrow \mathtt{CompanyEmployee}(\mathbf{x},\mathbf{y})$$

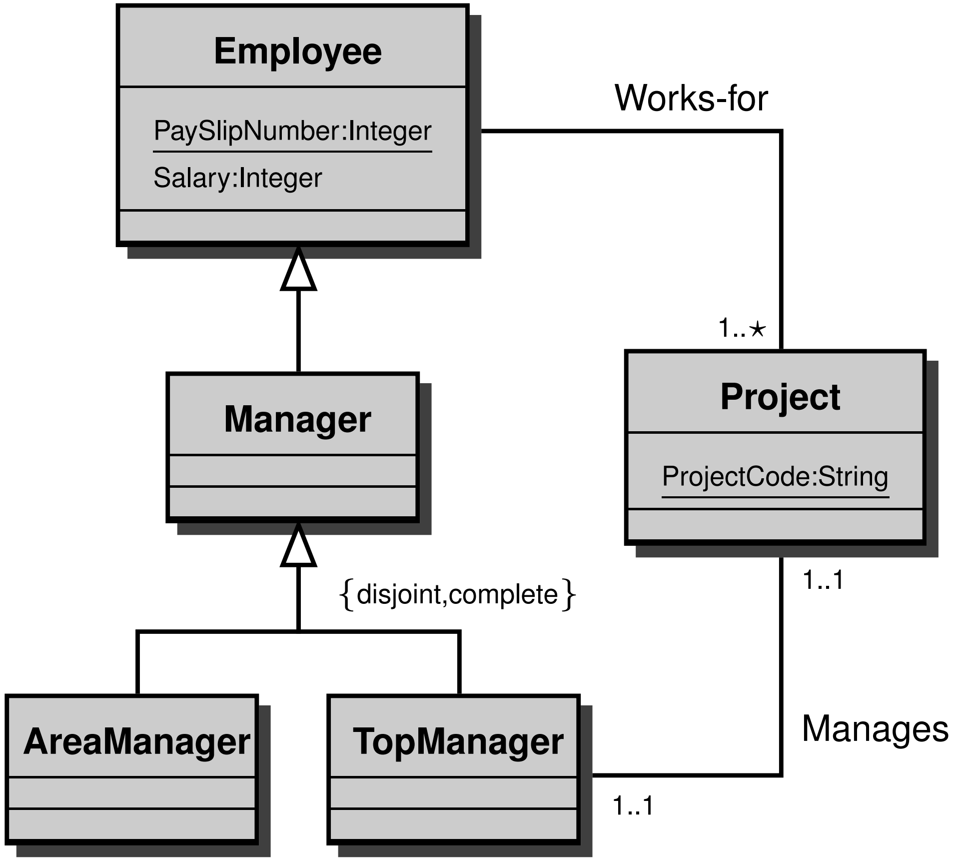
$$\mathtt{TopManager}(\mathbf{x}) \Leftarrow \mathtt{CompanyProject}(y,\mathbf{x},z)$$

$$\mathtt{Manages}(\mathbf{x},\mathbf{y}) \Leftarrow \mathtt{CompanyProject}(\mathbf{y},\mathbf{x},z)$$

$\dots$

Querying via the Ontology (global-as-view)

$$Q(x, y, z) \Leftarrow \texttt{Project}(x) \wedge \texttt{Works-for}(john, x) \wedge \texttt{TopManager}(y) \wedge \texttt{Manages}(y, x) \wedge \neg\texttt{InterestGroup}(z) \wedge \texttt{Resp-for}(z, x).$$



$$\texttt{Project}(\mathbf{x}) \Leftarrow \texttt{CompanyEmployee}(y, \mathbf{x}) \cup \texttt{CompanyProject}(\mathbf{x}, y, z)$$

$$\texttt{Works-for}(\mathbf{x}, \mathbf{y}) \Leftarrow \texttt{CompanyEmployee}(\mathbf{x}, \mathbf{y})$$

$$\texttt{TopManager}(\mathbf{x}) \Leftarrow \texttt{CompanyProject}(y, \mathbf{x}, z)$$

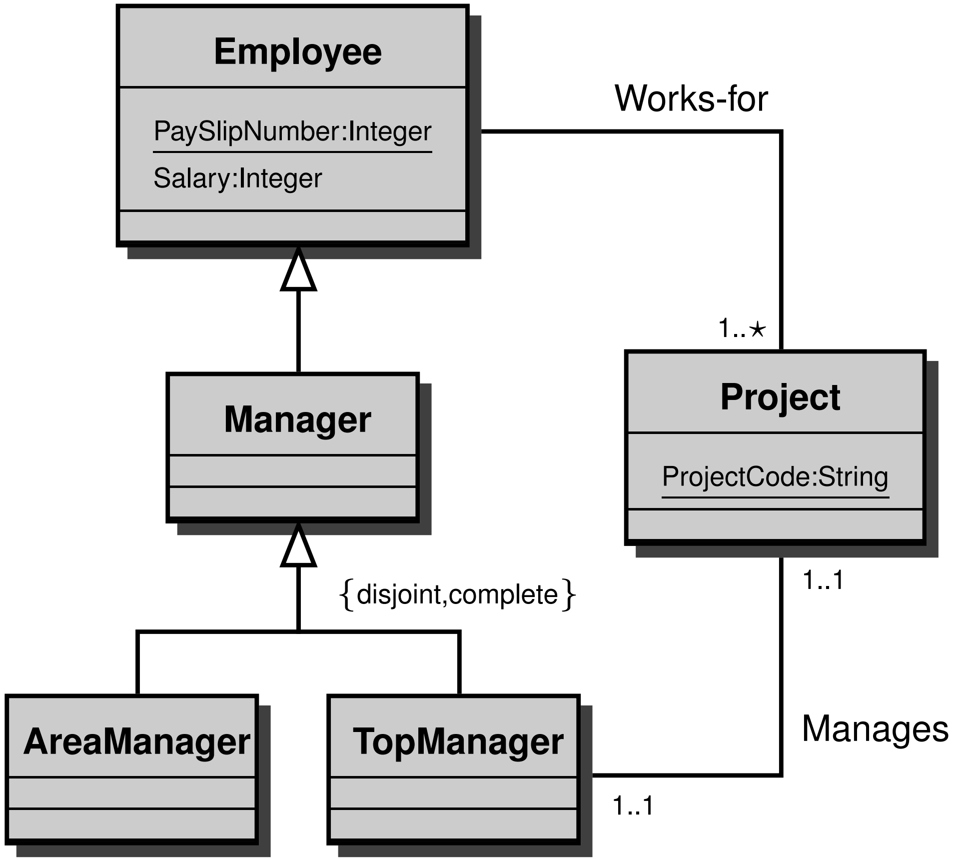
$$\texttt{Manages}(\mathbf{x}, \mathbf{y}) \Leftarrow \texttt{CompanyProject}(\mathbf{y}, \mathbf{x}, z)$$

...

$$\rightsquigarrow Q(x, y, z) \Leftarrow \texttt{CompanyEmployee}(john, x) \wedge \texttt{CompanyProject}(x, y, z)$$

Reasoning over queries

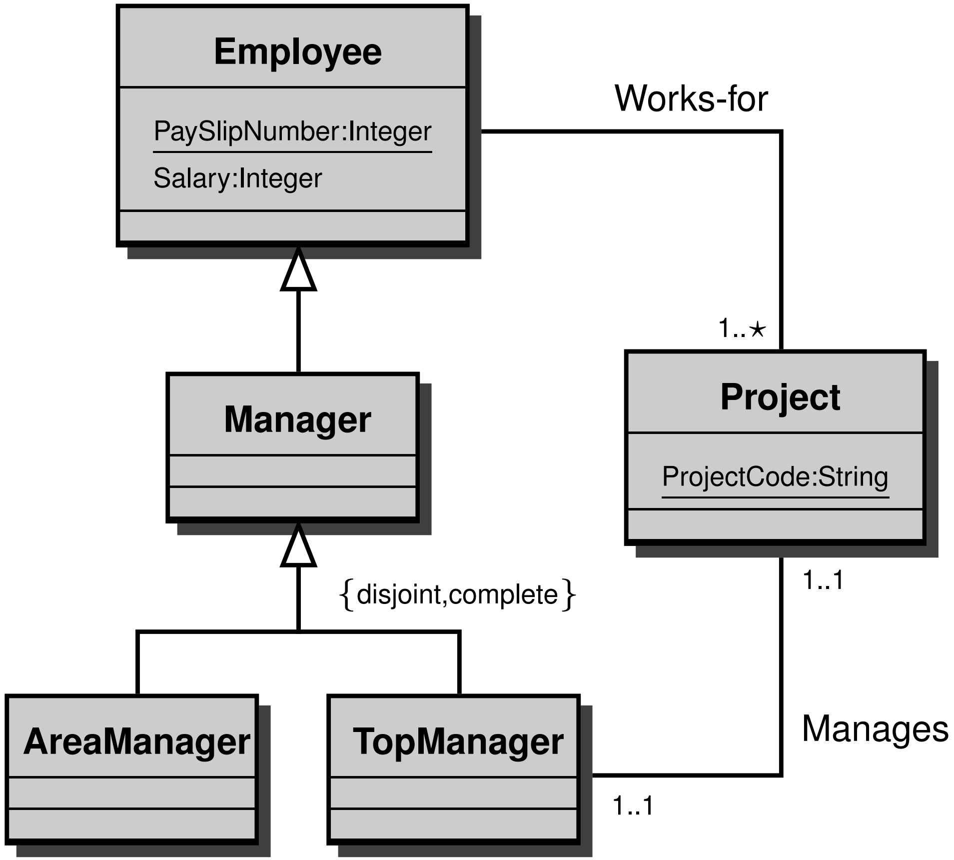
$$\mathtt{Q}(\mathtt{x},\mathtt{y}) \Leftarrow \mathtt{Employee}(\mathtt{x}) \wedge \mathtt{Works\text{-}for}(\mathtt{x},\mathtt{y}) \wedge \mathtt{Manages}(\mathtt{x},\mathtt{y})$$



$$\mathtt{Manager} \sqsubseteq \neg(\exists^{\geq 1}[\mathtt{emp}]\mathtt{Works\text{-}for})$$

Reasoning over queries

$$\mathtt{Q}(x, y) \Leftarrow \mathtt{Employee}(x) \wedge \mathtt{Works\text{-}for}(x, y) \wedge \mathtt{Manages}(x, y)$$



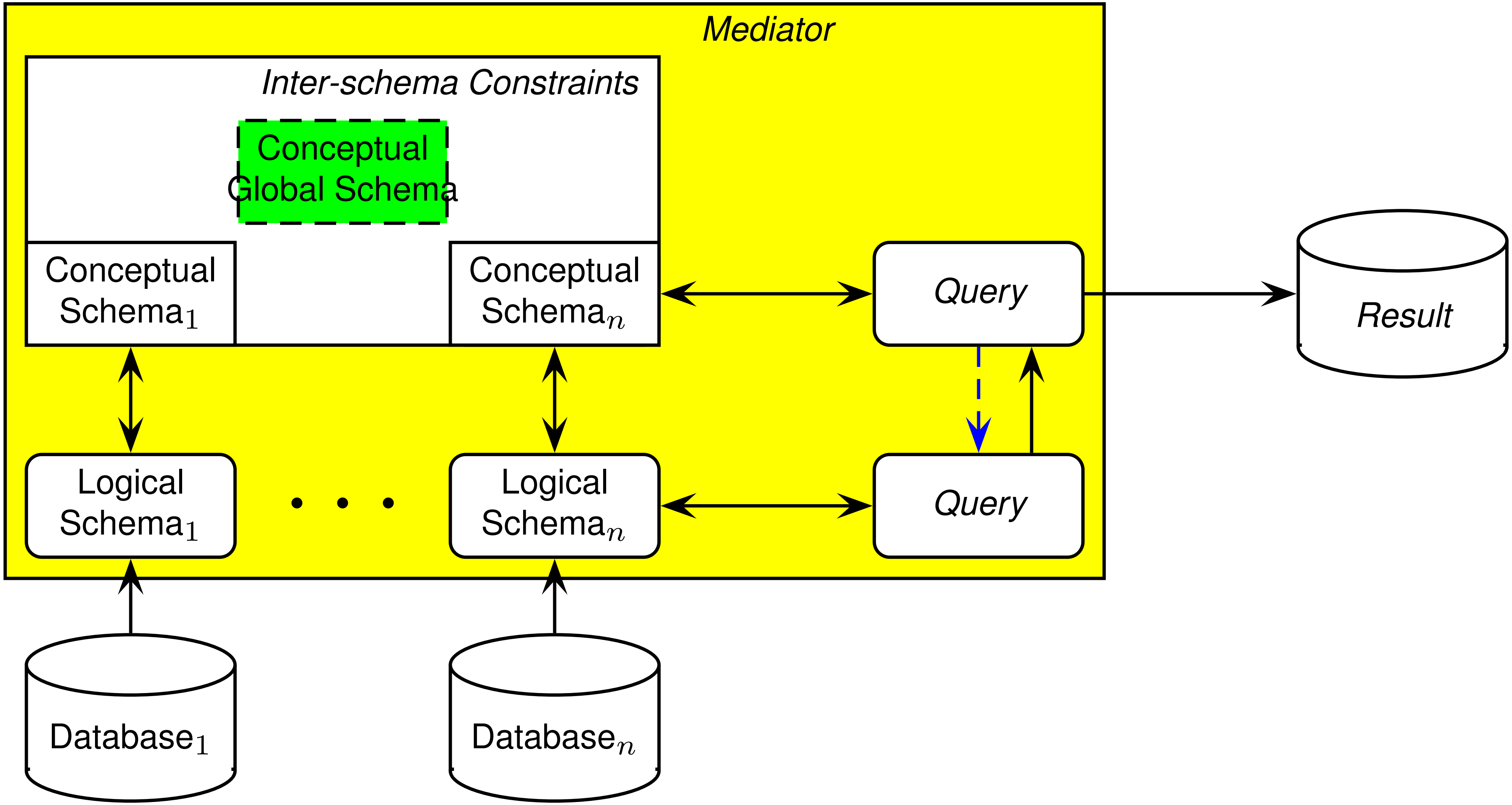
$$\mathtt{Manager} \sqsubseteq \neg(\exists^{\geq 1}[\mathtt{emp}]\mathtt{Works\text{-}for})$$

$\rightsquigarrow$ INCONSISTENT QUERY!

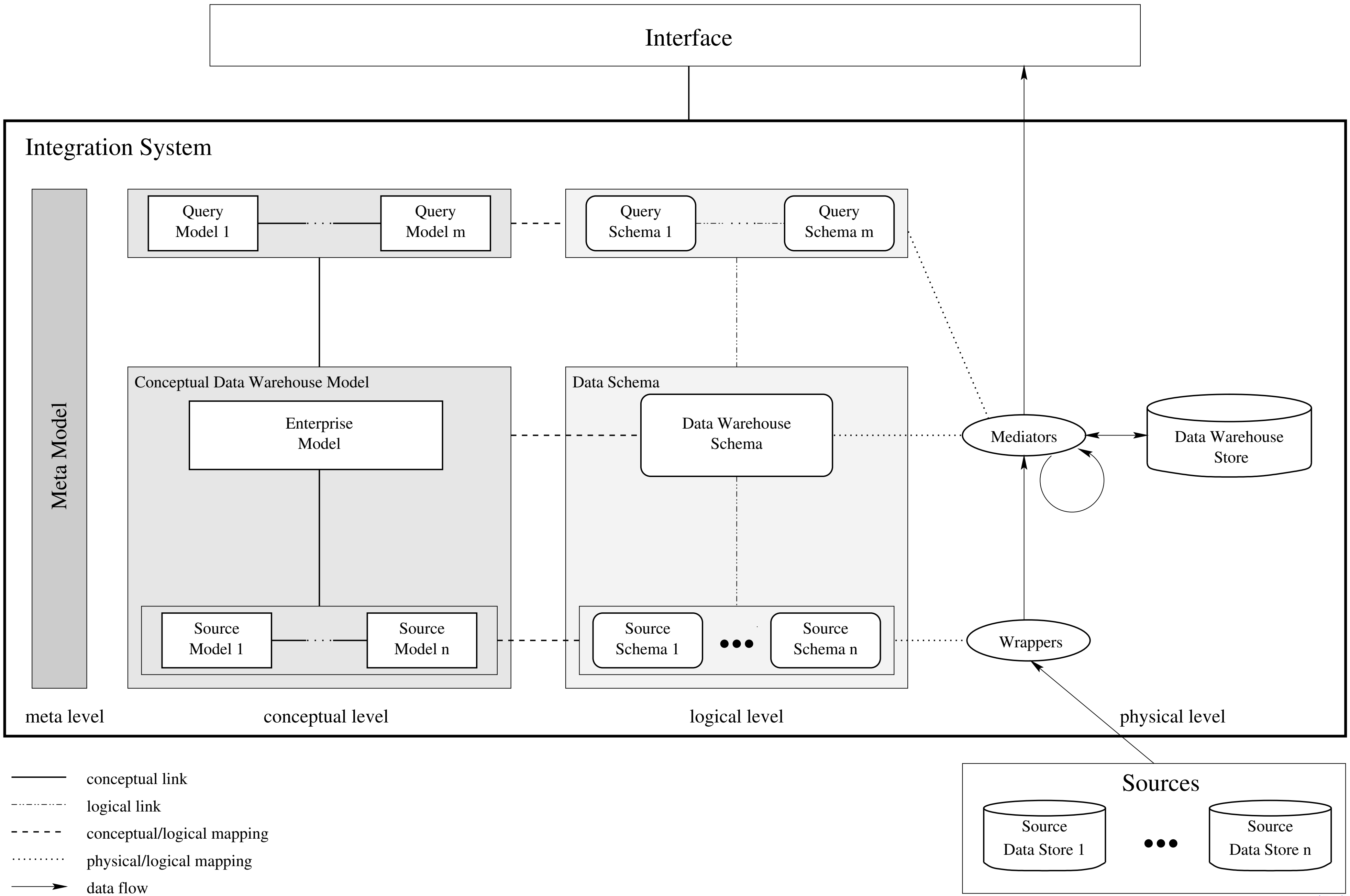
Summary

- Logic and Conceptual Modelling
- Description Logics for Conceptual Modelling
- Queries with an Ontology
- Ontology Integration

Mediator Architecture for Ontology Integration



DWQ Ontology Integration Architecture



Local-as-view vs. Global-as-view

Local-as-view (Information Manifold, DWQ, Picsel):

- High modularity and reusability (when a source changes, only its view definition is changed).
- Relationships betwen sources can be inferred.
- Computationally more difficult (query reformulation).

Global-as-view (Carnot, SIMS, TSIMMIS, Tambis, Observer, . . .):

- Whenever the source changes or a new one is added, the view needs to be reconsidered.
- Needs to understand the relationships between the sources.
- Query processing sometimes easy (unfolding), when the ontology is *very simple*. Otherwise it requires sophisticated query evaluation procedures.

Possible scenarios

- Empty ontology / very simple Ontology
  - Global-as-view
  - Local-as-view
- Full Ontology / Integrity Constraints
  - Global-as-view
  - Local-as-view

Possible scenarios

- Empty ontology / very simple Ontology
  - Global-as-view
    - The problem reduces to standard DB technology.
    - Can not express Ontology Integration needs.
    - Not modular.
  - Local-as-view
- Full Ontology / Integrity Constraints
  - Global-as-view
  - Local-as-view

Possible scenarios

- Empty ontology / very simple Ontology
  - Global-as-view
    - The problem reduces to standard DB technology.
    - Can not express Ontology Integration needs.
    - Not modular.
  - Local-as-view
    - “Standard” view-based query processing.
    - Can express only few Ontology Integration needs.
    - Modular.
- Full Ontology / Integrity Constraints
  - Global-as-view
  - Local-as-view

Possible scenarios

- Empty ontology / very simple Ontology
  - Global-as-view
    - The problem reduces to standard DB technology.
    - Can not express Ontology Integration needs.
    - Not modular.
  - Local-as-view
    - “Standard” view-based query processing.
    - Can express only few Ontology Integration needs.
    - Modular.
- Full Ontology / Integrity Constraints
  - Global-as-view
    - Requires sophisticated query evaluation procedures (involving deduction).
    - Can express Ontology Integration needs.
    - Not modular.
  - Local-as-view

Possible scenarios

- Empty ontology / very simple Ontology
  - Global-as-view
    - The problem reduces to standard DB technology.
    - Can not express Ontology Integration needs.
    - Not modular.
  - Local-as-view
    - “Standard” view-based query processing.
    - Can express only few Ontology Integration needs.
    - Modular.
- Full Ontology / Integrity Constraints
  - Global-as-view
    - Requires sophisticated query evaluation procedures (involving deduction).
    - Can express Ontology Integration needs.
    - Not modular.
  - Local-as-view
    - View-based query processing under constraints.
    - Can express Ontology Integration needs.
    - Modular.

Current (sad) Practice

- Most implemented Ontology Integration systems:

Current (sad) Practice

- Most implemented Ontology Integration systems:
  - either assume no Ontology or a very simple Ontology with a global-as-view approach,

Current (sad) Practice

- Most implemented Ontology Integration systems:
  - either assume no Ontology or a very simple Ontology with a global-as-view approach,
  - or include an Ontology or Integrity Constraints in their framework, but adopt a naive query evaluation procedure, based on query unfolding: no correctness of the query answering can be proved.

Conclusions

- All the things presented in this tutorial require heavy logical and technical machineries.
- Nonetheless, we believe that
  - it is feasible in practice,
  - it will lead to more usable information systems,
  - it is a lot of fun from the point of view of research.

i•com: Intelligent Conceptual Modelling tool

- i•com allows for the specification of multiple EER (or UML) diagrams and inter- and intra-schema constraints;
- Complete logical reasoning is employed by the tool using a hidden underlying $\mathcal{DLR}$ inference engine;
- i•com verifies the specification, infers implicit facts and stricter constraints, and manifests any inconsistencies during the conceptual modelling phase.
- `www.cs.man.ac.uk/~franconi/icom/`