

Introduction to Semantic Web Ontology Languages

Using Ontologies

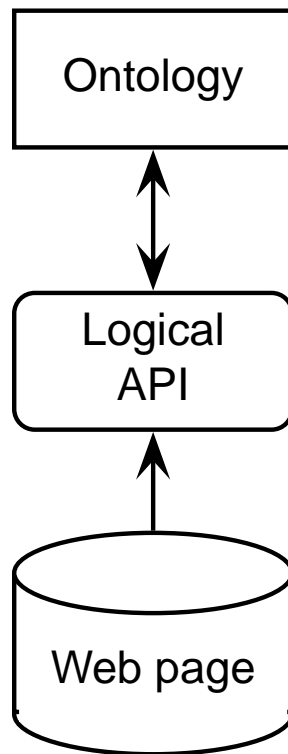
Enrico Franconi

franconi@inf.unibz.it

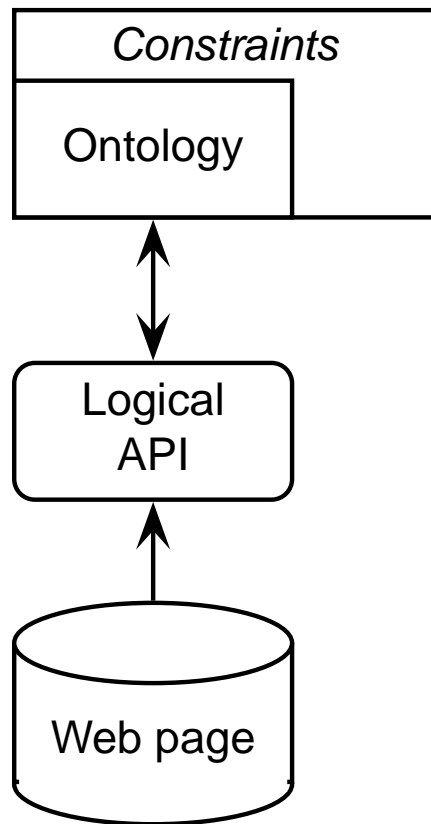
<http://www.inf.unibz.it/~franconi>

Faculty of Computer Science, Free University of Bozen-Bolzano, Italy

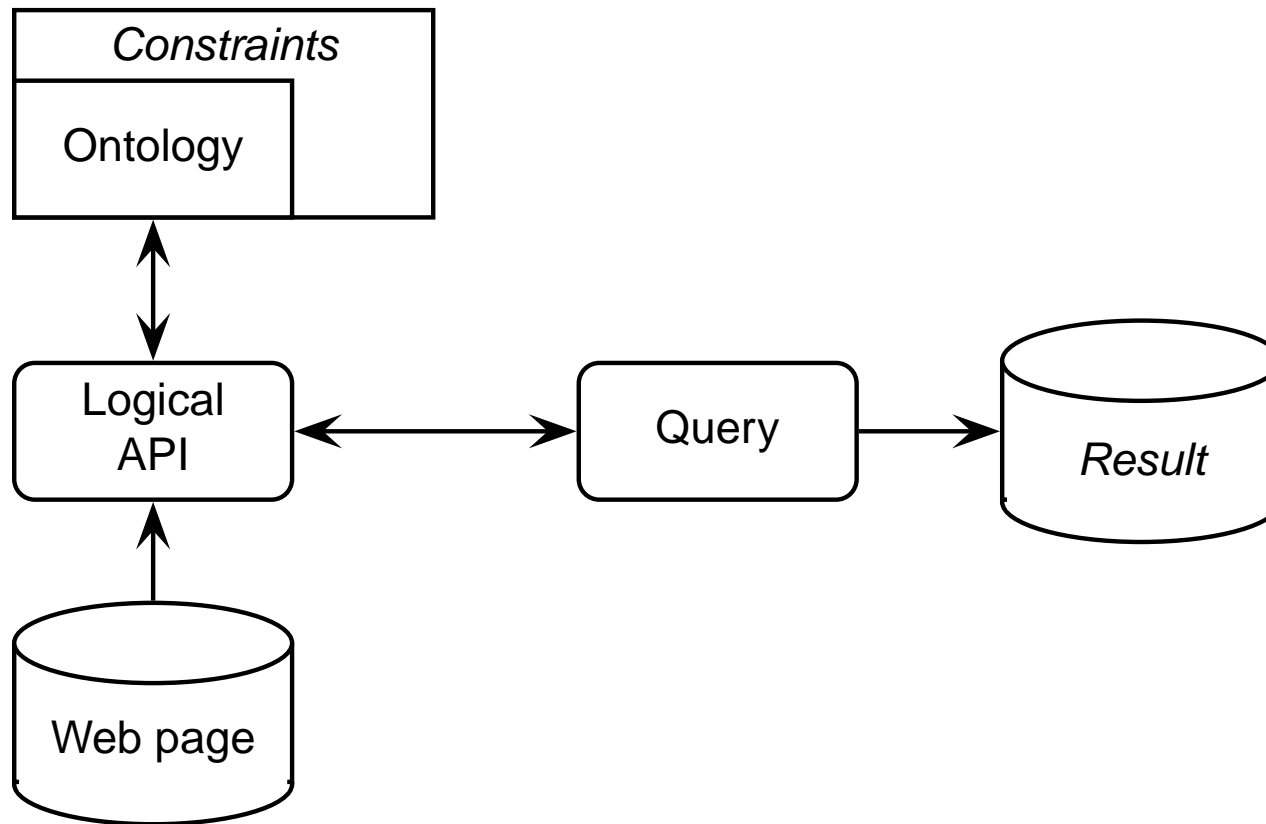
Ontologies linked with Information Sources



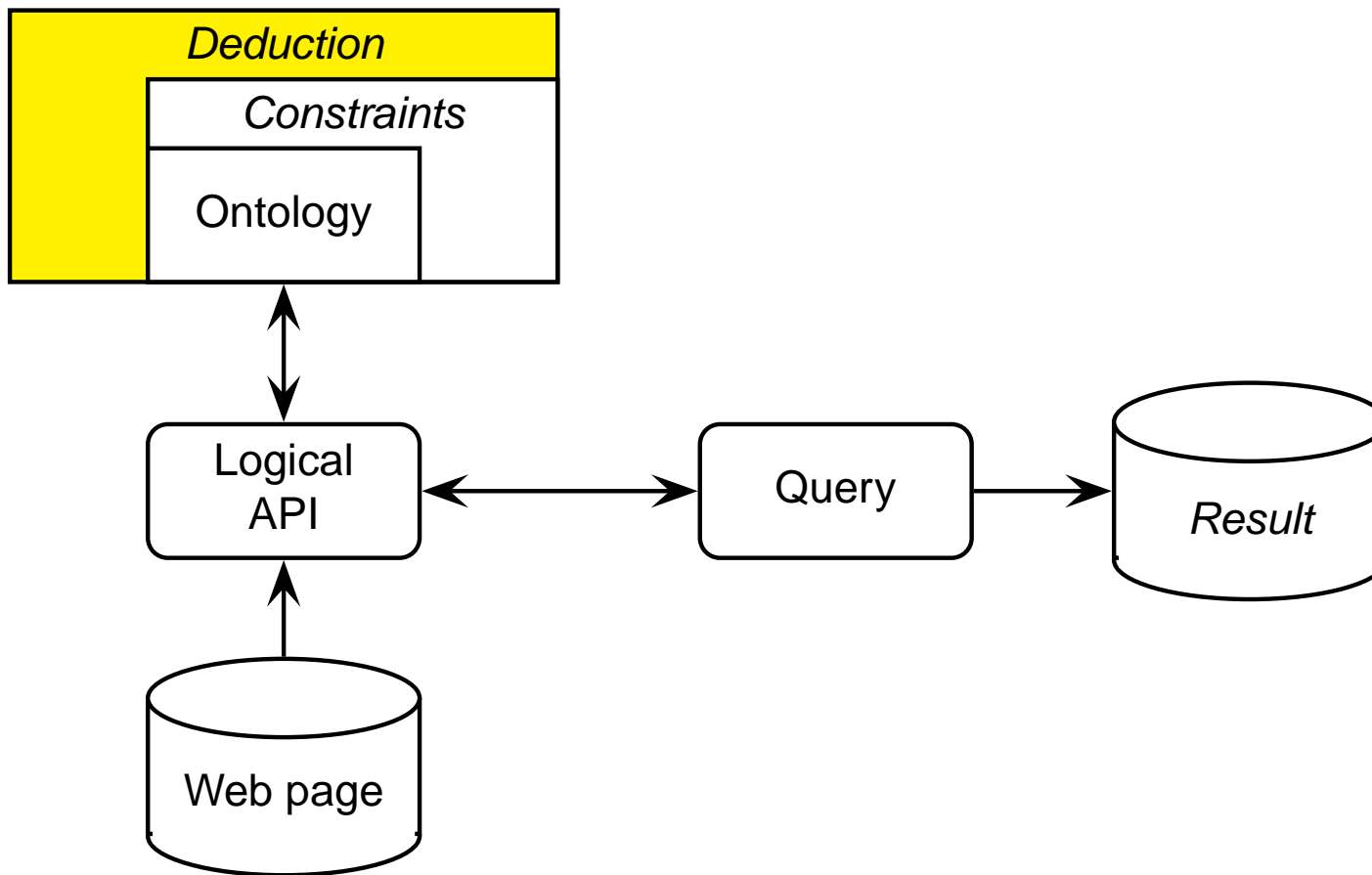
Ontologies linked with Information Sources



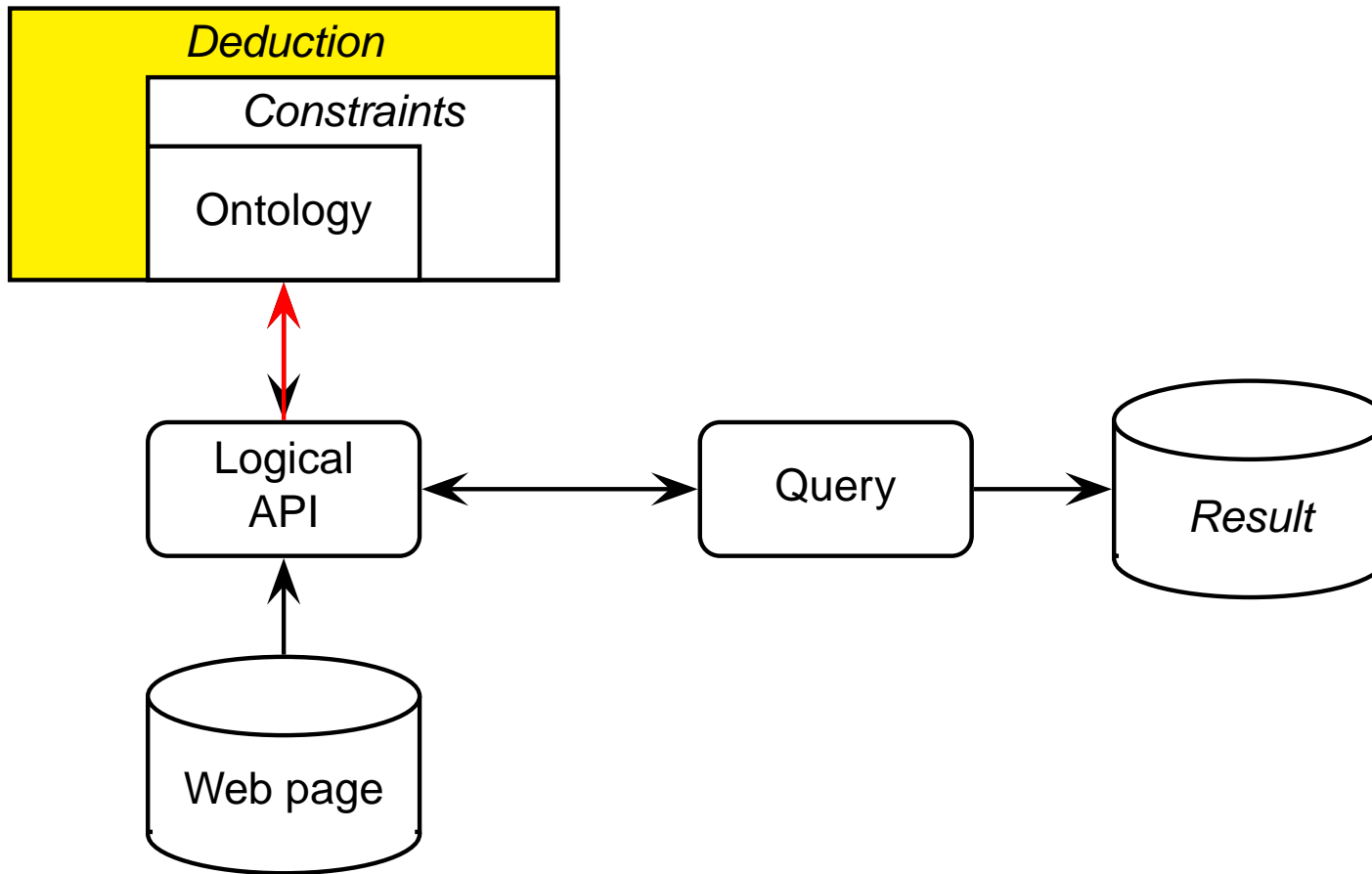
Ontologies linked with Information Sources



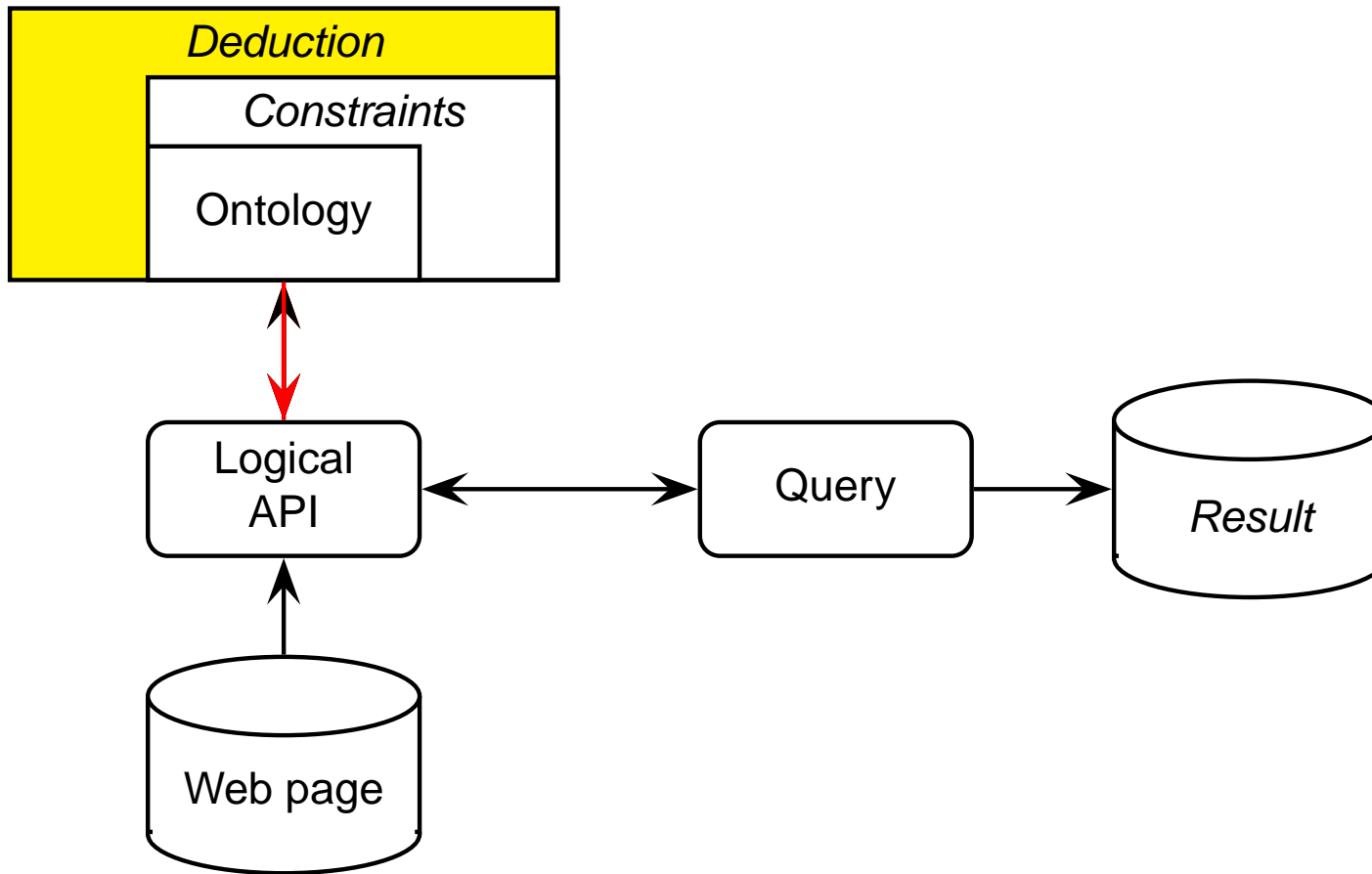
Ontologies linked with Information Sources



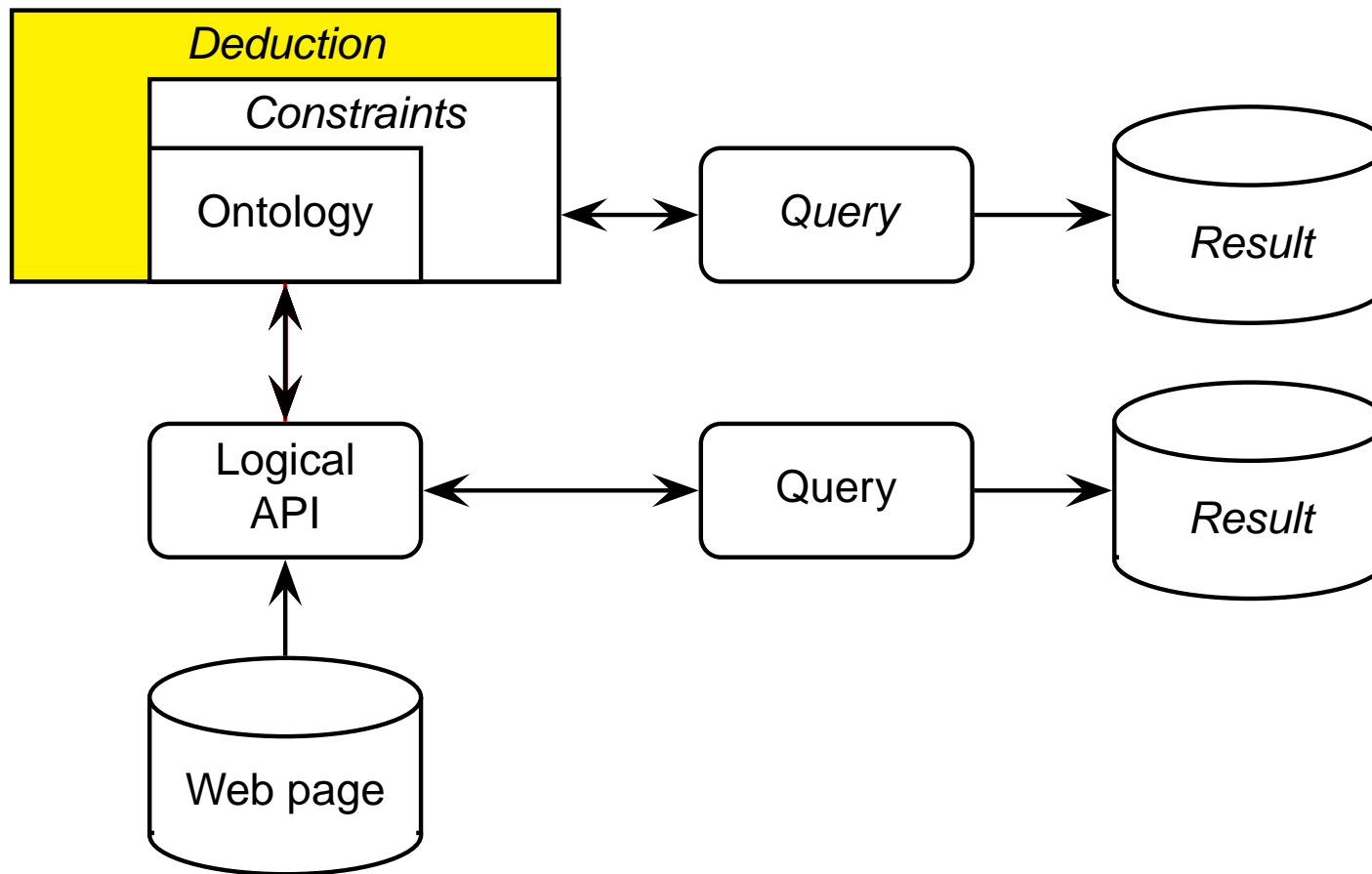
Ontologies linked with Information Sources



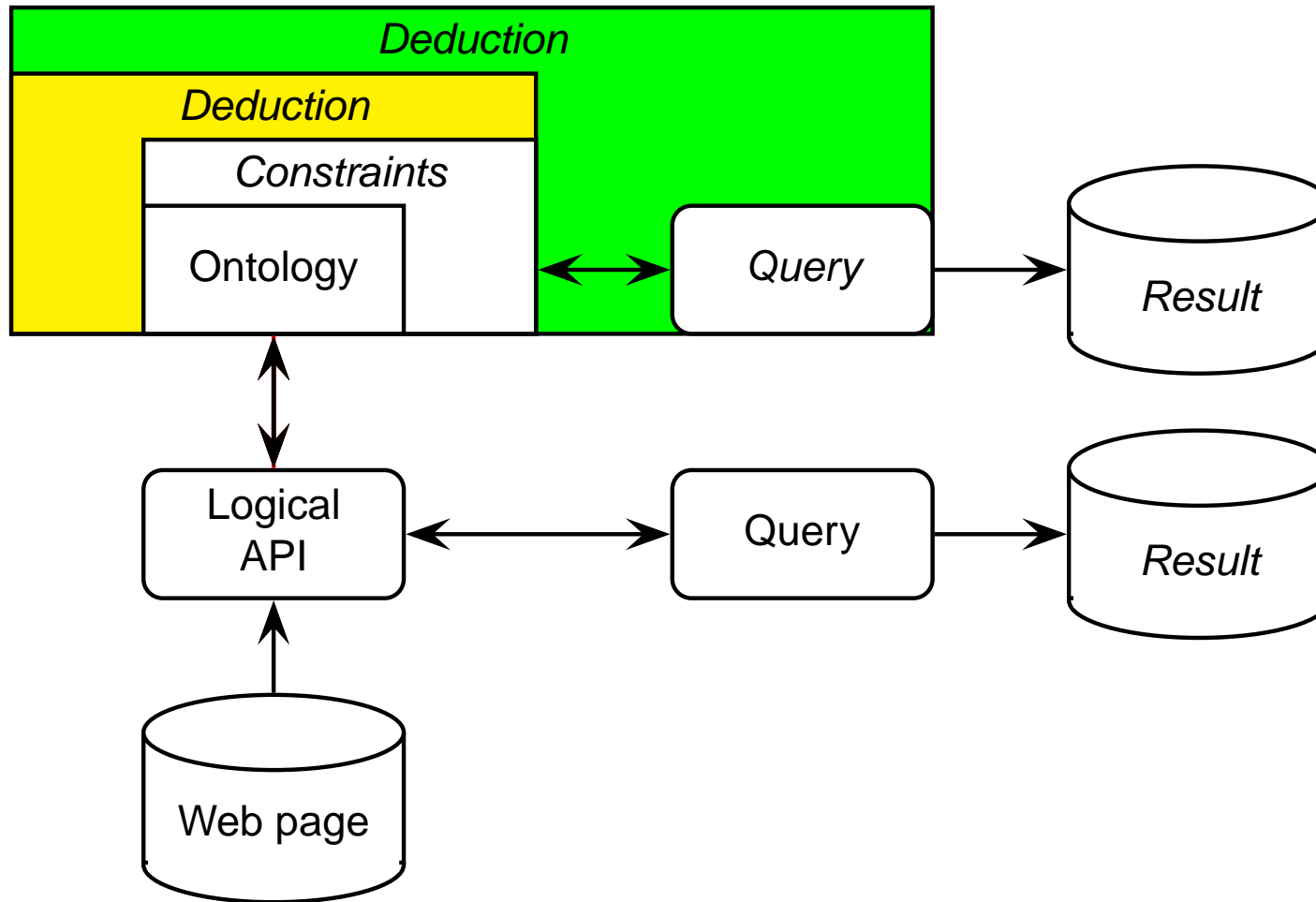
Ontologies linked with Information Sources



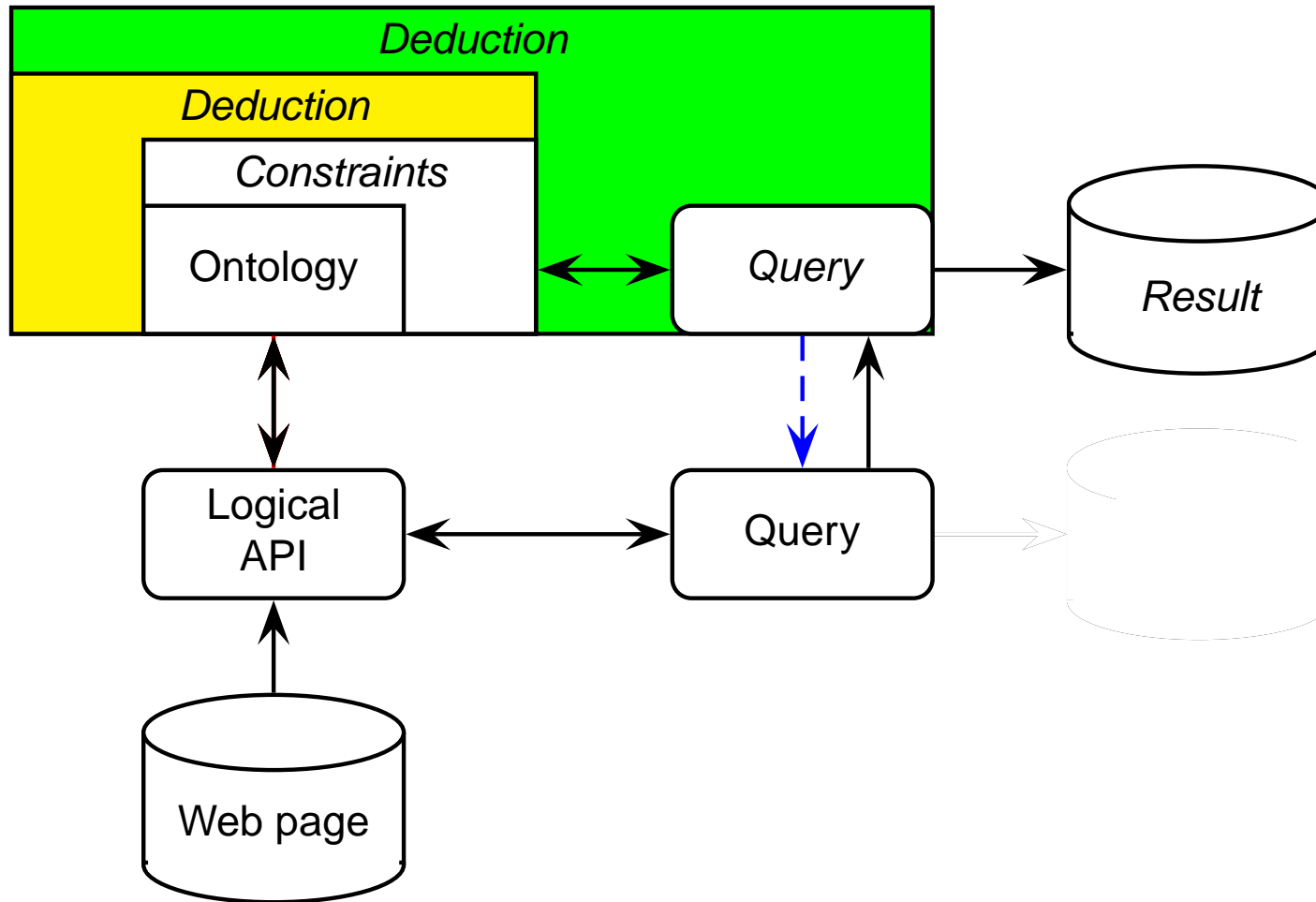
Ontologies linked with Information Sources



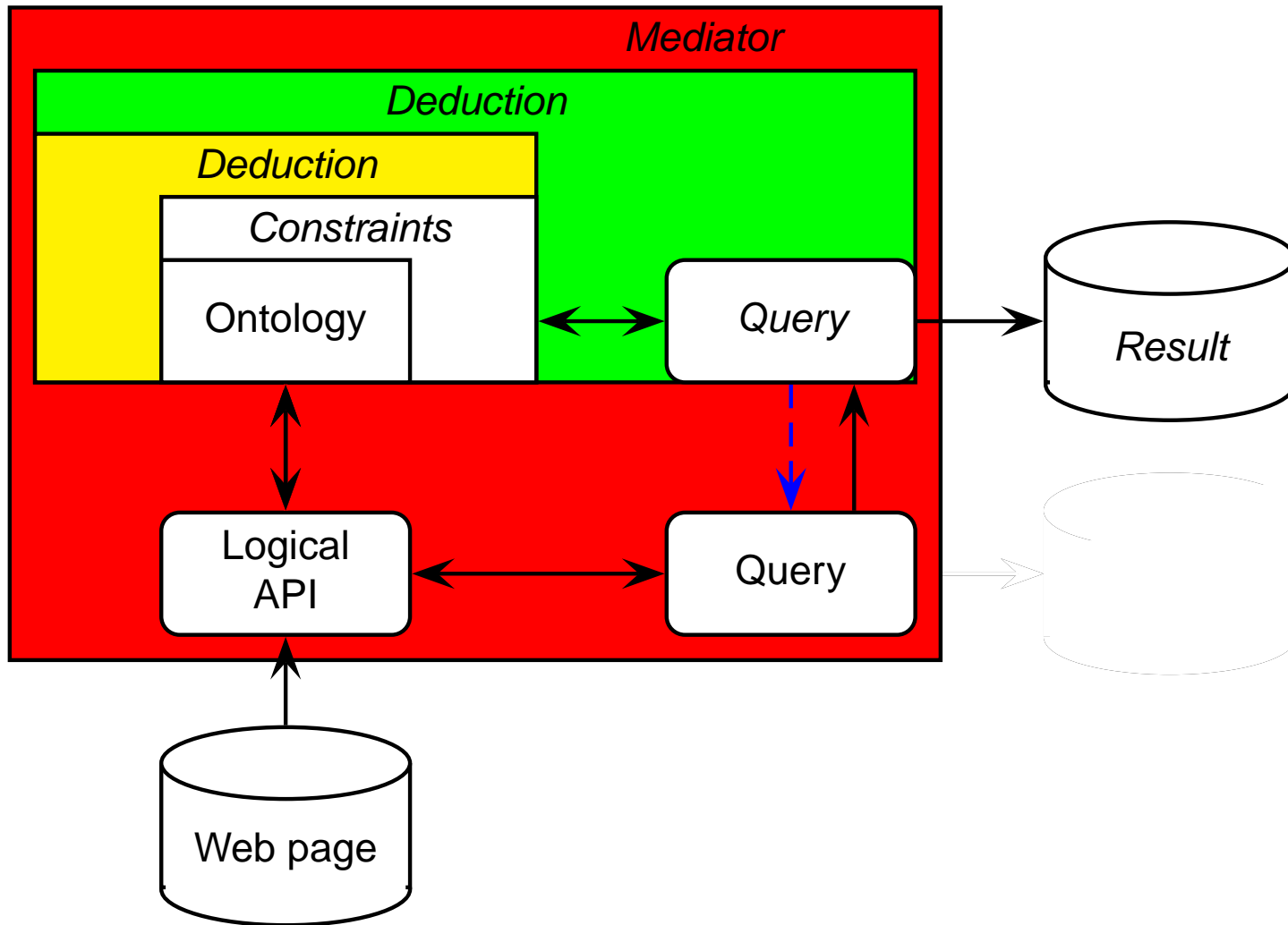
Ontologies linked with Information Sources



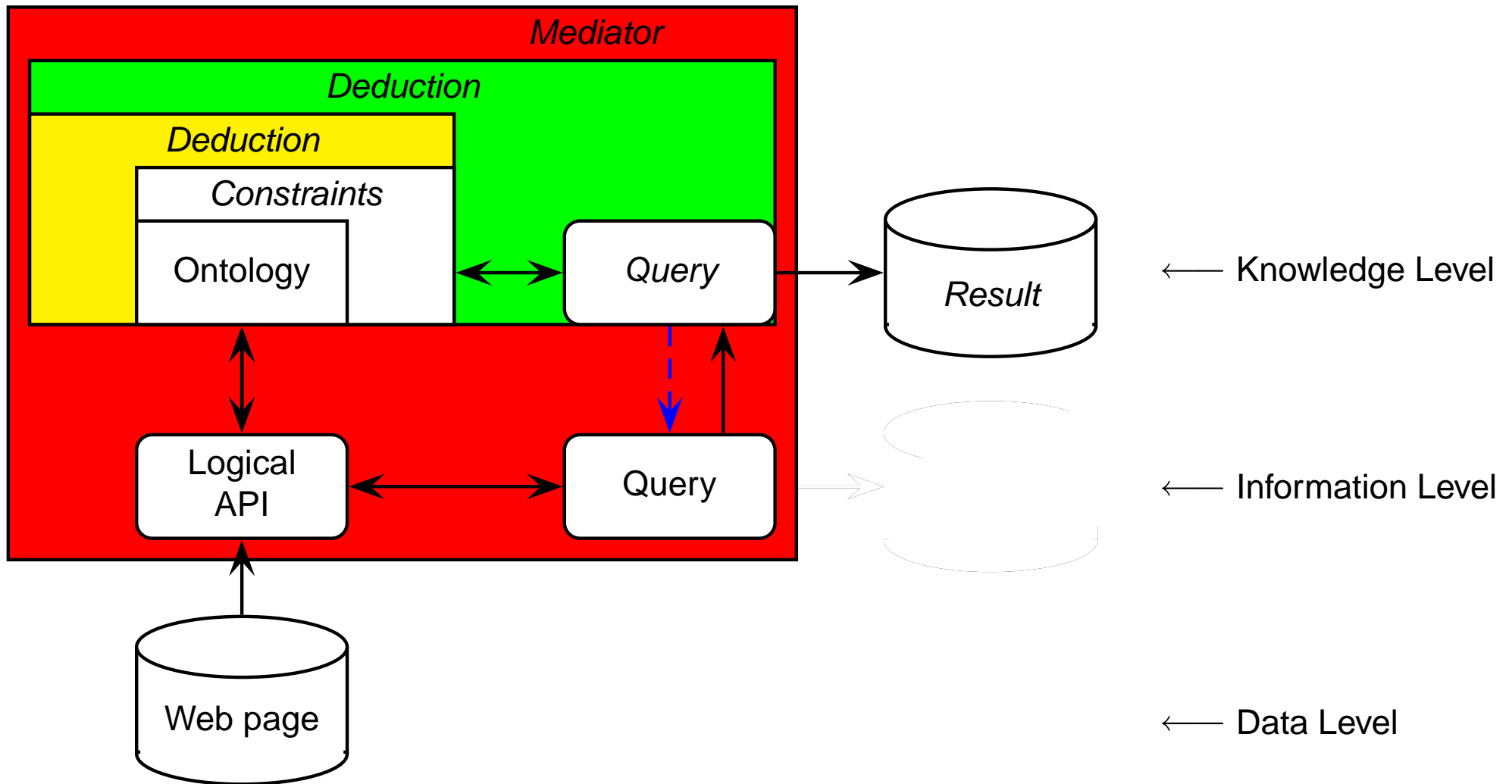
Ontologies linked with Information Sources



Ontologies linked with Information Sources



Ontologies linked with Information Sources



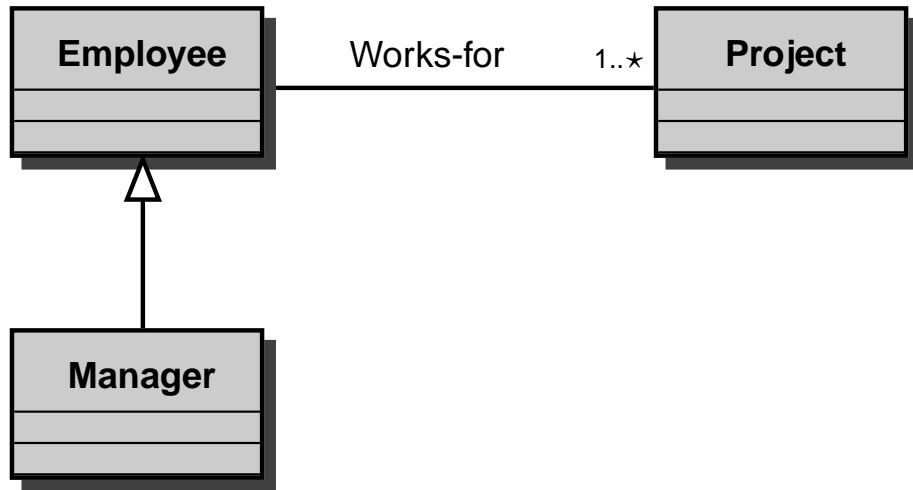
Queries with Ontologies: the DB assumption

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

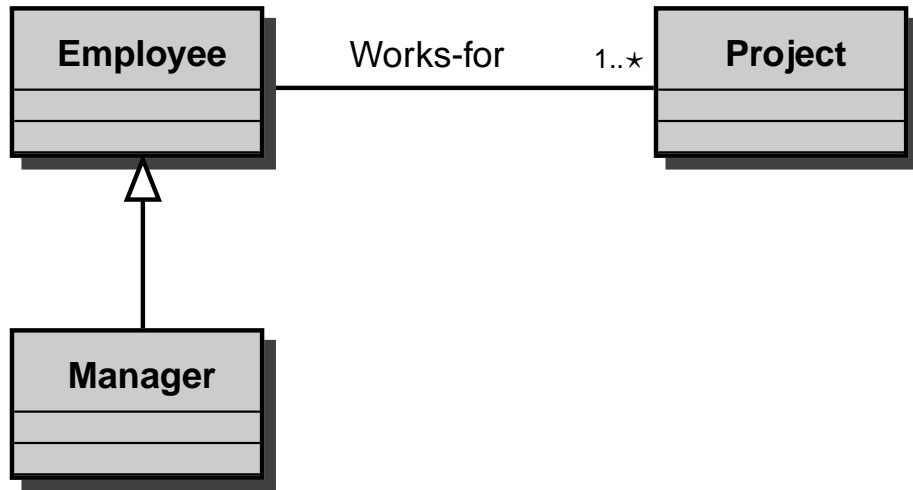
Queries with Ontologies: the DB assumption

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

Example with DB assumption



Example with DB assumption



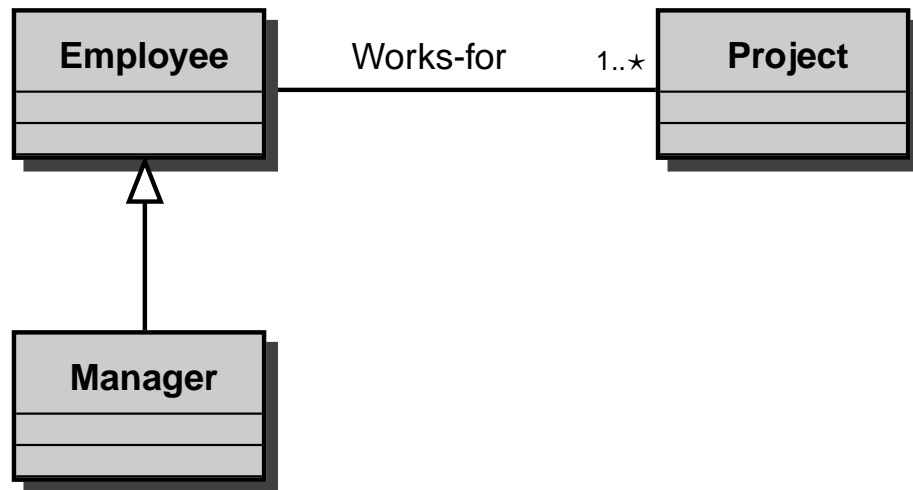
Employee = { **John**, **Mary**, **Paul** }

Manager = { **John**, **Paul** }

Works-for = { (**John**, **Prj-A**), (**Mary**, **Prj-B**) }

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

Example with DB assumption



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

Manager = { **John**, **Paul** }

Works-for = { (**John**, **Prj-A**), (**Mary**, **Prj-B**) }

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

$Q(X) \text{ :- Manager}(X), \text{ Works-for}(X, Y), \text{ Project}(Y)$

$\implies \{ \text{John} \}$

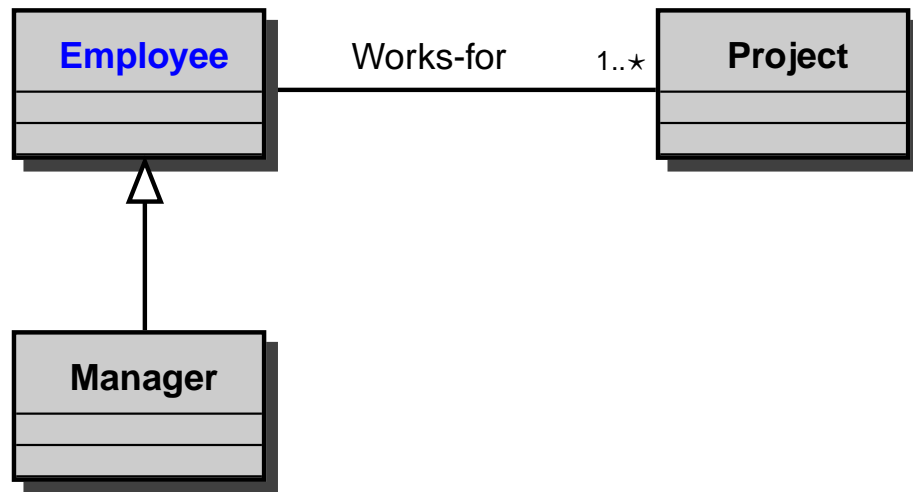
Weakening the DB assumption

- The DB assumption is against the principle that an ontology presents a richer vocabulary than the data stores.

Weakening the DB assumption

- The DB assumption is against the principle that an ontology presents a richer vocabulary than the data stores.
- Partial DB assumption: **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 with partial DB assumption

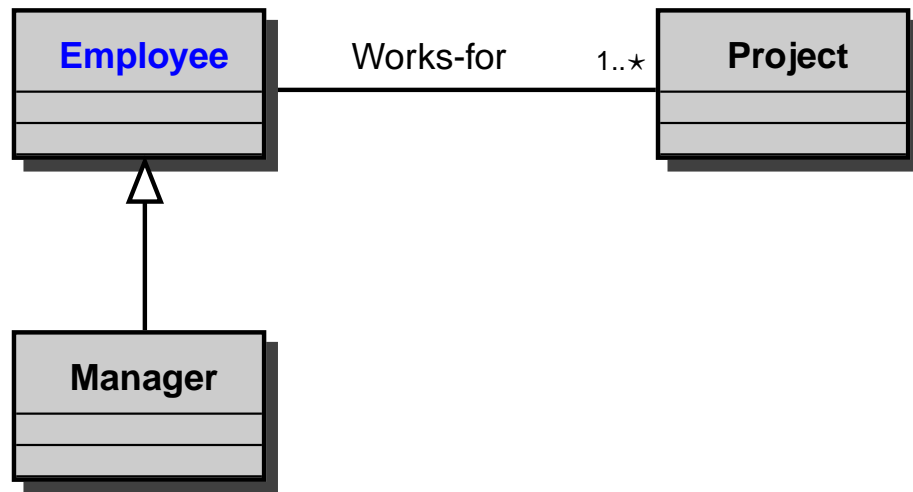


Manager = { **John**, **Paul** }

Works-for = { (**John**, **Prj-A**), (**Mary**, **Prj-B**) }

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

Example with partial DB assumption



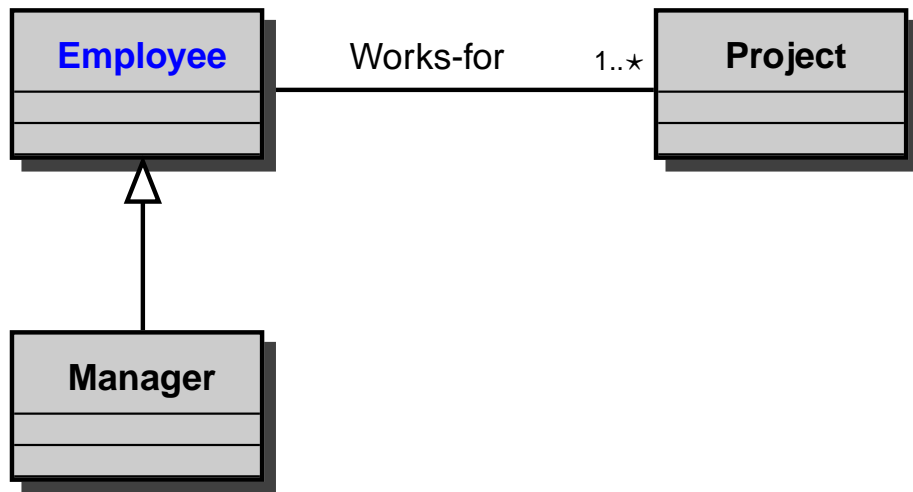
Manager = { **John, Paul** }

Works-for = { (**John, Prj-A**), (**Mary, Prj-B**) }

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

Q(X) :- Employee(X)

Example with partial DB assumption



Manager = { **John**, **Paul** }

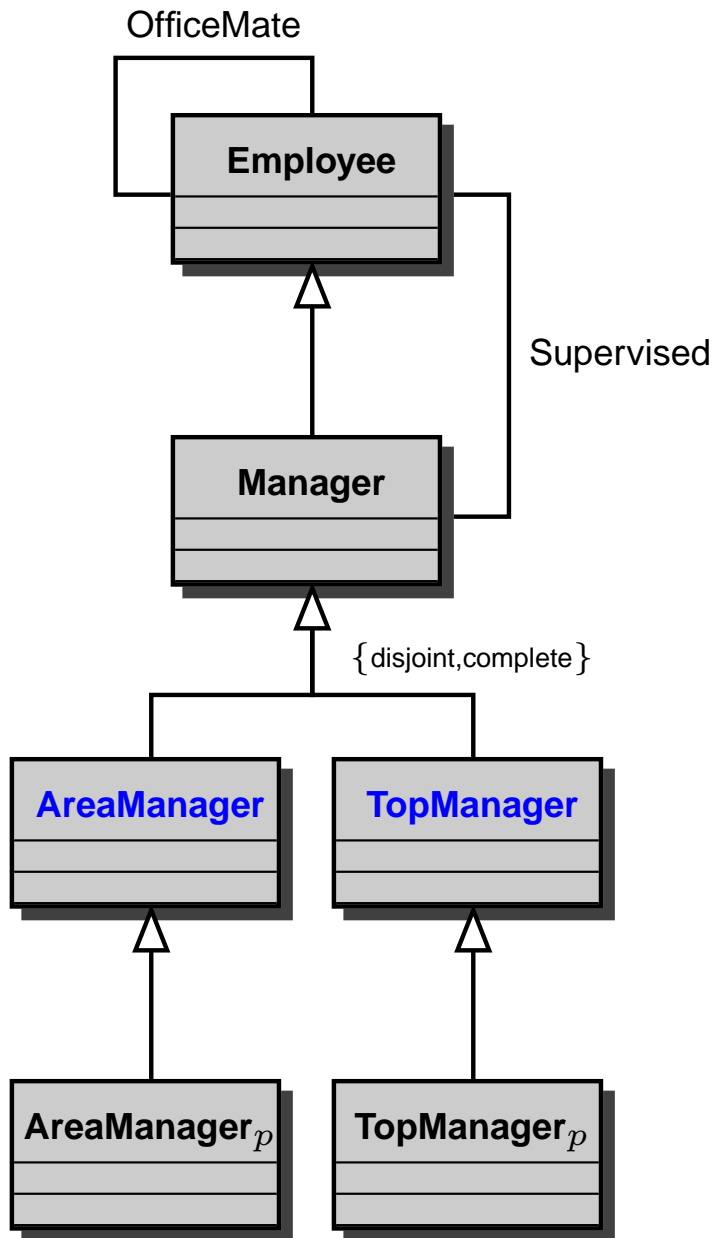
Works-for = { (**John**, **Prj-A**), (**Mary**, **Prj-B**) }

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

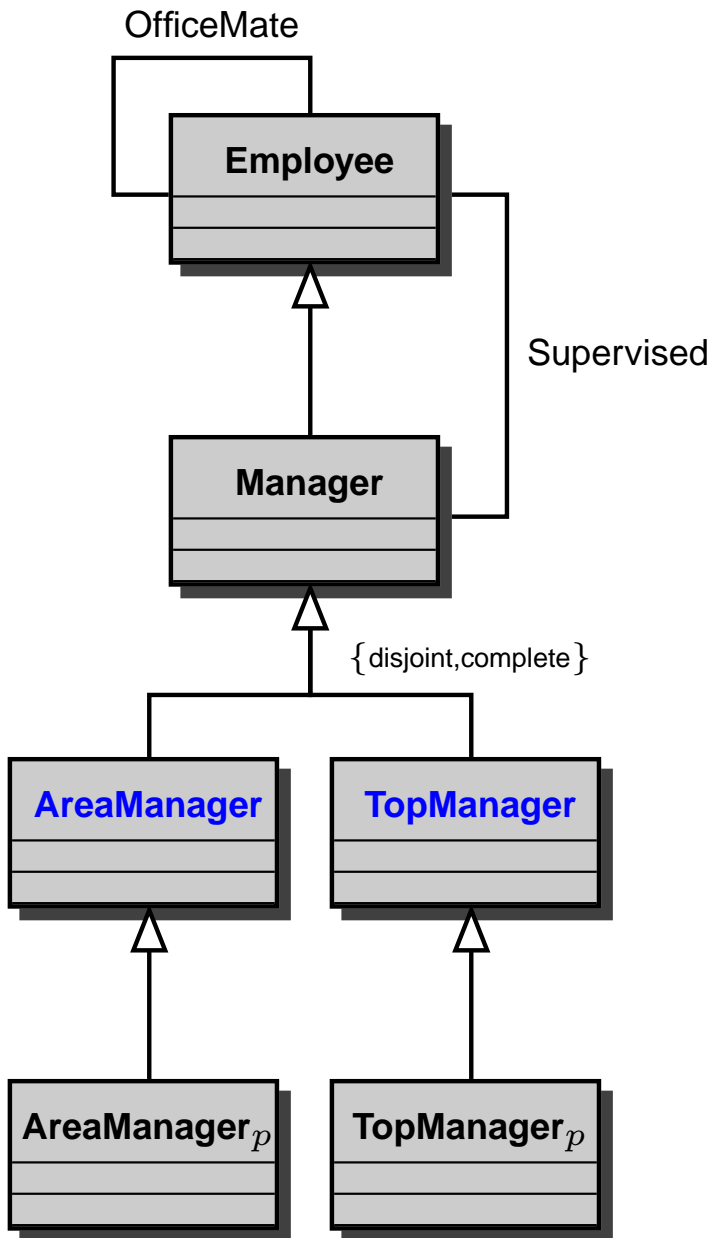
$Q(X) \text{ :- Employee}(X)$

\Rightarrow { **John**, **Paul**, **Mary** }

Andrea's Example



Andrea's Example



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

Manager = { **Andrea**, **Paul**, **Mary** }

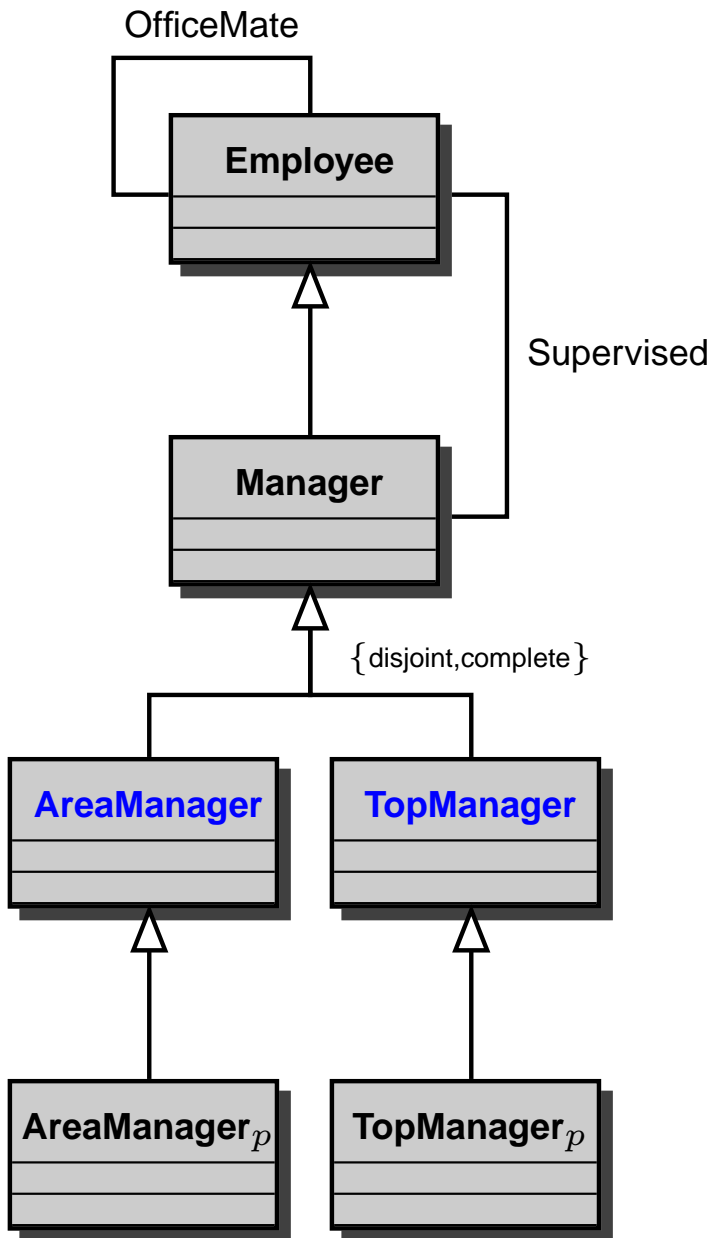
AreaManager_p = { **Paul** }

TopManager_p = { **Mary** }

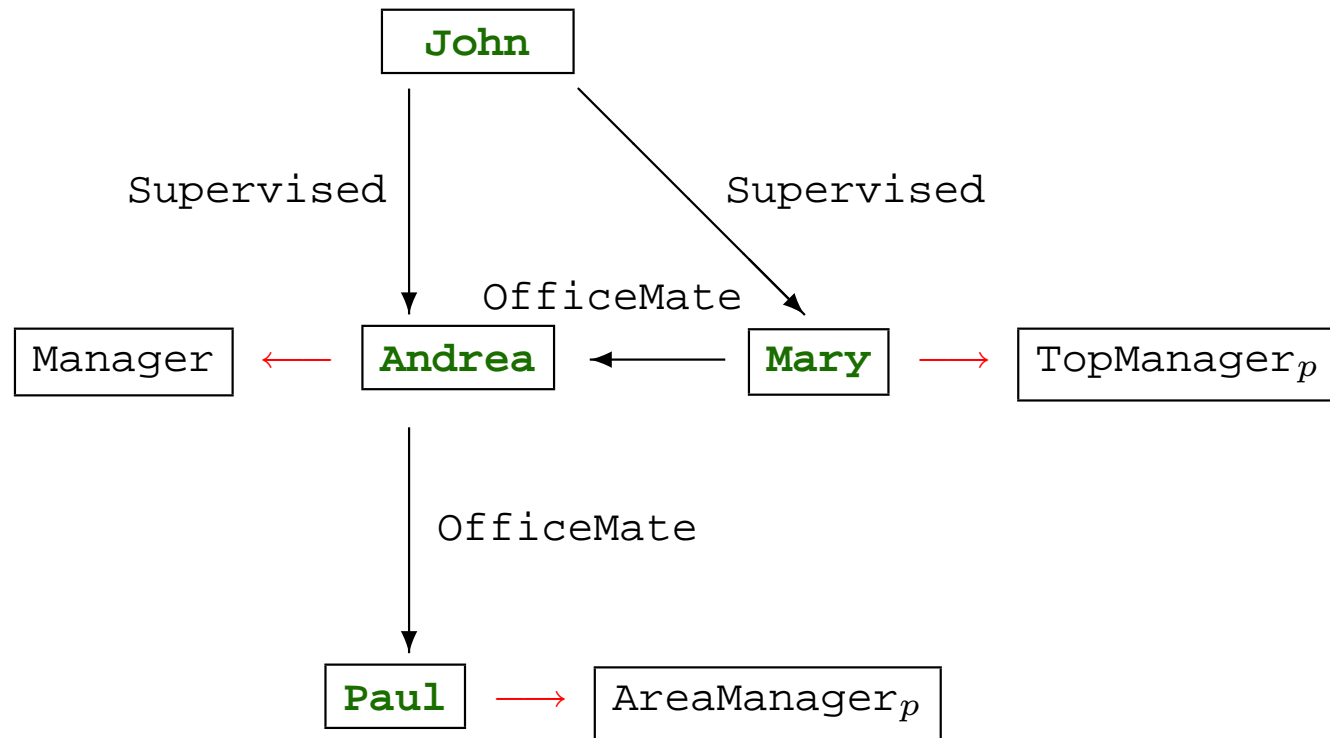
Supervised = { (**John**, **Andrea**), (**John**, **Mary**) }

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

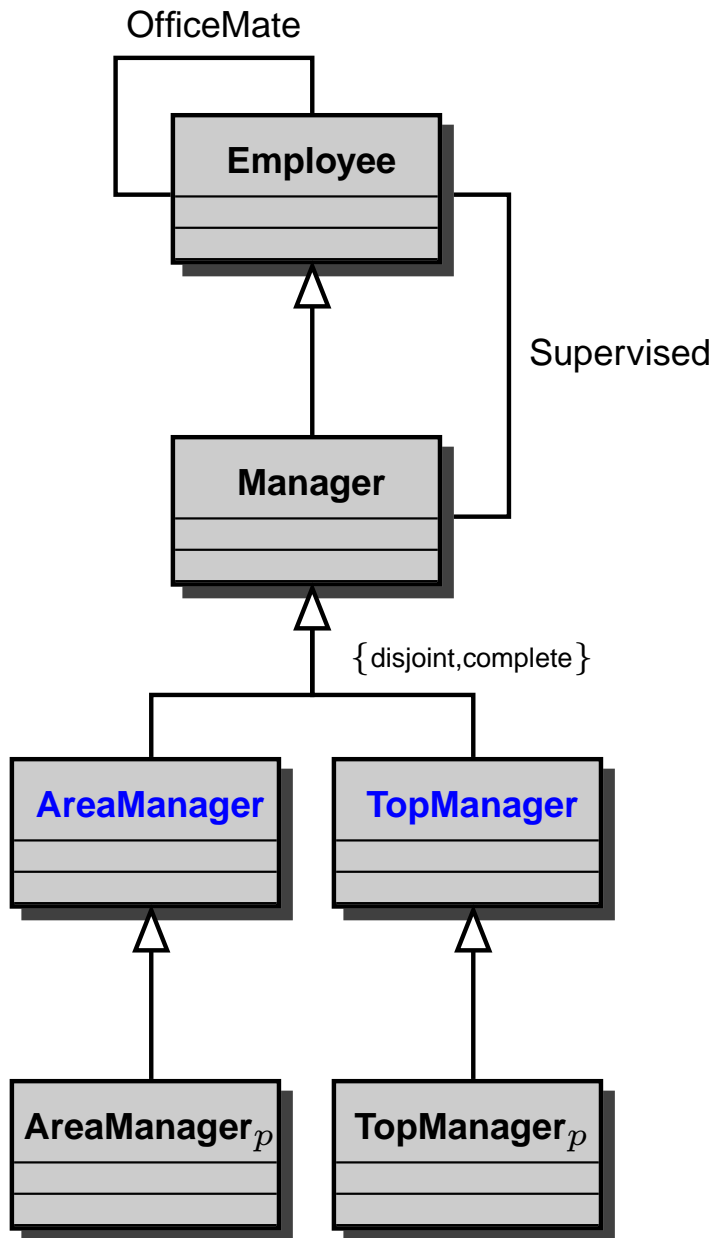
Andrea's Example



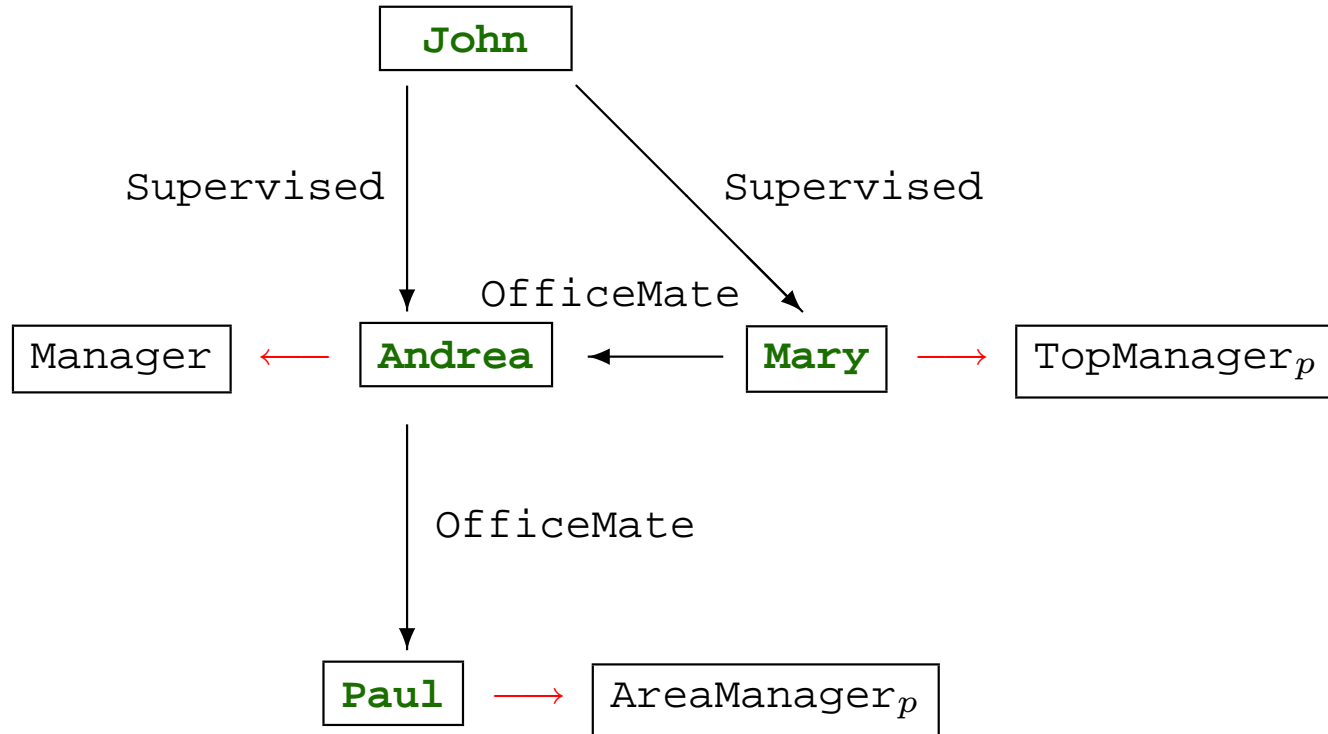
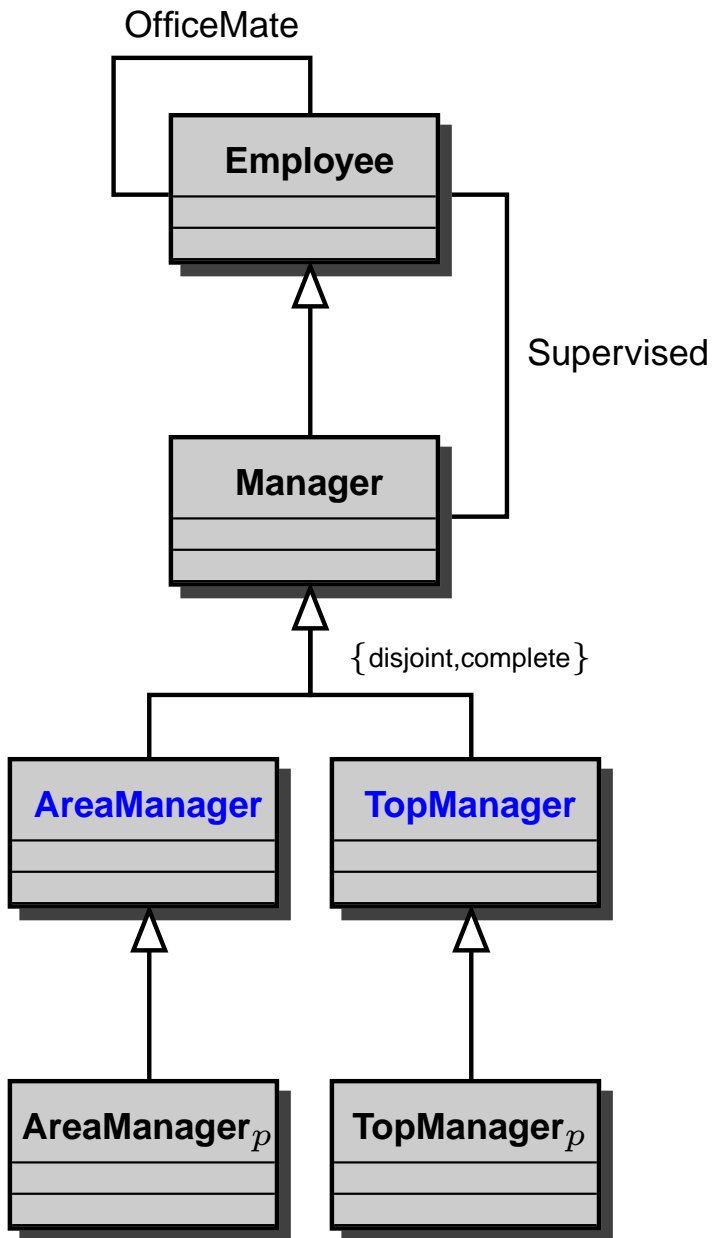
Employee = { **Andrea**, **Paul**, **Mary**, **John** }
 Manager = { **Andrea**, **Paul**, **Mary** }
 AreaManager_p = { **Paul** }
 TopManager_p = { **Mary** }
 Supervised = { (**John**, **Andrea**), (**John**, **Mary**) }
 OfficeMate = { (**Mary**, **Andrea**), (**Andrea**, **Paul**) }



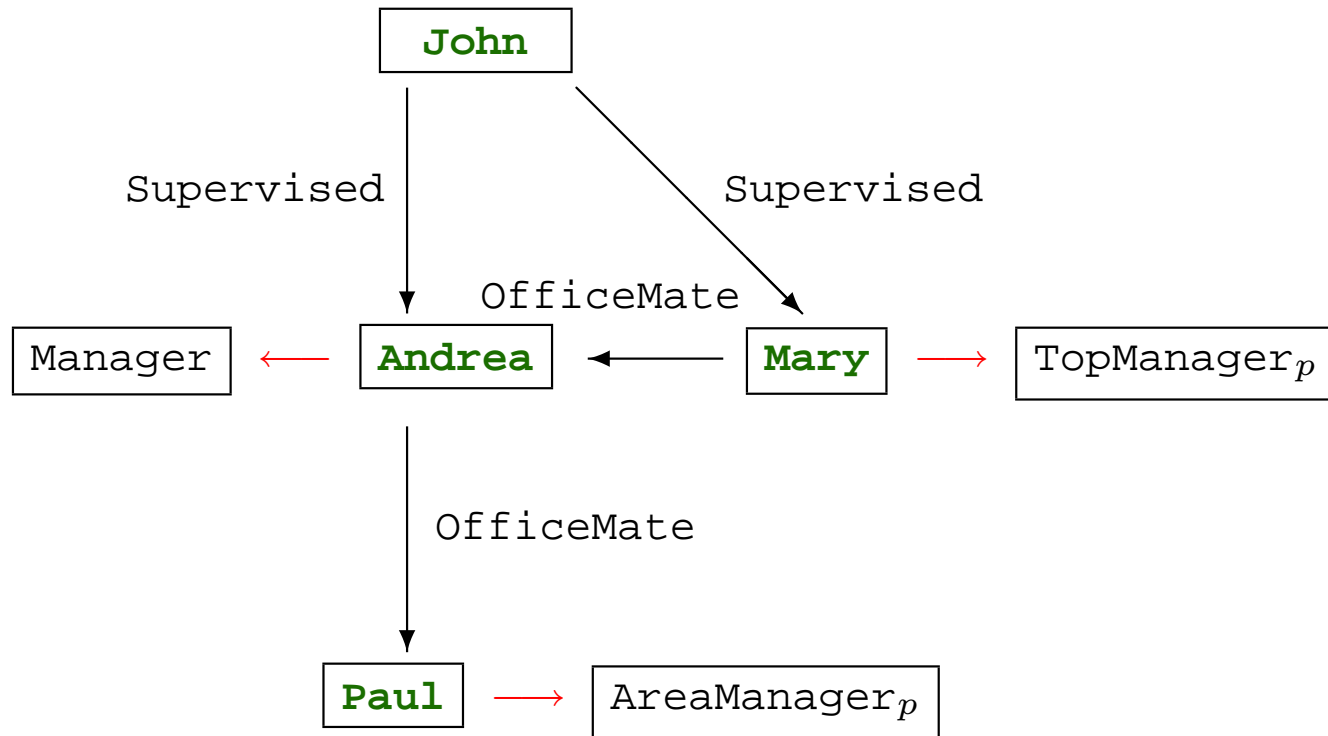
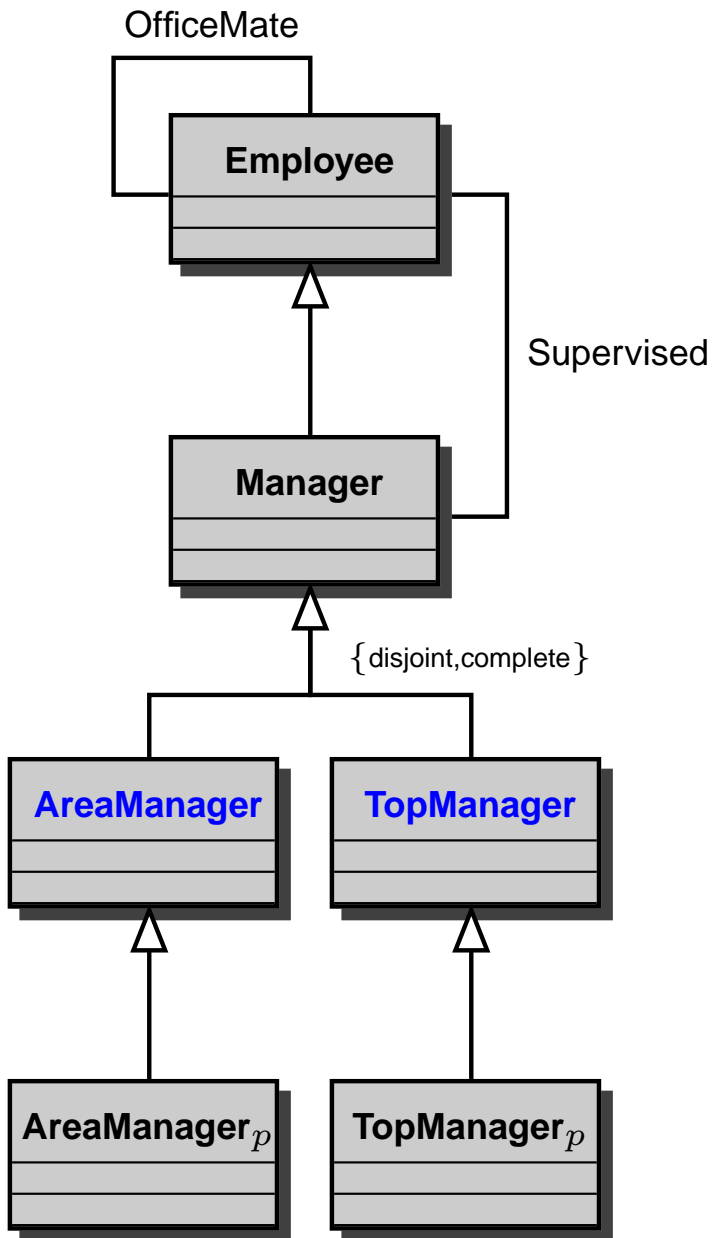
Andrea's Example (cont.)



Andrea's Example (cont.)

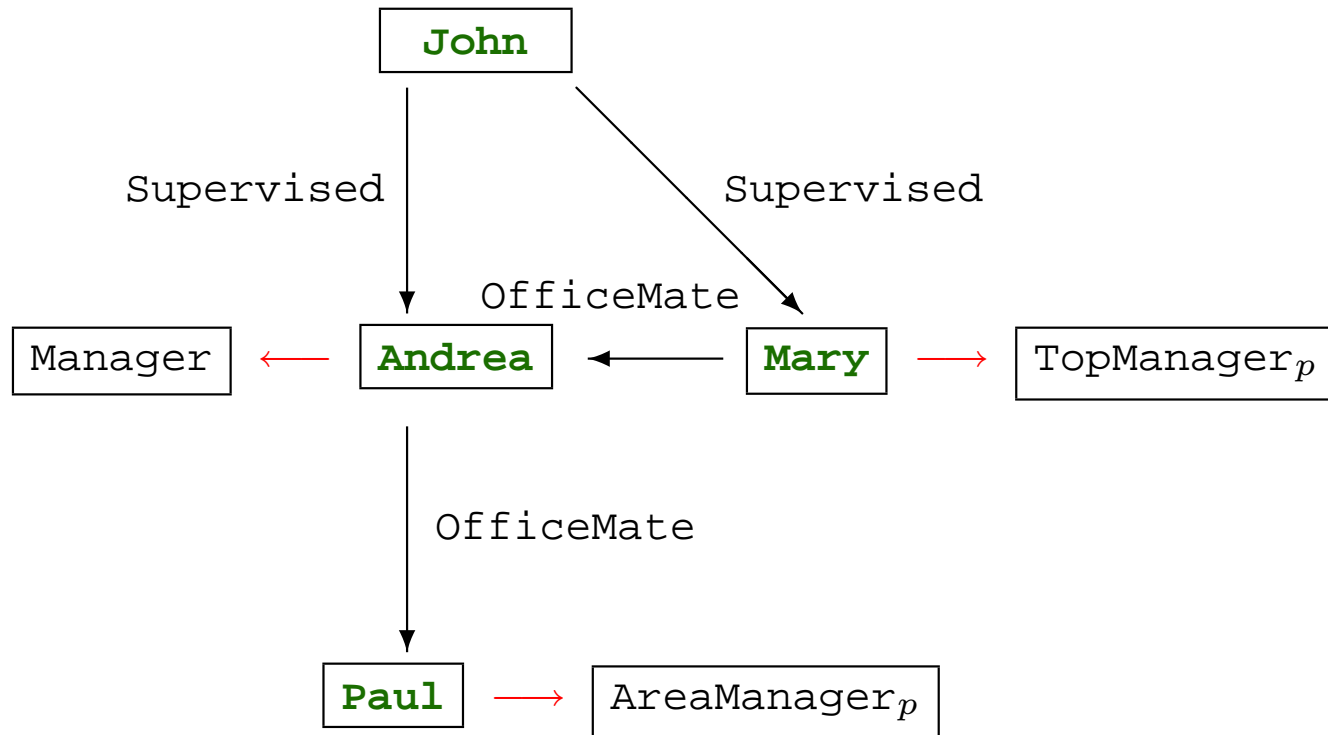
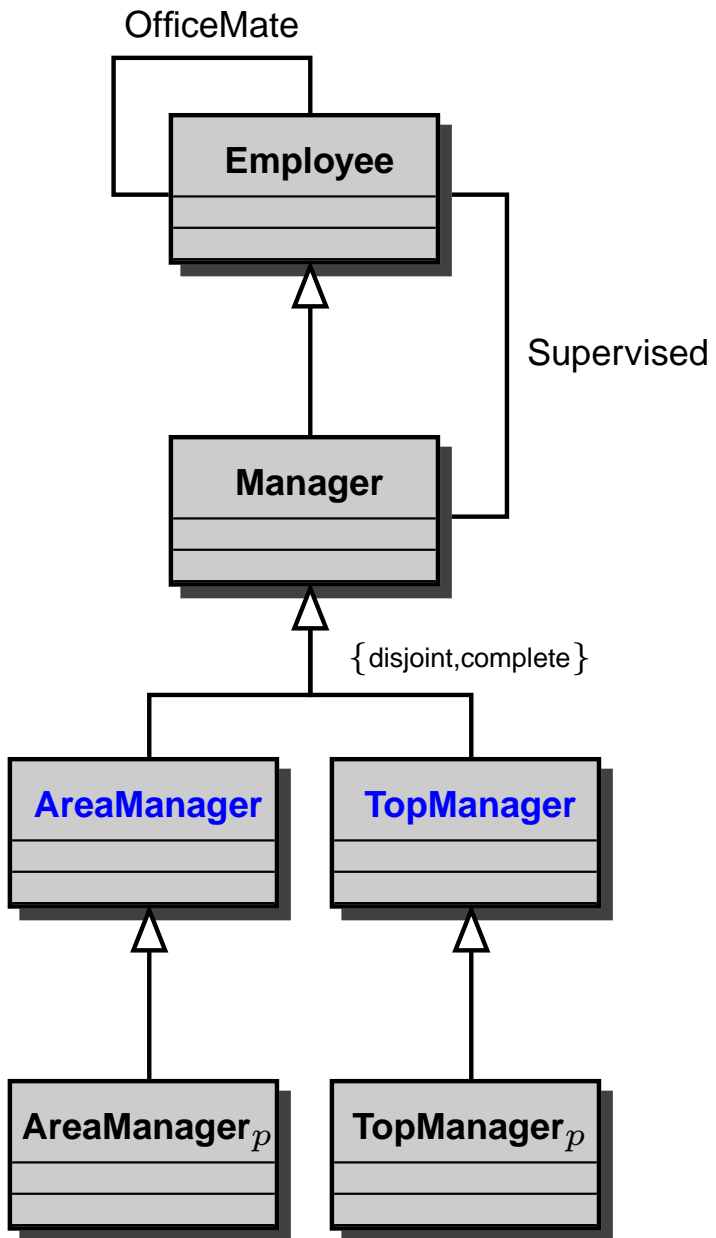


Andrea's Example (cont.)



$Q(X) :- Supervised(X, Y), TopManager(Y),$
 $Officemate(Y, Z), AreaManager(Z)$

Andrea's Example (cont.)



$Q(X) :- Supervised(X, Y), TopManager(Y),$
 $Officemate(Y, Z), AreaManager(Z)$

$\Rightarrow \{ John \}$

The general case: View based Query Processing

In general, the **mapping** between the ontology and the information source terms can be given in terms of a set of **sound** or **exact views**:

The general case: View based Query Processing

In general, the **mapping** between the ontology and the information source terms can be given in terms of a set of **sound** or **exact views**:

- **GAV** (global-as-view): a view over the information source is given for some term in the ontology

The general case: View based Query Processing

In general, the **mapping** between the ontology and the information source terms can be given in terms of a set of **sound** or **exact views**:

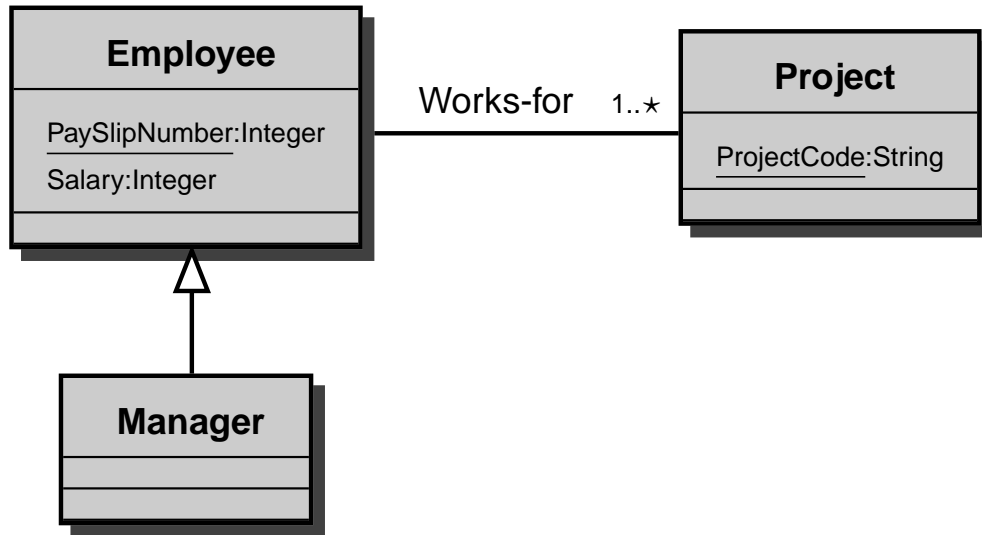
- **GAV** (global-as-view): a view over the information source is given for some term in the ontology
 - both the DB and the partial DB assumptions are special cases of GAV
 - an ER schema can be easily mapped to its corresponding relational schema in normal form via a GAV mapping

The general case: View based Query Processing

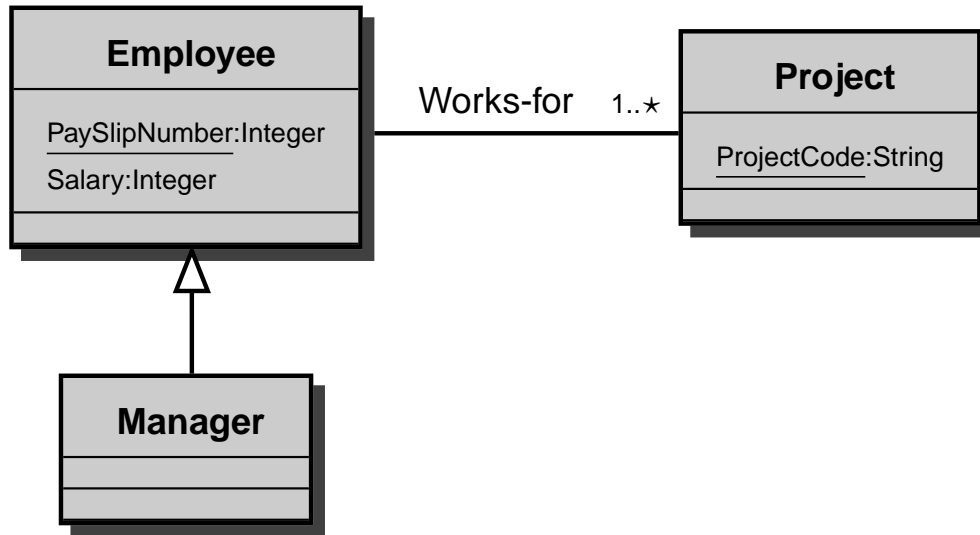
In general, the **mapping** between the ontology and the information source terms can be given in terms of a set of **sound** or **exact views**:

- **GAV** (global-as-view): a view over the information source is given for some term in the ontology
 - both the DB and the partial DB assumptions are special cases of GAV
 - an ER schema can be easily mapped to its corresponding relational schema in normal form via a GAV mapping
- **LAV** (local-as-view): a view over the ontology terms is given for each term in the information source;
- **GLAV**: mixed from the above.

Sound GAV mapping



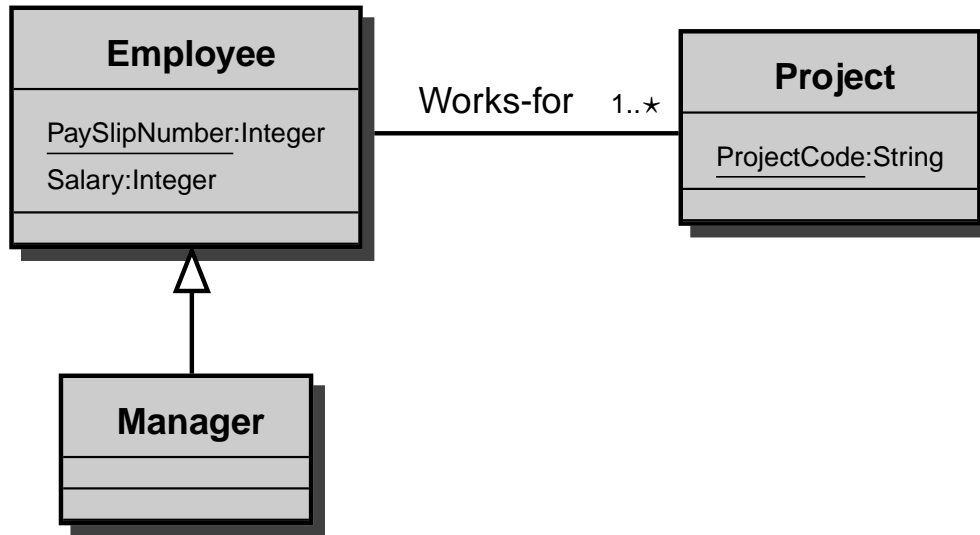
Sound GAV mapping



1-Employee(PaySlipNumber, Salary, ManagerP)

2-Works-for(PaySlipNumber, ProjectCode)

Sound GAV mapping



1-`Employee`(PaySlipNumber, Salary, ManagerP)

2-`Works-for`(PaySlipNumber, ProjectCode)

`Employee(X) :- 1-Employee(X,Y,Z)`

`Employee(X) :- 2-Works-for(X,Y)`

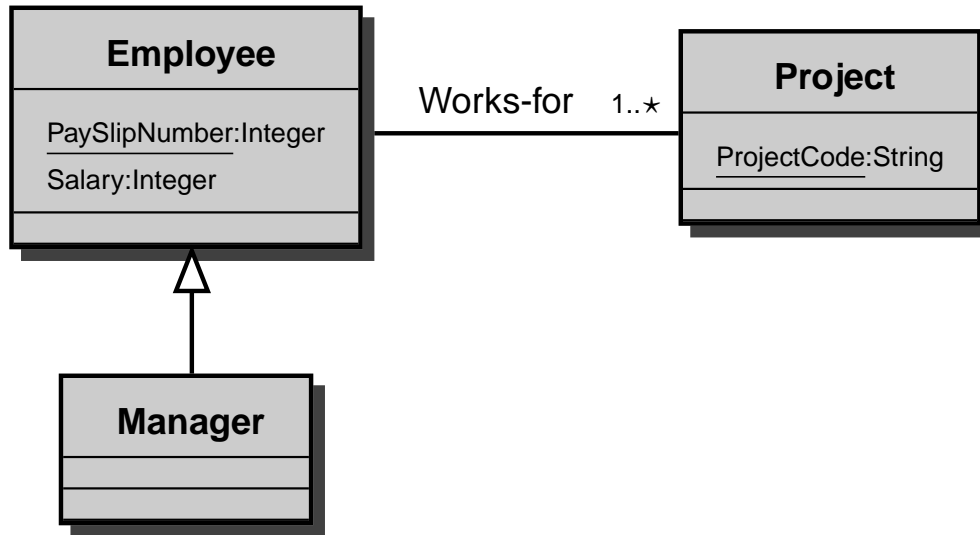
`Manager(X) :- 1-Employee(X,Y, true)`

`Project(Y) :- 2-Works-for(X,Y)`

`Works-for(X,Y) :- 2-Works-for(X,Y)`

`Salary(X,Y) :- 1-Employee(X,Y,Z)`

Queries with Sound GAV mapping



1-`Employee`(PaySlipNumber, Salary, ManagerP)

2-`Works-for`(PaySlipNumber, ProjectCode)

`Employee(X) :- 1-Employee(X,Y,Z)`

`Employee(X) :- 2-Works-for(X,Y)`

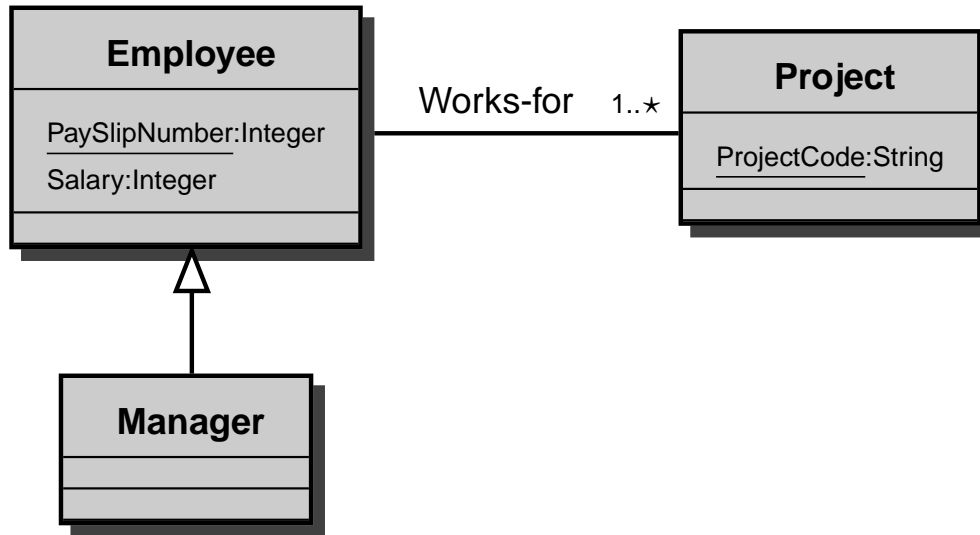
`Manager(X) :- 1-Employee(X,Y, true)`

`Project(Y) :- 2-Works-for(X,Y)`

`Works-for(X,Y) :- 2-Works-for(X,Y)`

`Salary(X,Y) :- 1-Employee(X,Y,Z)`

Queries with Sound GAV mapping



1-`Employee` (PaySlipNumber, Salary, ManagerP)

2-`Works-for` (PaySlipNumber, ProjectCode)

`Employee(X) :- 1-Employee(X,Y,Z)`

`Employee(X) :- 2-Works-for(X,Y)`

`Manager(X) :- 1-Employee(X,Y, true)`

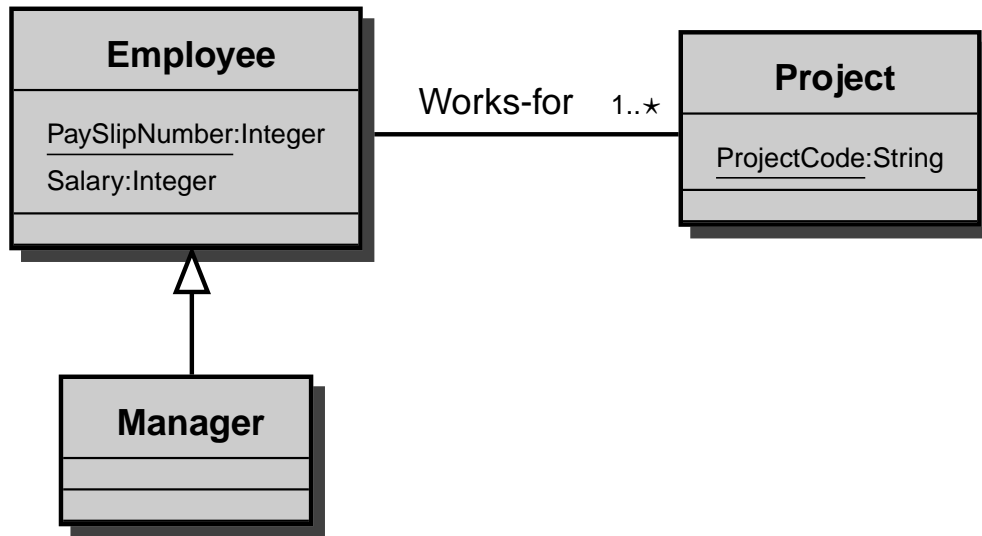
`Q(X) :- Employee(X)`

`Project(Y) :- 2-Works-for(X,Y)`

`Works-for(X,Y) :- 2-Works-for(X,Y)`

`Salary(X,Y) :- 1-Employee(X,Y,Z)`

Queries with Sound GAV mapping



1- $\text{Employee}(\underline{\text{PaySlipNumber}}, \text{Salary}, \text{ManagerP})$

2- $\text{Works-for}(\underline{\text{PaySlipNumber}}, \underline{\text{ProjectCode}})$

$\text{Employee}(X) \text{ :- } 1\text{-Employee}(X, Y, Z)$

$\text{Employee}(X) \text{ :- } 2\text{-Works-for}(X, Y)$

$\text{Manager}(X) \text{ :- } 1\text{-Employee}(X, Y, \text{true})$

$\text{Project}(Y) \text{ :- } 2\text{-Works-for}(X, Y)$

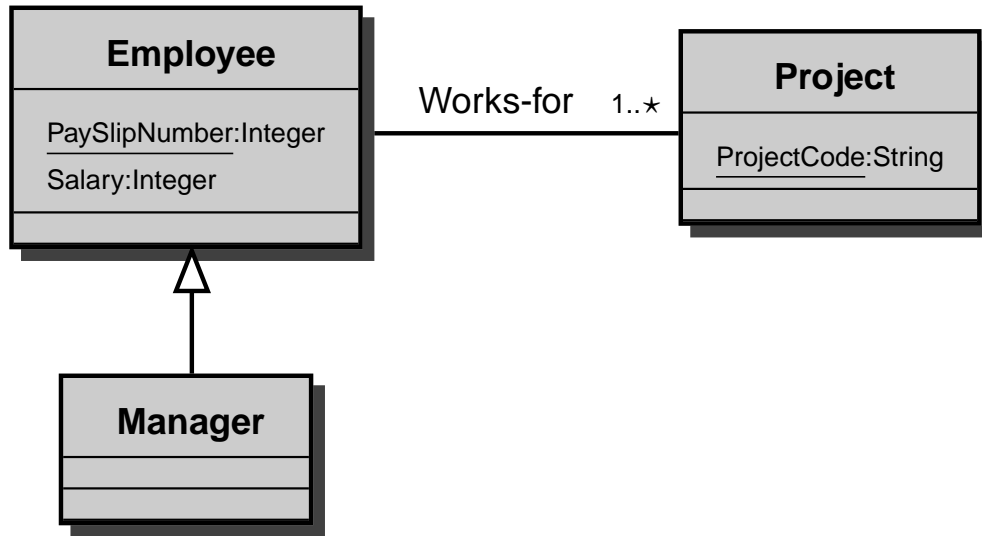
$\text{Works-for}(X, Y) \text{ :- } 2\text{-Works-for}(X, Y)$

$\text{Salary}(X, Y) \text{ :- } 1\text{-Employee}(X, Y, Z)$

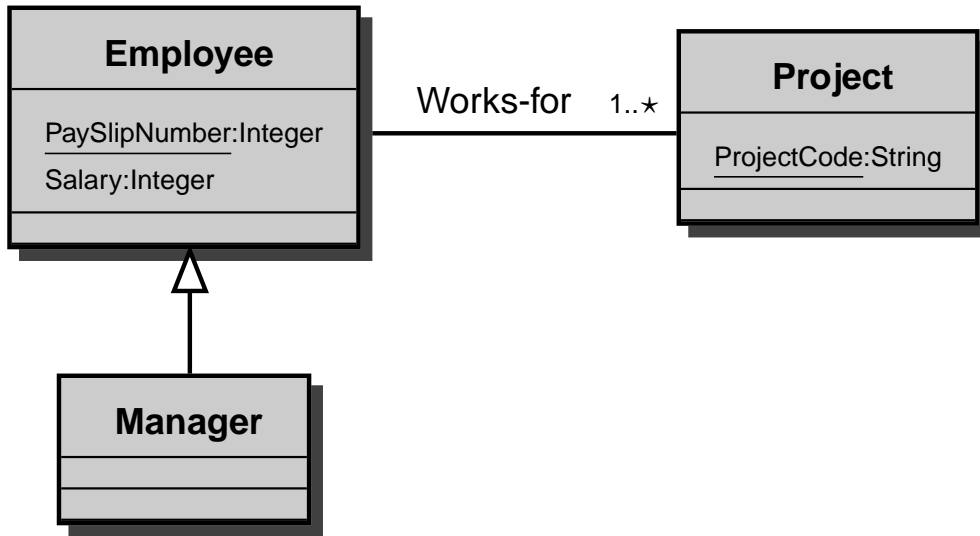
$Q(X) \text{ :- } \text{Employee}(X)$

$\Rightarrow Q'(X) \text{ :- } 1\text{-Employee}(X, Y, Z) \cup 2\text{-Works-for}(X, W)$

Sound LAV mapping



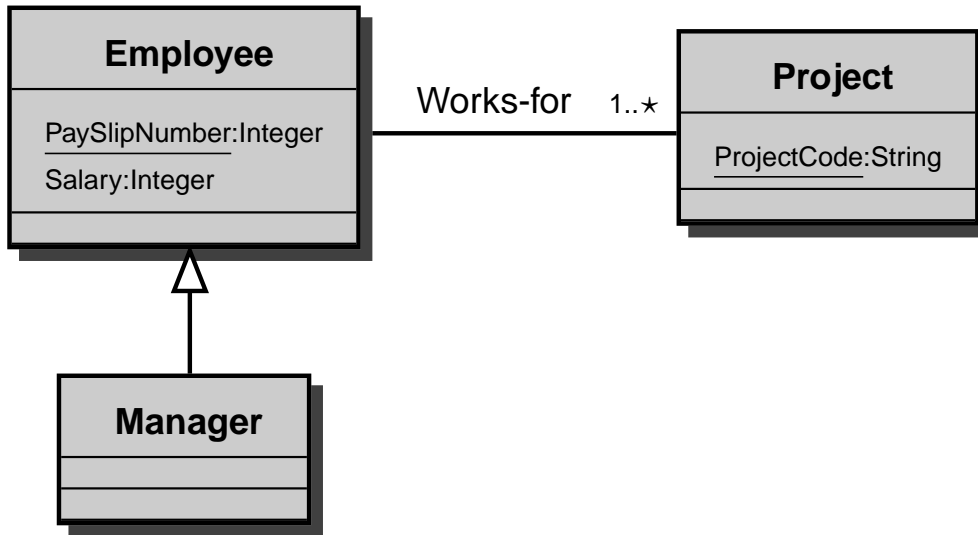
Sound LAV mapping



1-Employee (PaySlipNumber, Salary, ManagerP)

2-Works-for (PaySlipNumber, ProjectCode)

Sound LAV mapping



1-`Employee(PaySlipNumber, Salary, ManagerP)`

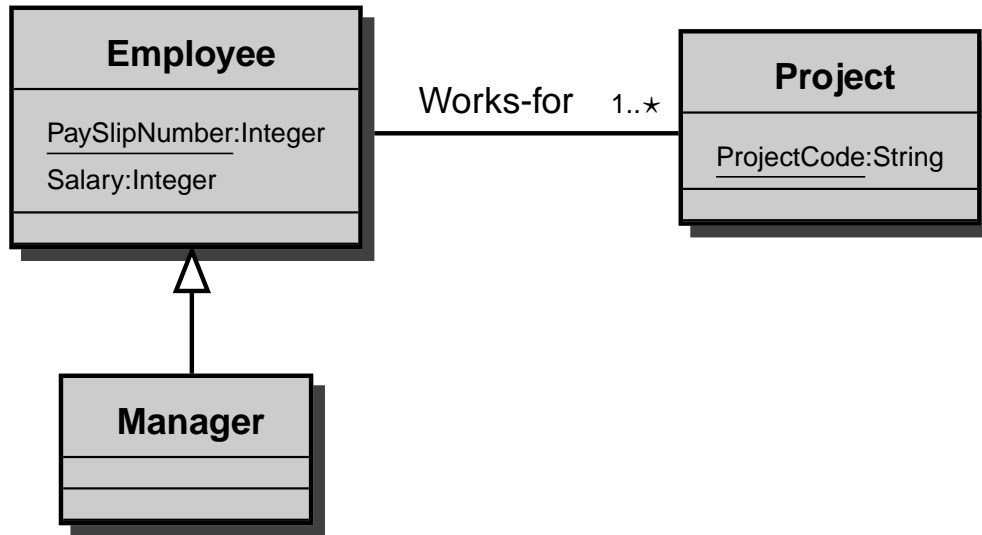
2-`Works-for(PaySlipNumber, ProjectCode)`

1-`Employee(X,Y,Z) :- Manager(X), Salary(X,Y), Z=true`

1-`Employee(X,Y,Z) :- Employee(X), ¬Manager(X), Salary(X,Y), Z=false`

2-`Works-for(X,Y) :- Works-for(X,Y)`

Queries with Sound LAV mapping



1-`Employee(PaySlipNumber, Salary, ManagerP)`

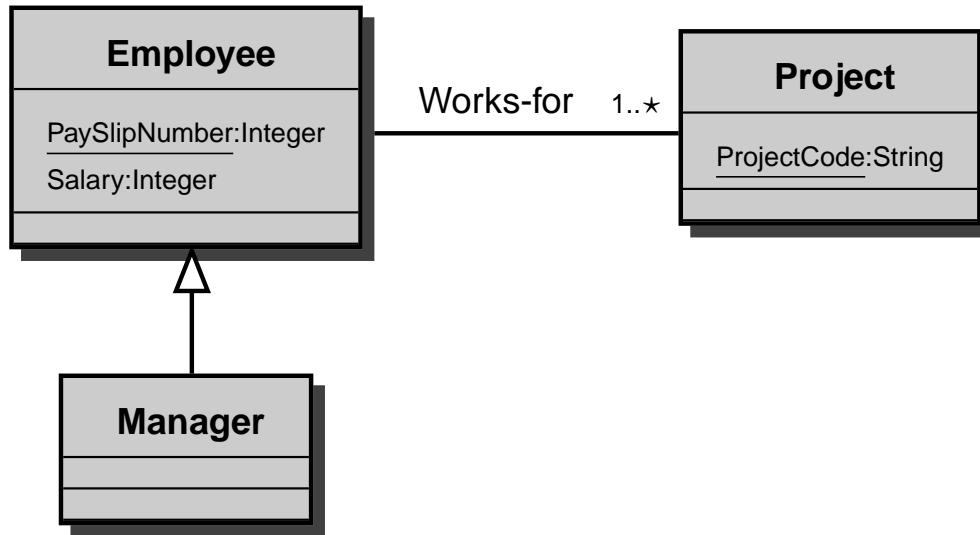
2-`Works-for(PaySlipNumber, ProjectCode)`

1-`Employee(X,Y,Z) :- Manager(X), Salary(X,Y), Z=true`

1-`Employee(X,Y,Z) :- Employee(X), ¬Manager(X), Salary(X,Y), Z=false`

2-`Works-for(X,Y) :- Works-for(X,Y)`

Queries with Sound LAV mapping



1-`Employee(PaySlipNumber, Salary, ManagerP)`

2-`Works-for(PaySlipNumber, ProjectCode)`

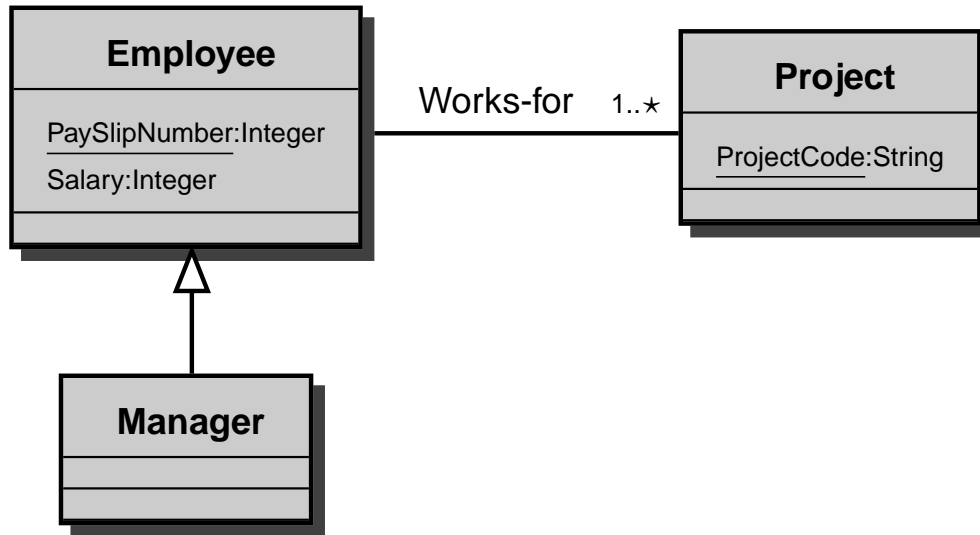
1-`Employee(X,Y,Z) :- Manager(X), Salary(X,Y), Z=true`

1-`Employee(X,Y,Z) :- Employee(X), ¬Manager(X), Salary(X,Y), Z=false`

2-`Works-for(X,Y) :- Works-for(X,Y)`

`Q(X) :- Manager(X), Works-for(X,Y), Project(Y)`

Queries with Sound LAV mapping



1-`Employee`(PaySlipNumber, Salary, ManagerP)

2-`Works-for`(PaySlipNumber, ProjectCode)

1-`Employee`(X,Y,Z) :- `Manager`(X), `Salary`(X,Y), Z=true

1-`Employee`(X,Y,Z) :- `Employee`(X), \neg `Manager`(X), `Salary`(X,Y), Z=false

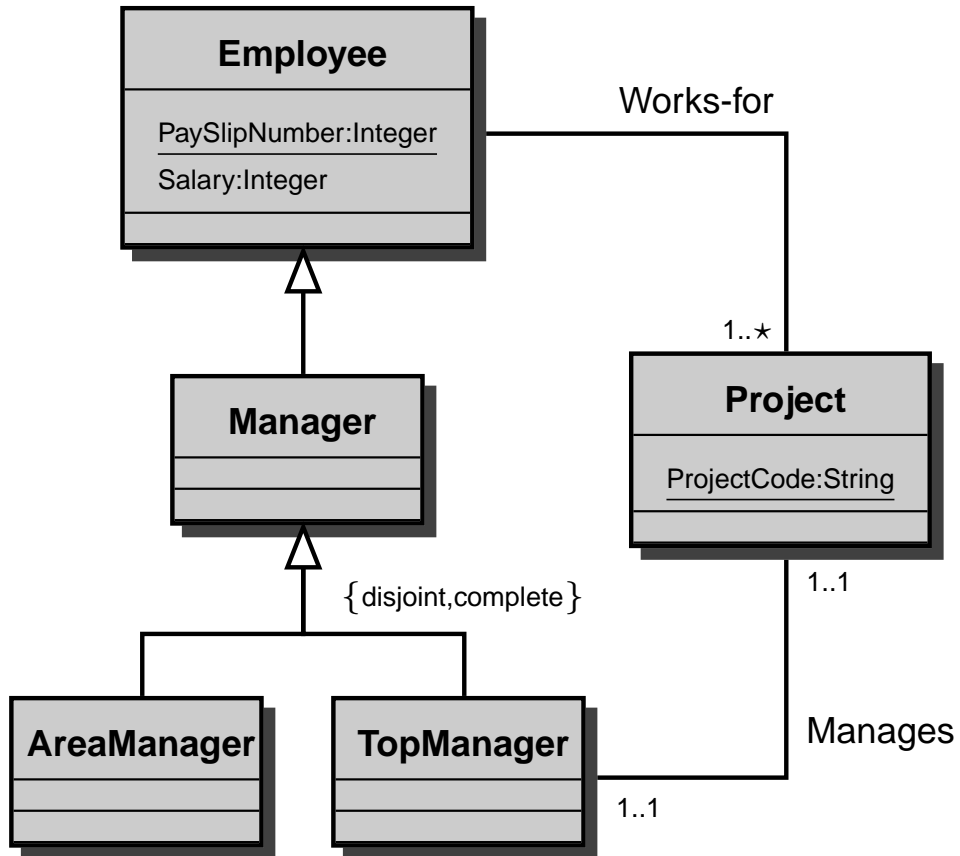
2-`Works-for`(X,Y) :- `Works-for`(X,Y)

`Q`(X) :- `Manager`(X), `Works-for`(X,Y), `Project`(Y)

\Rightarrow `Q'`(X) :- 1-`Employee`(X,Y,true), 2-`Works-for`(X,Z)

Reasoning over queries

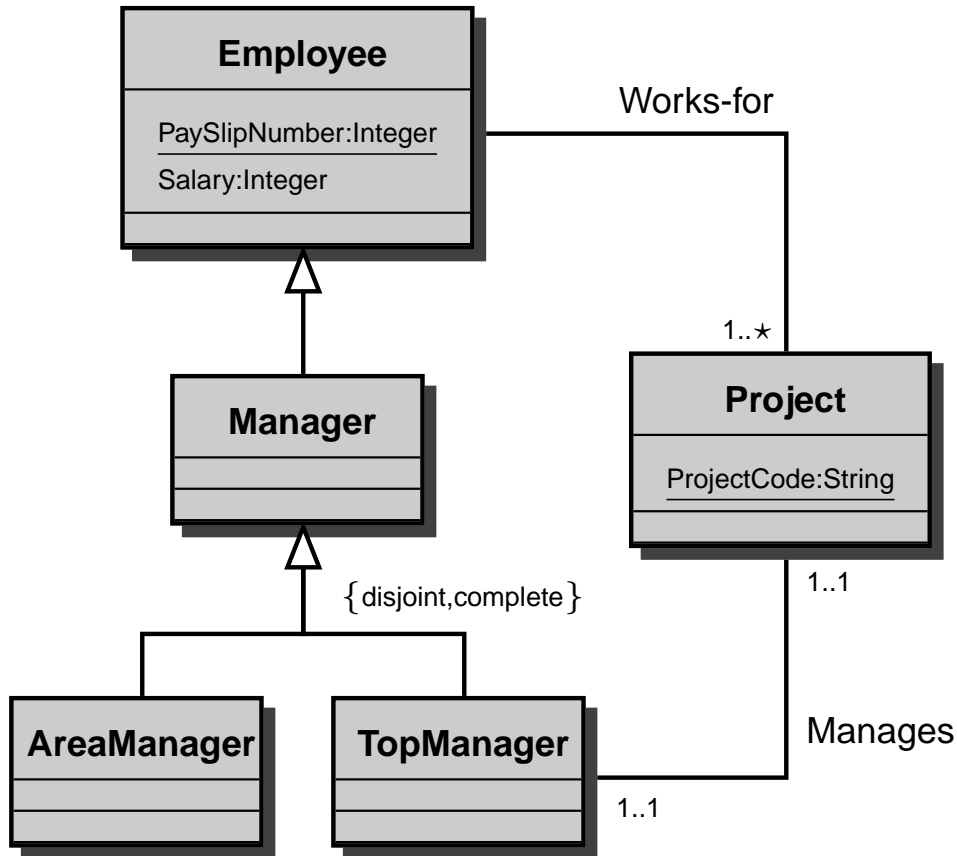
$Q(X, Y) :- \text{Employee}(X), \text{Works-for}(X, Y), \text{Manages}(X, Y)$



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

Reasoning over queries

$Q(X, Y) :- \text{Employee}(X), \text{Works-for}(X, Y), \text{Manages}(X, Y)$



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



INCONSISTENT QUERY!

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

Local-as-view

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

Global-as-view

- 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
 - “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 Practice

- Most implemented ontology based systems:

Current Practice

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

Current Practice

- Most implemented ontology based 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

Conclusions

Do you have an ontology in your application?

Conclusions

Do you have an ontology in your application?

Pay attention!