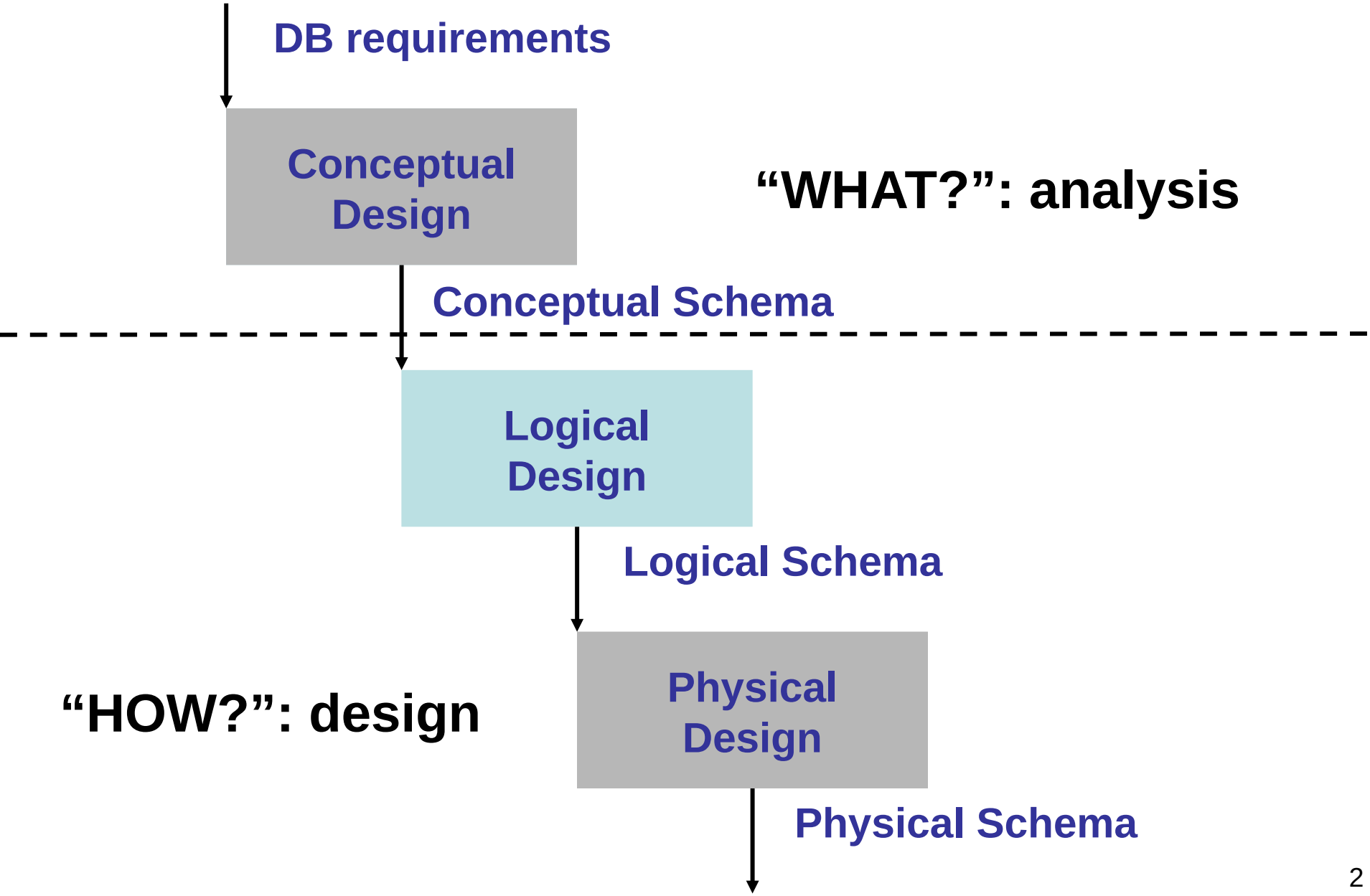
Databases

Logical Design

Danilo Montesi

danilo.montesi@unibo.it

Requirements & Design



Logical Design: Objective

- The goal is to “translate” the conceptual schema into a logical schema representing the same data in a correct and efficient way

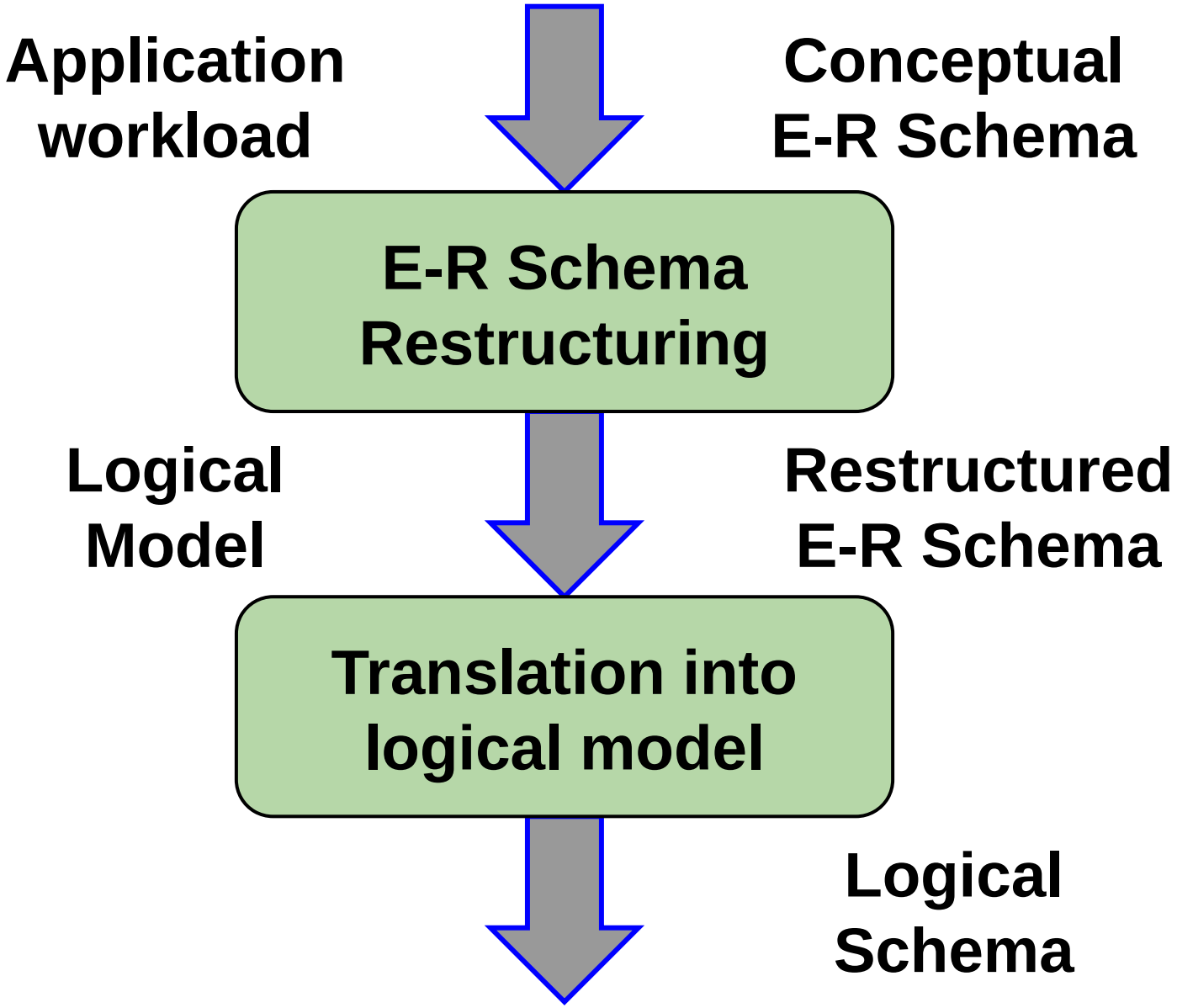
Logical Design: Input and Output Data

- Input:
  - Conceptual Schema
  - Application workload information
  - Logical Model
- Output:
  - Logical Schema (either relational, object-oriented, graph …)
  - Associated documentation

It is NOT a Simple Translation!

- Some aspects may not be represented directly
- In this phase we must also consider performance (efficiency)

Transformation (Sub-)Phases



E-R Schema Restructuring

- Why?
  - Simplify the translation
  - Optimize the performances
- Please note that:
  - A restructured E-R is no more a “conceptual schema” in the strict sense of the term

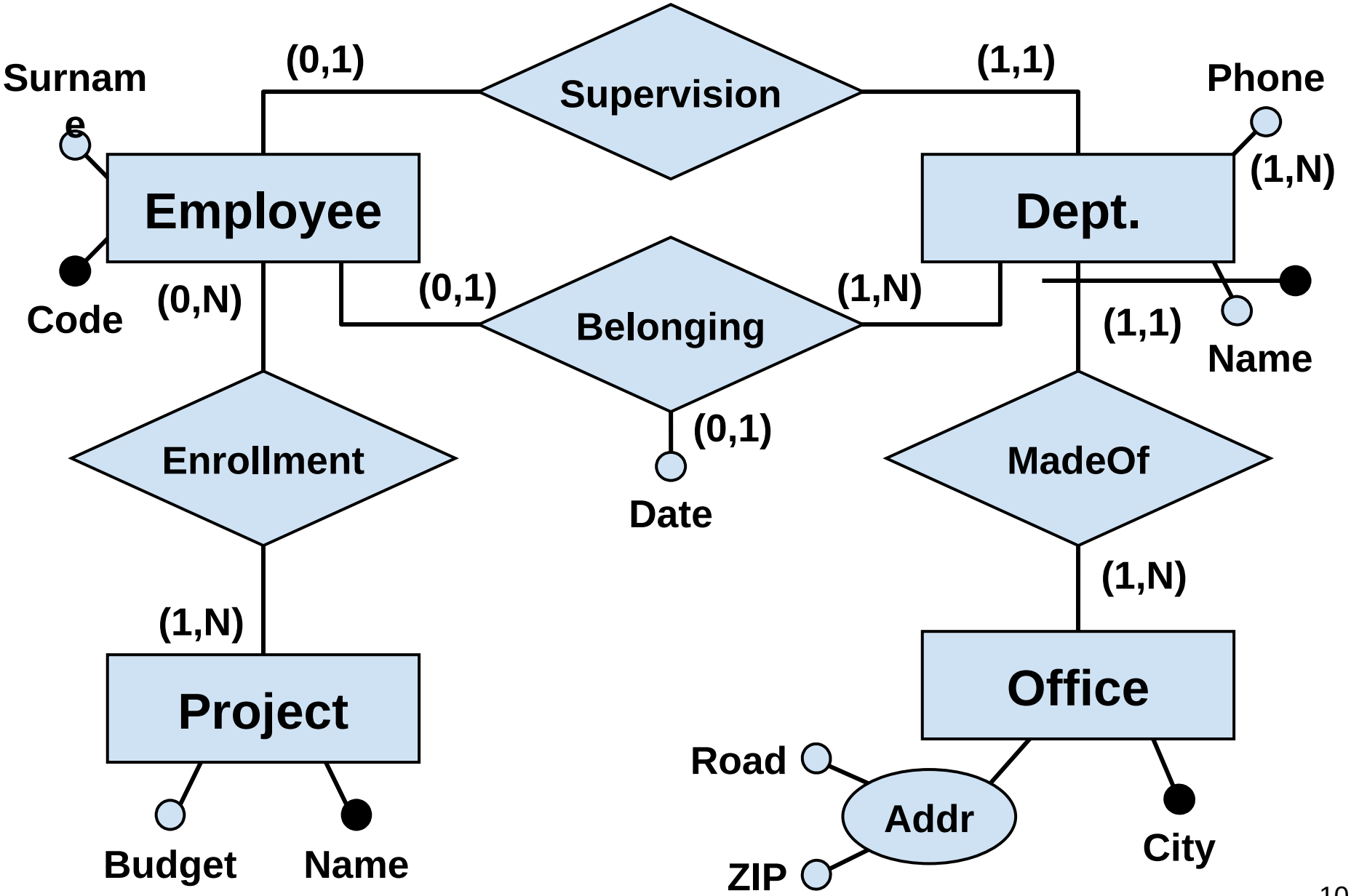
Performances?

- In order to optimize the results we need to analyze performances at this level
- However:
  - Performances cannot be evaluated precisely on a conceptual schema!

Performances Approximation

- Let us consider some performance "indicators": parameters on which the performances depend upon
  - **Space**: number of stored instances expected
  - **Time**: number of instances (of entities and relationships) visited during an operation

The Input E-R Schema



Size Table

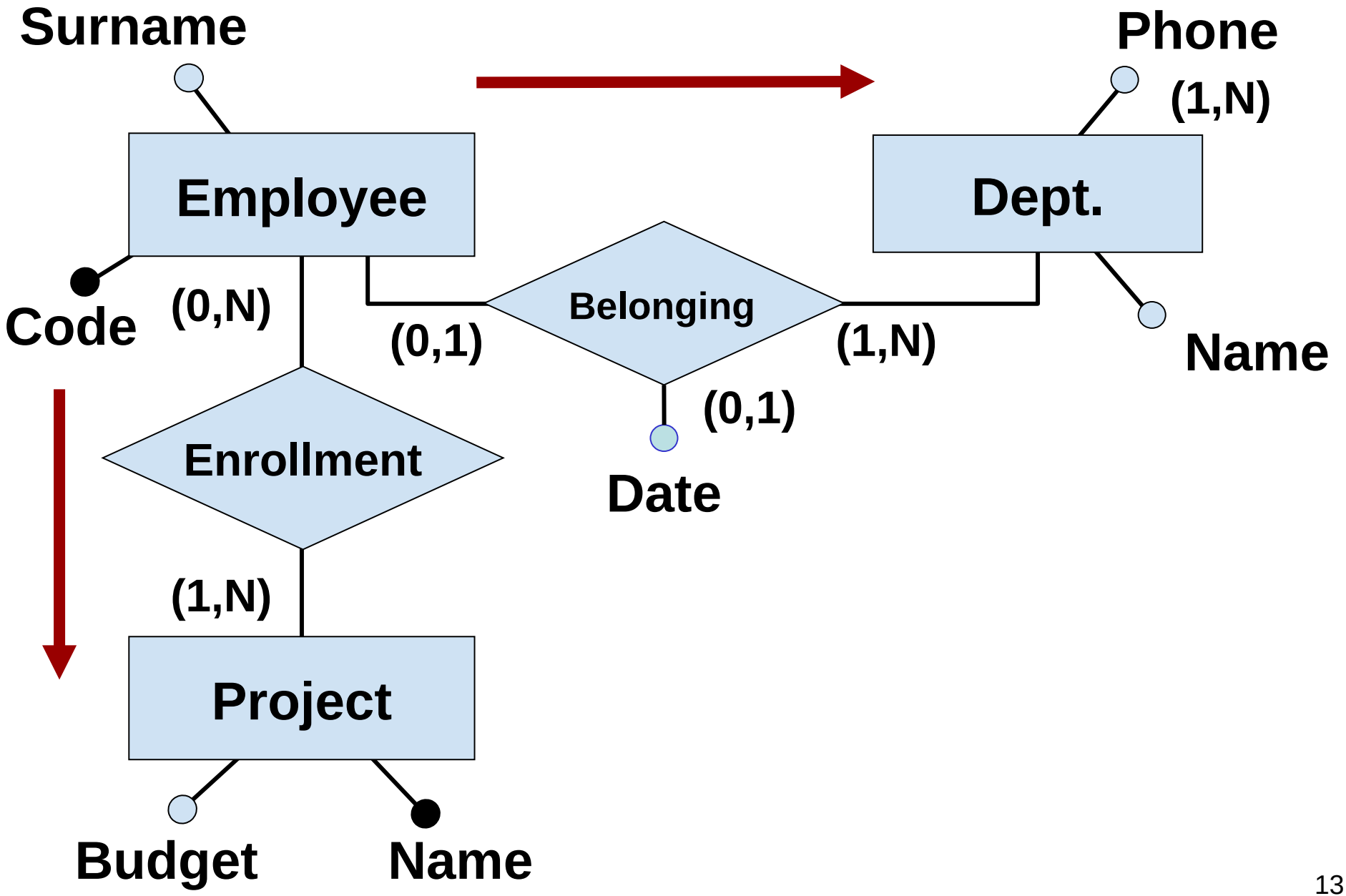
Name	Type	Size
Office	E	10
Dept.	E	80
Employee	E	2’000
Project	E	500
MadeOf	R	80
Belonging	R	1’900
Supervision	R	80
Enrollment	R	6’000

Example of Cost Evaluation

- Operation:
  - Return all the data of a given employee, the data of the department where he belongs, and the data of the projects he works on

- Build an **access table** following the **navigation schema**

E-R Schema: Follow the Path



Access Table

Name	Type	Accesses Number	Accesses Type	Accesses Order
Dept.	E	1	R	3
Employee	E	1	R	1
Project	E	3	R	5
Belonging	R	1	R	2
Enrollment	R	3	R	4

Restructuring Activities

- Redundancies Analysis
- Generalizations deletion
- Partitioning/grouping of entities and relationships
- Identifying the primary keys

Restructuring Activities

- **Redundancies Analysis**
- Generalizations deletion
- Partitioning/grouping of entities and relationships
- Identifying the primary keys

Redundancy Analysis

- A redundancy in a E-R schema is an information that is relevant but can be derived from others
- In this phase we have to decide whether we have to keep, remove or create new redundancies

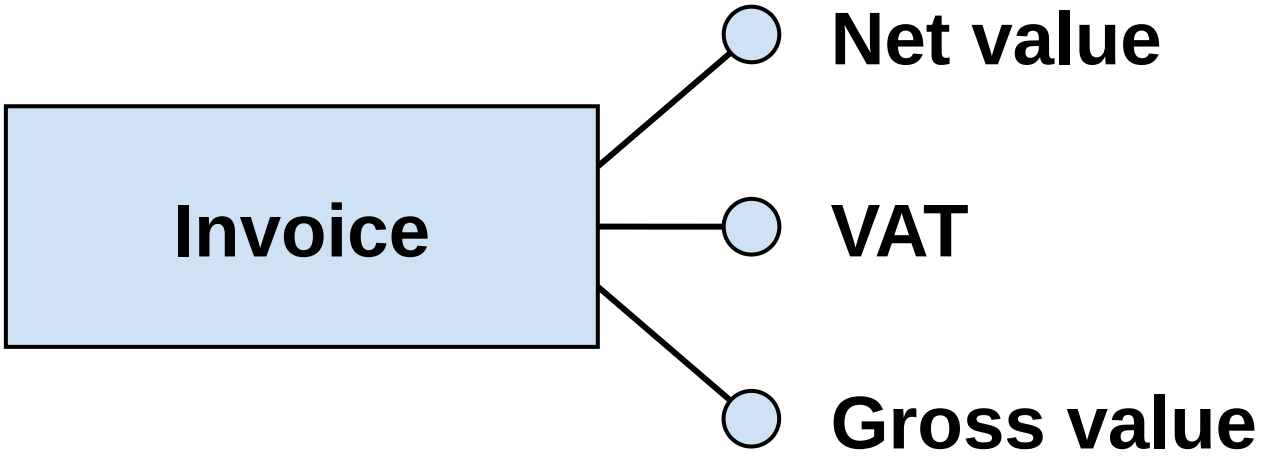
Redundancies

- Pros
  - They simplify queries
- Cons
  - Updates take more time
  - Storage size is increased

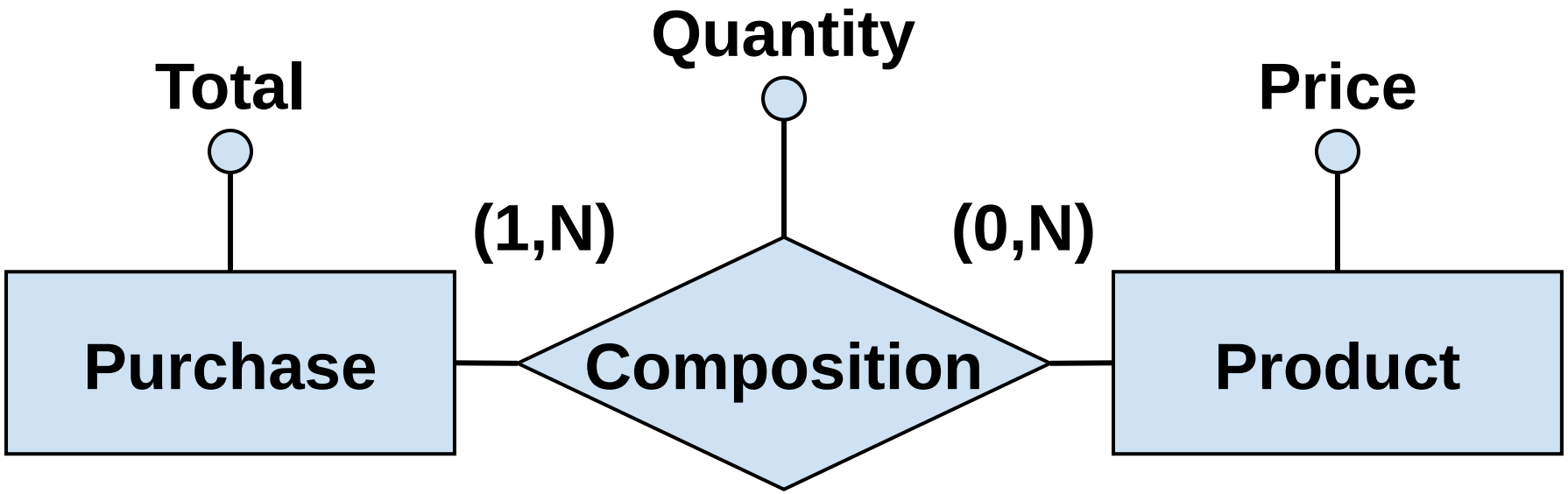
E-R: Types of Redundancies

- Derivable attributes:
  - From other attributes within the same entity (or relationship)
  - From attributes of other entities (or relationships)
- Relationships derivable from the composition of different relationships (generally speaking: cycles of relationships)

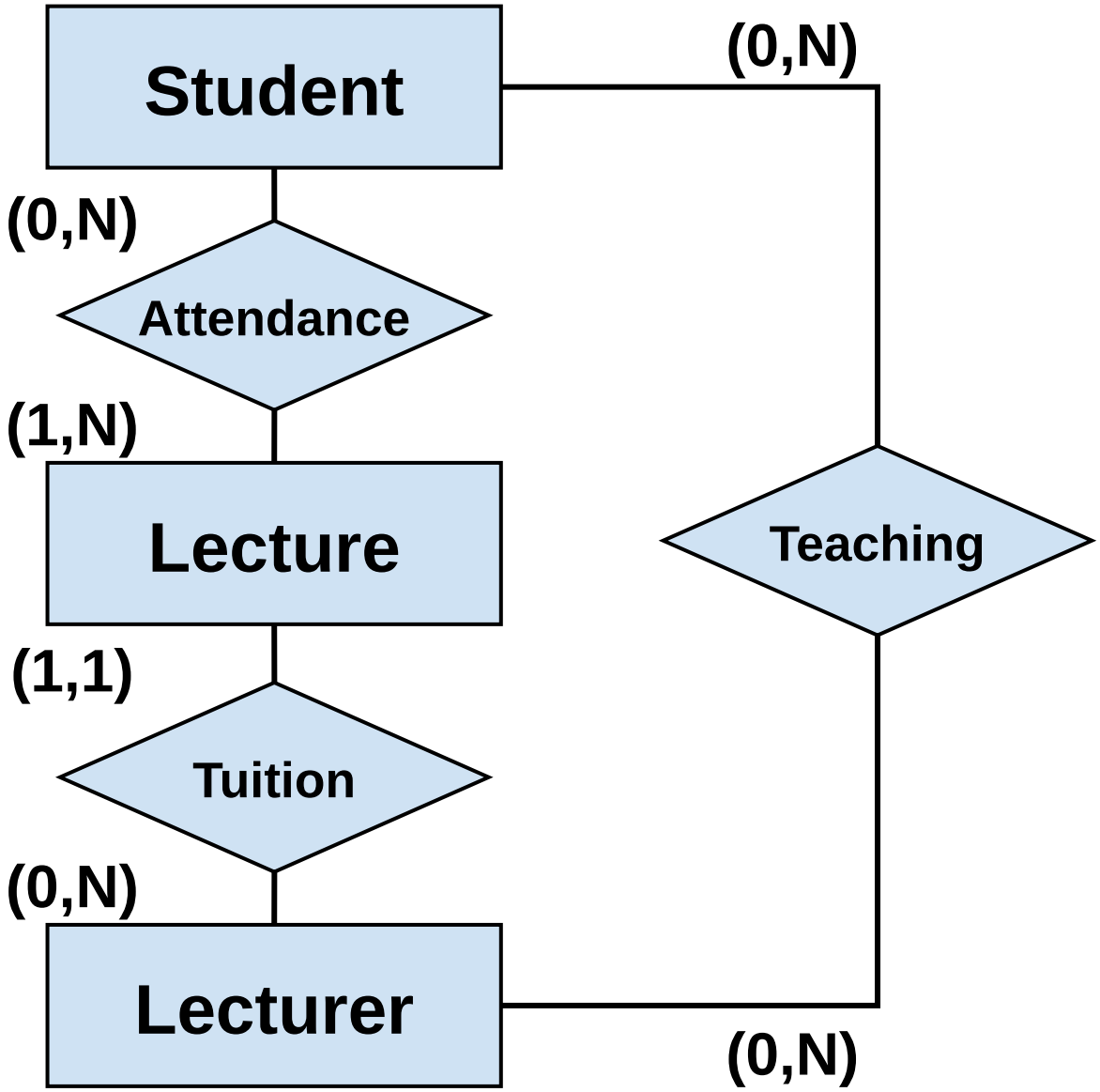
Derivable Attributes: within the Entity



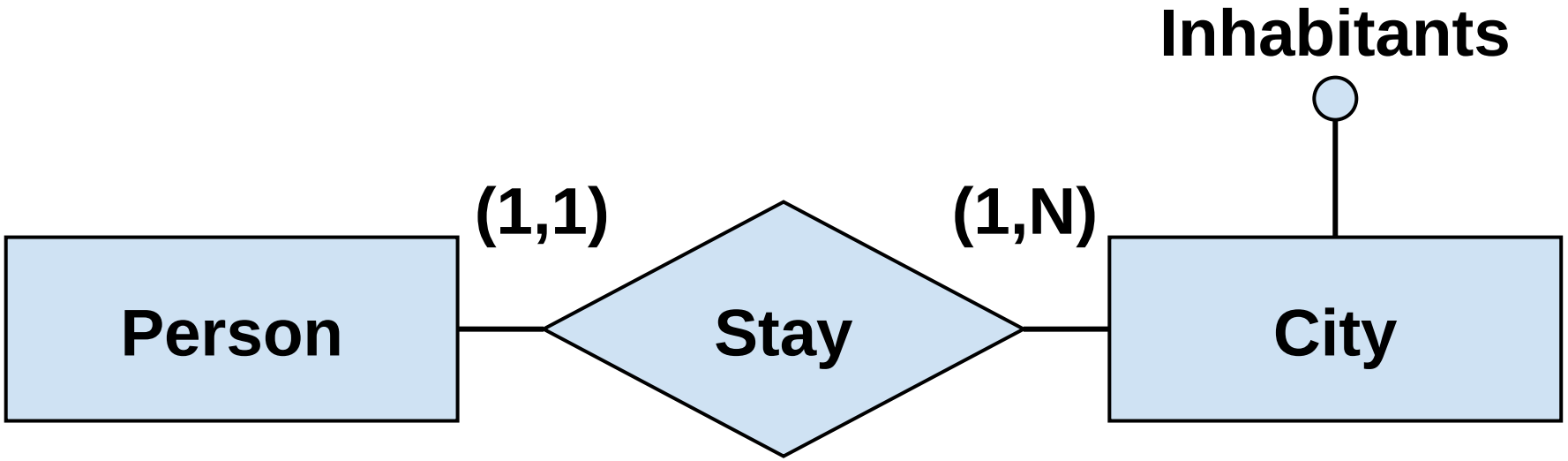
Derivable Attributes: from other Entities



Redundancy from a Cycle



Redundancy Analysis



- Attributes derivable from occurrence counting operations

Redundancy Analysis: Operations

Name	Type	Size
City	E	200
Person	E	1’000’000
Stay	R	1’000’000

- **Operation #1**: store a new person with his city of staying (500 times a day)
- **Operation #2**: print all the cities’ data, including number of inhabitants (2 times a day)

Operation Details: With Redundancy

Operation #1			
Name	**Type**	**Accesses Number**	**Accesses Type**
Person	E	1	W
Stay	R	1	W
City	E	1	R
City	E	1	W

Operation #2			
Name	**Type**	**Accesses Number**	**Accesses Type**
City	E	1	R

Operation Details: Without Redundancy

Operation #1			
Name	**Type**	**Accesses Number**	**Accesses Type**
Person	E	1	W
Stay	R	1	W

Operation #2			
Name	**Type**	**Accesses Number**	**Accesses Type**
City	E	1	R
Stay	R	5’000	R

Costs: With Redundancy

- Costs:
  - Operation #1: 1’500 writes + 500 reads per day
  - Operation #2: negligible
- We count write accesses as doubles
  - A total amount of **3’500 (read equivalent) operations per day**

Costs: Without Redundancy

- Costs:
  - Operation #1: 1’000 writes per day
  - Operation #2: 10’000 reads per day
- Again, writings costs double
  - A total amount of **12’000 (equivalent read) operations per day**

Restructuring Activities

- Redundancies Analysis
- **Generalizations deletion**
- Partitioning/grouping of entities and relationships
- Identifying the primary keys

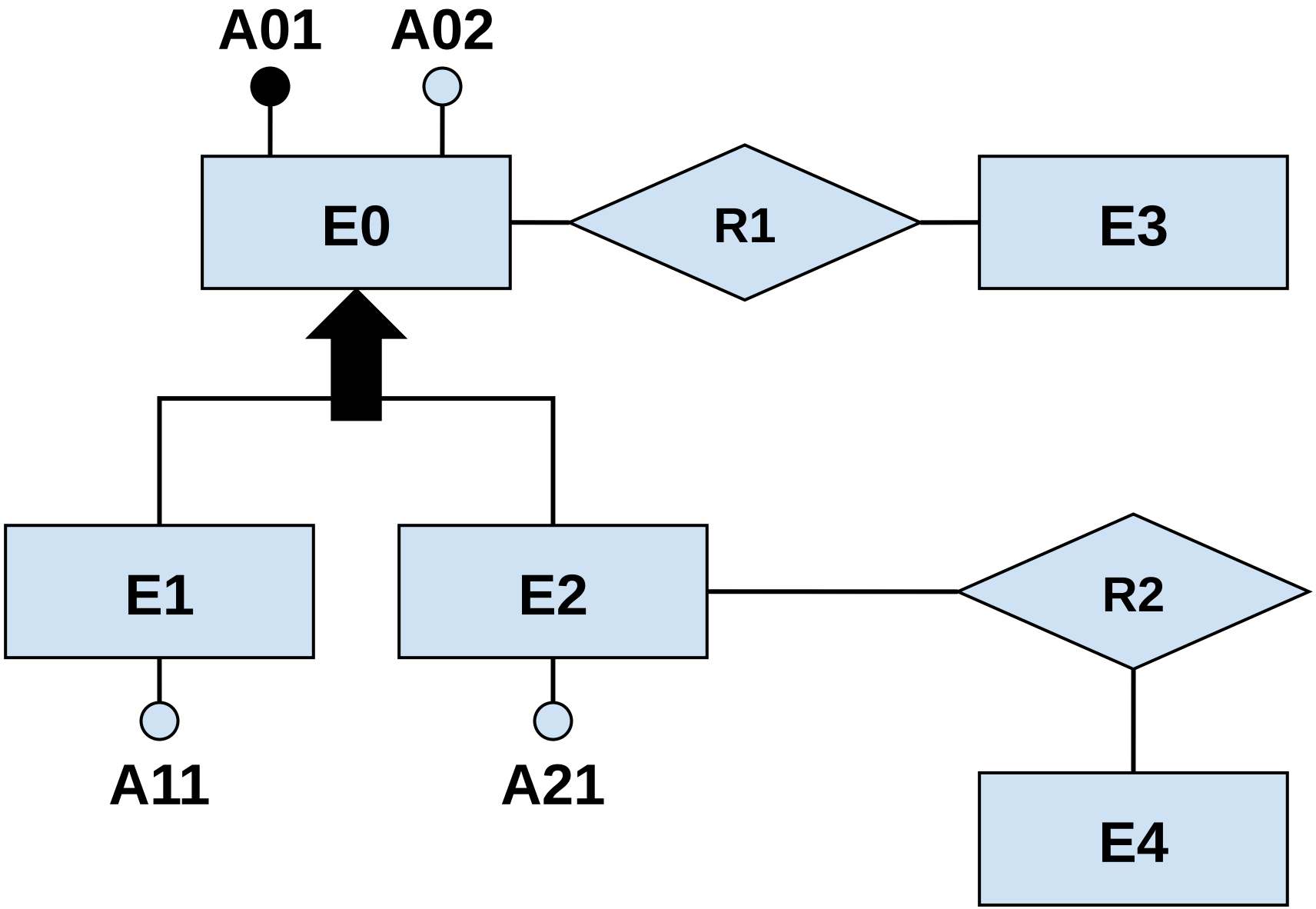
Hierarchies Deletion

- The relational data model does not directly support generalizations, they cannot be directly represented
  - … while entities and relationships are directly representable
- We therefore remove hierarchies replacing them with entities and relationships

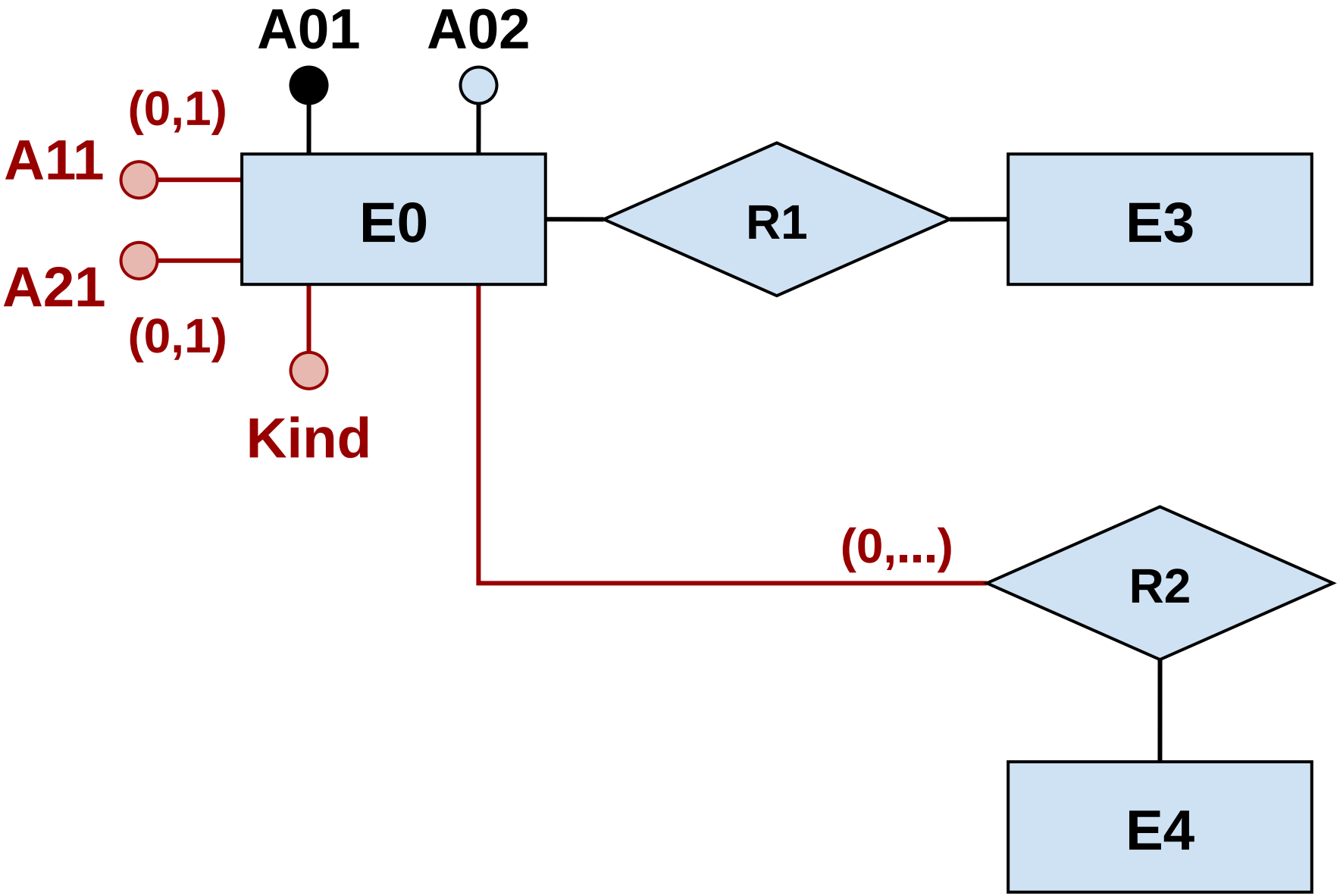
Three Possible Solutions

1. Embedding the children of the generalization into the parent
2. Embedding the parent of the generalization into the children
3. Replacing the generalization with a relationship

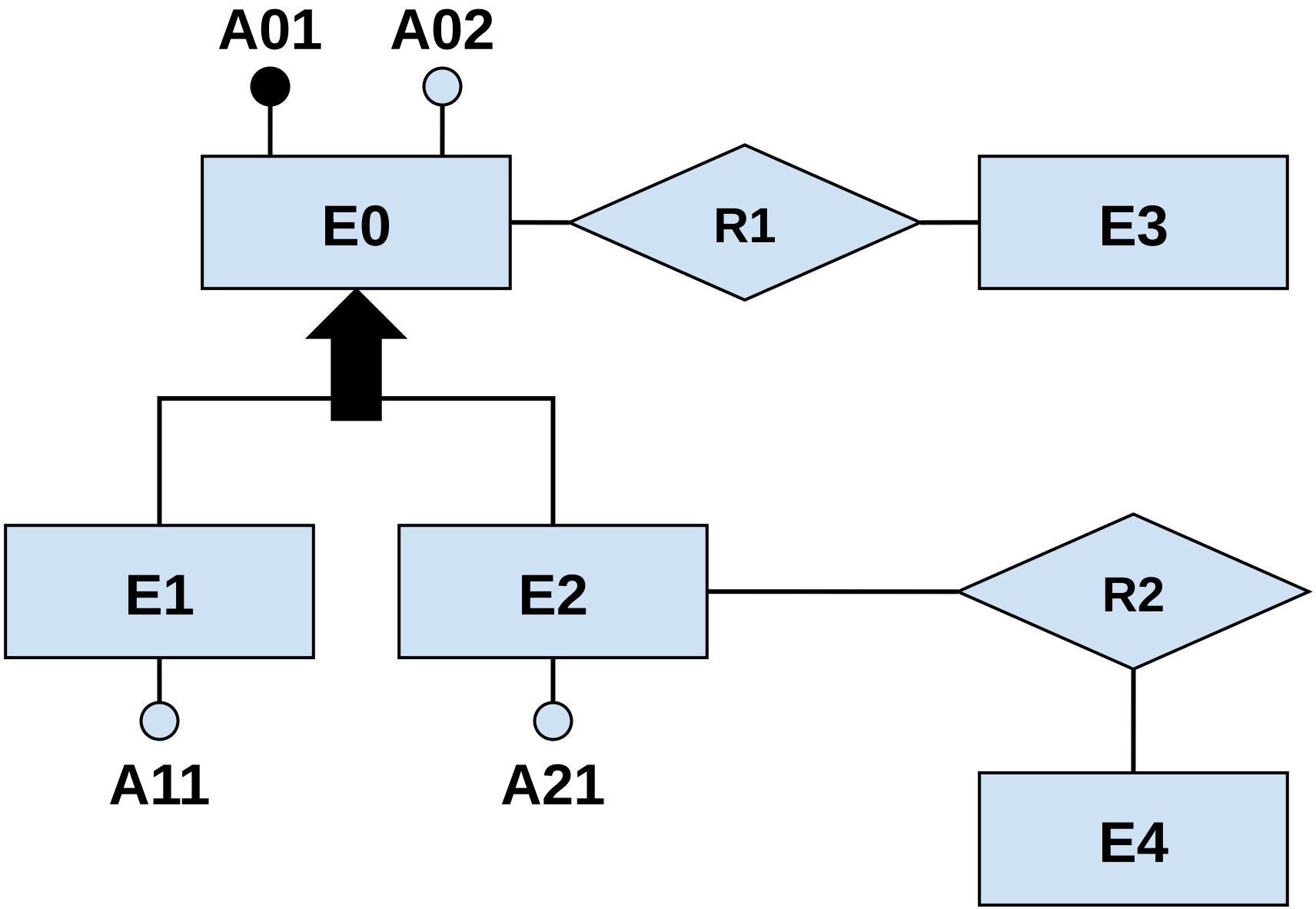
Hierarchies Deletion: an Example



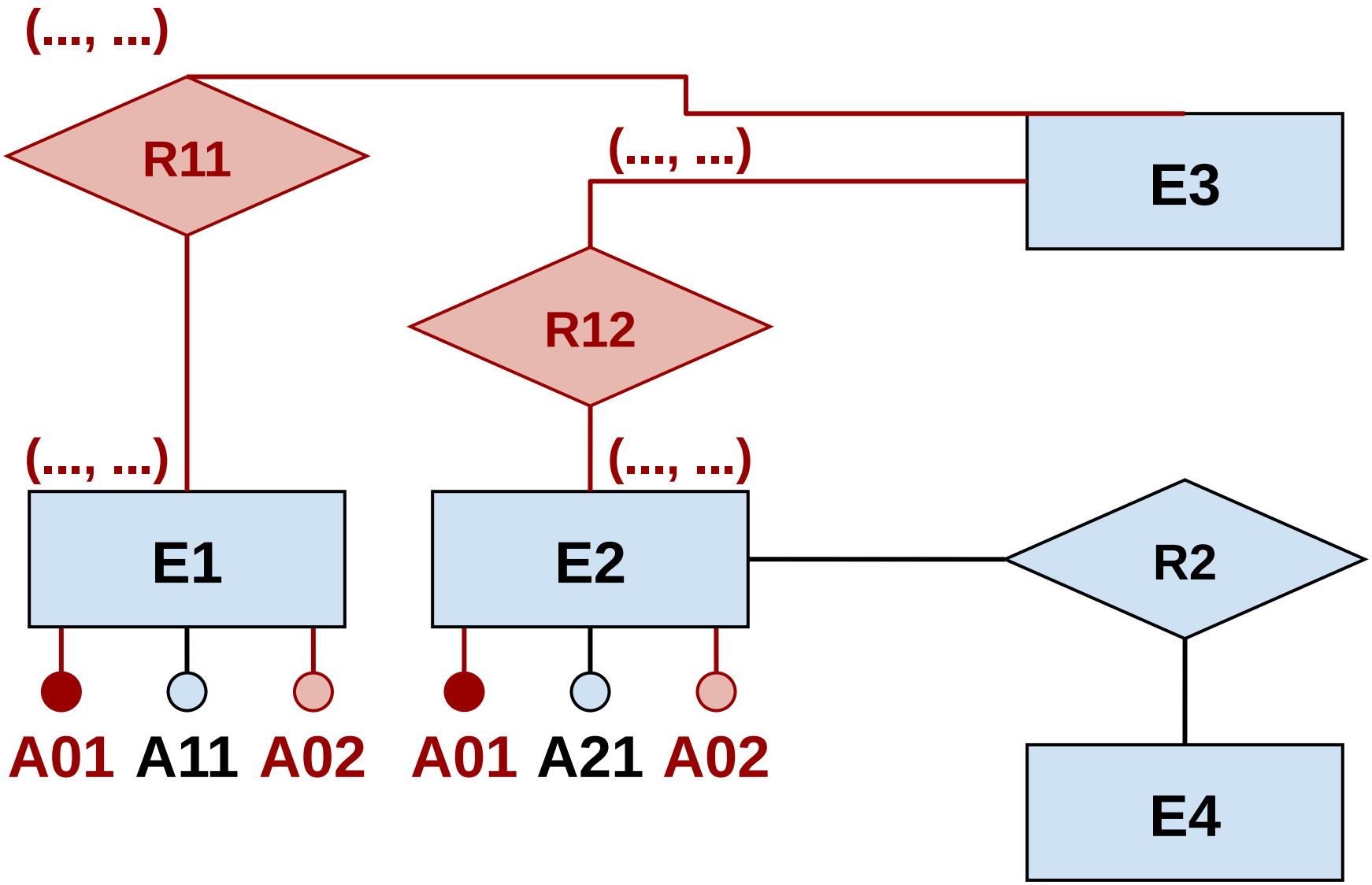
Hierarchies Deletion (1): Parent Embedding



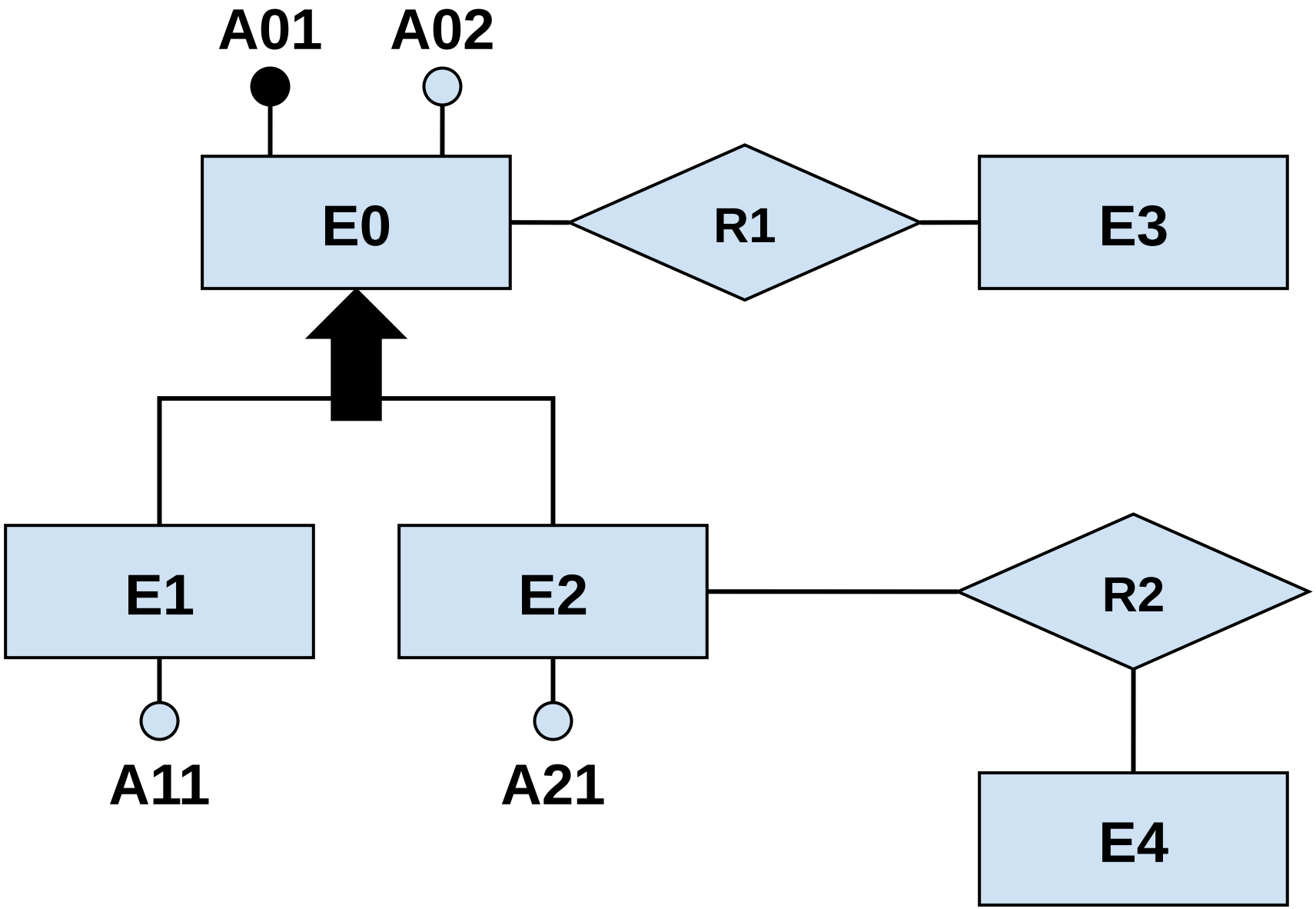
Hierarchies Deletion: an Example



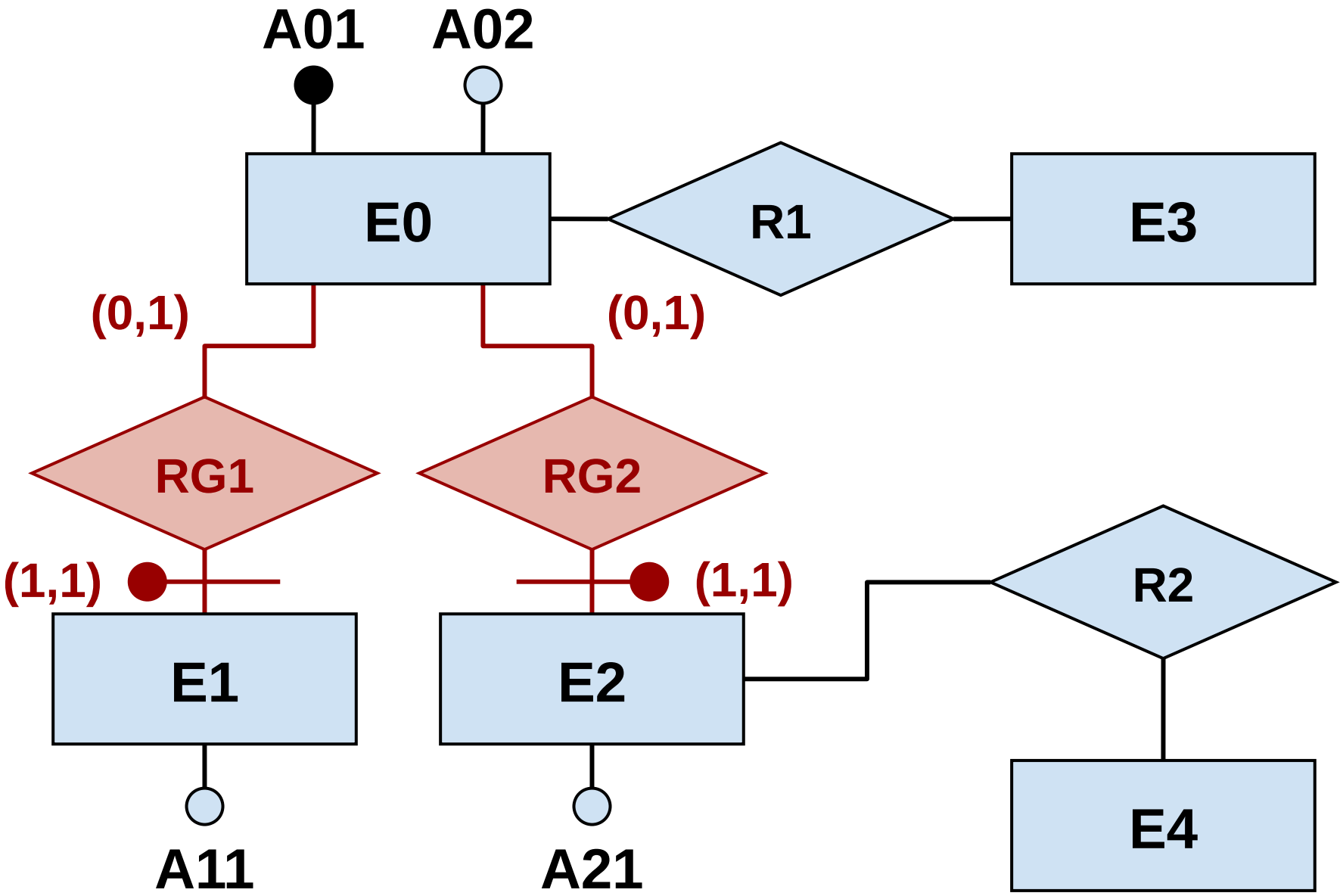
Hierarchies Deletion (2): Children Embedding



Hierarchies Deletion: an Example



Hierarchies Deletion (3): Using Relationship



Observations (1)

- We can decide between those three alternatives depending on the sizes and access tables (considering only the number of accesses is not enough)
- We now provide some general rules that can be followed

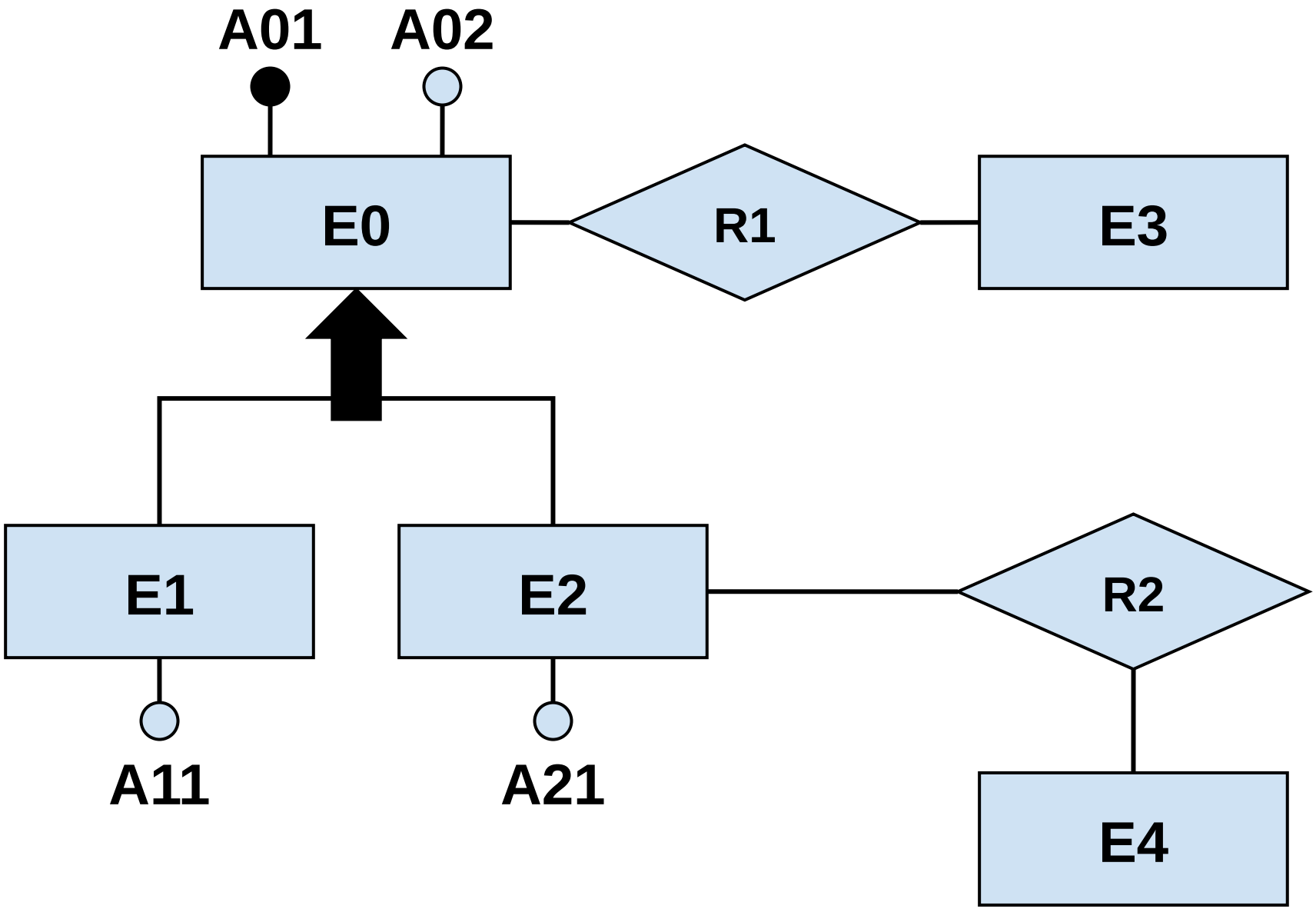
Observations (2)

The three solutions (1, 2 and 3):

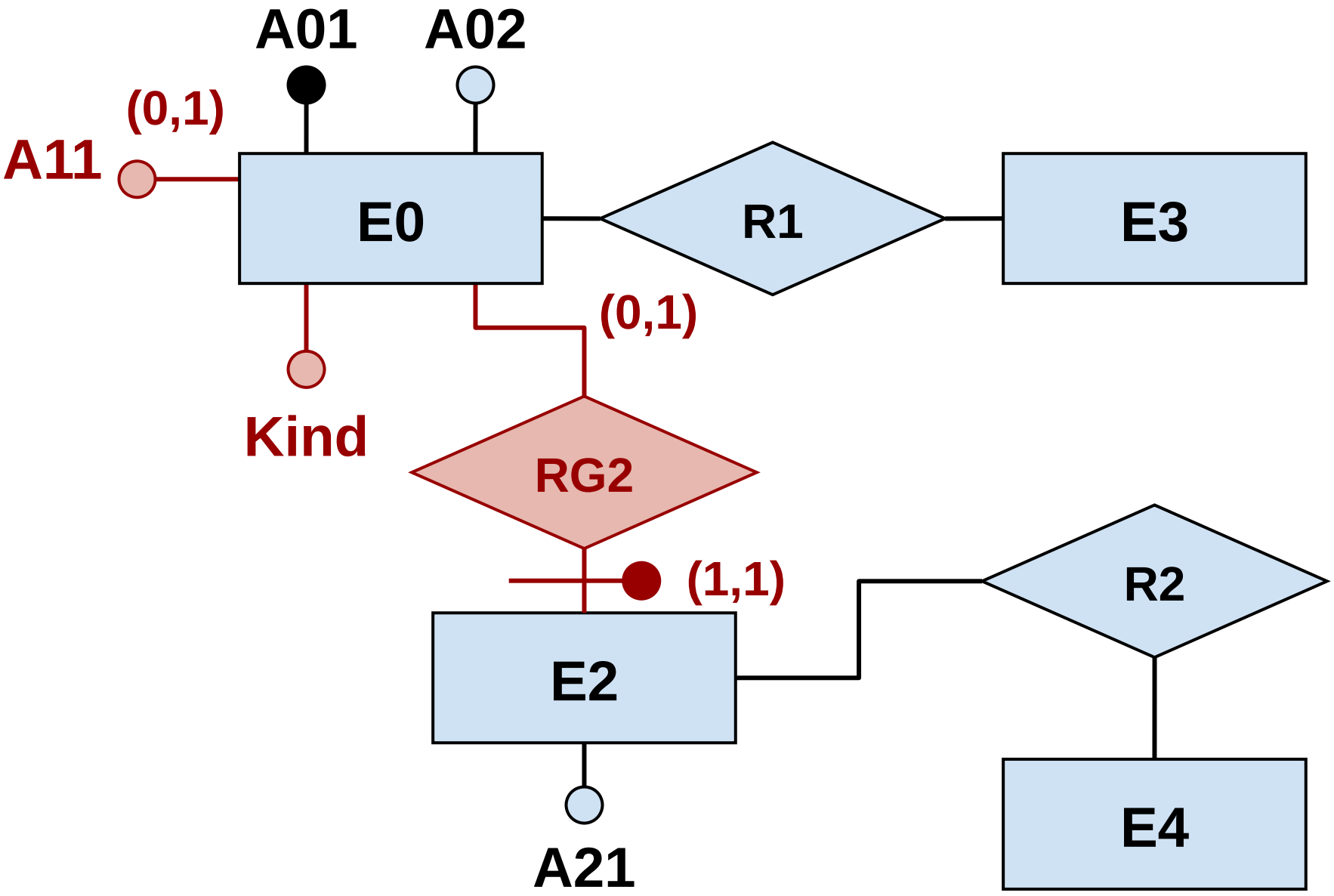
1. **Parent Embedding** should be used when children and father are accessed at the same time
2. **Children Embedding** should be used when children are accessed independently from one another
3. **Using Relationship** should be used when children are accessed independently from the father

- We can also apply “hybrid” solutions, especially in hierarchies with more levels

Hierarchies Deletion: an Example



Hierarchies Deletion (4): Hybrid Solutions



Restructuring Activities

- Redundancies Analysis
- Generalizations deletion
- **Partitioning/grouping of entities and relationships**
- Identifying the primary keys

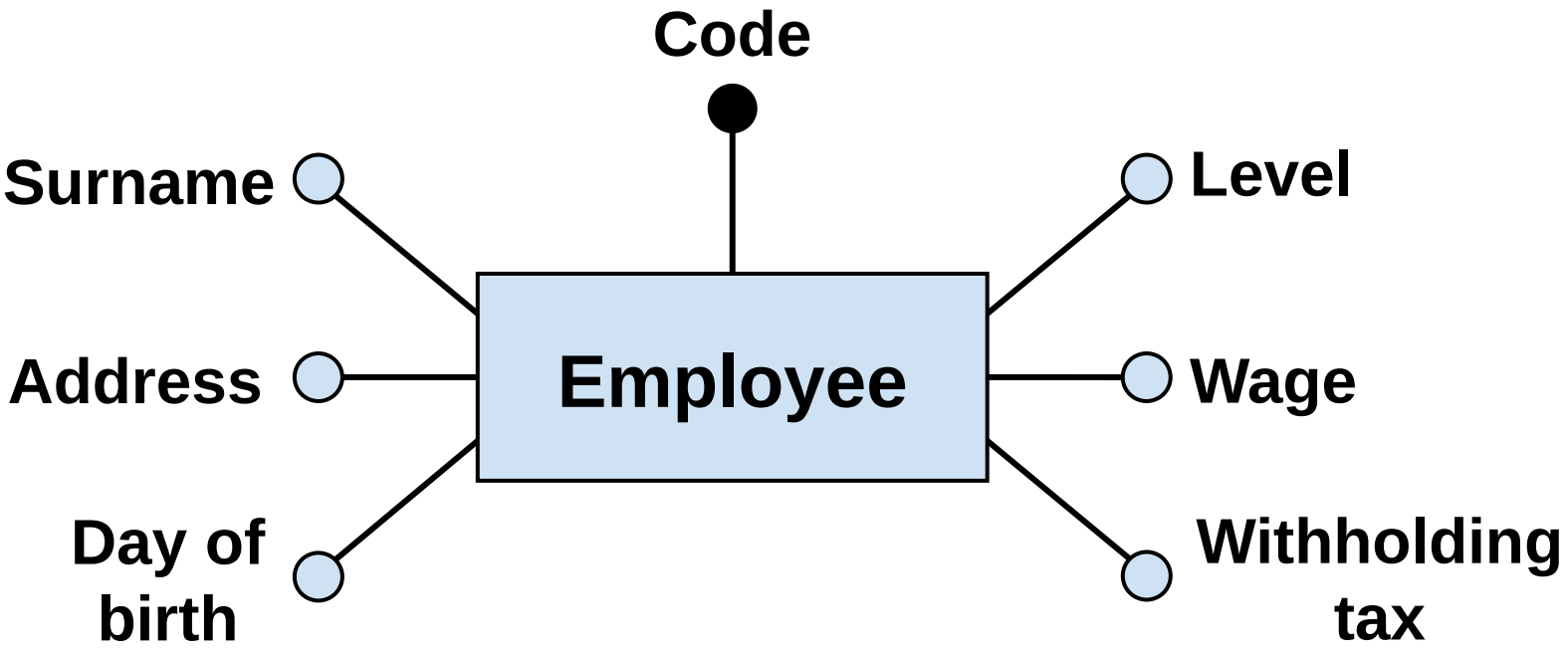
Attributes Restructuring

- Restructuring can provide more efficient operations by simply reducing the number of accesses:
  - Attributes accessed separately are splitted
  - Attributes accessed on the same time are grouped, even when they belong to different entities/relationships

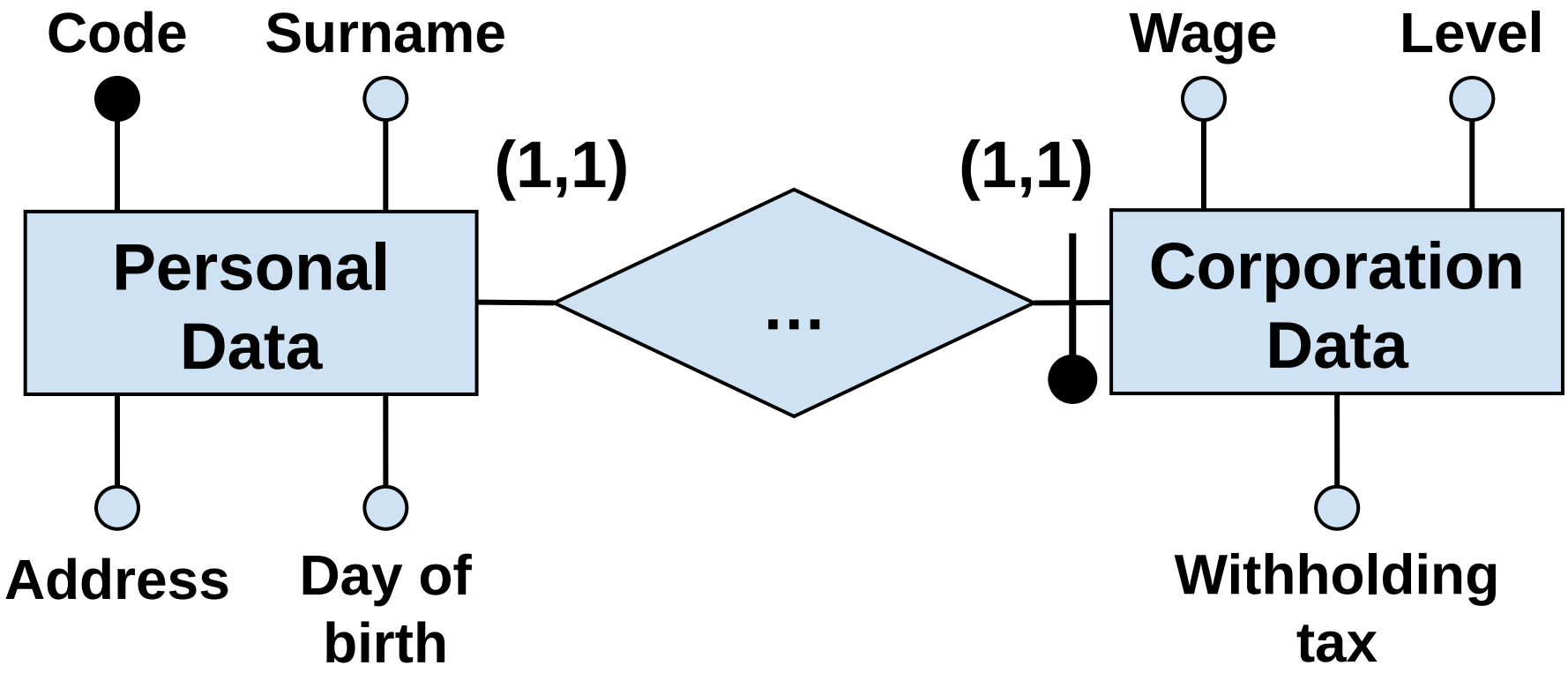
Restructuring: Main Cases

1. Entity vertical partitioning
2. Restructuring multi-valued attributes
3. Grouping of entities/relationships
4. Relationship horizontal partitioning

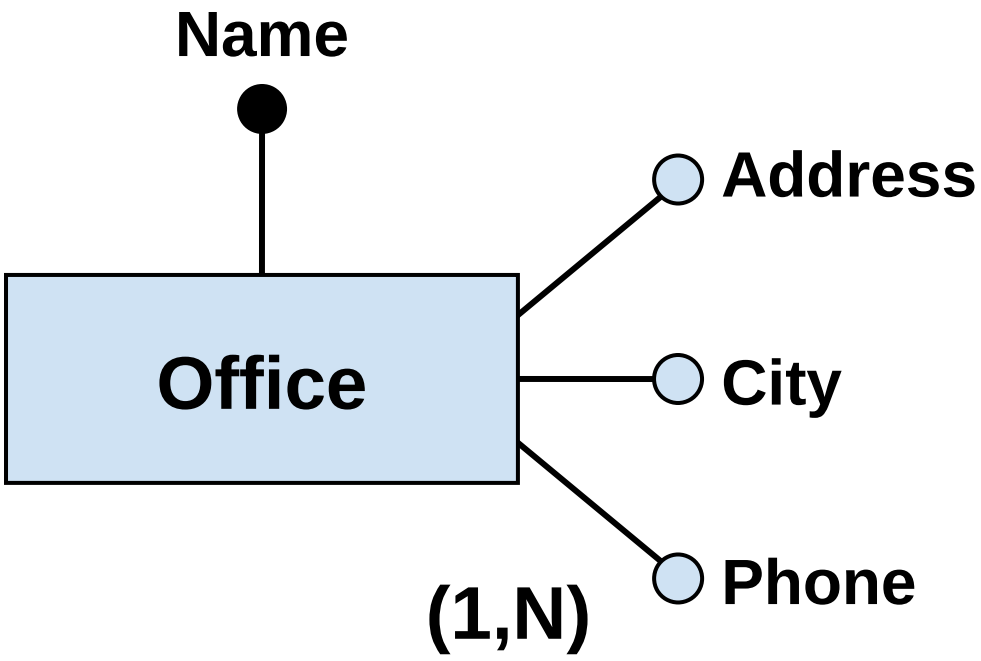
Entity Vertical Partitioning (1a)



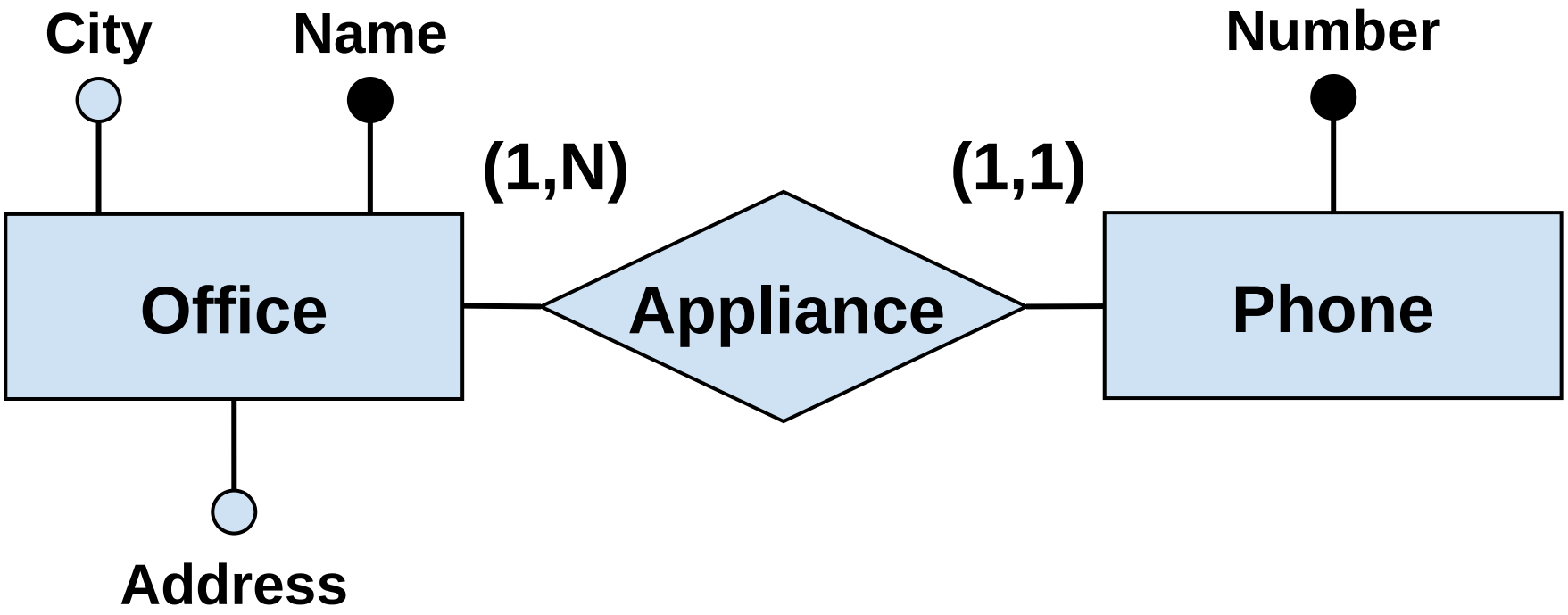
Entity Vertical Partitioning (1b)



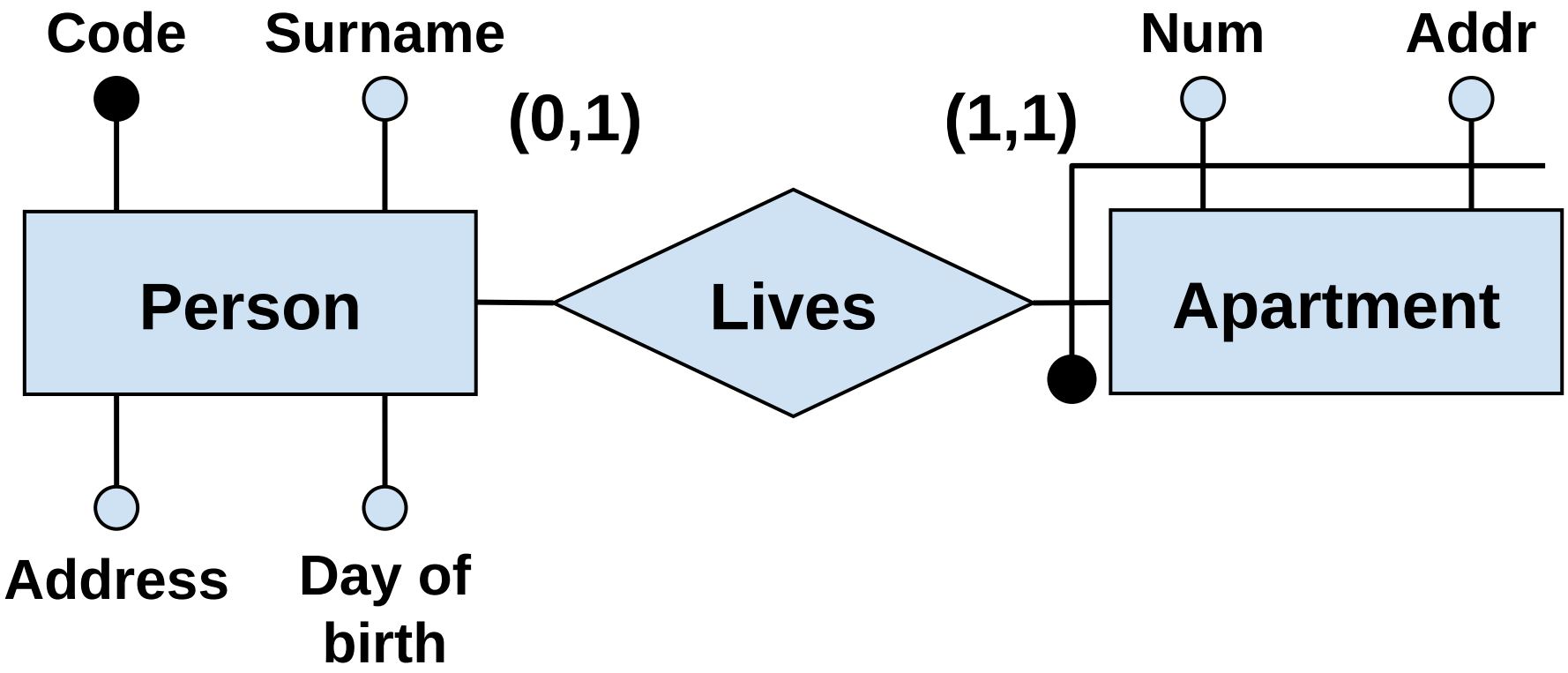
Restructuring Multi-valued Attributes (2a)



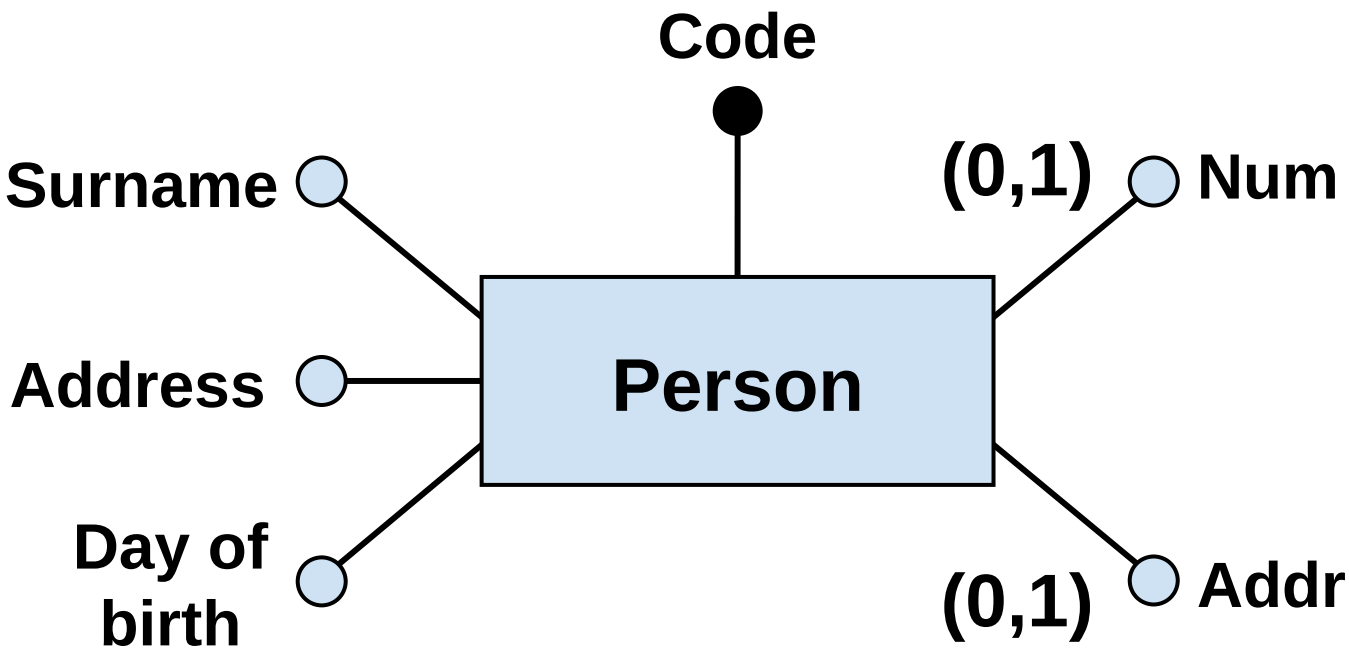
Restructuring Multi-valued Attributes (2b)



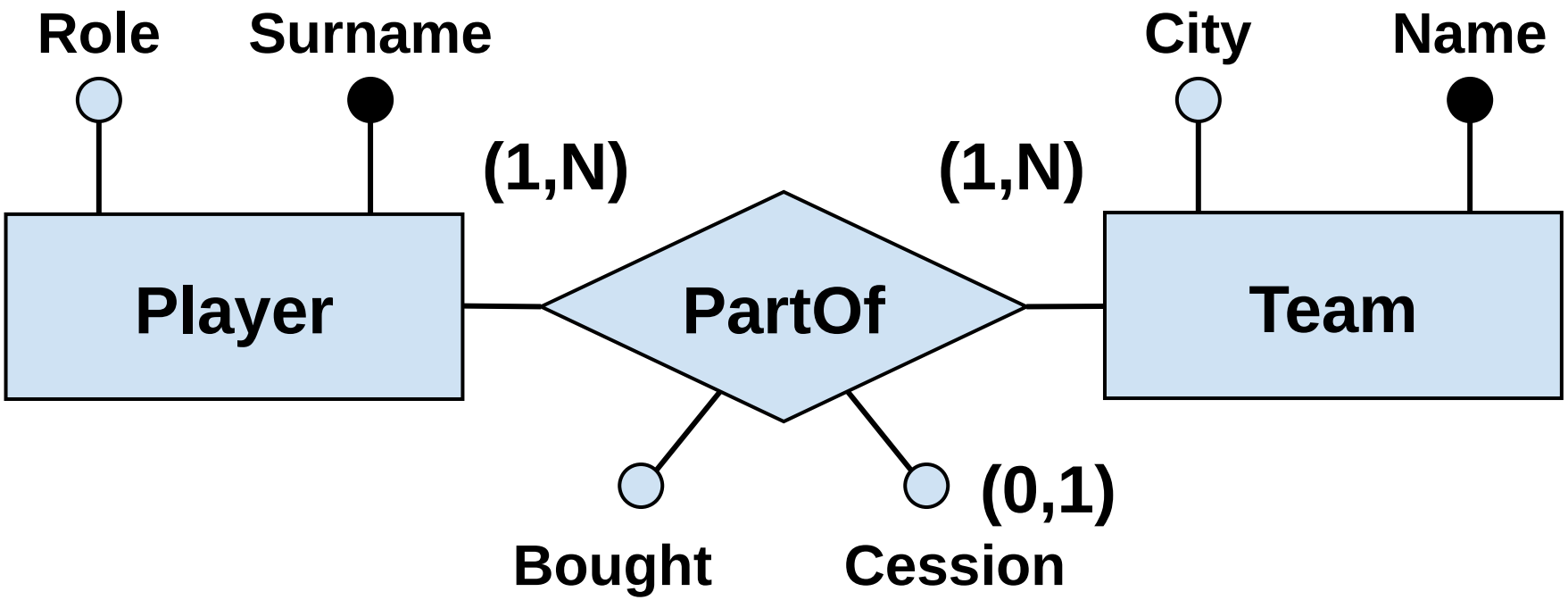
Grouping of Entities (3a)



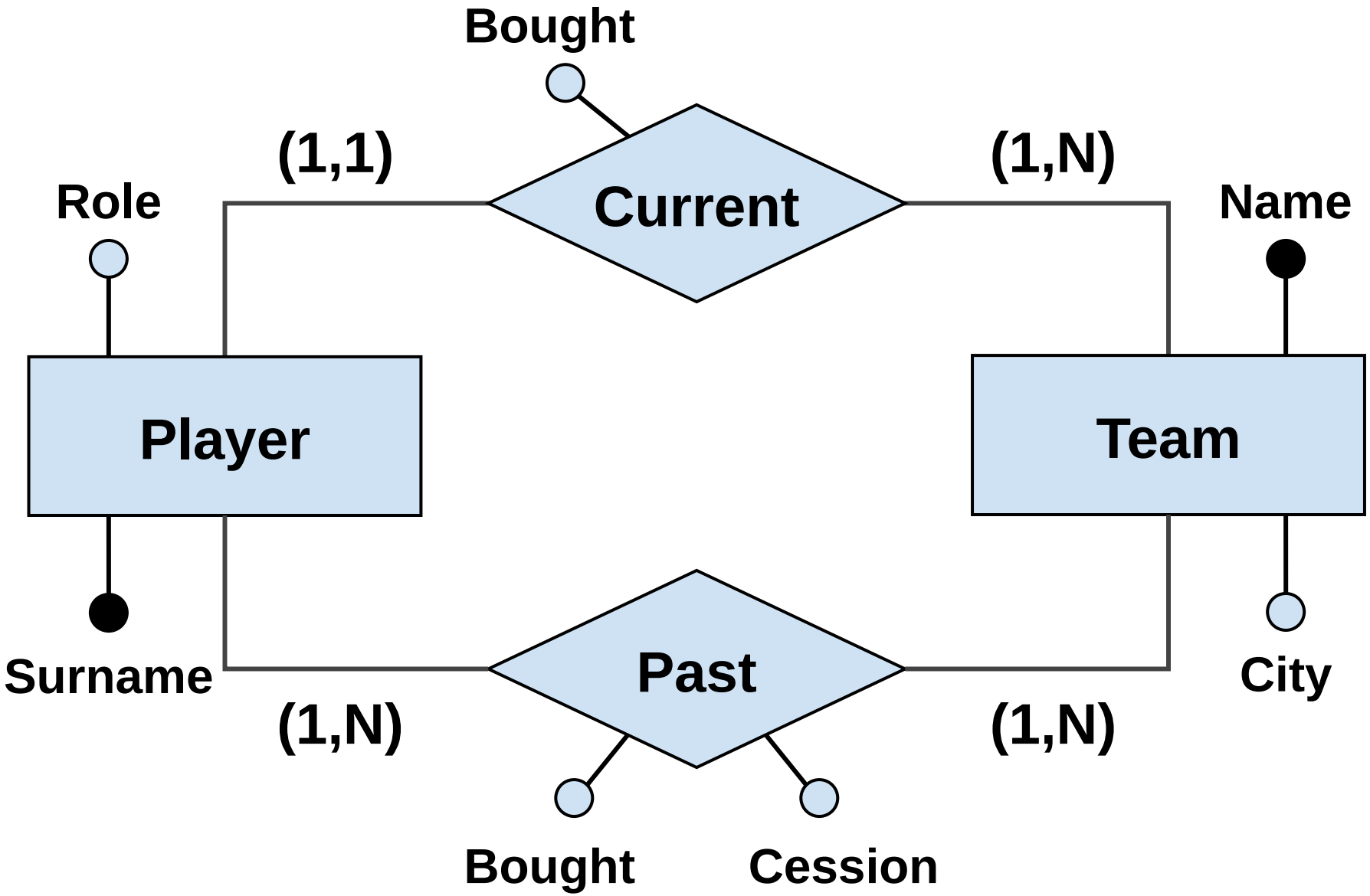
Grouping of Entities (3b)



Relationship Horizontal Partitioning (4a)



Relationship Horizontal Partitioning (4b)



Restructuring Activities

- Redundancies Analysis
- Generalizations deletion
- Partitioning/grouping of entities and relationships
- **Identifying the primary keys**

Identifying Primary Keys

- A mandatory operation for the translation into a relational model
- Criteria
  - Compulsory information
  - Simplicity
  - Used within the most frequent/relevant operation

Primary Keys: New Attributes

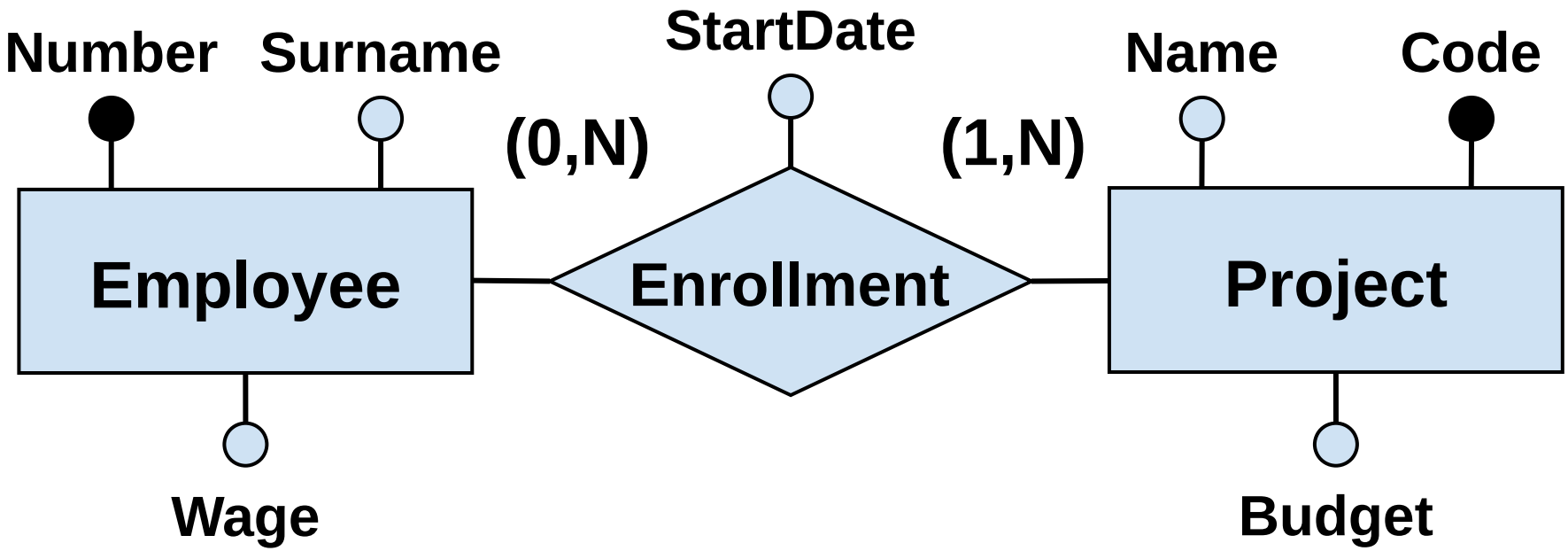
- What if none of the aforementioned conditions are met?
- New attributes are introduced using specifically generated **codes**

Translation to the Relational Model

Rules of thumb:

- **Entities** become “tables”, which schema corresponds to the entities’ attributes
- **Relationships** become “tables”, their schema correspond to the entities’ attributes, plus the foreign identifiers for the involved entities

Many-to-Many Relationships



EMPLOYEE(Number, Surname, Wage)

PROJECT(Code, Name, Budget)

ENROLLMENT(Number, Code, StartDate)

Many-to-Many Relationships

```
EMPLOYEE(Number, Surname, Wage)
PROJECT(Code, Name, Budget)
ENROLLMENT(Number, Code, StartDate)
```

- Referential Integrity Constraint between:
  - `Number` in `ENROLLMENT` and `EMPLOYEE`'s key
  - `Code` in `ENROLLMENT` and `PROJECT`'s key

Foreign Keys: use more Expressive Names

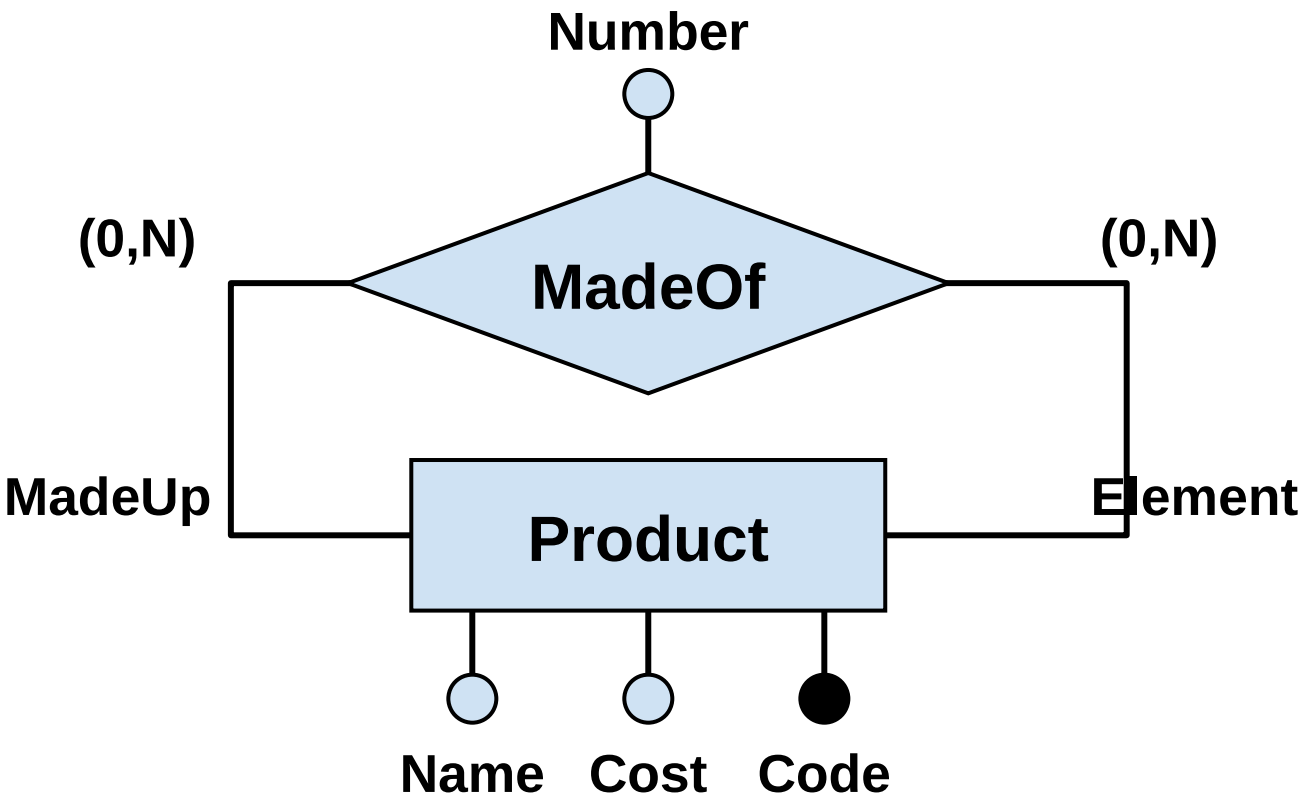
EMPLOYEE(Number, Surname, Wage)
PROJECT(Code, Name, Budget)
ENROLLMENT(Number, Code, StartDate)

ENROLLMENT(Employee, Project, StartDate)

Please Note

- Such translation does not keep into account the many-to-many relationships' minimal cardinality
- Even if we could use very uncommon and complex CHECKs

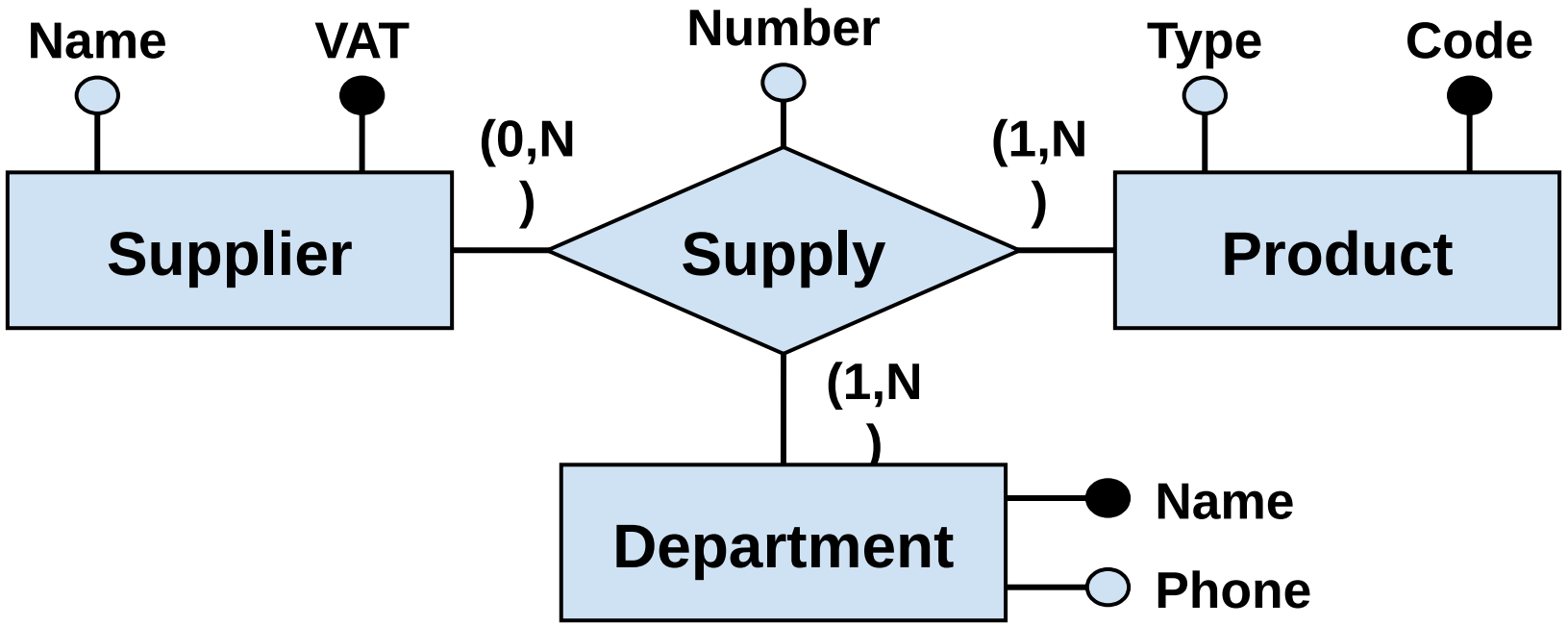
Recursive Relationships



PRODUCT(Code, Name, Cost)

MADEOF(MadeUp, Element, Number)

N-ary Relationships



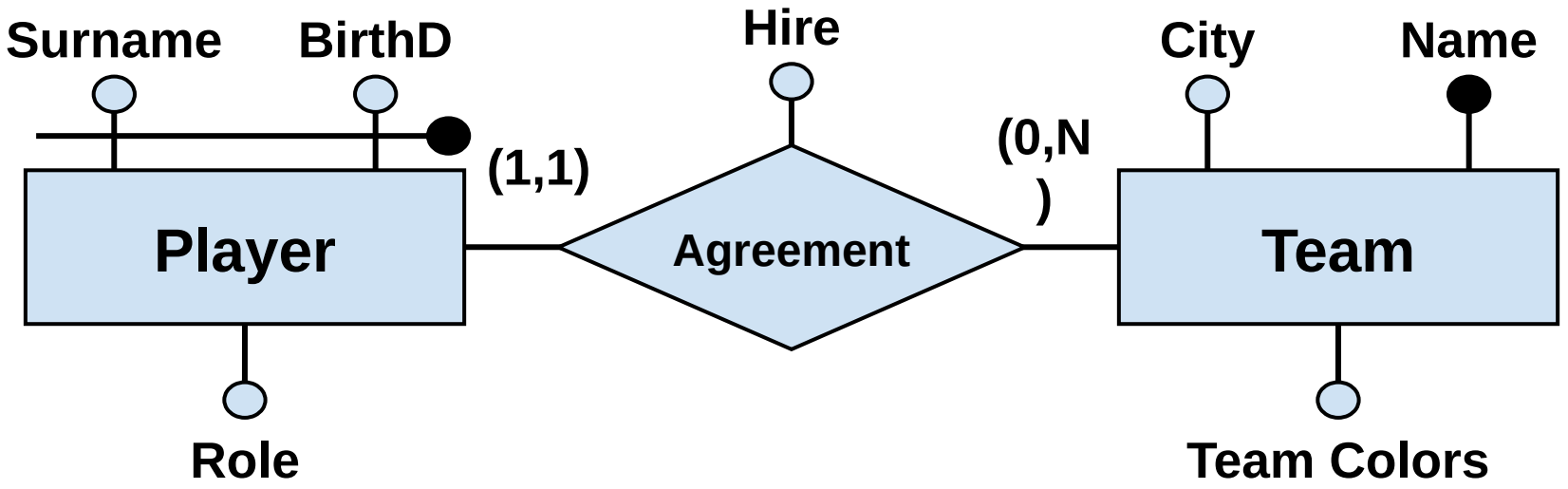
SUPPLIER(VAT, Name)

PRODUCT(Code, Type)

DEPARTMENT(Name, Phone)

SUPPLY(Supplier, Product, Department, Number)

One-to-Many Relationship



PLAYER(Surname, BirthD, Role)

AGREEMENT(SurnameP, BirthDP, Team, Hire)

TEAM(Name, City, TeamColors)

- **Is it correct?**

A less Redundant Solution

PLAYER(Surname, BirthD, Role)
AGREEMENT(SurnameP, BirthDP, Team, Hire)
TEAM(Name, City, TeamColors)

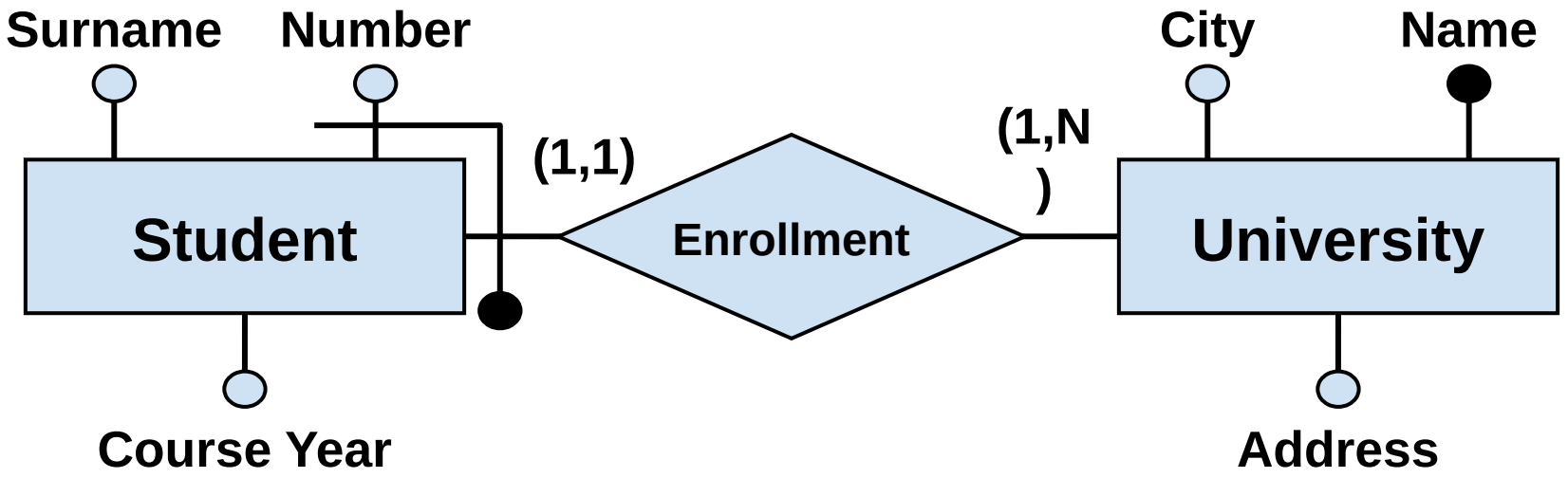
PLAYER(Surname, BirthD, Team, Role, Hire)
TEAM(Name, City, TeamColors)

- Referential Integrity Constraint between **Team** in **PLAYER** and **TEAM's key**
- If the relationship's minimal cardinality is 0, then **Team** in **PLAYER** must allow **NULL** values

Please Note

- Such translation could represent the case when 0 is the minimal cardinality and 1 is the maximum one:
  - 0 : NULL values allowed
  - 1 : NULL values **NOT** allowed

Entity with External Identifier

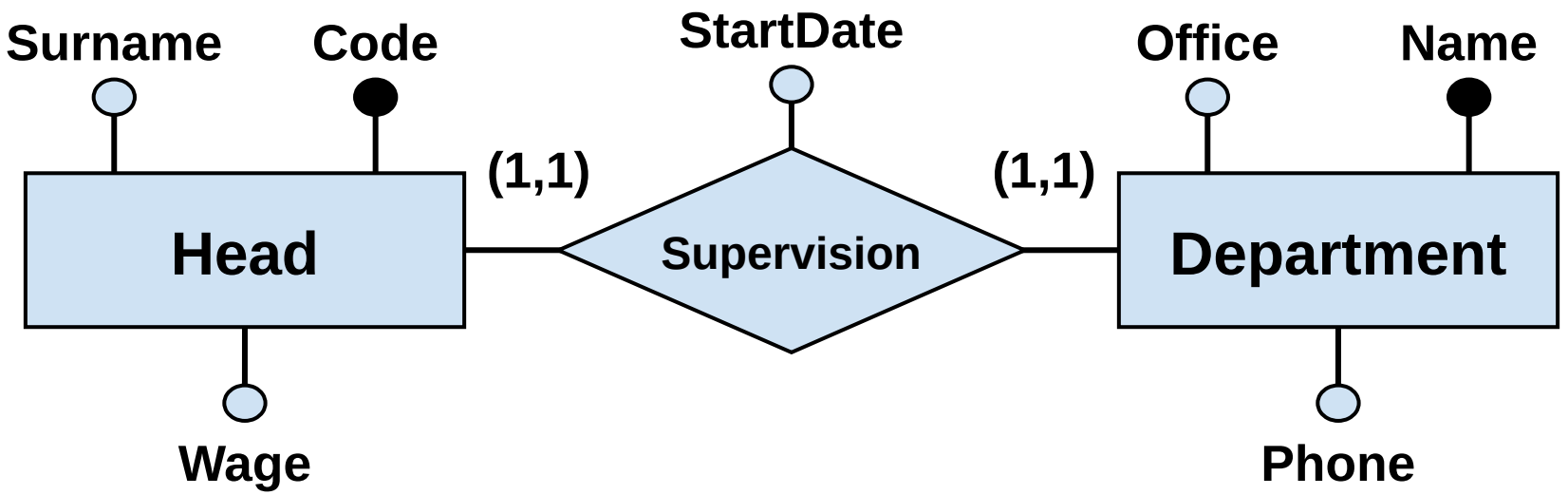


STUDENT(Number, University, Surname, CourseYear)

UNIVERSITY(Name, City, Addr)

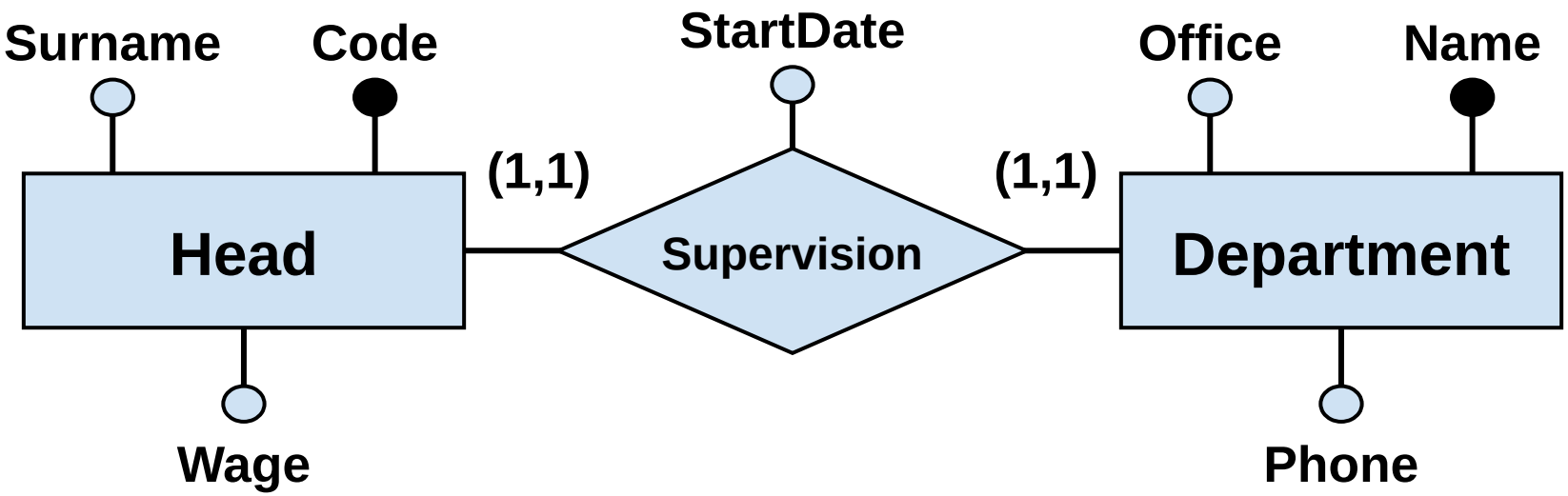
- **Constraint**: each student is enrolled to only one university

One-to-One Relationship (1)



- Different options:
  - Merge an entity with a relationship (either side)
  - Merge everything together, maybe?

One-to-One Relationship (2)



HEAD(Code, Surname, Wage, Department, StartDate)

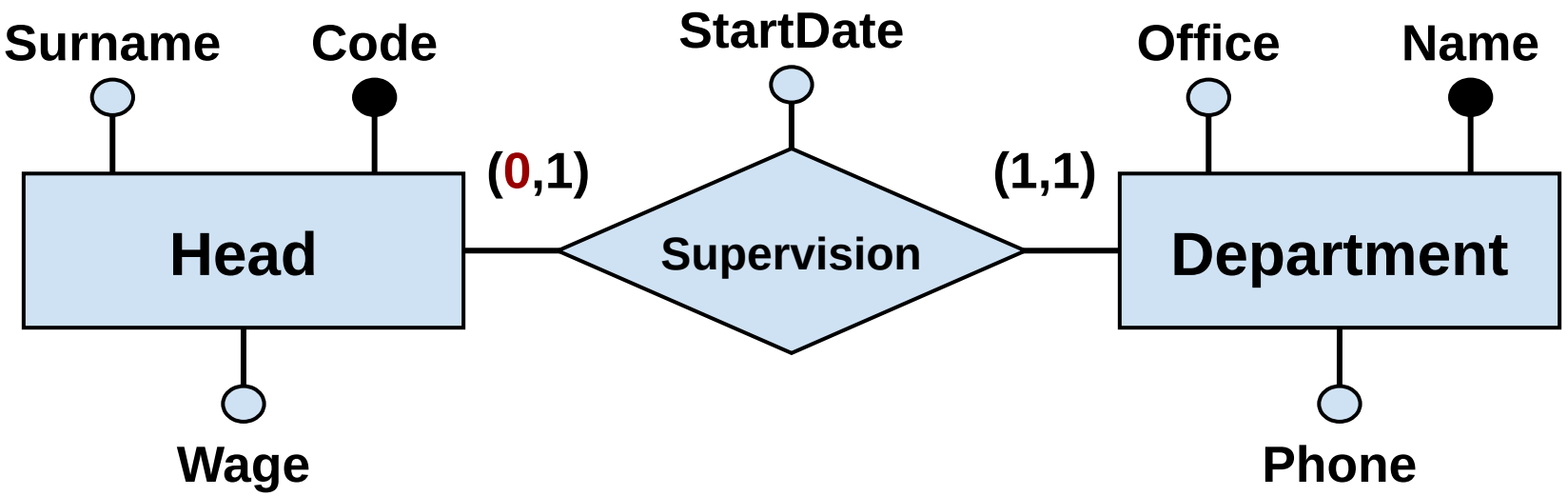
DEPARTMENT(Name, Office, Phone)

OR

HEAD(Code, Surname, Wage)

DEPARTMENT(Name, Office, Phone, Head, StartDate)

One-to-One: a Special Case

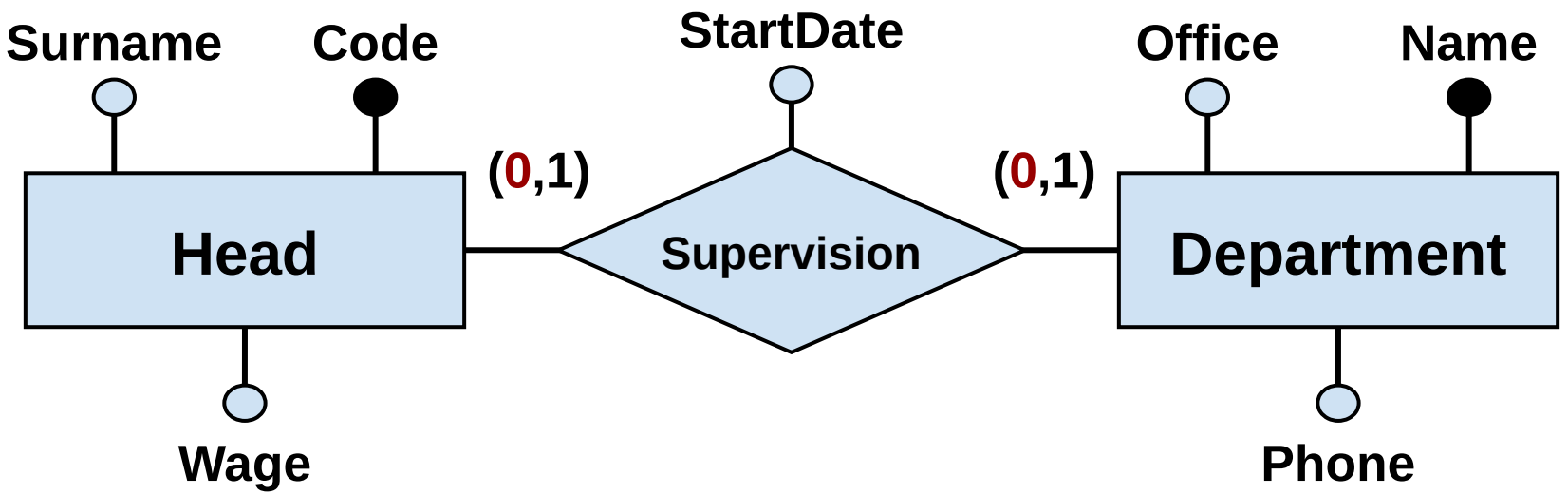


HEAD(Code, Surname, Wage)

DEPARTMENT(Name, Office, Phone, Head, StartDate)

- Referential Integrity Constraint
- No NULLs allowed

One-to-One: Another Special Case



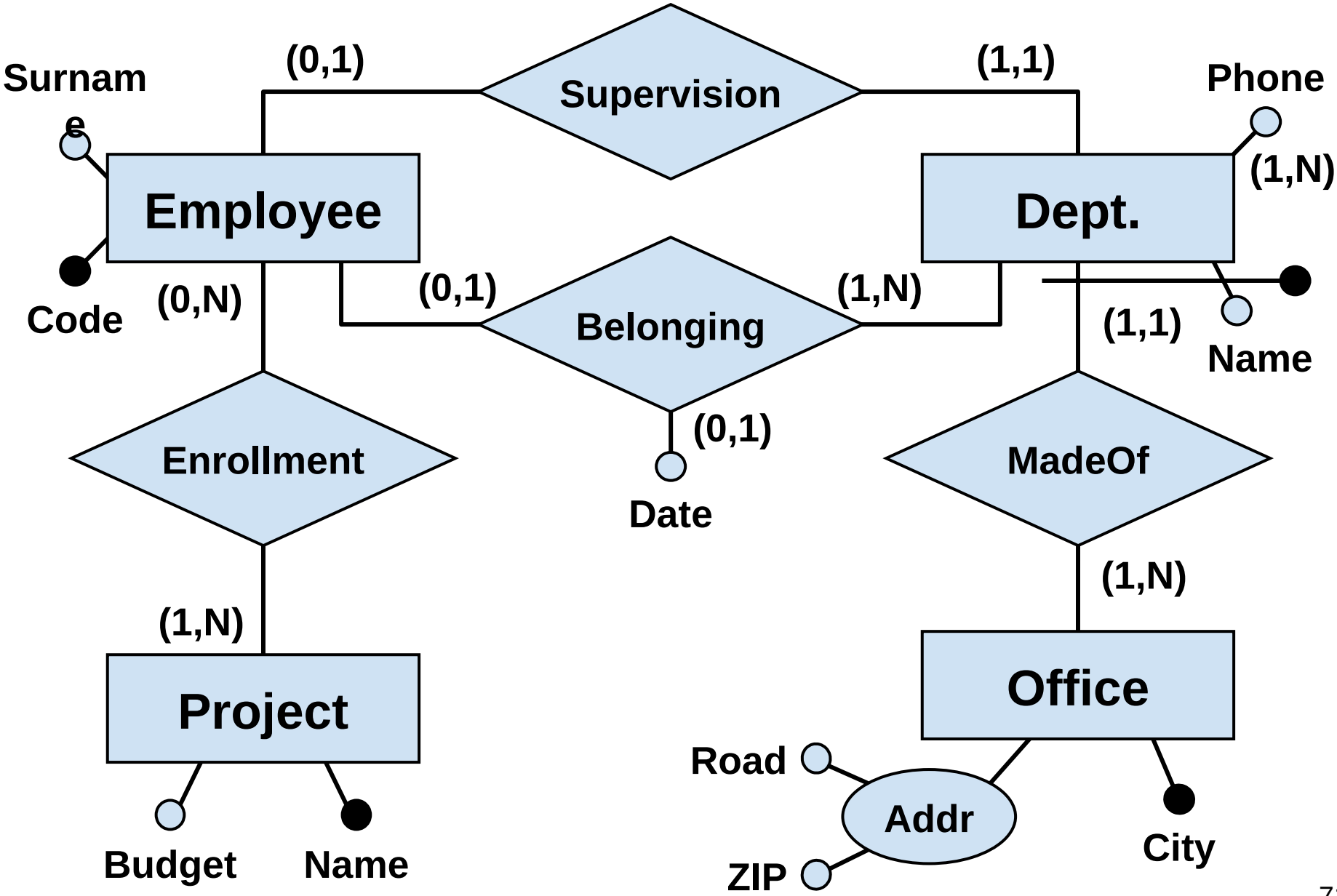
HEAD(Code, Surname, Wage)

DEPARTMENT(Name, Office, Phone)

SUPERVISION(Head, Department, StartDate)

- Two Referential Integrity Constraints
- No NULLs allowed

Final Schema (1)



Final Logical Schema

EMPLOYEE(Code, Surname, Dept,
Office, Date*)

DEPT(Name, City, Phone, Head)

OFFICE(City, Road, Zip)

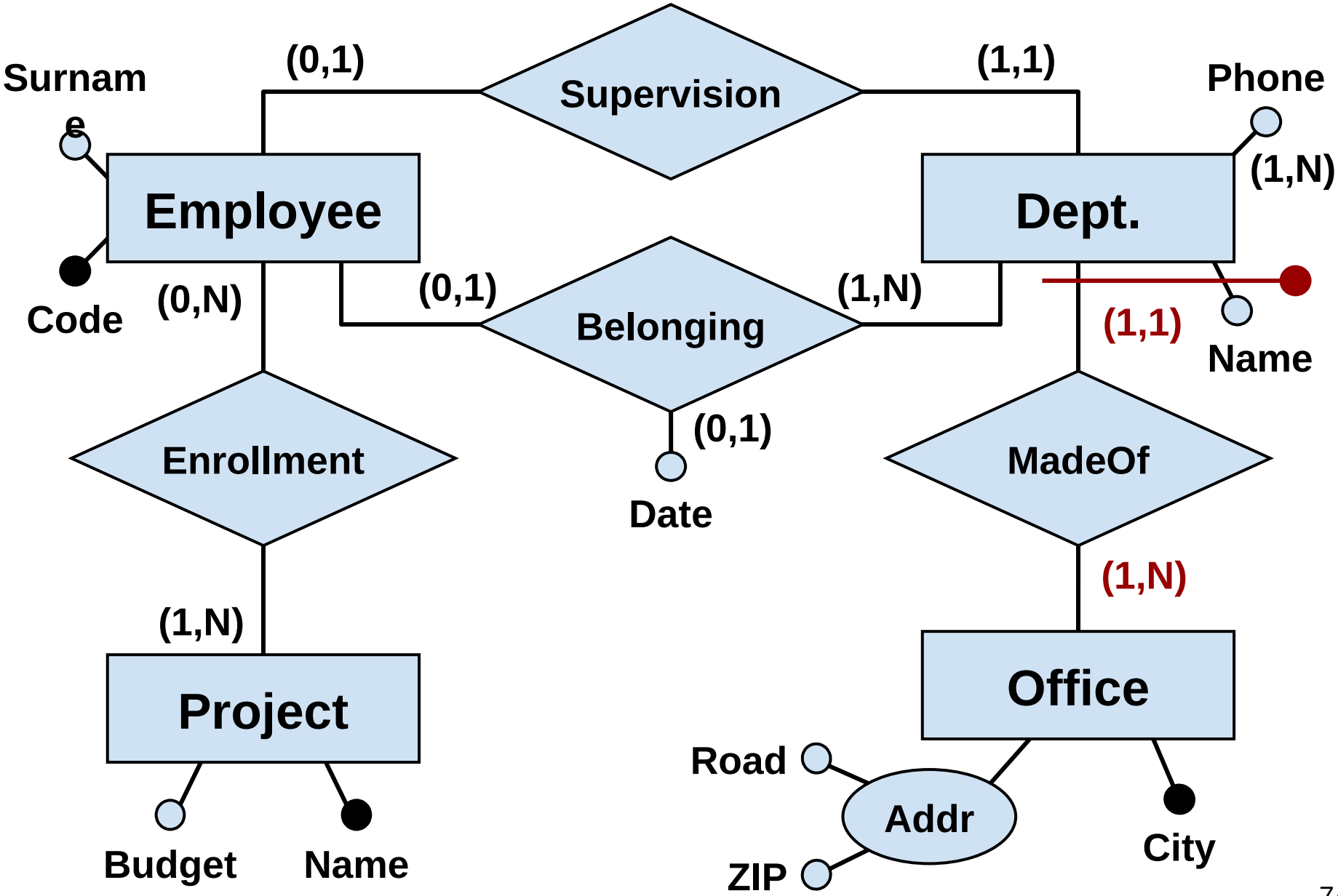
PROJECT(Name, Budget)

ENROLLMENT(Employee, Project)

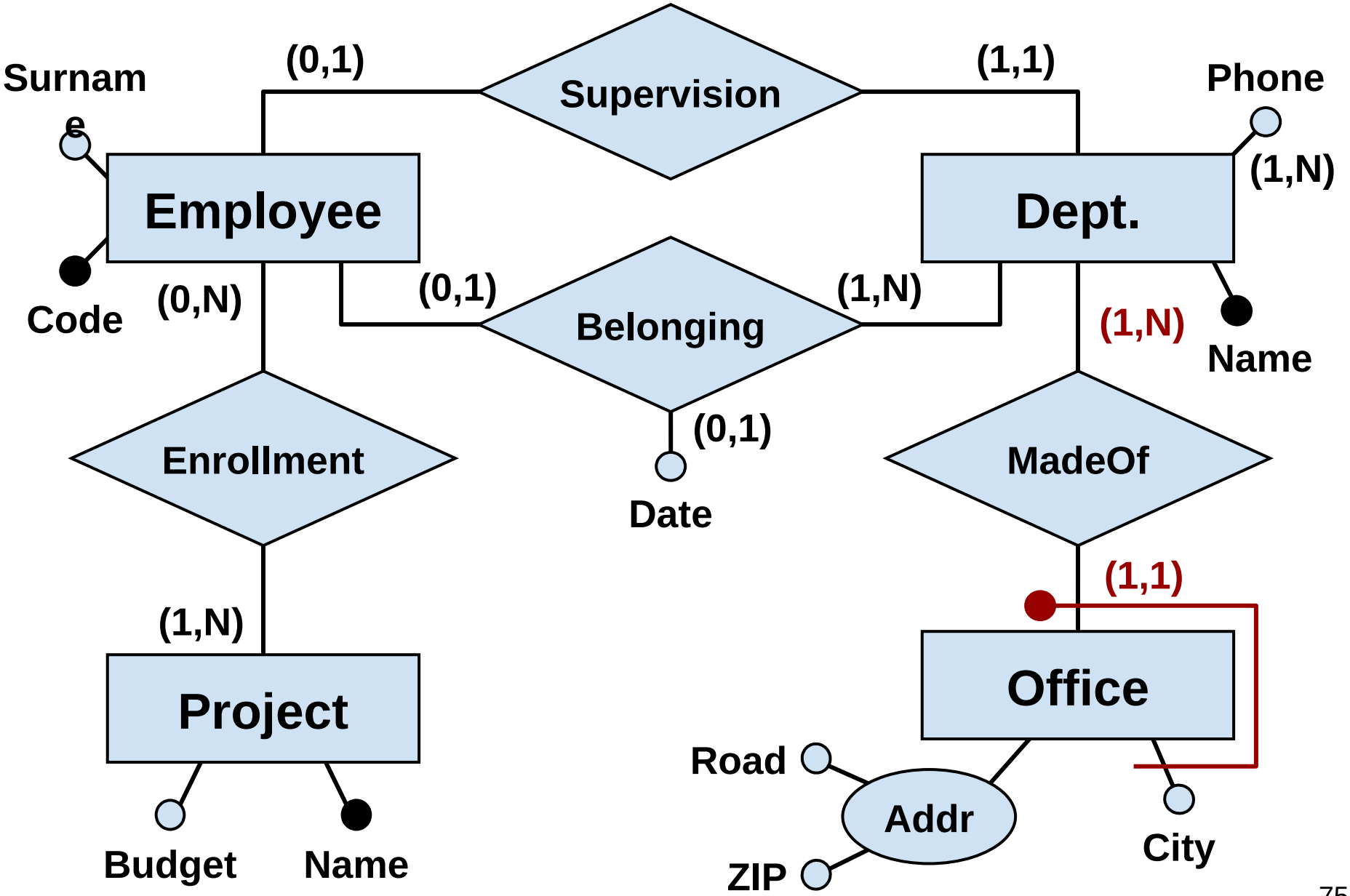
Warning

- Apparently small differences in cardinality and identifier choices could lead to very different meanings

Final Schema (2)



Final Schema (3)



Second Final Schema

EMPLOYEE(Code, Surname, Dept, Office, Date*)

DEPT(Name, Phone, Head)

OFFICE(City, Dept, Road, Zip)

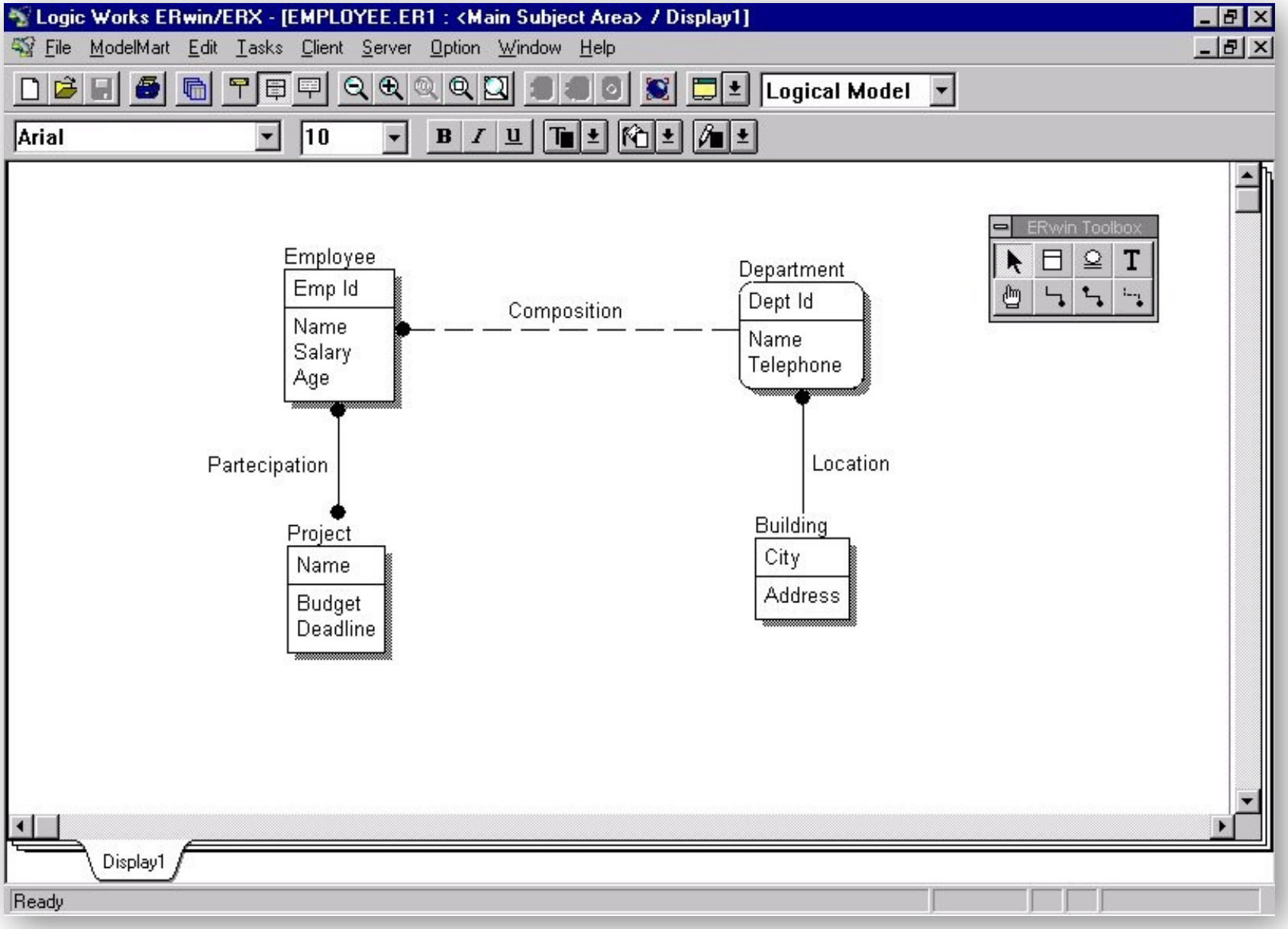
PROJECT(Name, Budget)

ENROLLMENT(Employee, Project)

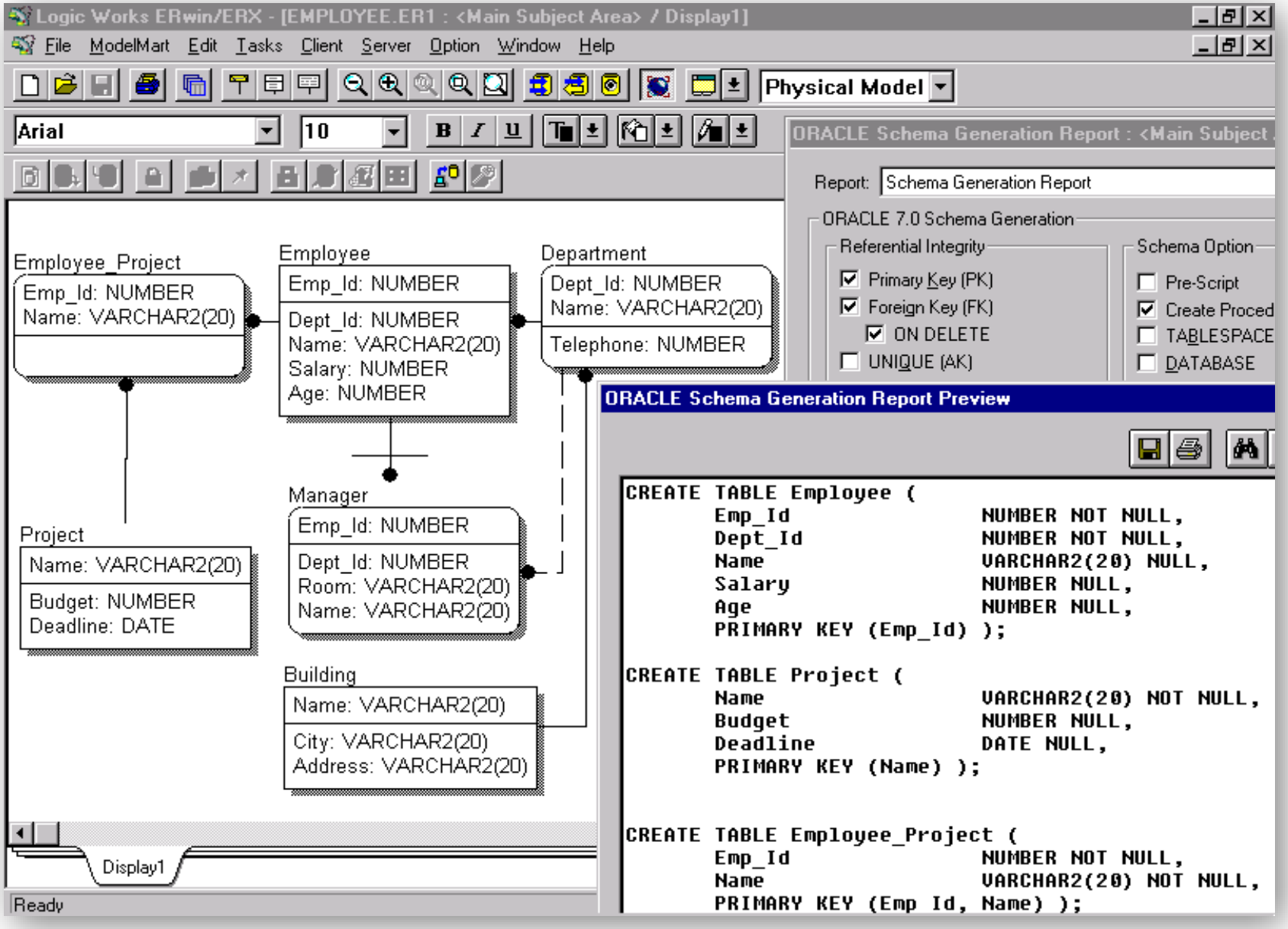
Tools

- There are some **Computer-Aided Software Engineering** (CASE) off-the-shelf software that provide support throughout all the modelling phases of designing a database

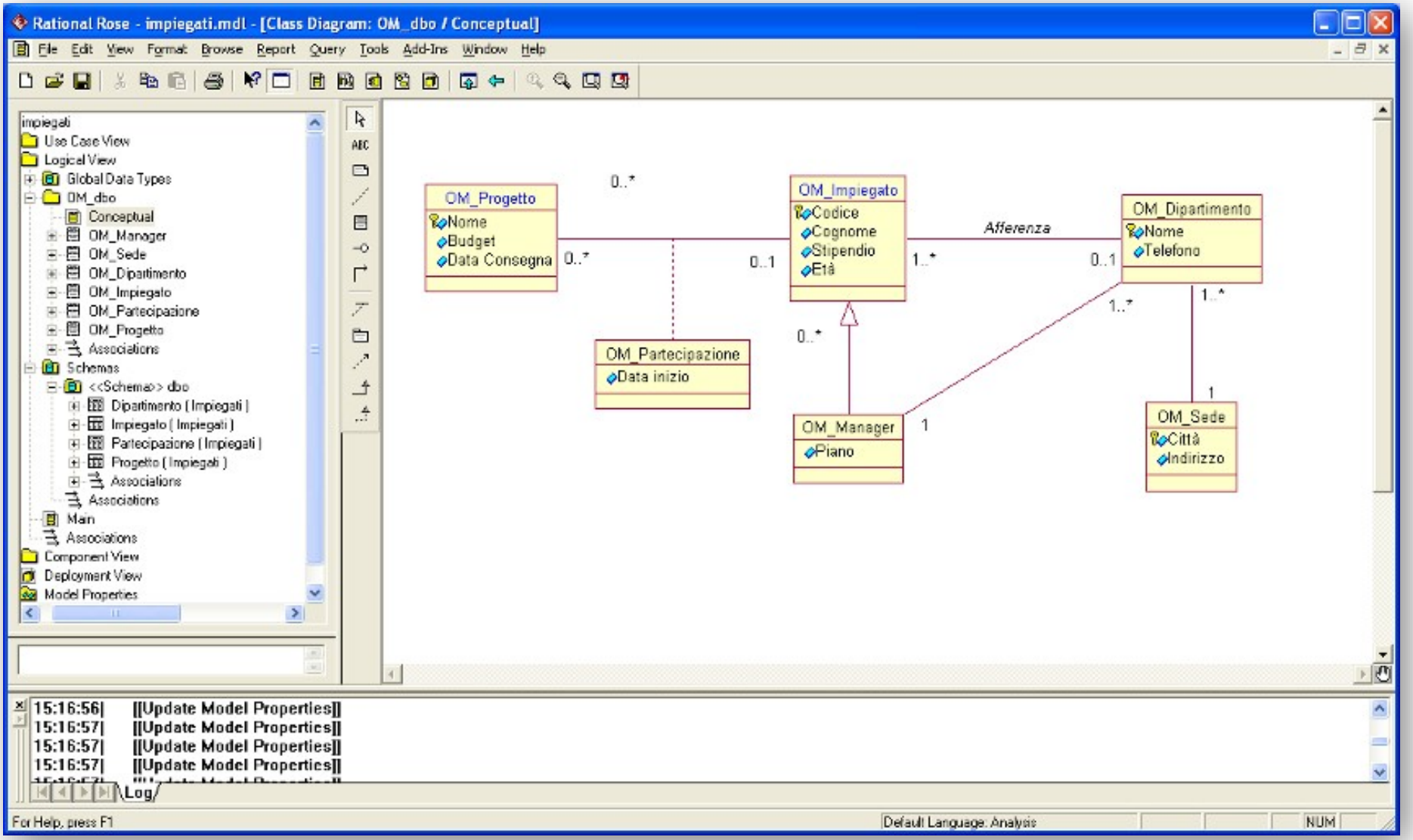
ERwin/ERX (1)


Logic Works ERwin/ERX - [EMPLOYEE.ER1 : <Main Subject Area> / Display1]
File ModelMart Edit Tasks Client Server Option Window Help
Logical Model
Arial
10
ERwin Toolbox
Employee
Emp Id
Name
Salary
Age
Composition
Department
Dept Id
Name
Telephone
Partecipation
Location
Project
Name
Budget
Deadline
Building
City
Address
Display1
Ready

ERwin/ERX (2)


Logic Works ERwin/ERX - [EMPLOYEE.ER1 : <Main Subject Area> / Display1]
File ModelMart Edit Tasks Client Server Option Window Help
Physical Model
Arial
10
Employee_Project
Emp_Id: NUMBER
Name: VARCHAR2(20)
Employee
Emp_Id: NUMBER
Dept_Id: NUMBER
Name: VARCHAR2(20)
Salary: NUMBER
Age: NUMBER
Department
Dept_Id: NUMBER
Name: VARCHAR2(20)
Telephone: NUMBER
Manager
Emp_Id: NUMBER
Dept_Id: NUMBER
Room: VARCHAR2(20)
Name: VARCHAR2(20)
Project
Name: VARCHAR2(20)
Budget: NUMBER
Deadline: DATE
Building
Name: VARCHAR2(20)
City: VARCHAR2(20)
Address: VARCHAR2(20)
Display1
Ready
ORACLE Schema Generation Report : <Main Subject
Report: Schema Generation Report
ORACLE 7.0 Schema Generation
Referential Integrity
Primary Key (PK)
Foreign Key (FK)
ON DELETE
UNIQUE (AK)
Schema Option
Pre-Script
Create Proced
TABLESPACE
DATABASE
ORACLE Schema Generation Report Preview
CREATE TABLE Employee (
Emp_Id NUMBER NOT NULL,
Dept_Id NUMBER NOT NULL,
Name VARCHAR2(20) NULL,
Salary NUMBER NULL,
Age NUMBER NULL,
PRIMARY KEY (Emp_Id));
CREATE TABLE Project (
Name VARCHAR2(20) NOT NULL,
Budget NUMBER NULL,
Deadline DATE NULL,
PRIMARY KEY (Name));
CREATE TABLE Employee_Project (
Emp_Id NUMBER NOT NULL,
Name VARCHAR2(20) NOT NULL,
PRIMARY KEY (Emp Id, Name));

IBM Rational Rose (1)



IBM Rational Rose (2)