Logic

The Meaning of Entity-Relationship Diagrams

Enrico Franconi

franconi@inf.unibz.it
http://www.inf.unibz.it/~franconi

Faculty of Computer Science, Free University of Bozen-Bolzano

What is a Conceptual Schema

- A conceptual schema is a formal conceptualisation of the world.

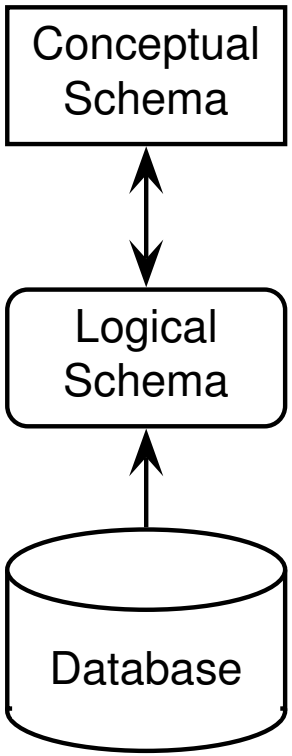
What is a Conceptual Schema

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

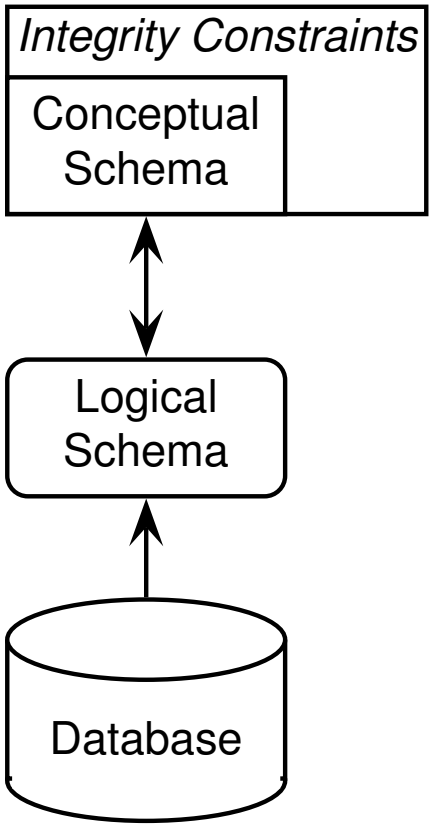
What is a Conceptual Schema

- A conceptual schema is a formal conceptualisation of the world.
- A conceptual schema specifies a set of *constraints*, which declare what should necessarily hold in any possible database.
- Given a conceptual schema, a *legal database* is a database satisfying the constraints.

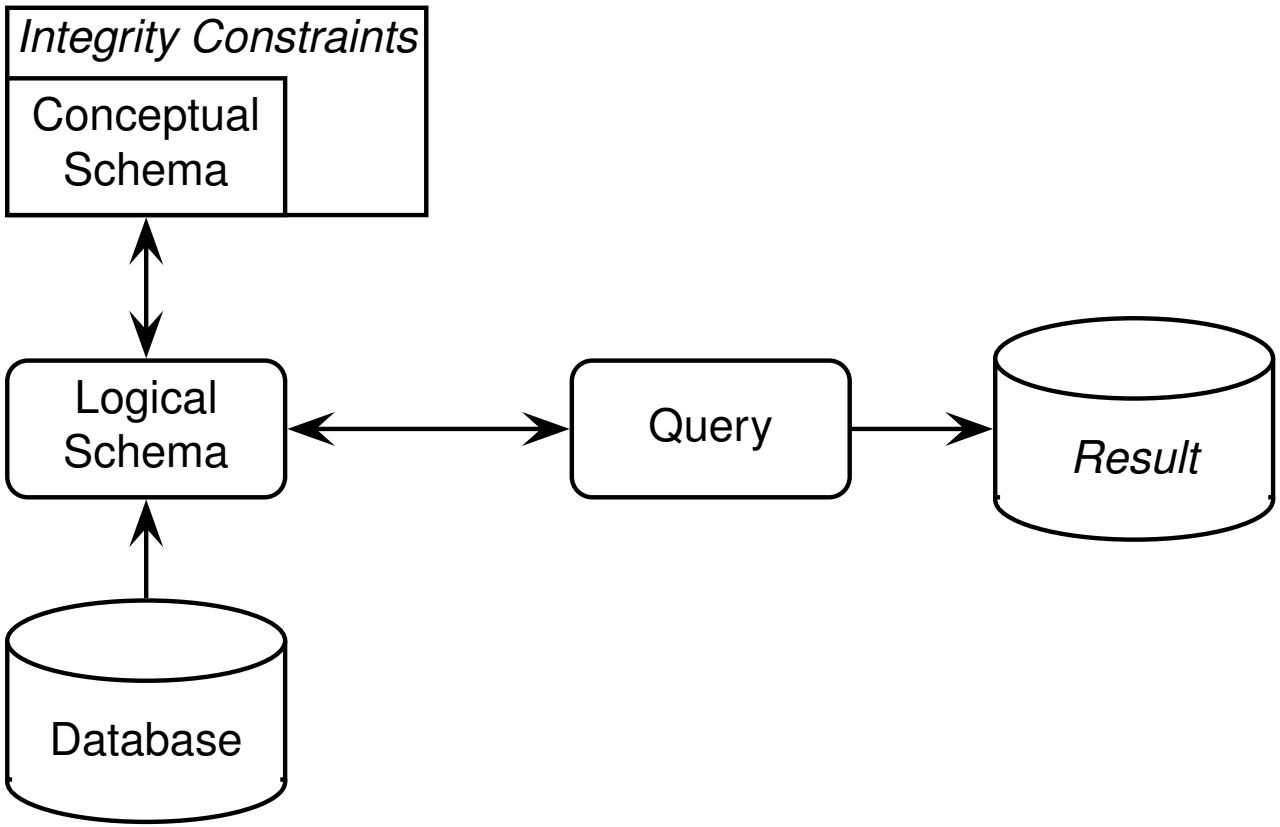
The Architecture of a Database



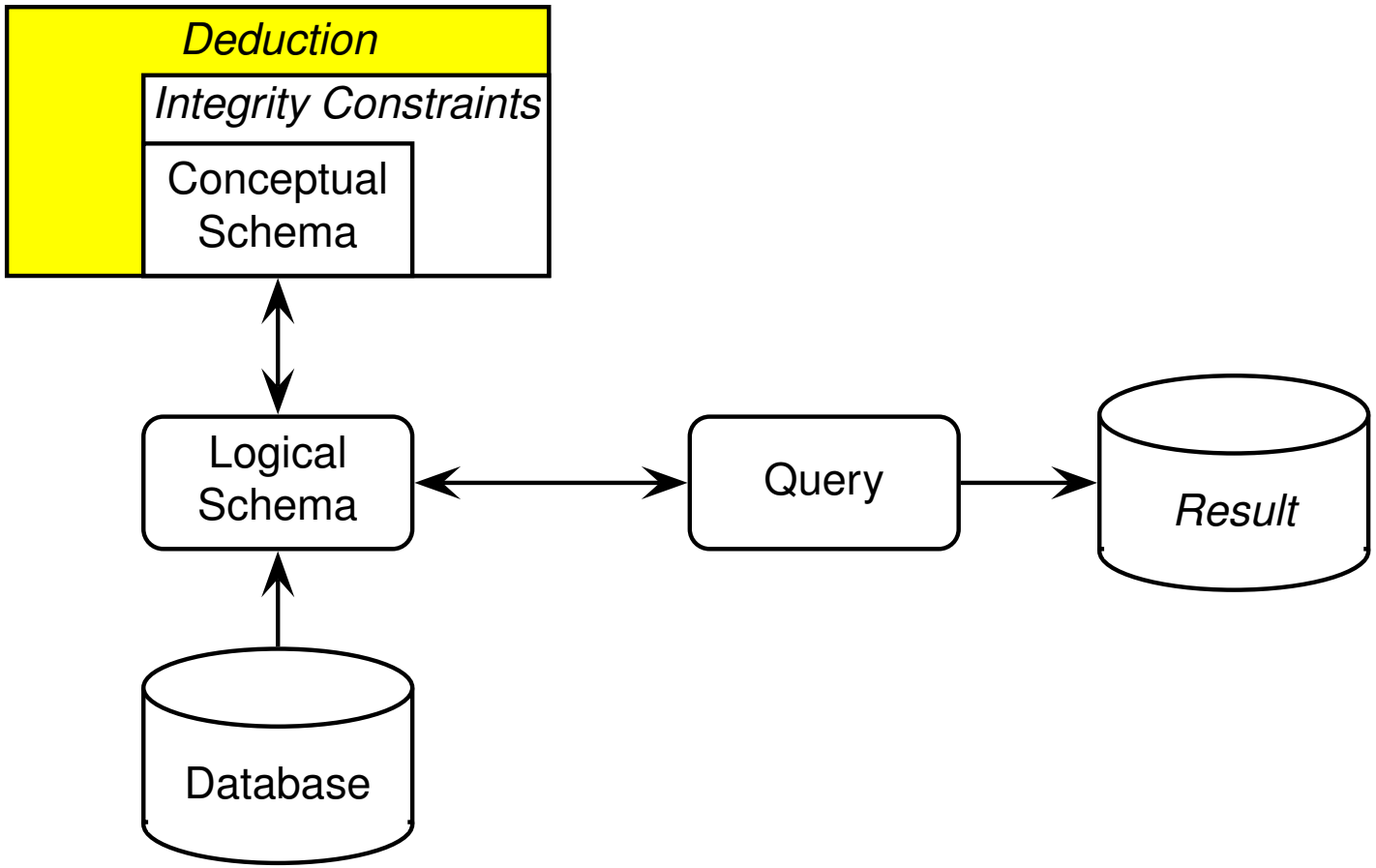
The Architecture of a Database



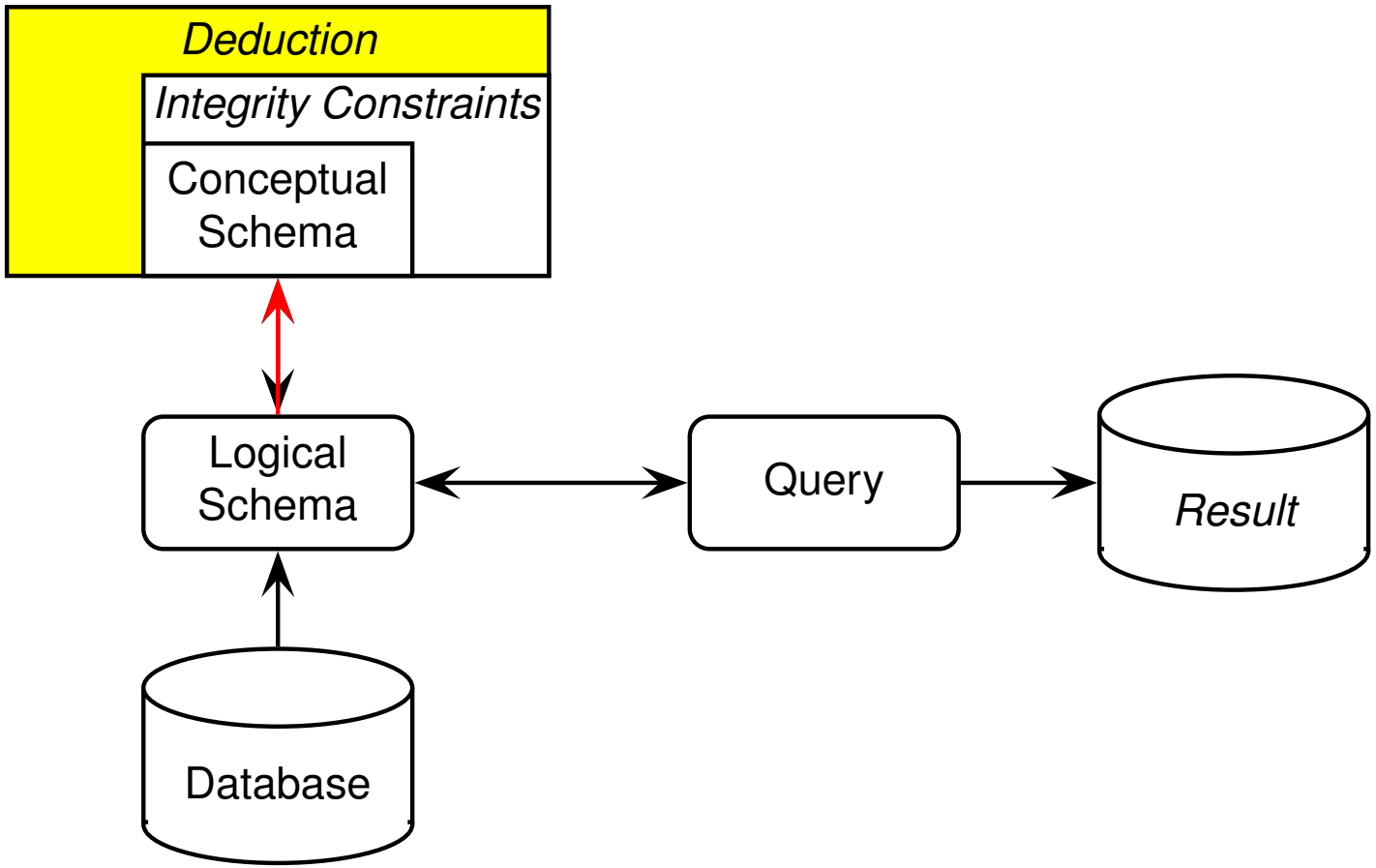
The Architecture of a Database



The Architecture of a Database



The Architecture of a Database



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), frame-based (having only concepts and properties), or logic-based (e.g. Ontolingua and DAML+OIL).

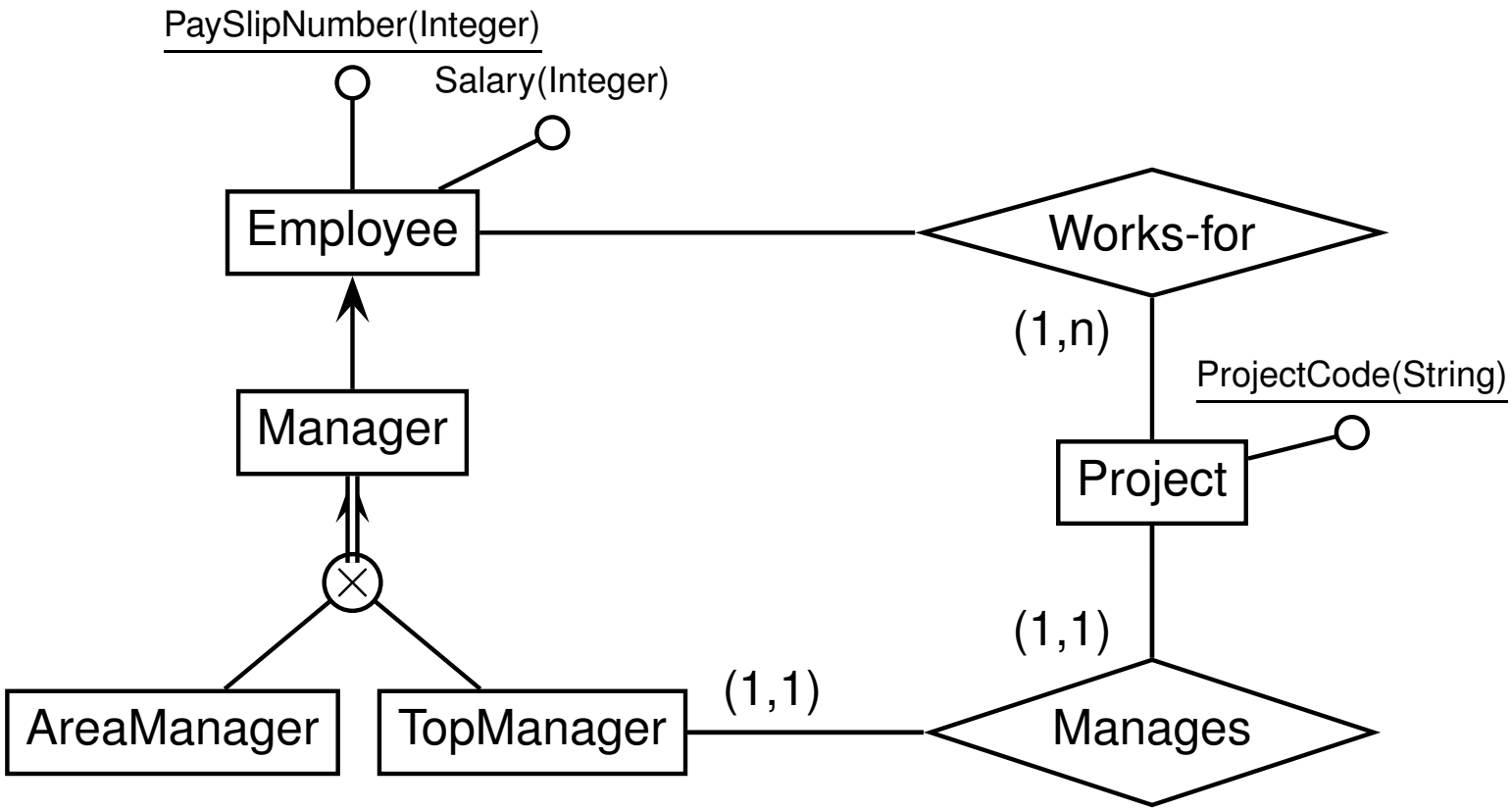
Ontology languages and Conceptual Data Models

- 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), 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.

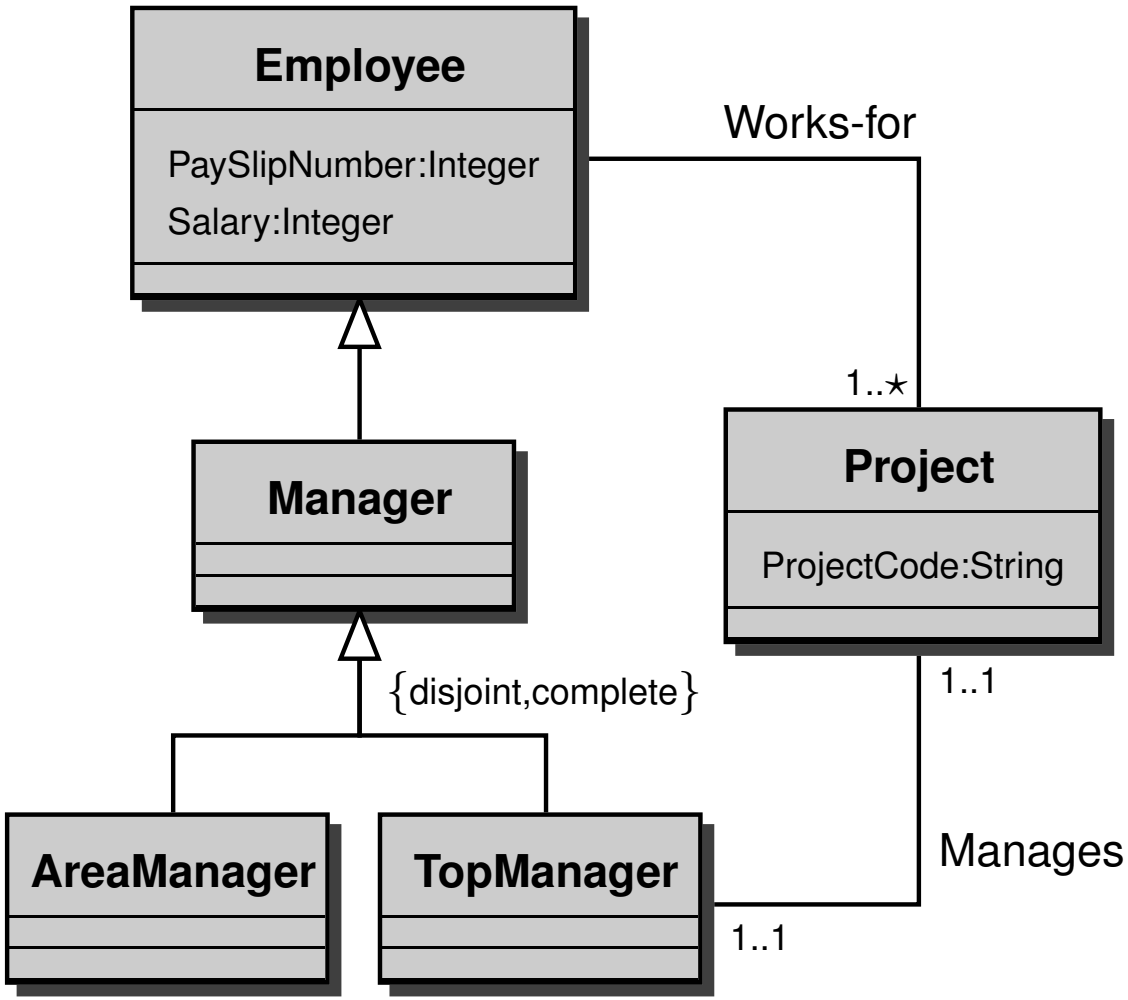
Ontology languages and Conceptual Data Models

- 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), 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.
- The Entity-Relationship conceptual data model and UML Class Diagrams can be considered as ontology languages.

Entity-Relationship Diagram



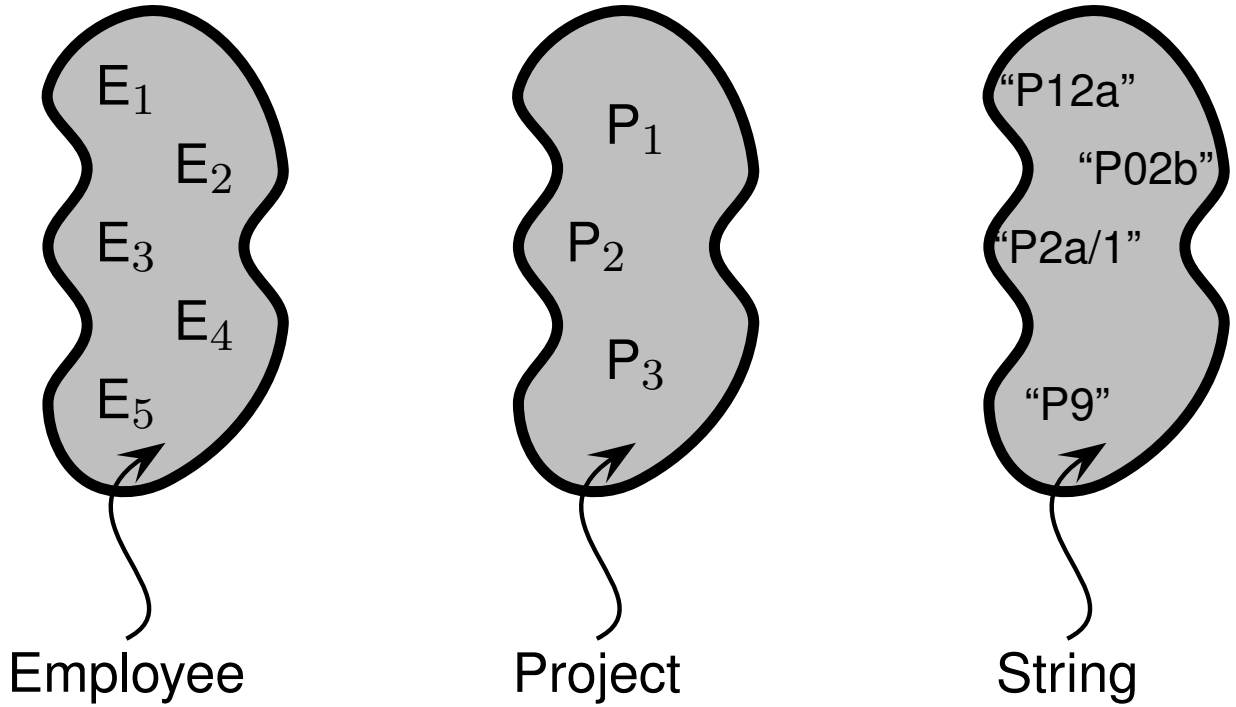
UML Class Diagram



Meaning of basic constructs

In a specific legal database:

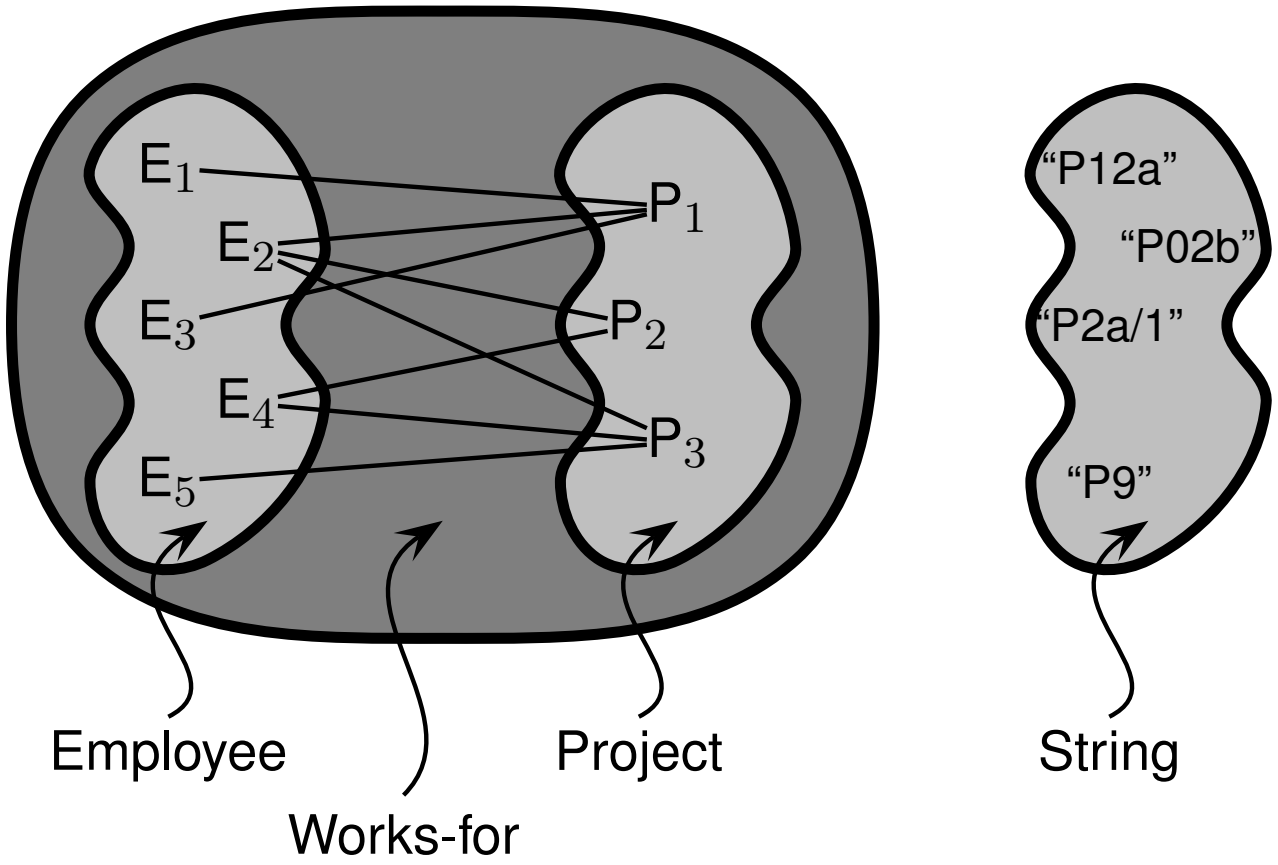
- An entity is a ***set of instances***;
- a n-ary relationship is a ***set of n-tuples of instances***;
- an attribute is a ***set of pairs of an instance and a domain element***.



Meaning of basic constructs

In a specific legal database:

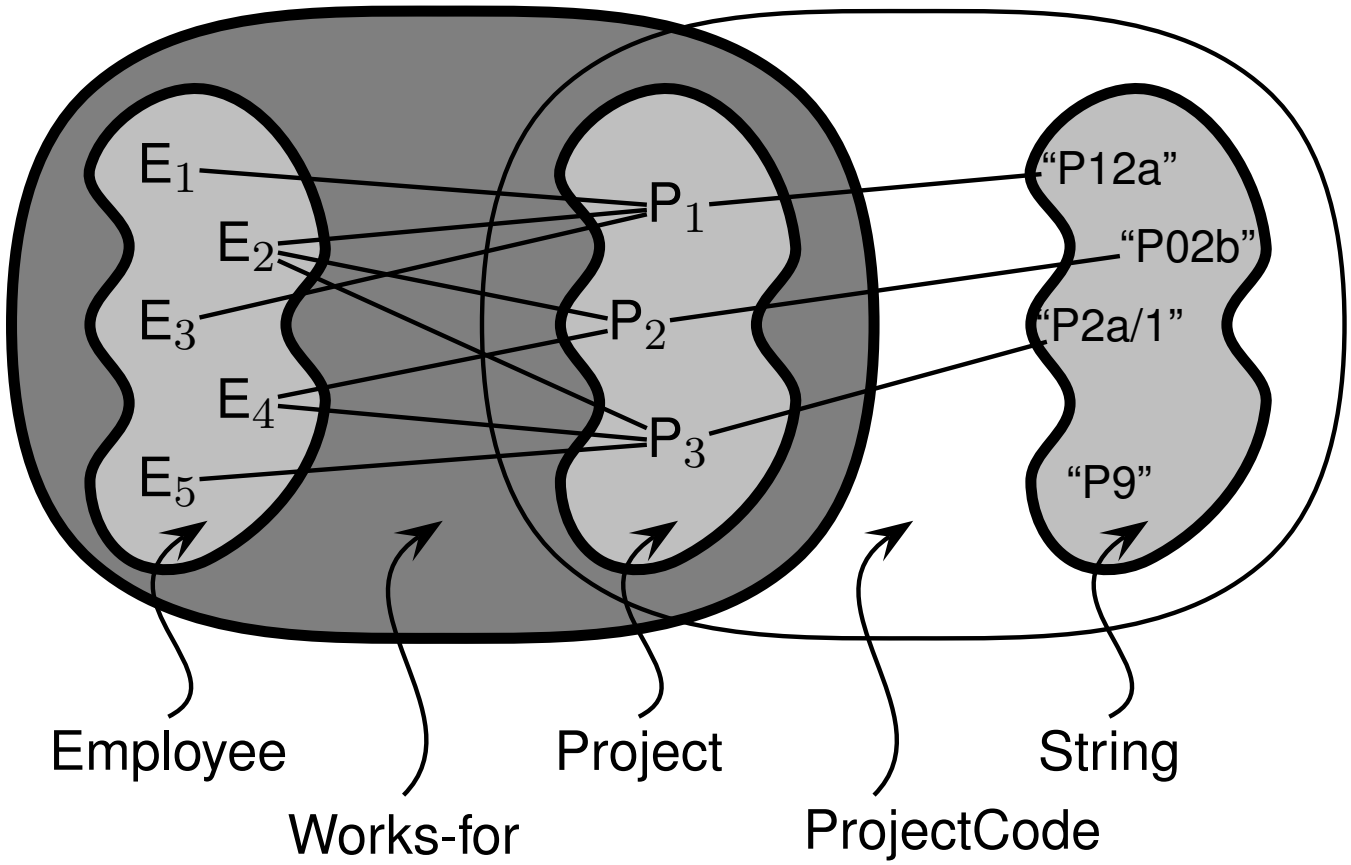
- An entity is a ***set of instances***;
- a n-ary relationship is a ***set of n-tuples of instances***;
- an attribute is a ***set of pairs of an instance and a domain element***.



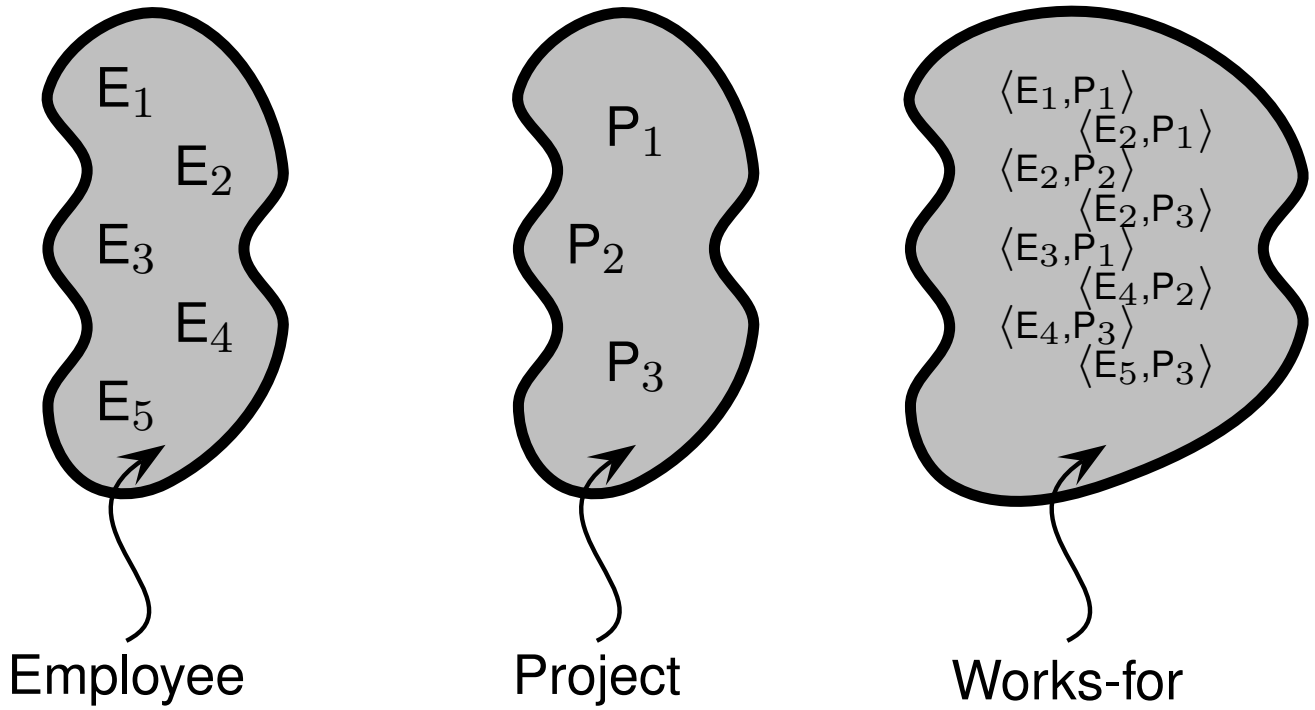
Meaning of basic constructs

In a specific legal database:

- An entity 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***.



Relations as sets of tuples



The relational representation

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”
. . .

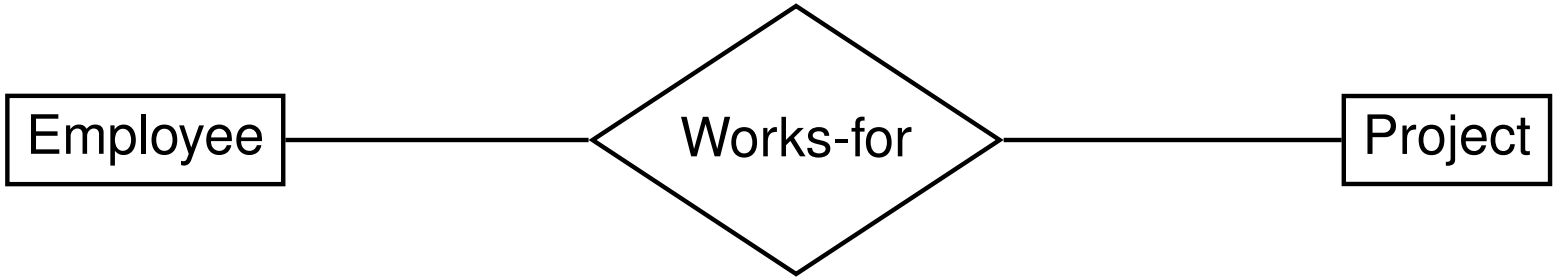
Works-for

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

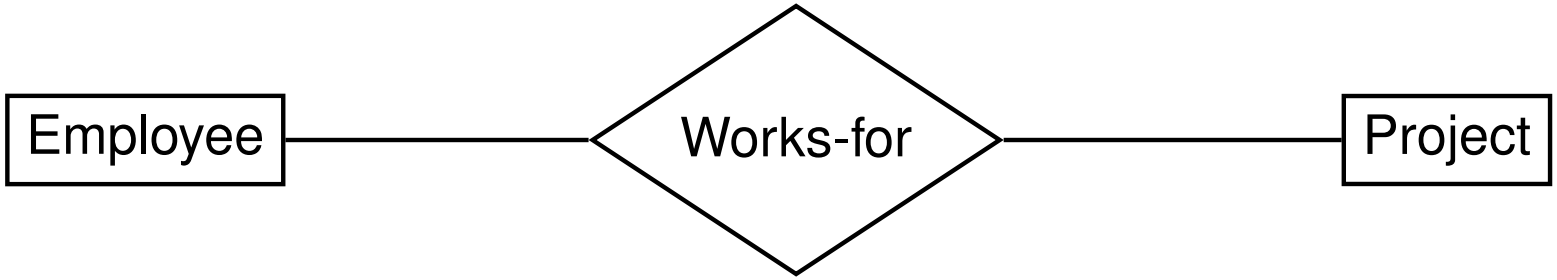
ProjectCode

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

Meaning of Relationships

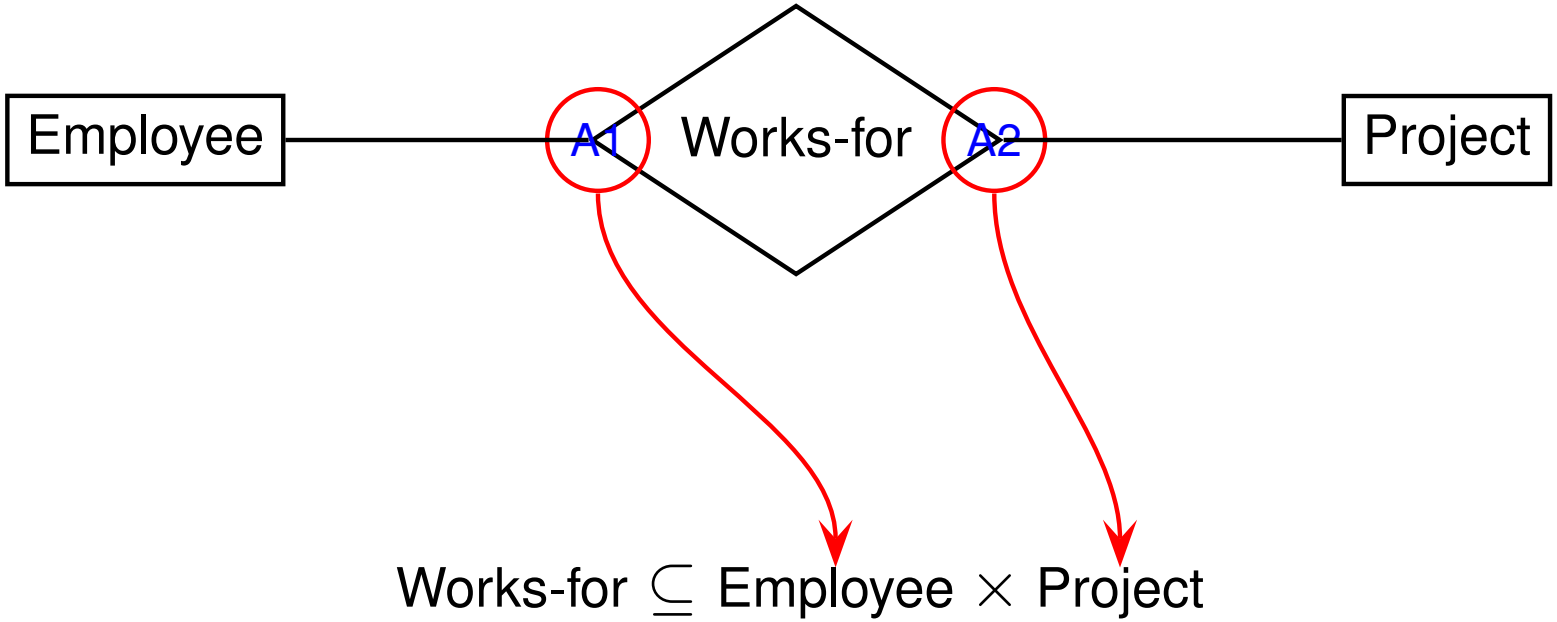


Meaning of Relationships

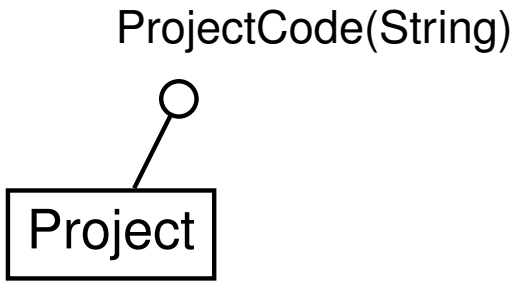


$$\text{Works-for} \subseteq \text{Employee} \times \text{Project}$$

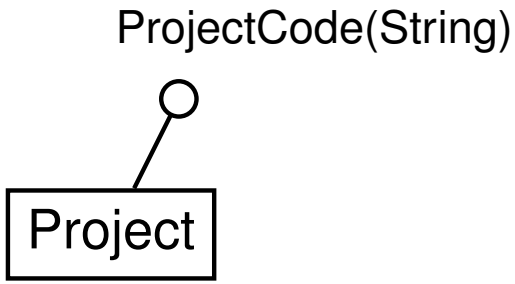
Meaning of Relationships



Meaning of Attributes

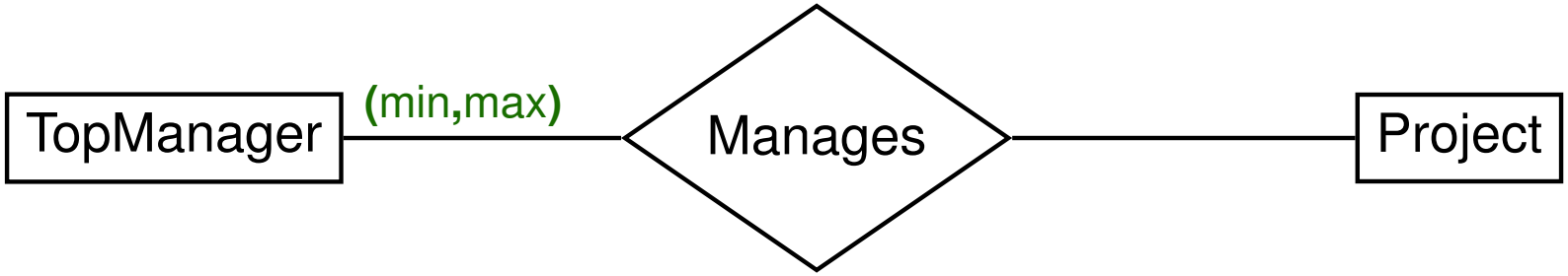


Meaning of Attributes

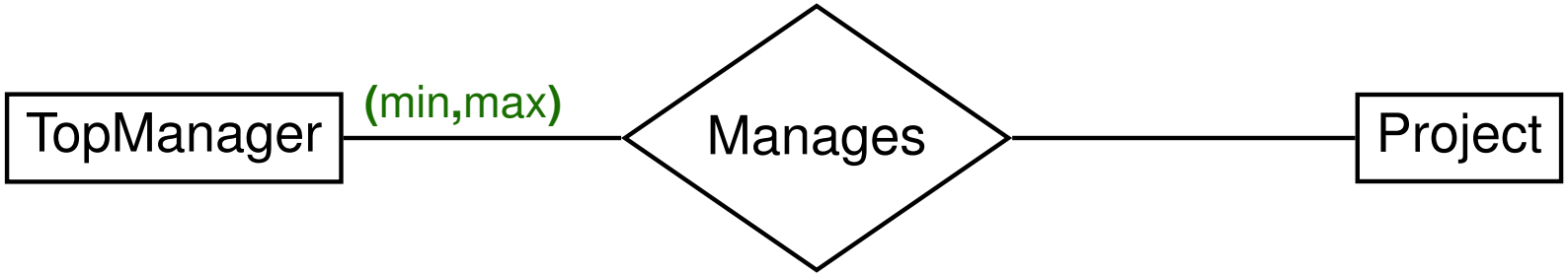


$$\mathsf{Project} \subseteq \{p \mid \sharp(\mathsf{ProjectCode} \cap (\{p\} \times \mathtt{String})) \geq 1\}$$

Meaning of Cardinality Constraints



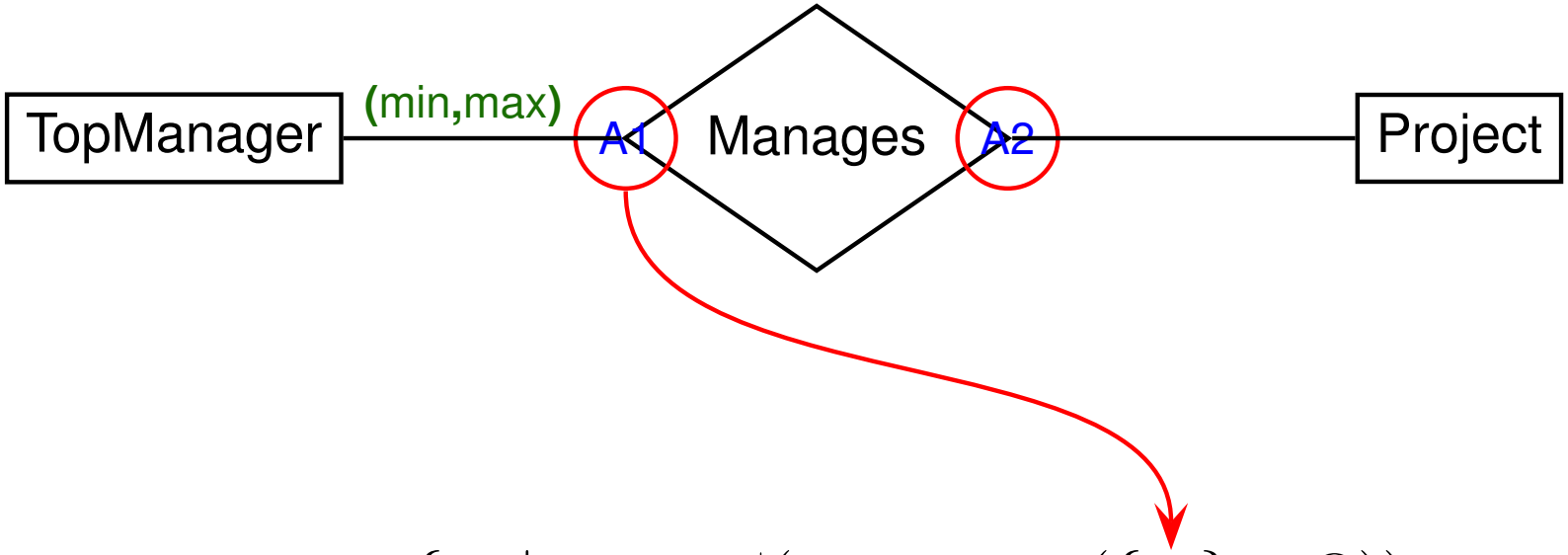
Meaning of Cardinality Constraints



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

(where Ω is the set of all instances)

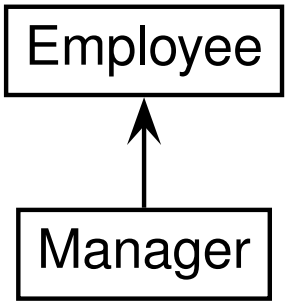
Meaning of Cardinality Constraints



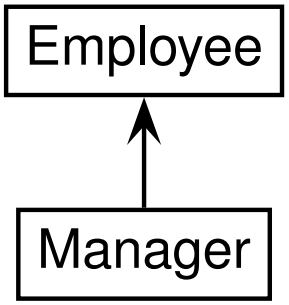
$$\text{TopManager} \subseteq \{m \mid \text{max} \geq \sharp(\text{Manages} \cap (\{m\} \times \Omega)) \geq \text{min}\}$$

(where Ω is the set of all instances)

Meaning of ISA

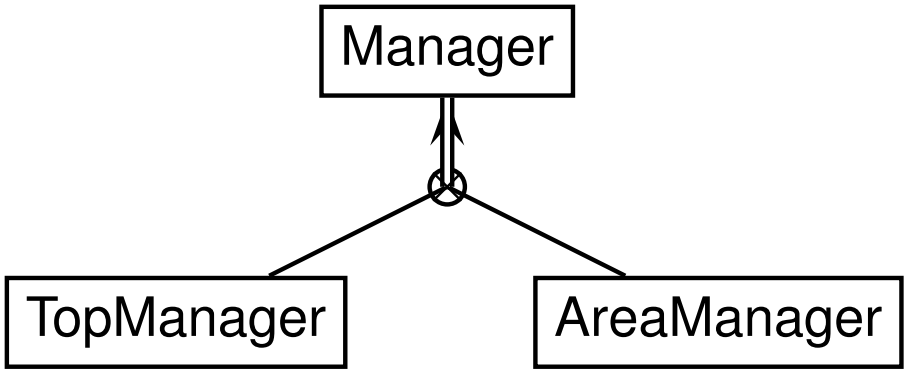


Meaning of ISA

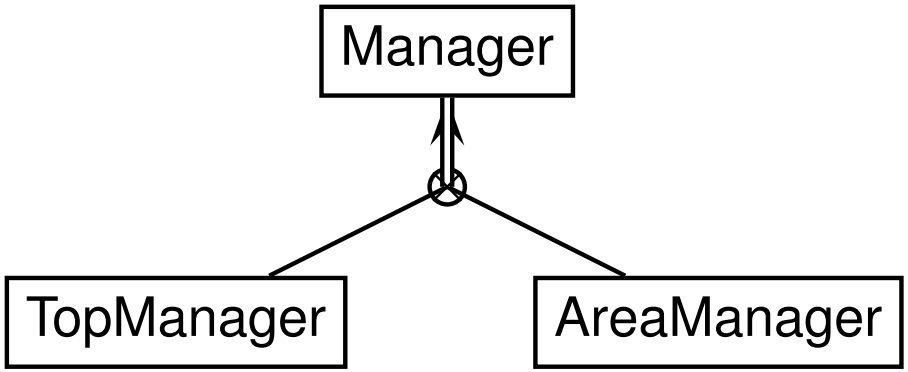


$$\text{Manager} \subseteq \text{Employee}$$

Meaning of *disjoint* and *total* constraints



Meaning of *disjoint* and *total* constraints



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

Meaning of the initial diagram

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

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

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

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

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

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

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

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

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

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

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

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

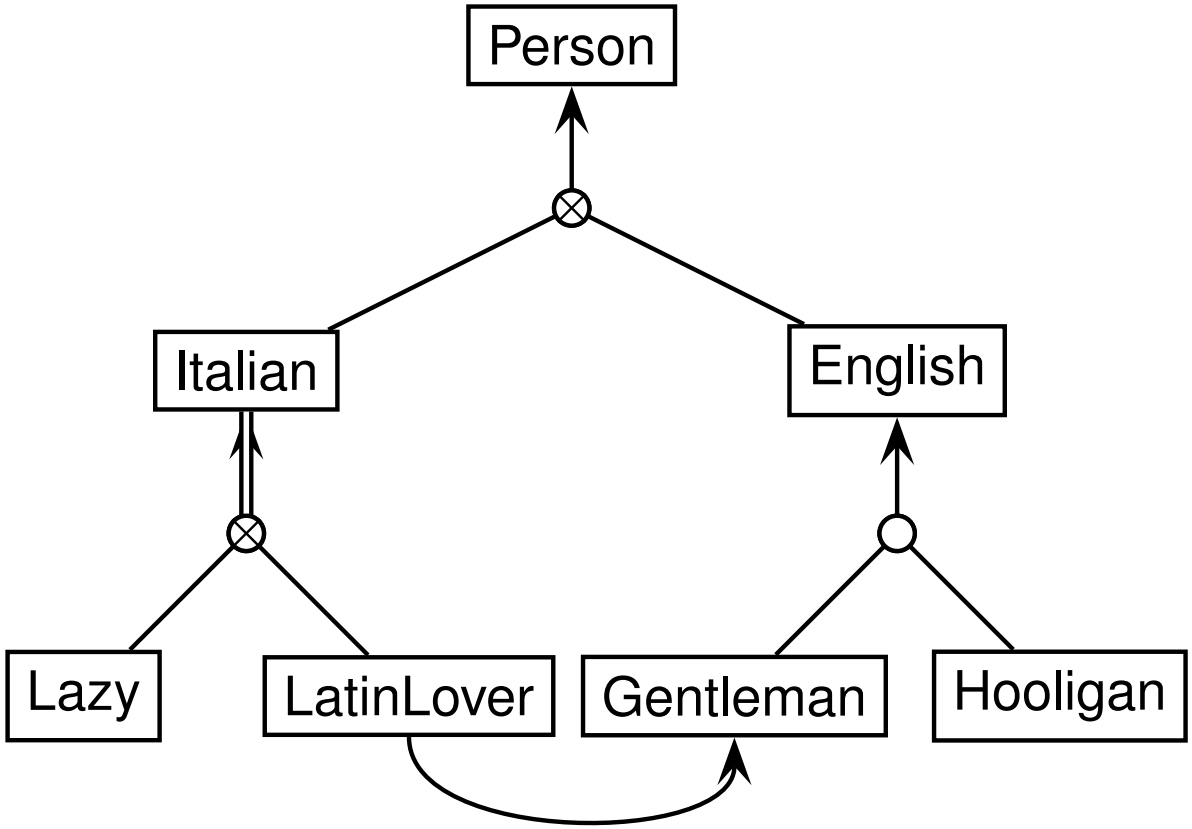
$$\text{Manager} \subseteq \text{AreaManager} \cup \text{TopManager}$$

Inference

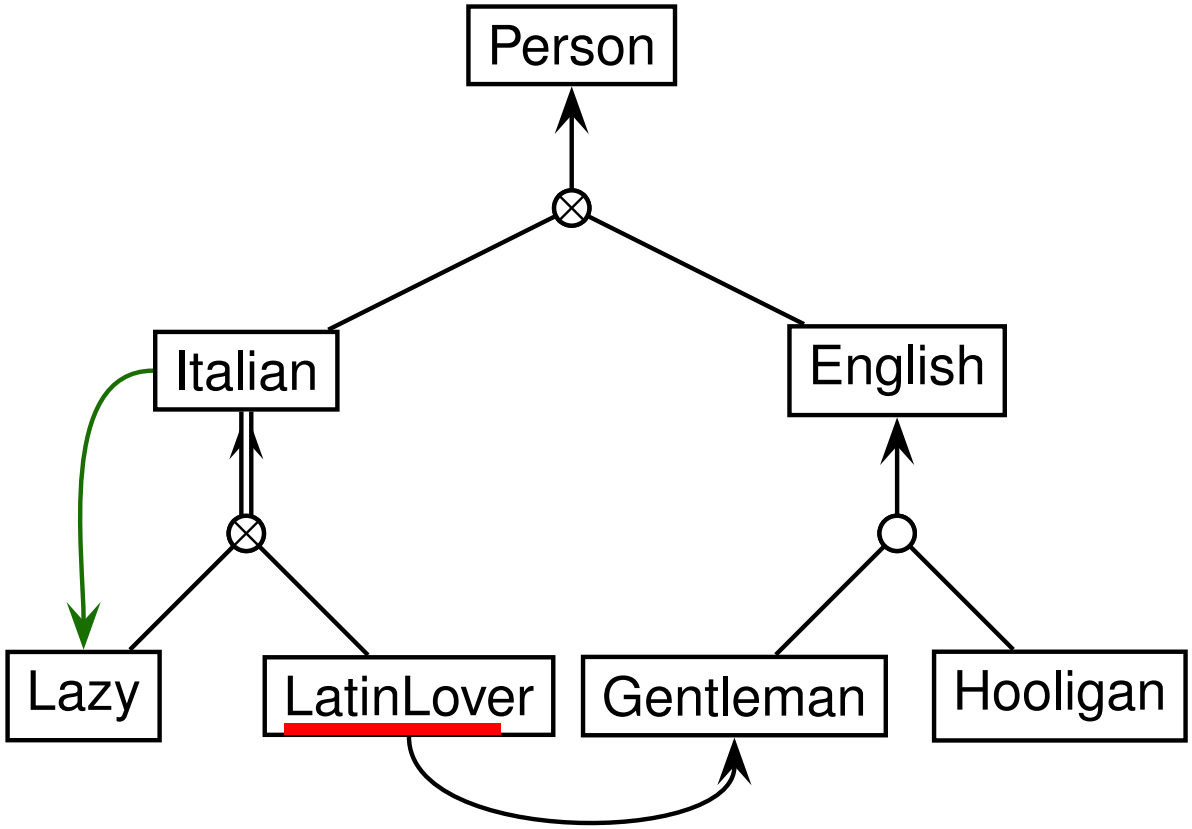
Given a collection of constraints, such as an Entity-Relationship diagram, it is possible that additional contraints can be inferred.

- An entity is inconsistent if it denotes the empty set in any legal database.
- An entity is a subentity of another entity if the former denotes a subset of the set denoted by the latter in any legal database.
- Two entities are equivalent if they denote the same set in any legal database.
- A stricter contraint is inferred – e.g., a **cardinality** contraint – if it holds in in any legal database.
- . . .

Inference


Person
Italian
English
Lazy
LatinLover
Gentleman
Hooligan

Inference



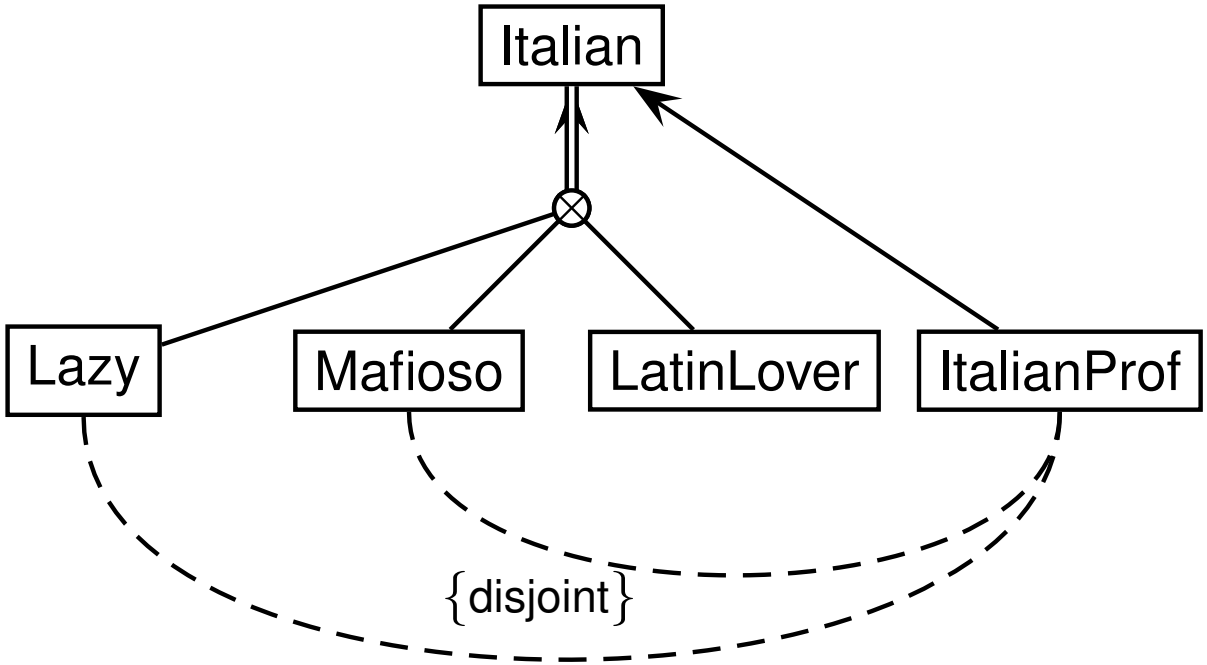
implies

$\text{LatinLover} = \emptyset$

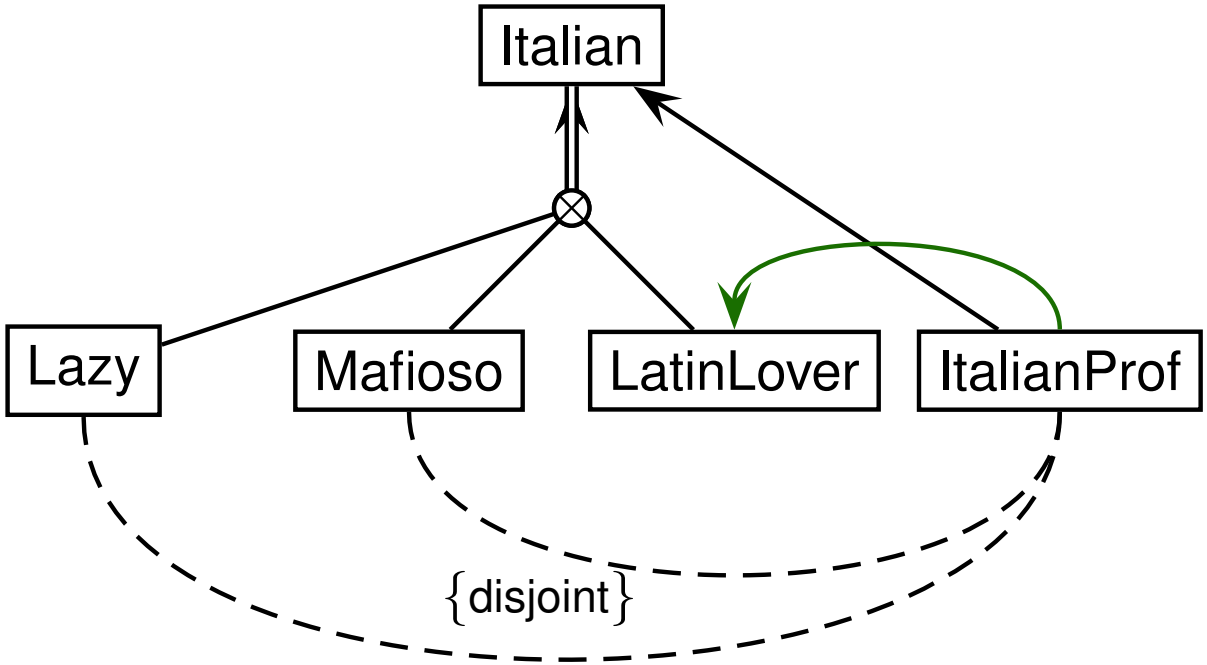
$\text{Italian} \subseteq \text{Lazy}$

$\text{Italian} \equiv \text{Lazy}$

Reasoning by cases



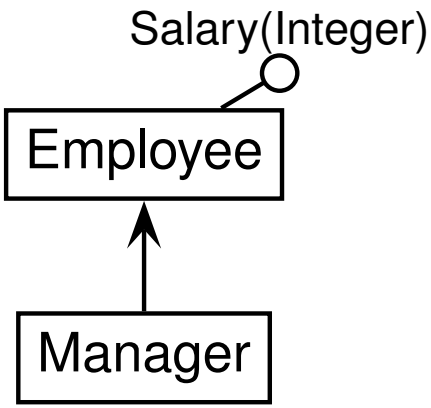
Reasoning by cases



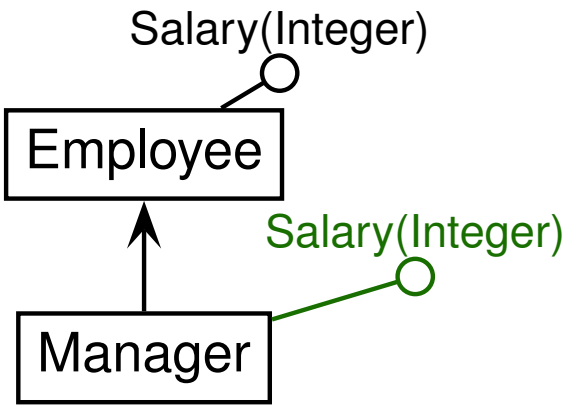
implies

ItalianProf $\subseteq$ LatinLover

ISA and Inheritance



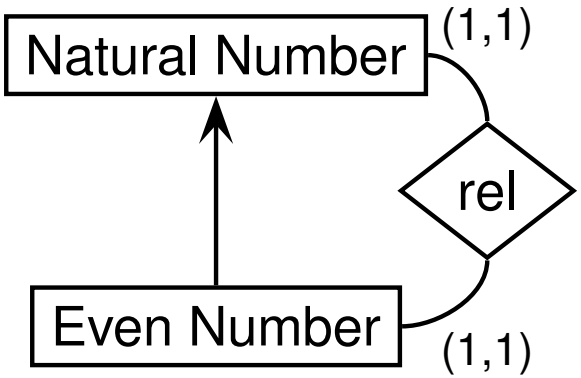
ISA and Inheritance



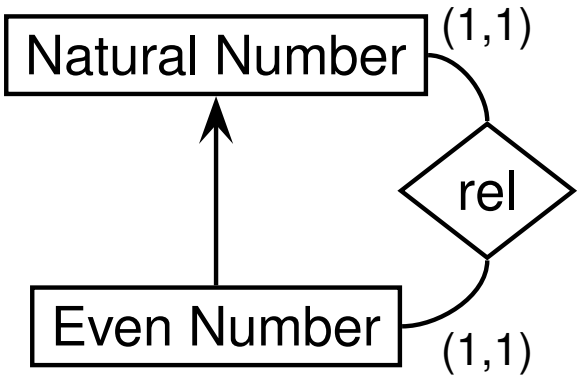
implies

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

Bijection bewteen Entities



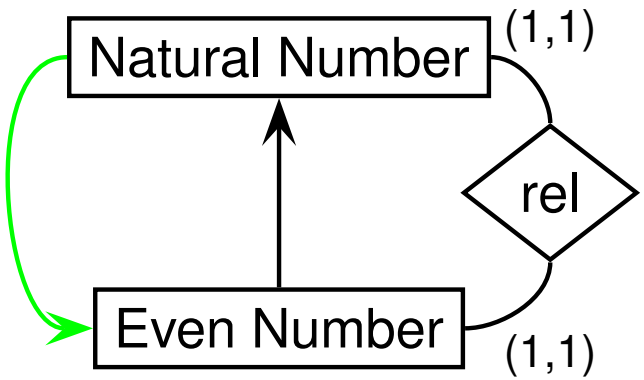
Bijection bewteen Entities



implies

“the entities *’Natural Number’* and *’Even Number’* contain the same number of instances”.

Bijection bewteen Entities

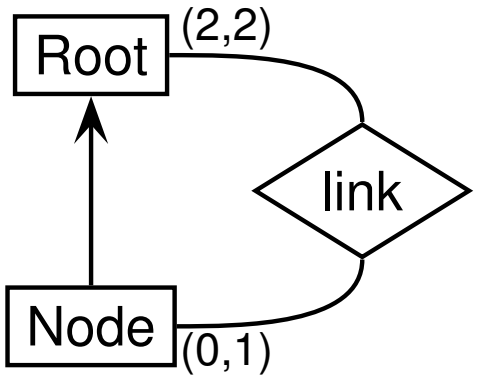


implies

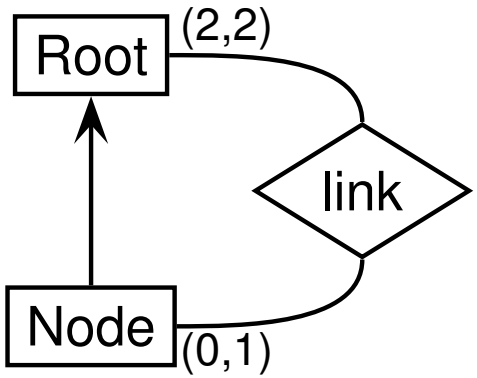
“the entities *’Natural Number’* and *’Even Number’* contain the same number of instances”.

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

Infinite Databases



Infinite Databases



implies

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