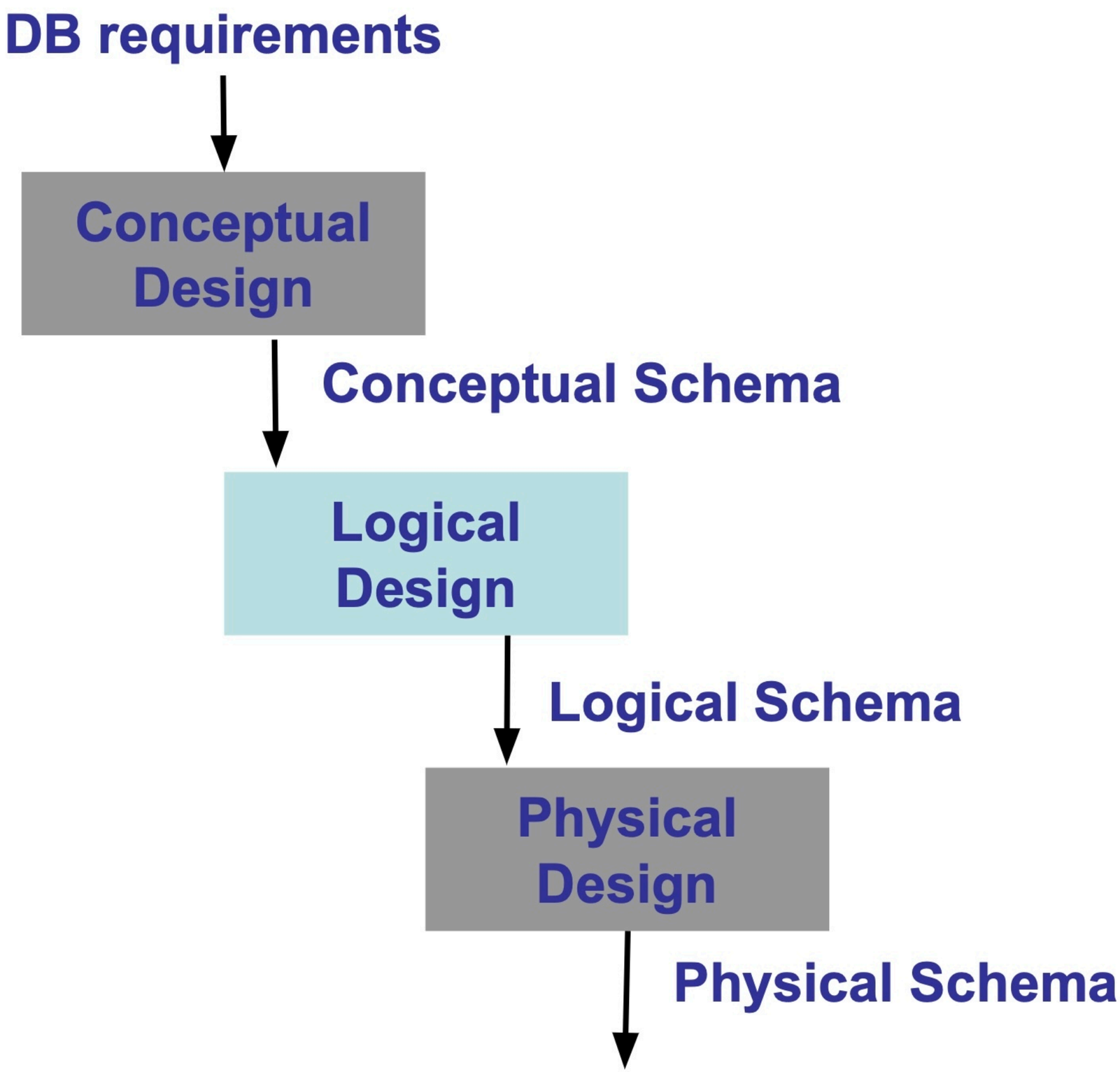
Databases

Logical Design

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.

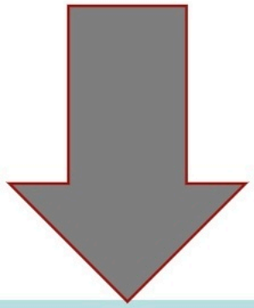
Logical design: input and output data

- Input:
  - Conceptual Schema
  - Application workload information
  - Logical Model
- Uscita:
  - 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).

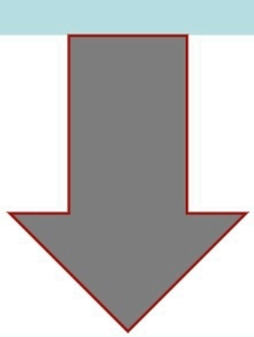
Application workload



Conceptual E-R Schema

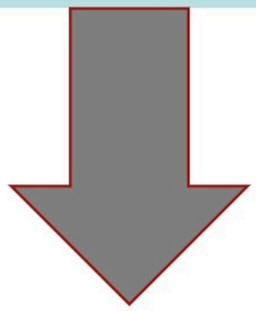
E-R Schema Restructuring

Logical Model



Restructured E-R Schema

Translation into logical model



Logical Schema

E-R Schema restructuring

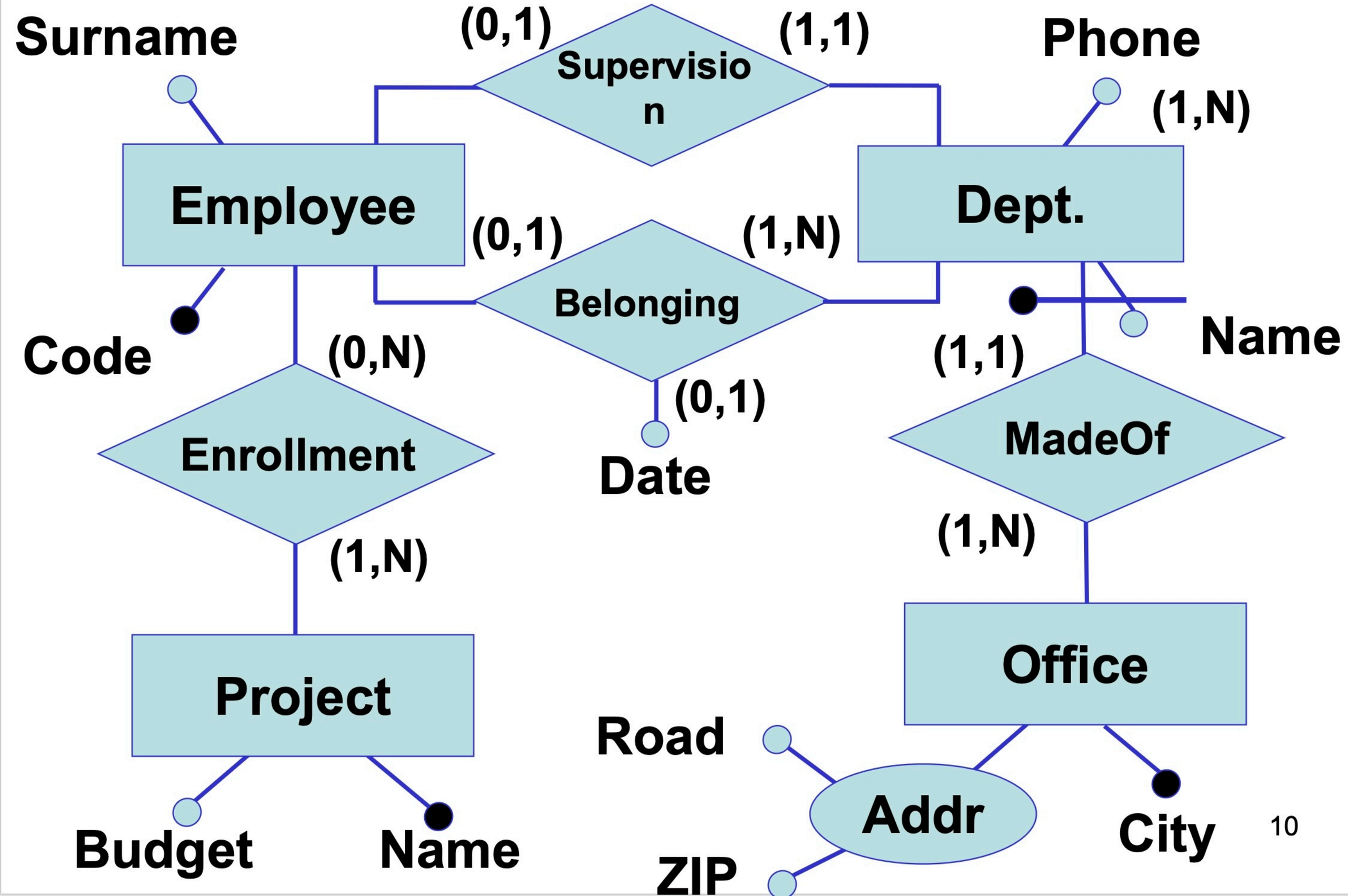
- Why?
  - Simplify the translation
  - Optimize the performances
- Please note that:
  - A restructured E-R is no more a “conceptual schema” in the strict sense of the term.

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, approximated

- 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


Surname
(0,1)
Supervisio
n
(1,1)
Phone
(1,N)
Employee
(0,1)
(1,N)
Dept.
Belonging
Code
(0,N)
(1,1)
Name
(0,1)
Enrollment
Date
MadeOf
(1,N)
(1,N)
Office
Project
Road
Addr
City
Budget
Name
ZIP

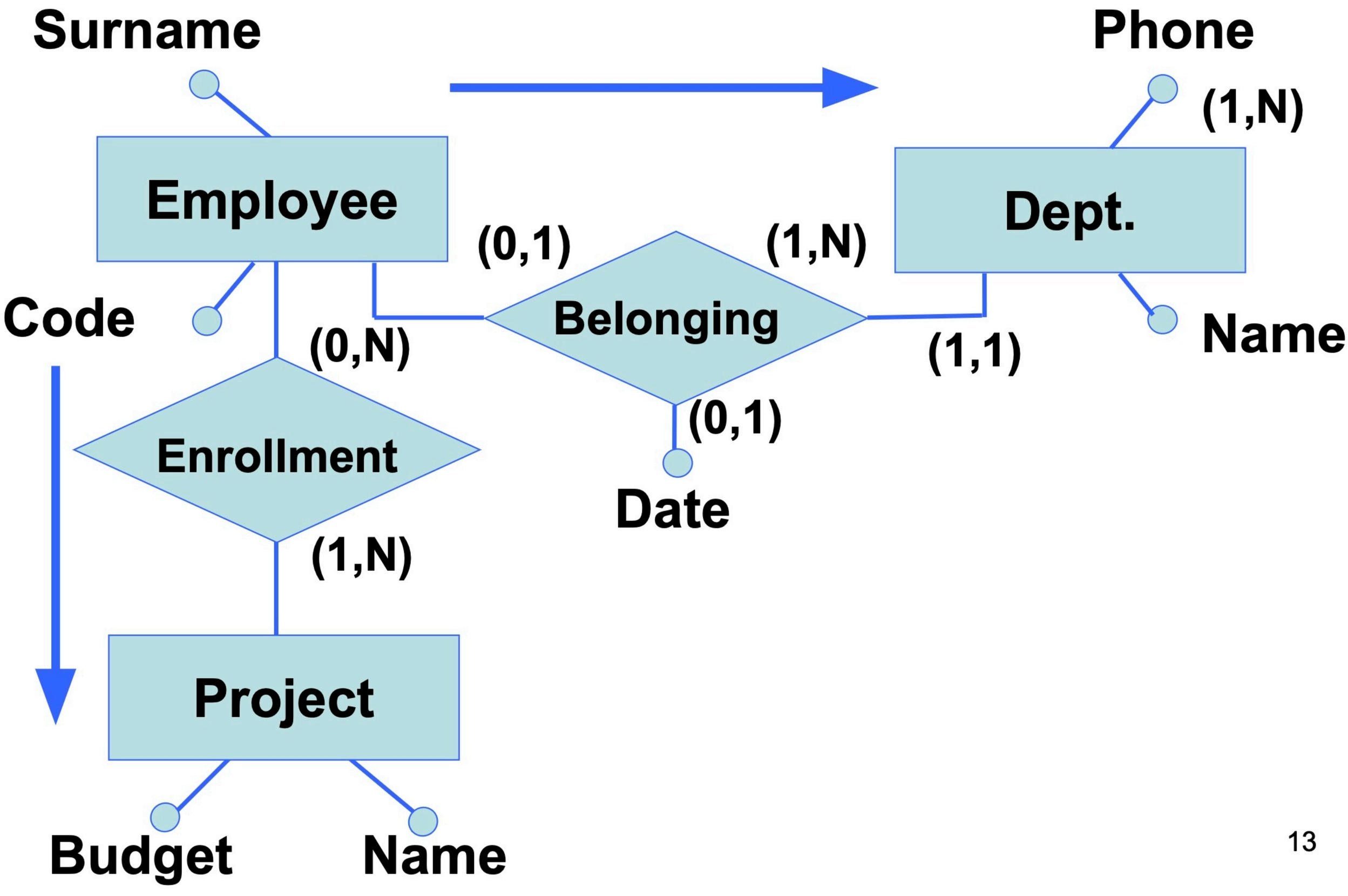
Size Table

Name	Kind	Size
Office	E	10
Dept.	E	80
Employee	E	2000
Project	E	500
MadeOf	R	80
Belonging	R	1900
Supervision	R	80
Enrollment	R	6000

Example of cost evaluation

- Operation:
  - Return all the data of an 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


Surname
Phone
(1,N)
Employee
Dept.
(0,1)
(1,N)
Code
Belonging
Name
(0,N)
(1,1)
Enrollment
(0,1)
Date
(1,N)
Project
Budget
Name

Access Table

Name	Kind	#Accesses	Accesses Types	Access order
Dept.	Entity	1	R	3
Employee	Entity	1	R	1
Project	Entity	3	R	5
Belonging	Rel.	1	R	2
Enrollment	Rel.	3	R	4

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.

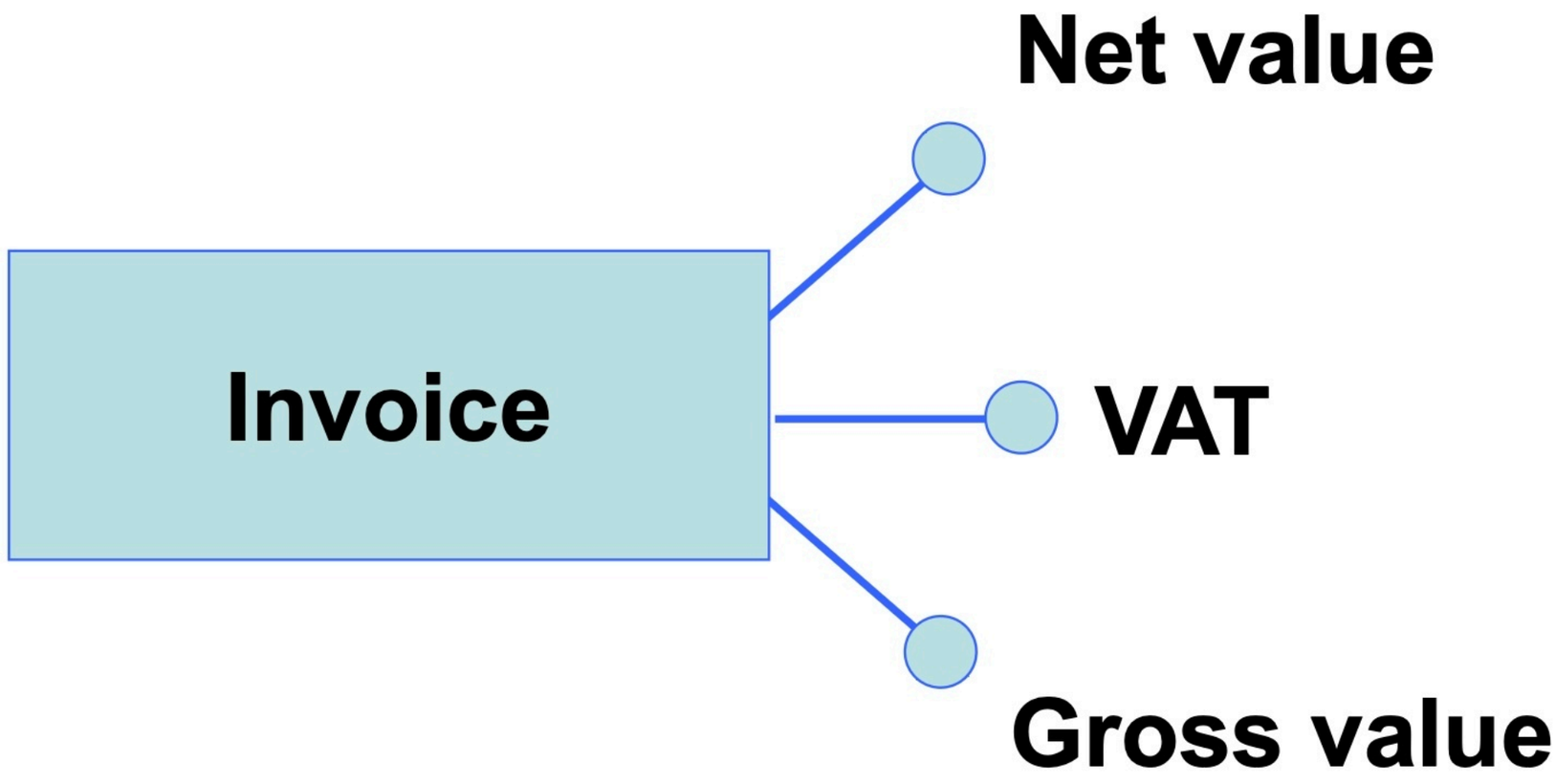
Redundances

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

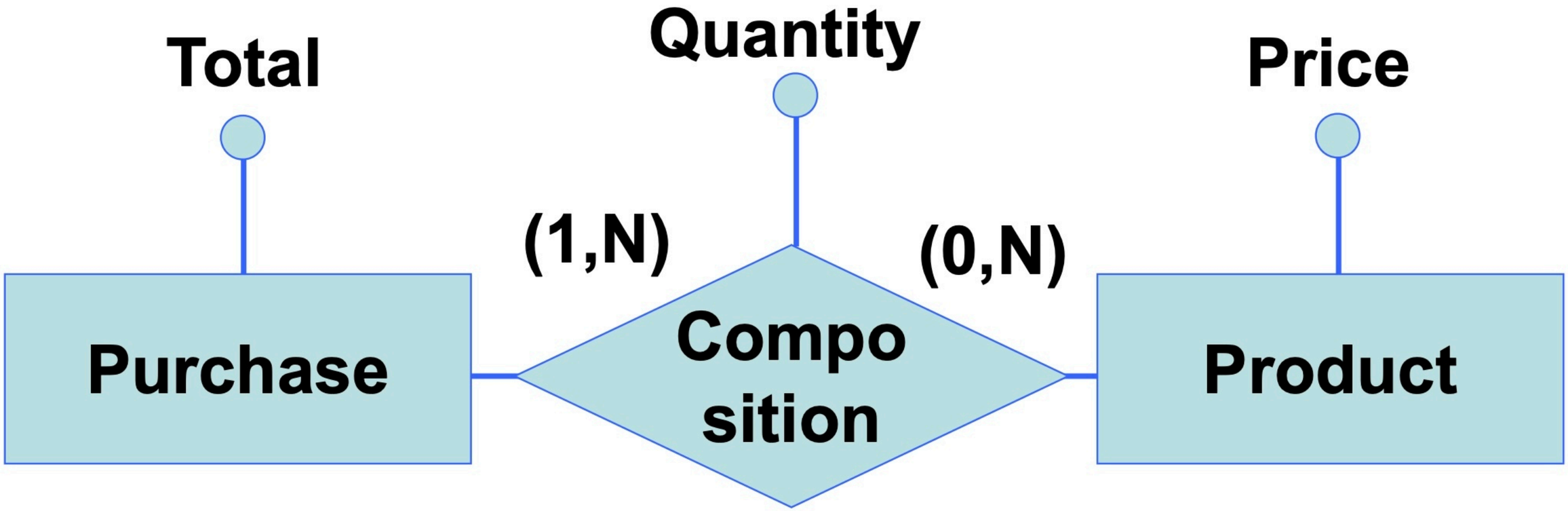
E-R: Types of Redundancies

- 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)

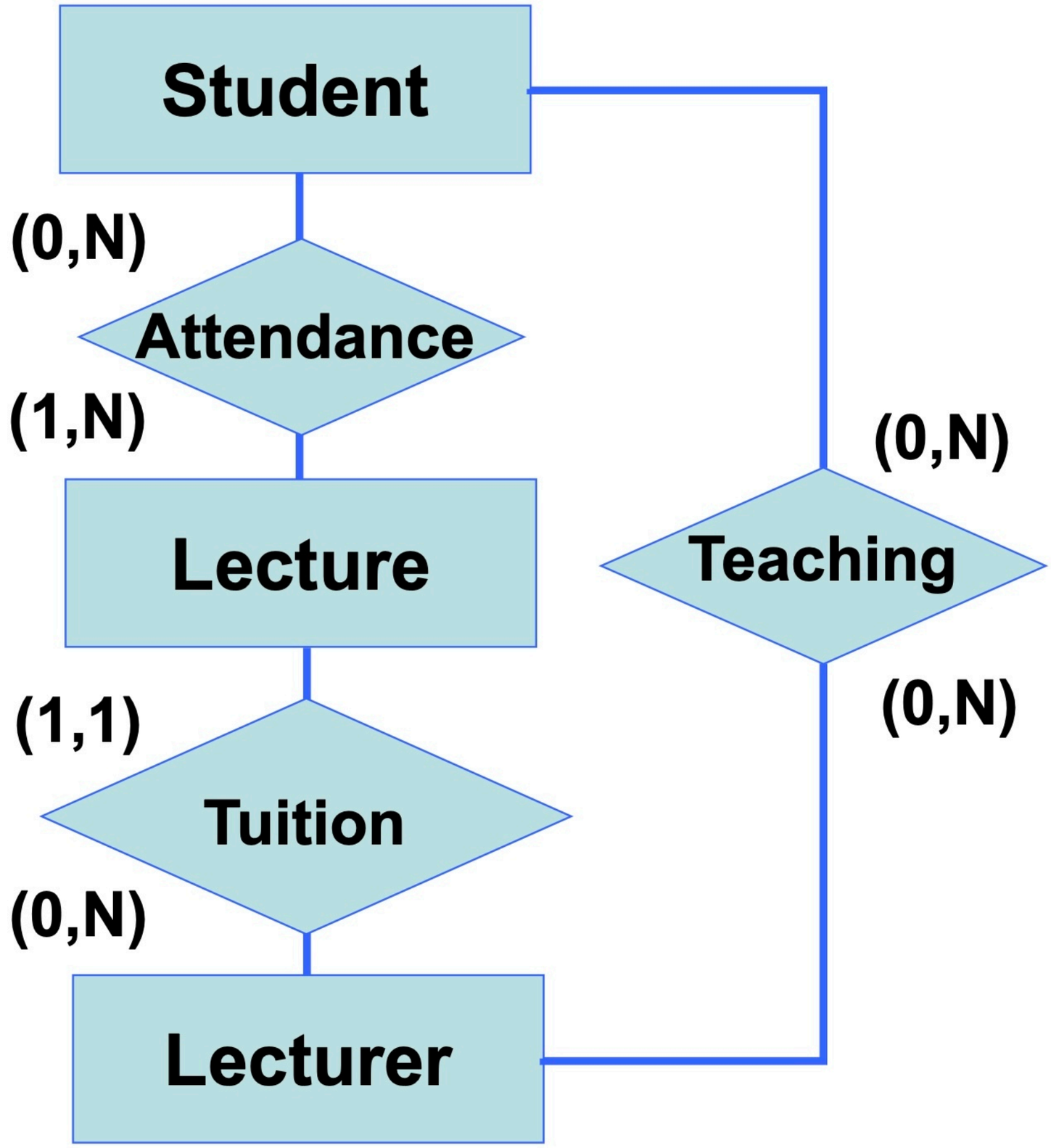
Derivable attributes



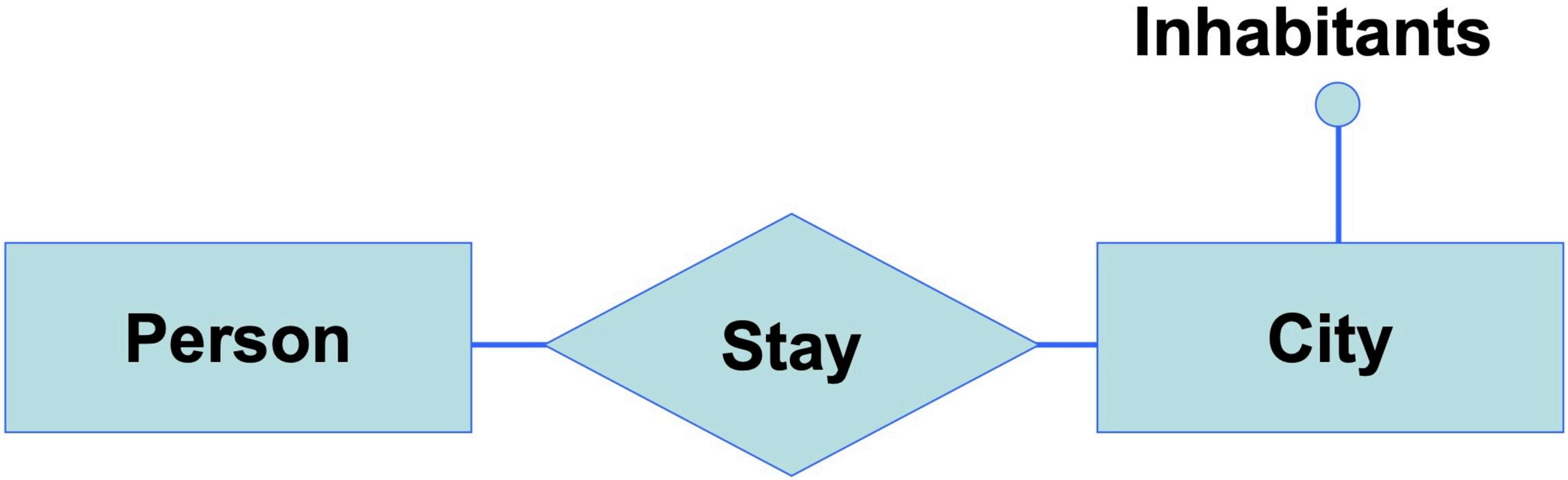
Attribute derivable from other entities



Redundancy from a cycle



Redundancy Analysis



Name	Kind	Size
City	E	200
Person	E	1000000
Stay	R	1000000

- 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)

With Redundancy

Operation #1

Name	Kind	#Accesses	Accesses Types
Person	E	1	W
Stay	R	1	W
City	E	1	R
City	E	1	W

Operation #2

Name	Kind	#Accesses	Accesses Types
City	E	1	R

Without Redundancy

Operation #1

Name	Kind	#Accesses	Accesses Types
Person	E	1	W
Stay	R	1	W

Operation #2

Name	Kind	#Accesses	Accesses Types
City	E	1	R
Stay	R	5000	R

With Redundancy

- Costs:
  - Operation #1: 1500 writes and 500 reads per day
  - Operation #2: negligible.
- We count write accesses as doubles
  - A total amount of 3500 operations per day

Without Redundancy

- Costs:
  - Operation #1: 1000 writes
  - Operation #2: 10000 reads per day
- Writings costs double
  - A total amount of 12000 operations per day

Restructuring activities

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

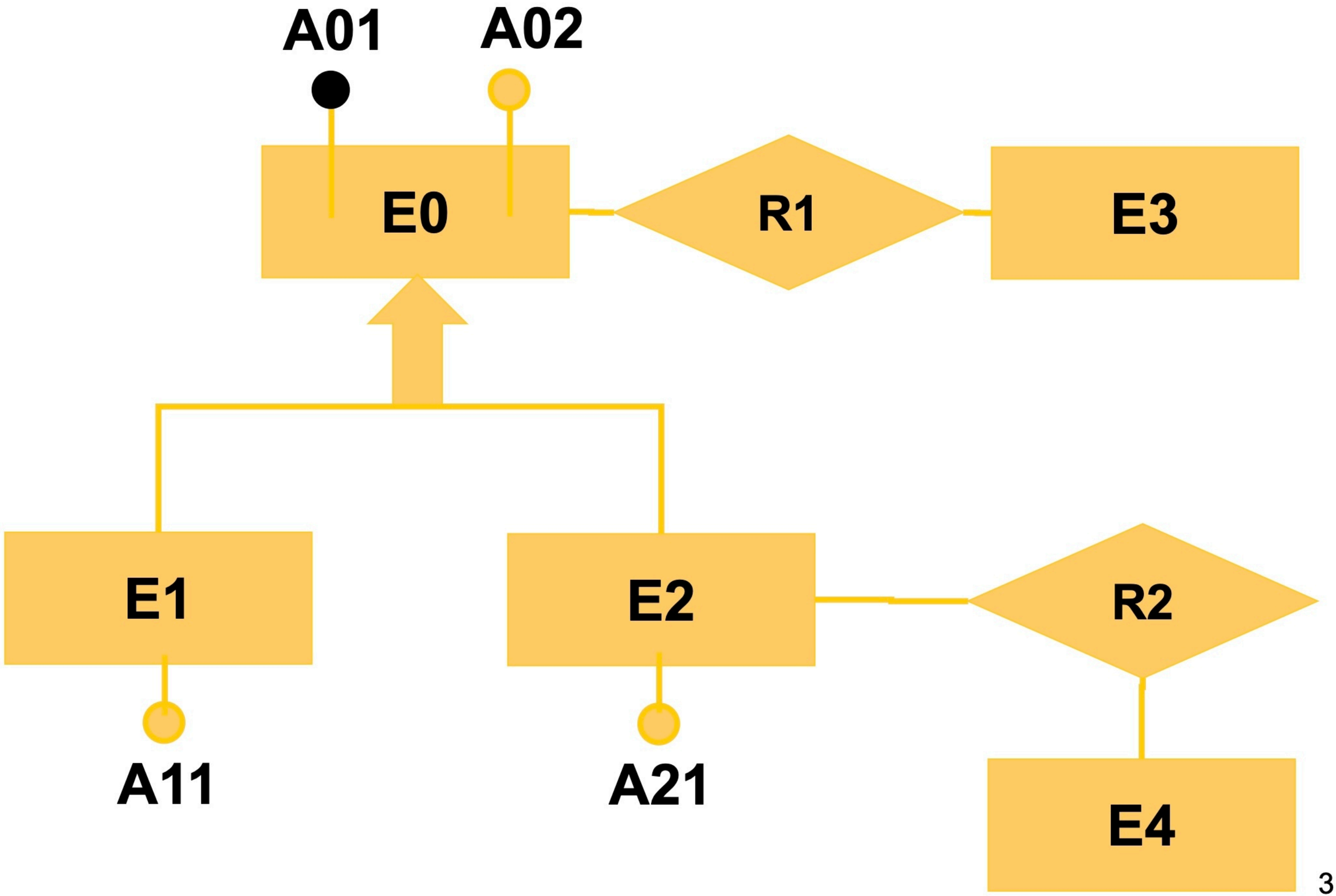
Hierarchies deletion

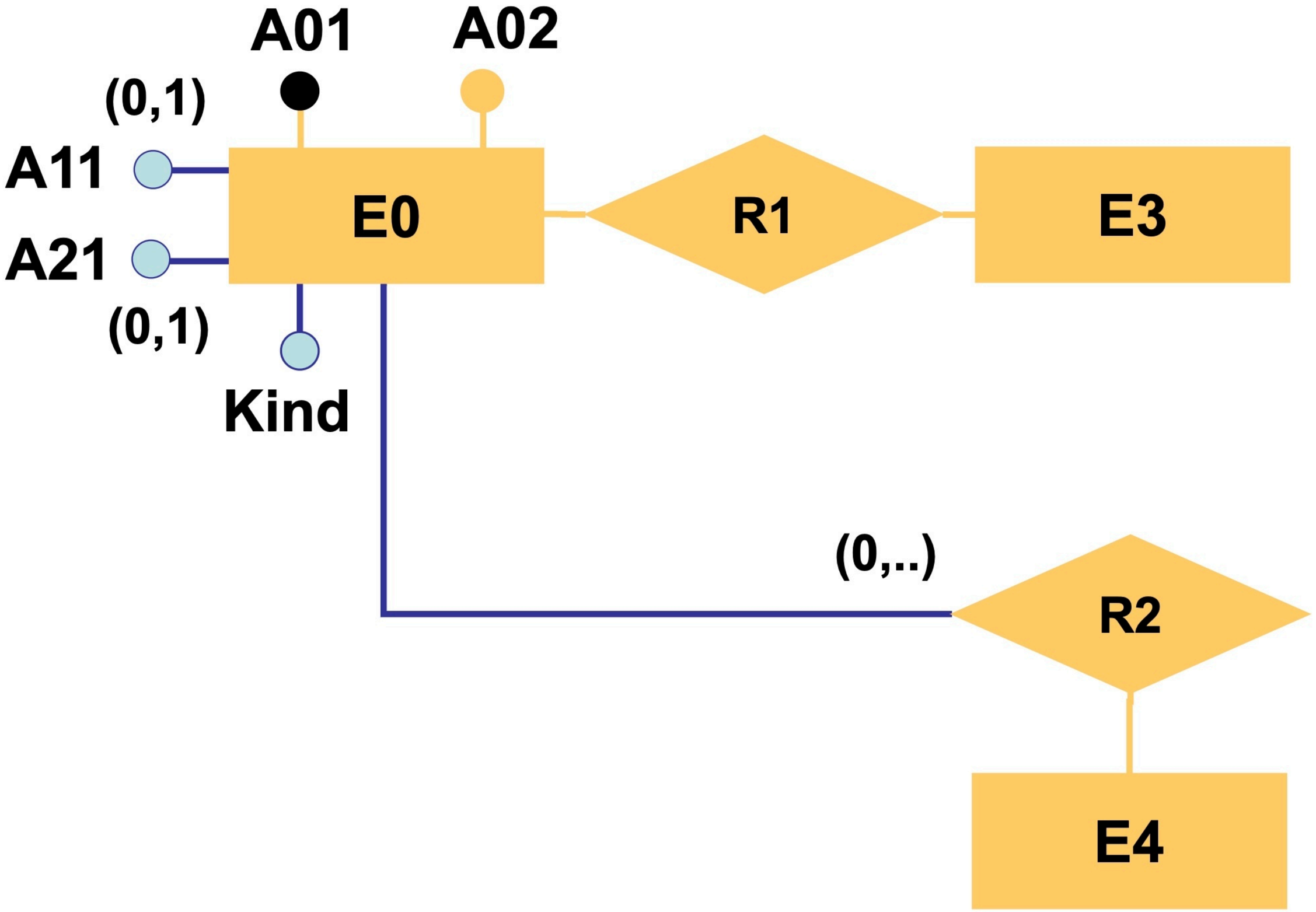
- The relational data model does not directly support generalizations, they cannot be directly represented
- …while entities and relationships are directly representable

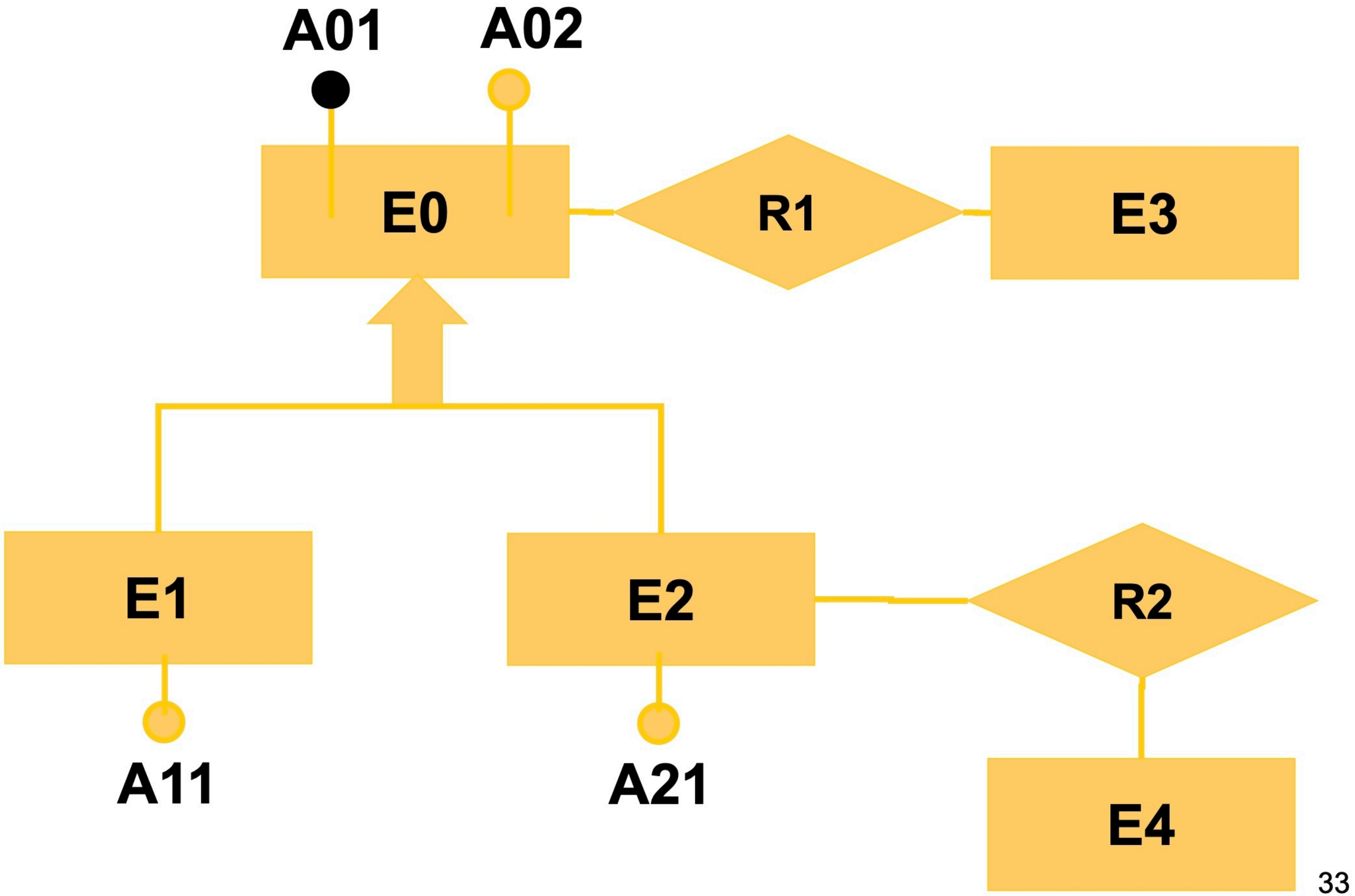
- We therefore remove hierarchies replacing them with entities and relationships

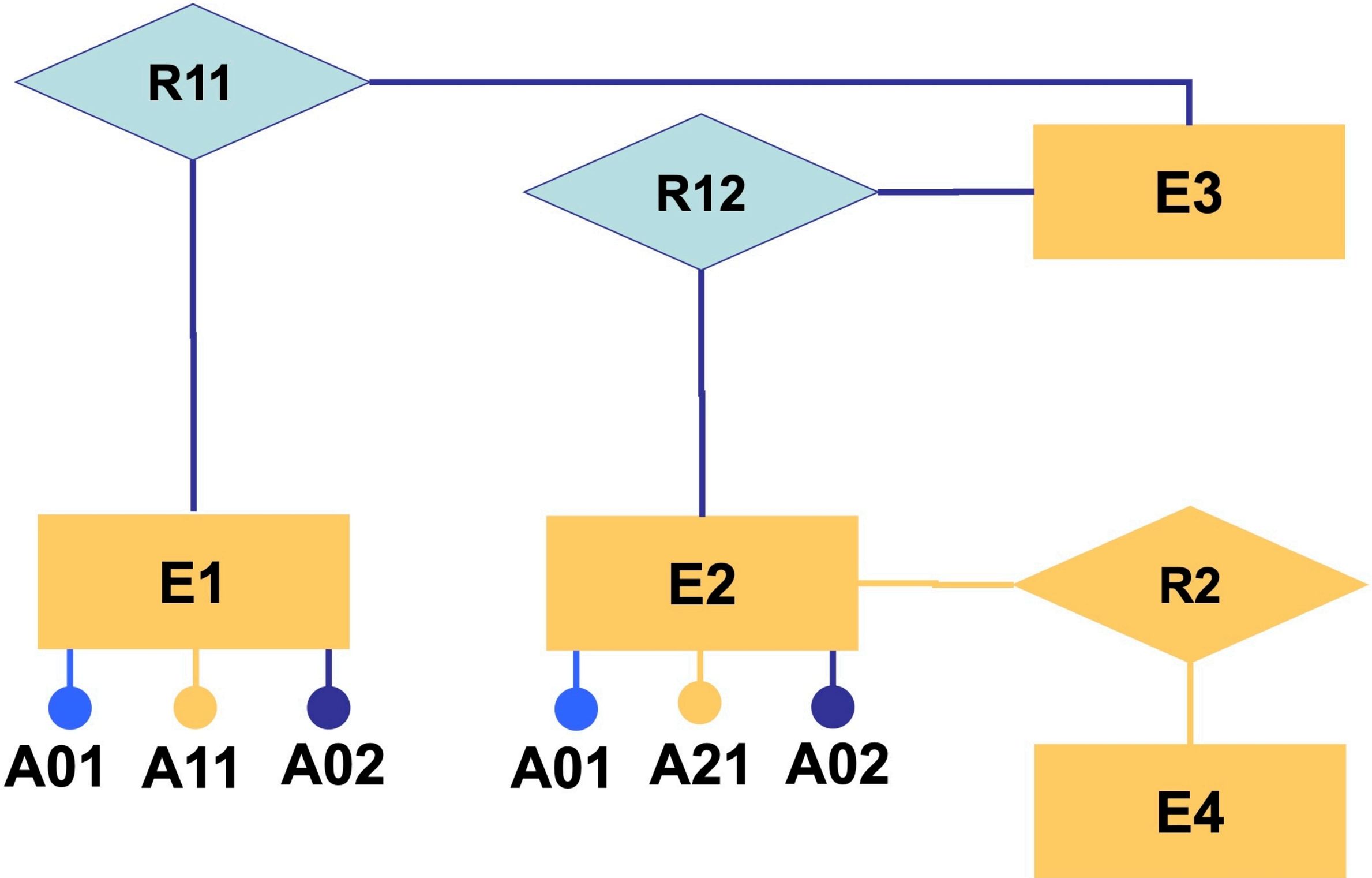
Three possible solutions

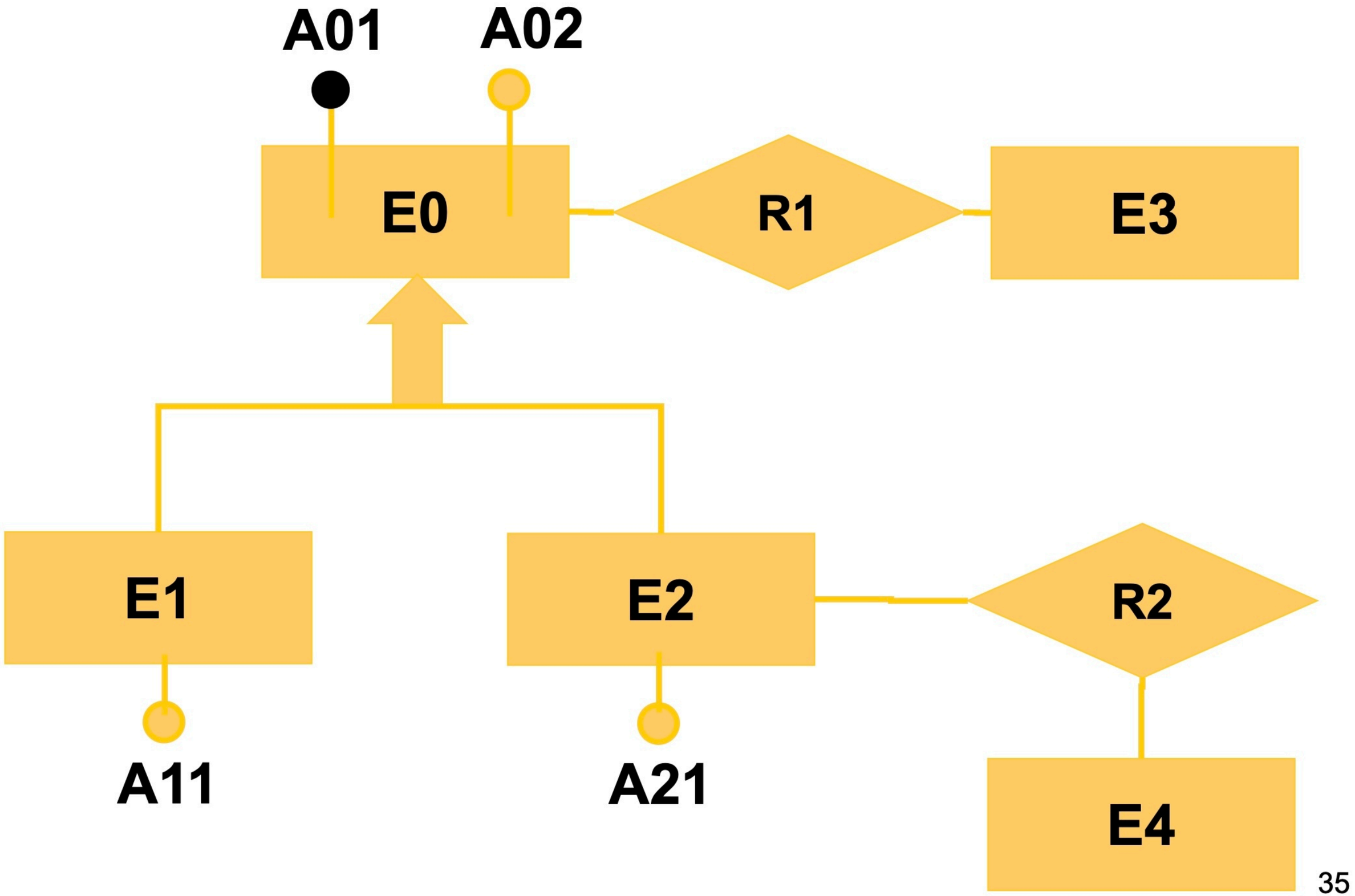
1. Embedding the children of the generalization into the parent.
2. Embedding the parent of the generalization into the children
3. Replacing the generalization with a relationship

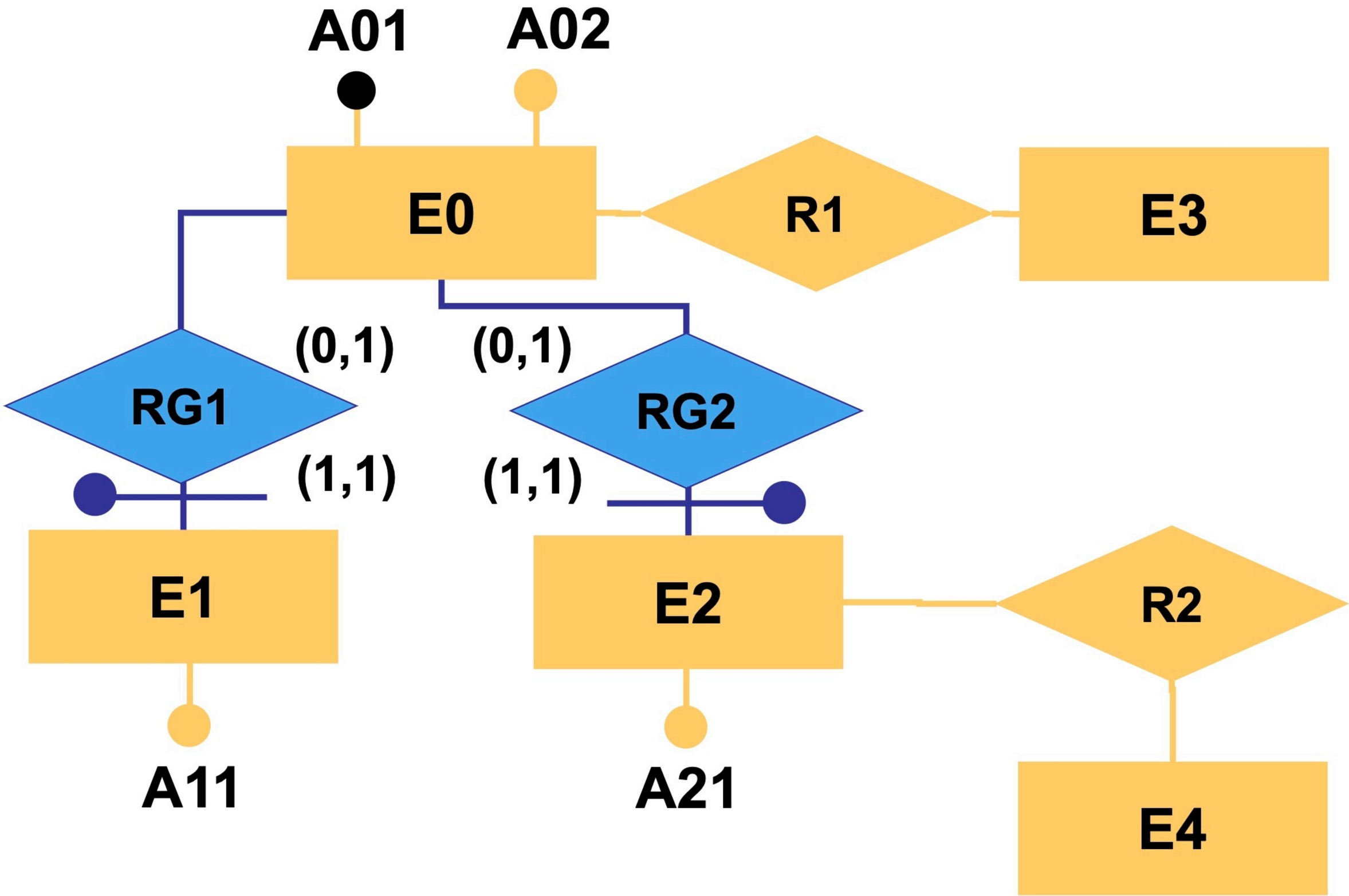

A01
A02
E0
R1
E3
E1
E2
R2
A11
A21
E4


A01
A02
(0,1)
A11
A21
(0,1)
Kind
E0
R1
E3
(0,..)
R2
E4


A01
A02
E0
R1
E3
E1
E2
R2
A11
A21
E4


R11
R12
E3
E1
E2
R2
E4
A01
A11
A02
A01
A21
A02


A01
A02
E0
R1
E3
E1
E2
R2
A11
A21
E4


A01
A02
E0
R1
E3
(0,1)
(0,1)
RG1
RG2
(1,1)
(1,1)
E1
E2
R2
A11
A21
E4

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)

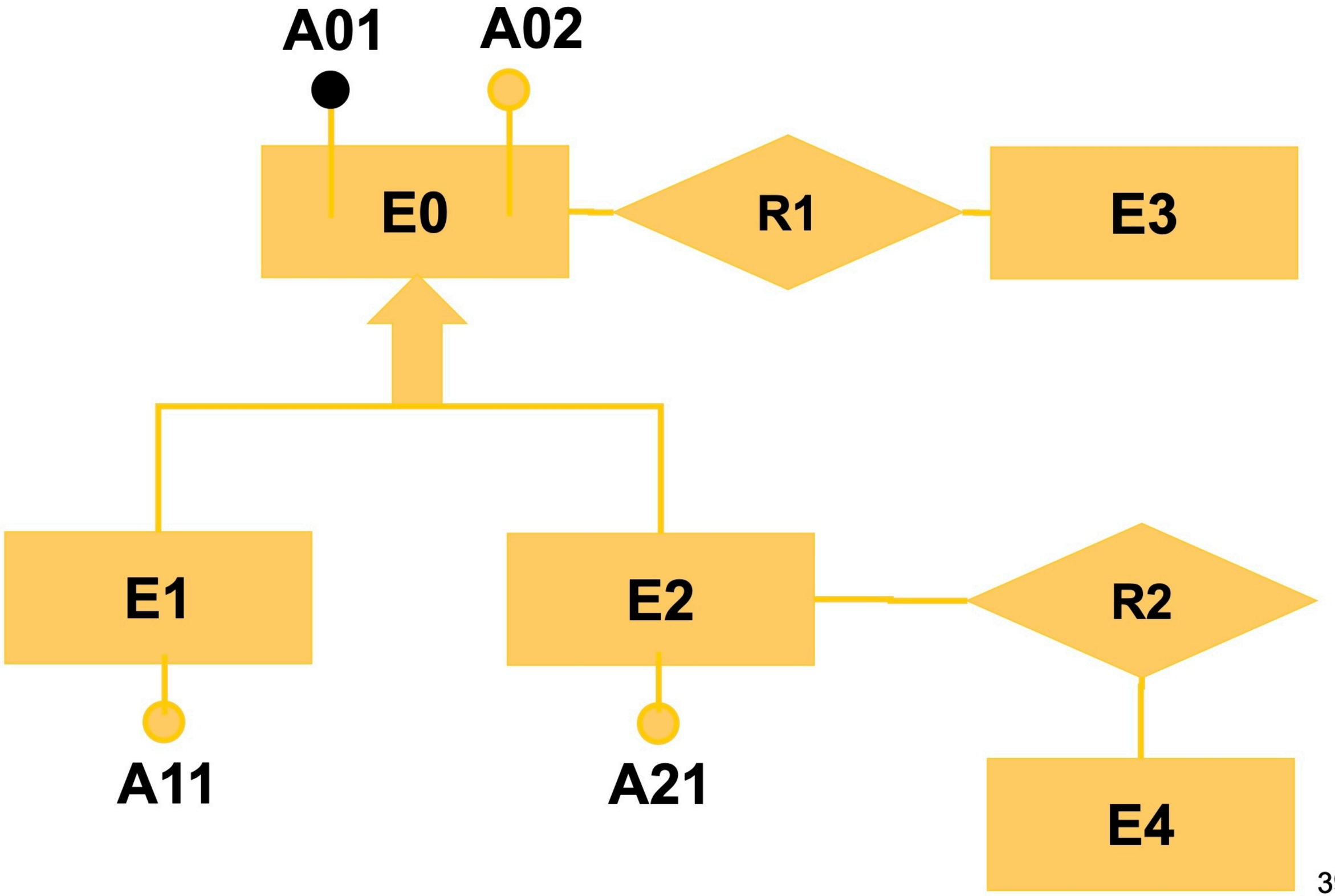
- We now provide some general rules that can be followed

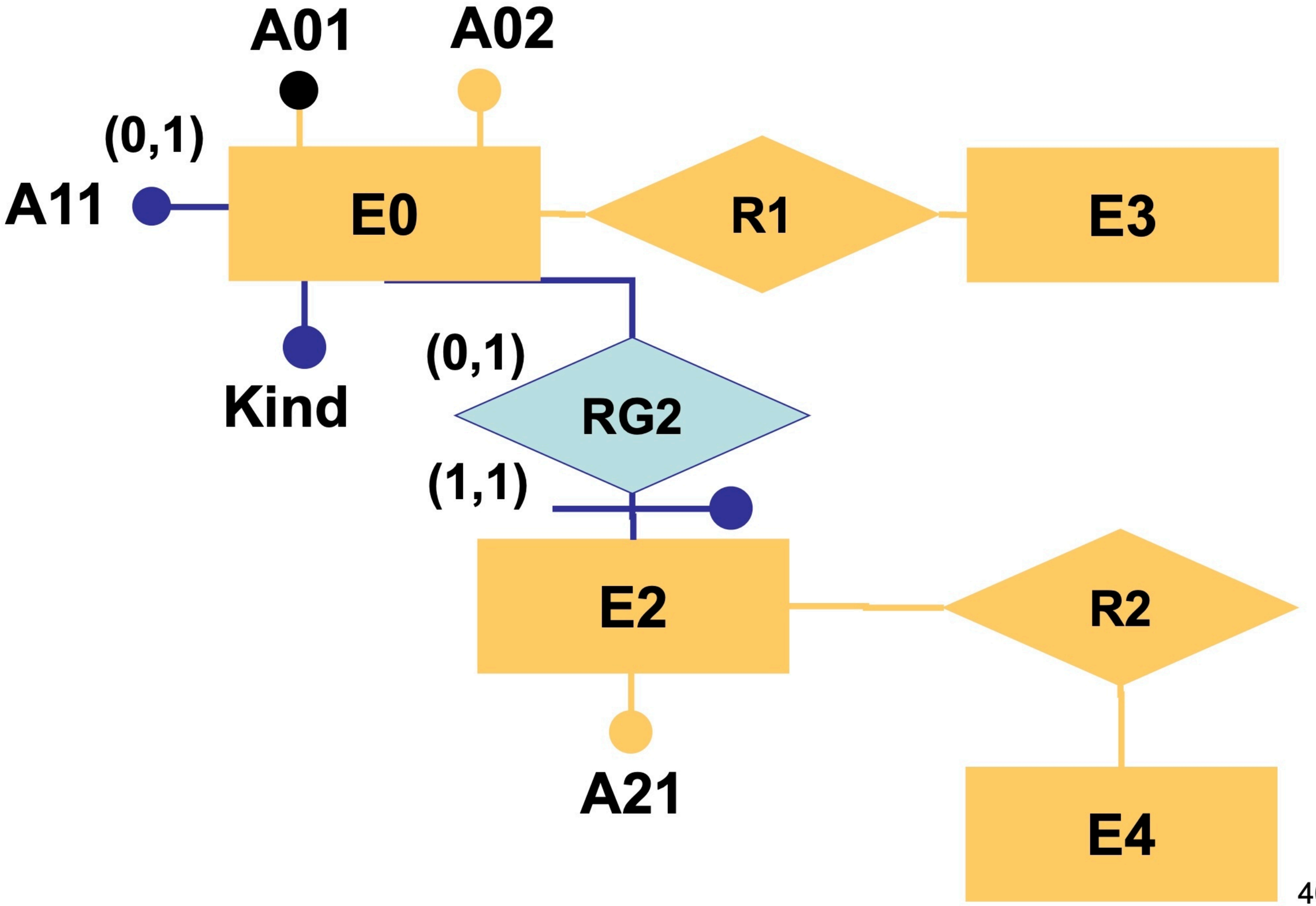
Observations (2)

The three solutions (1, 2 and 3):

1. Should be used when children and father are accessed at the same time
2. Should be used when children are accessed independently from one another
3. Should be used when children are accessed independently from the father

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


A01
A02
E0
R1
E3
E1
E2
R2
A11
A21
E4


A01
A02
(0,1)
A11
E0
R1
E3
Kind
(0,1)
RG2
(1,1)
E2
R2
A21
E4

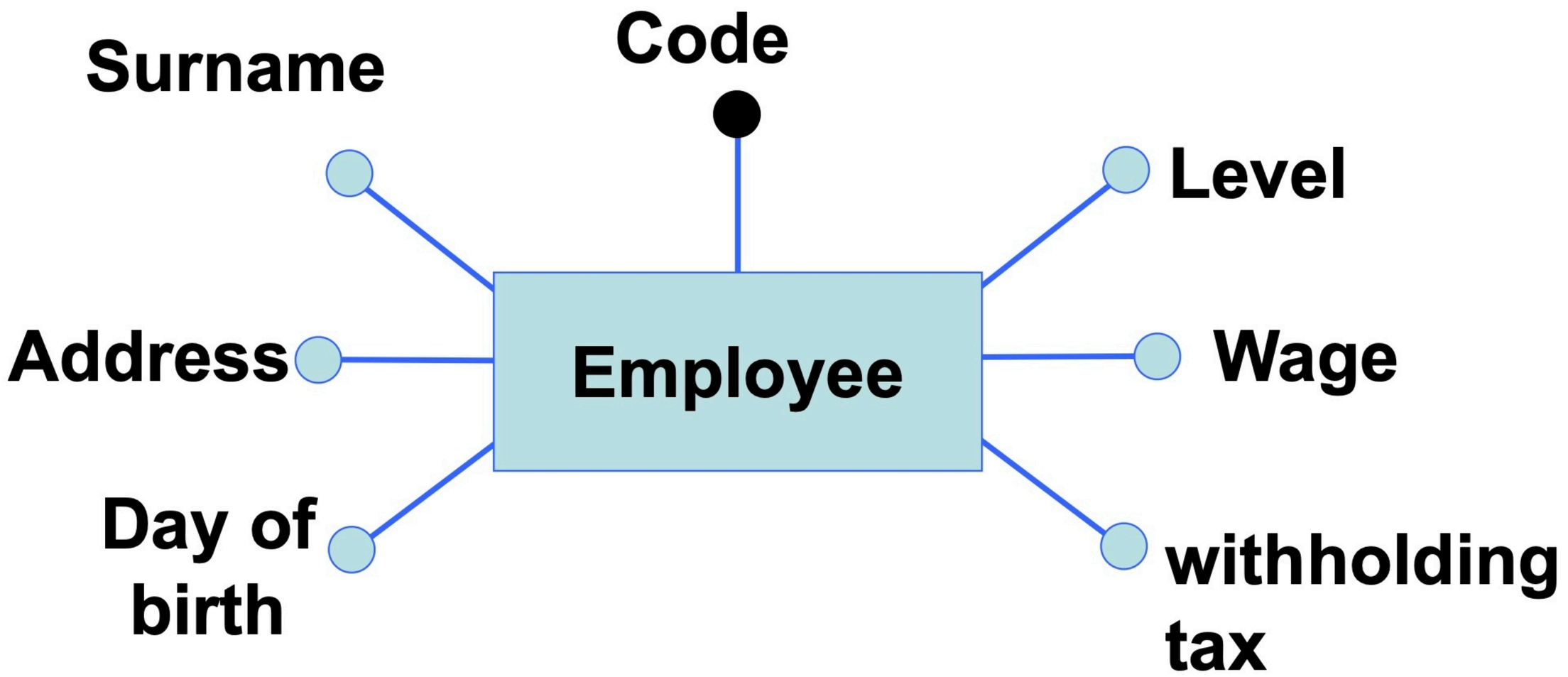
Restructuring activities

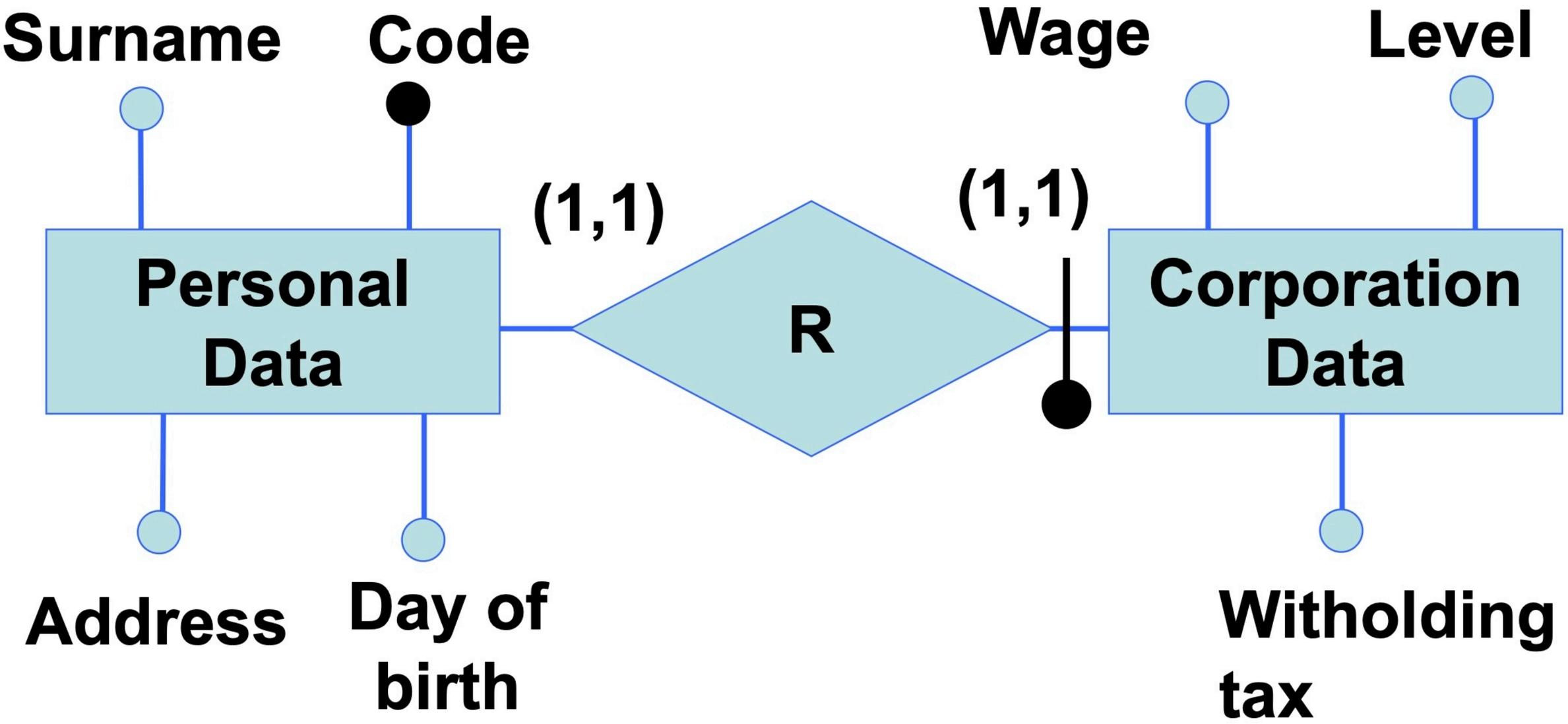
- Redundancies analysis
- Generalizations deletion
- Partitioning/grouping of entities and relationships
- Identifying the primary keys

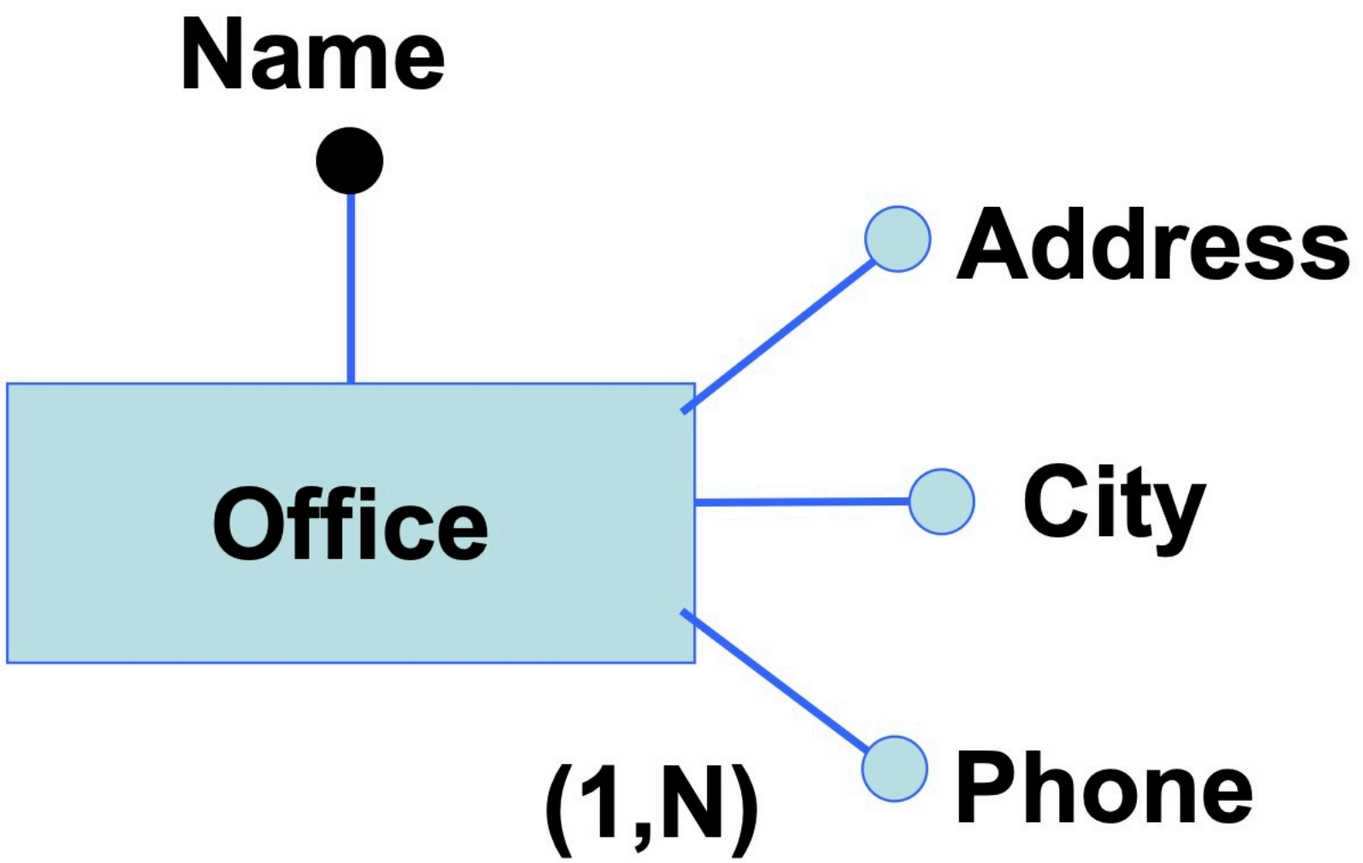
- Restructuring can provide more efficient operations by simply reducing the number of accesses:
  - Attributes accessed separately are splitted
  - Attributes accessed on the same time are grouped, even when they belong to different entities/relationships

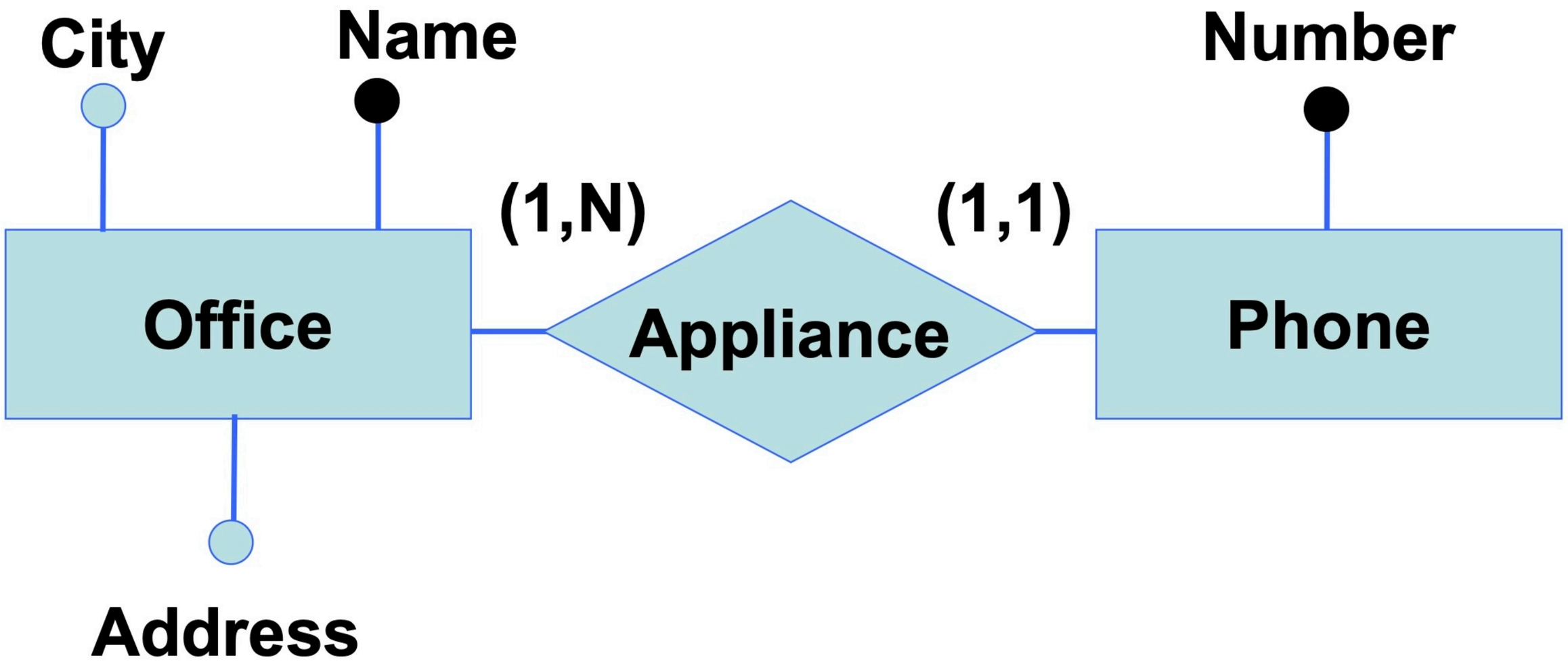
Restructuring, main cases

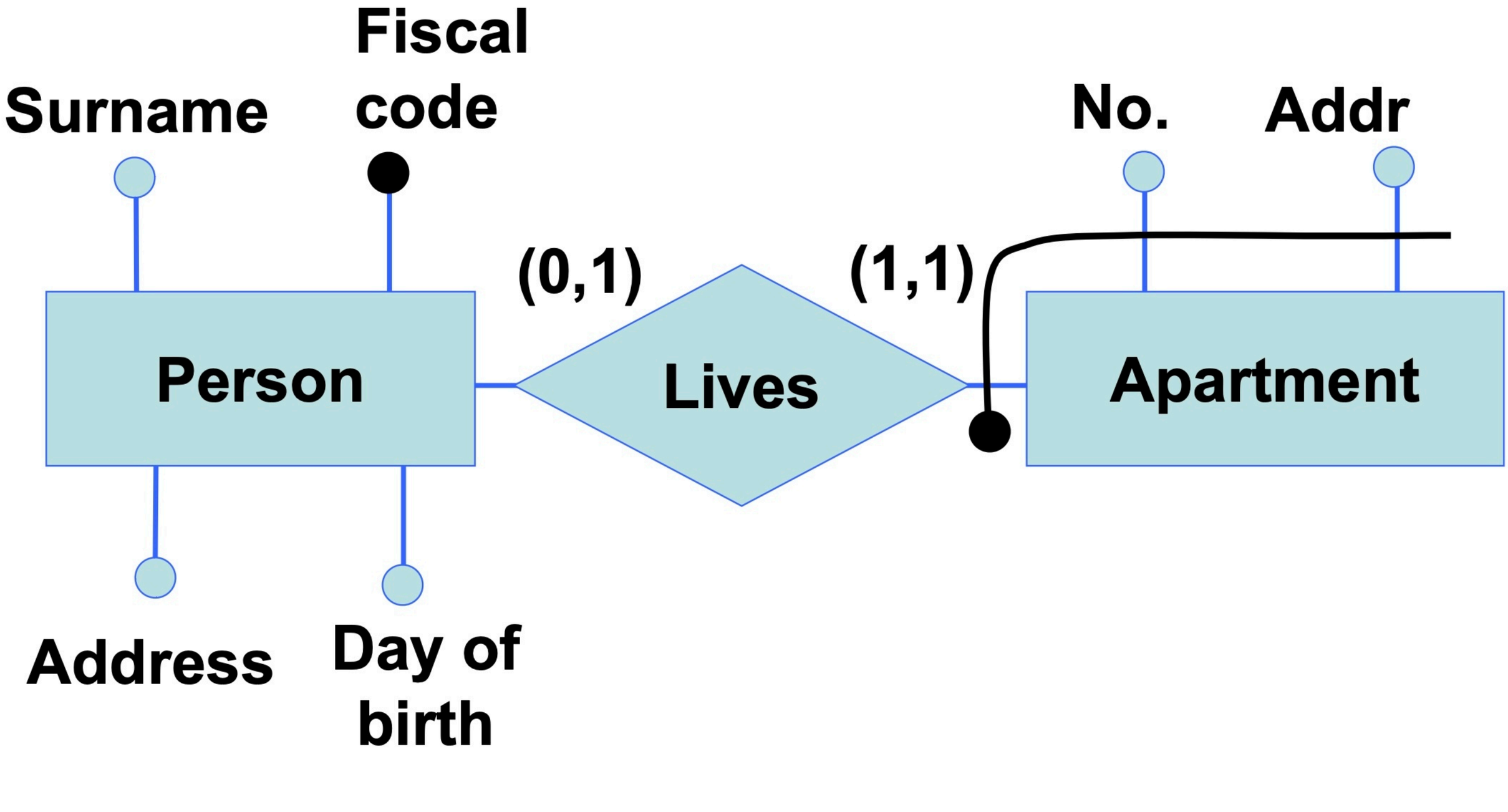
- Entity vertical partitioning
- Relationship horizontal partitioning
- Restructuring multi-valued attributes
- Grouping of entities/relationships

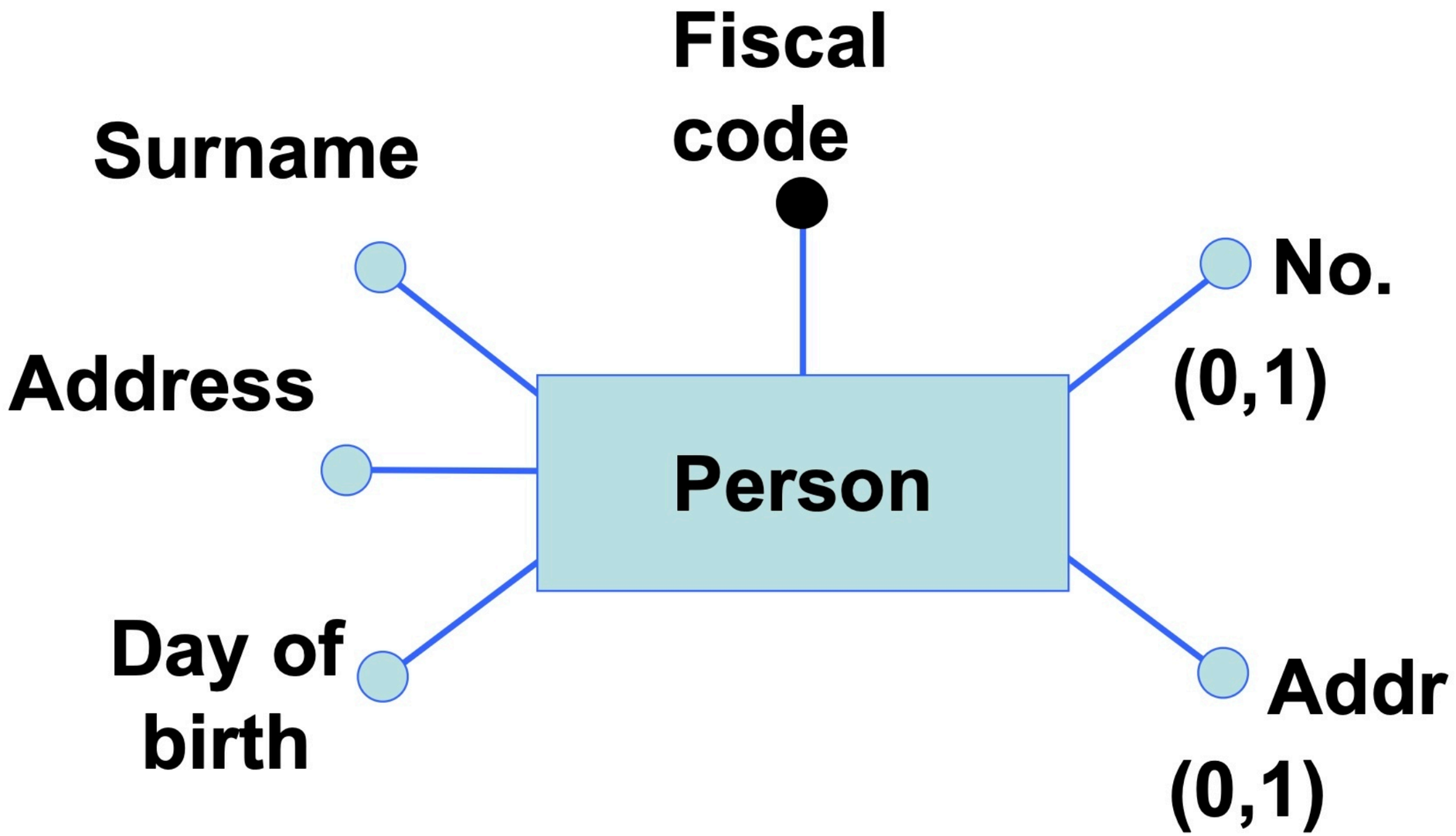

Surname
Code
Level
Address
Employee
Wage
Day of birth
withholding tax

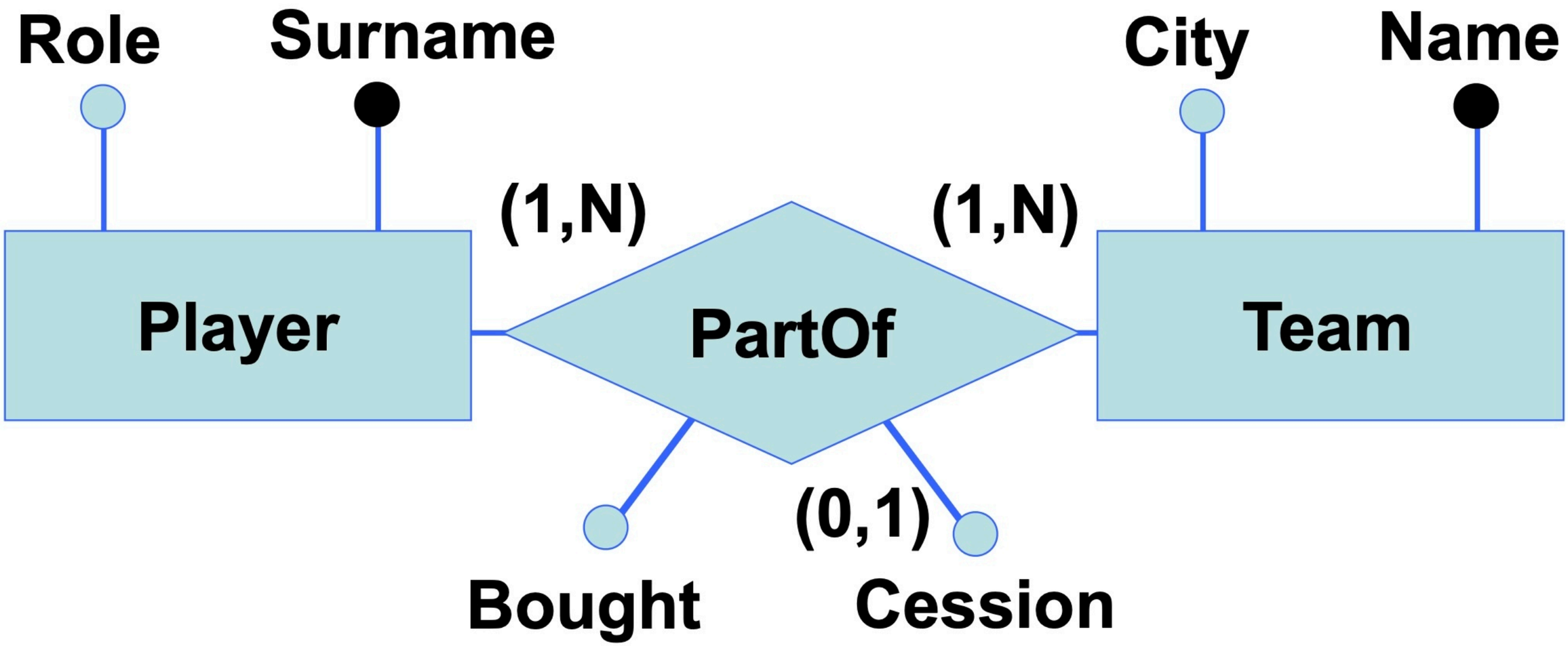

Surname
Code
Wage
Level
(1,1)
(1,1)
Personal Data
R
Corporation Data
Address
Day of birth
Witholding tax

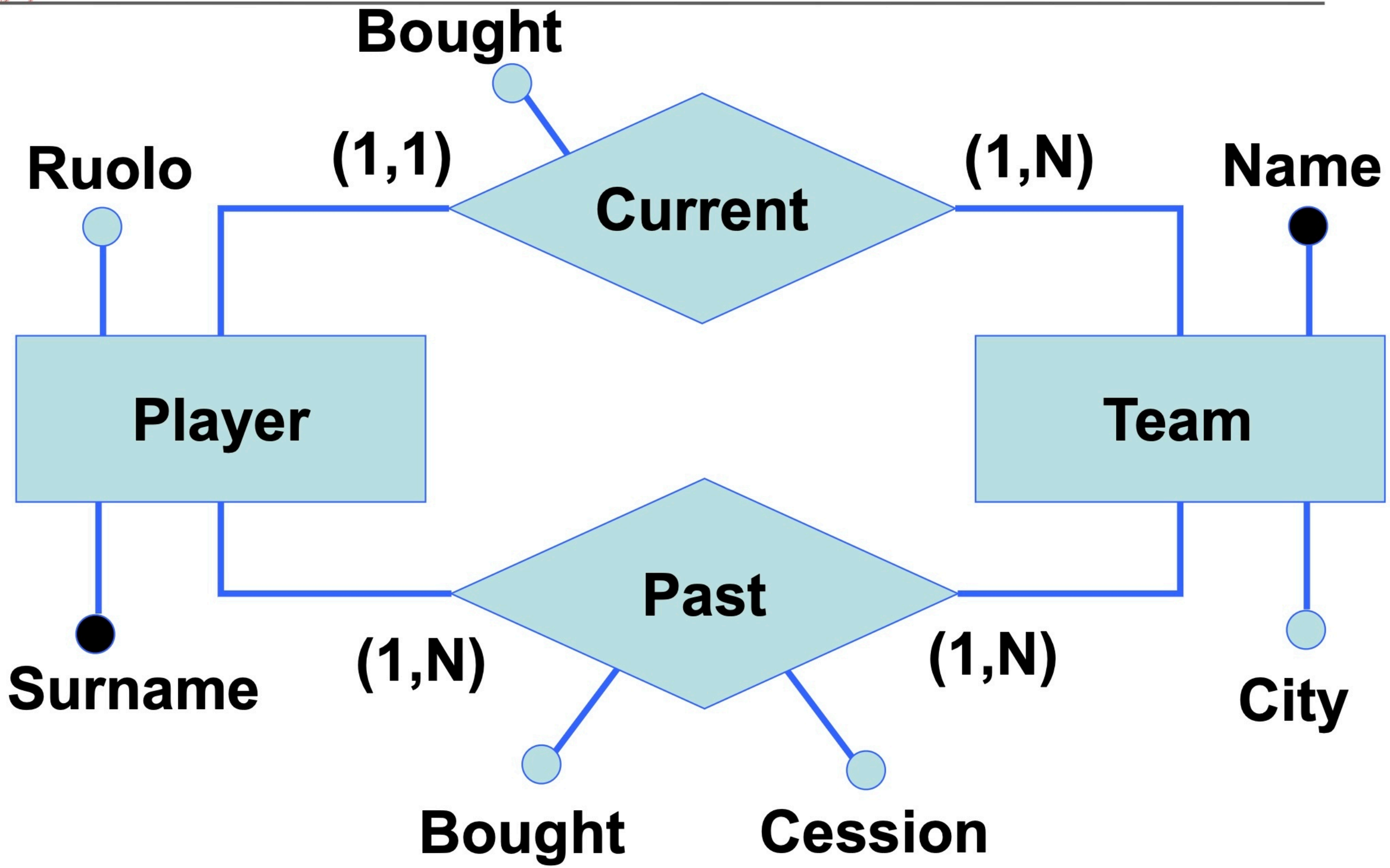

Name
Address
Office
City
(1,N)
Phone


City
Name
Number
(1,N)
(1,1)
Office
Appliance
Phone
Address


Surname
Fiscal code
No.
Addr
(0,1)
(1,1)
Person
Lives
Apartment
Address
Day of birth


Fiscal code
Surname
No.
(0,1)
Address
Person
Day of birth
Addr
(0,1)


Role
Surname
City
Name
(1,N)
(1,N)
Player
PartOf
Team
(0,1)
Bought
Cession


Bought
Ruolo
(1,1)
Current
(1,N)
Name
Player
Team
Surname
(1,N)
Past
(1,N)
City
Bought
Cession

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

Identifying primary keys

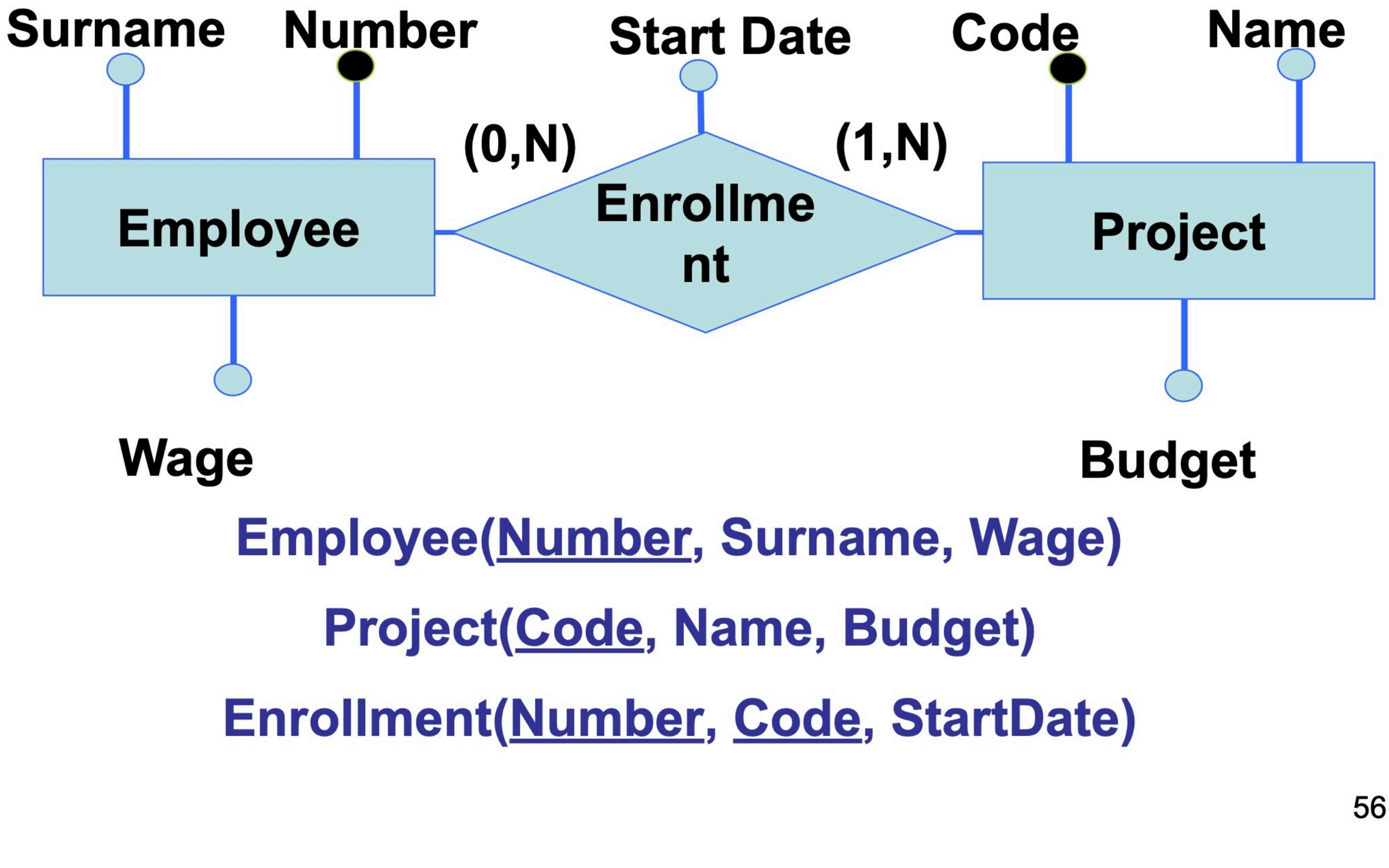
What if none of the aforementioned conditions are met?

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(<u>Number</u>, Surname, Wage)

Project(<u>Code</u>, Name, Budget)

Enrollment(<u>Number</u>, <u>Code</u>, 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, Stipendio)

Project(Code, Name, Budget)

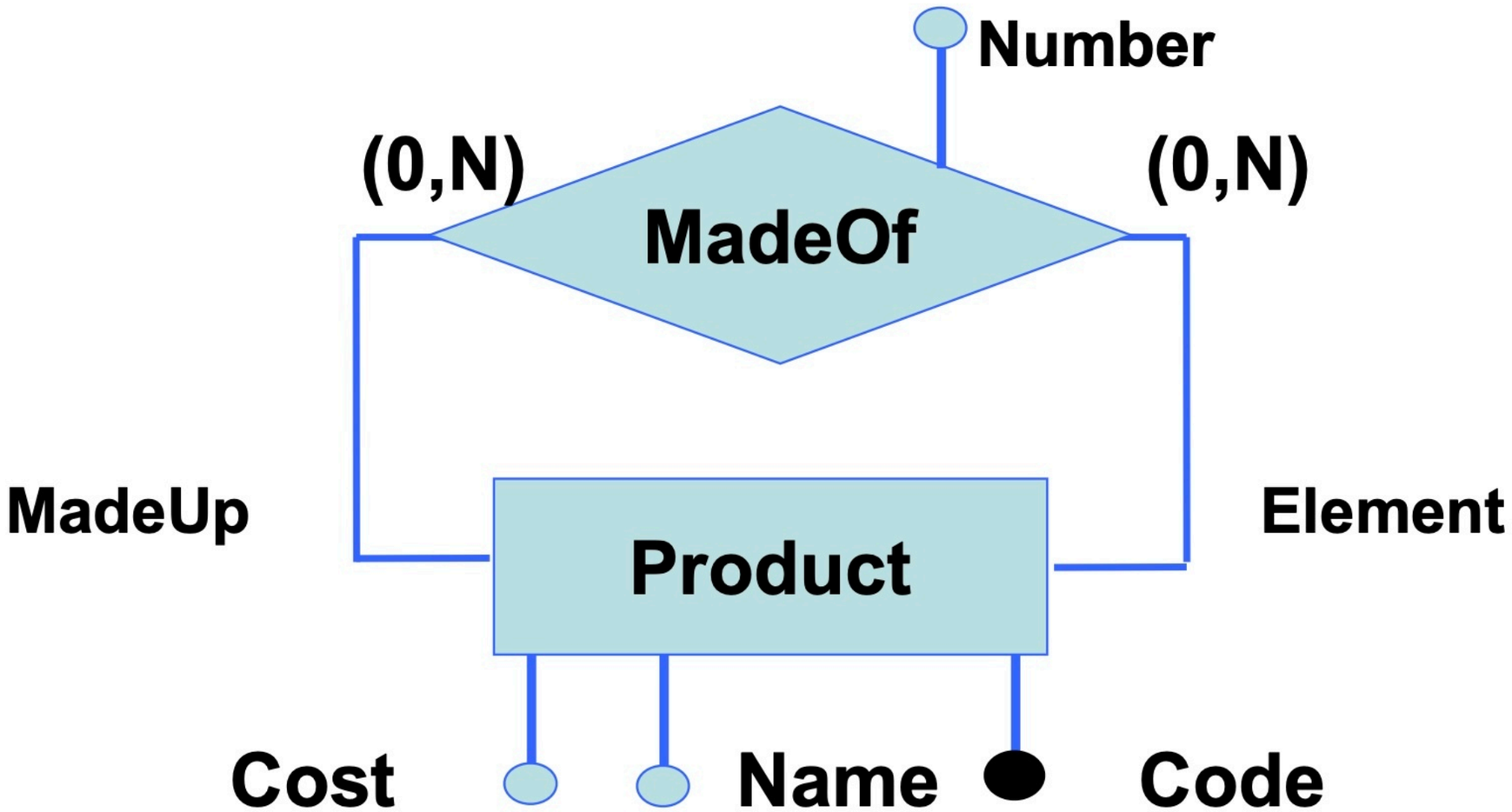
Enrollment(Number, Code, StartDate)

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)

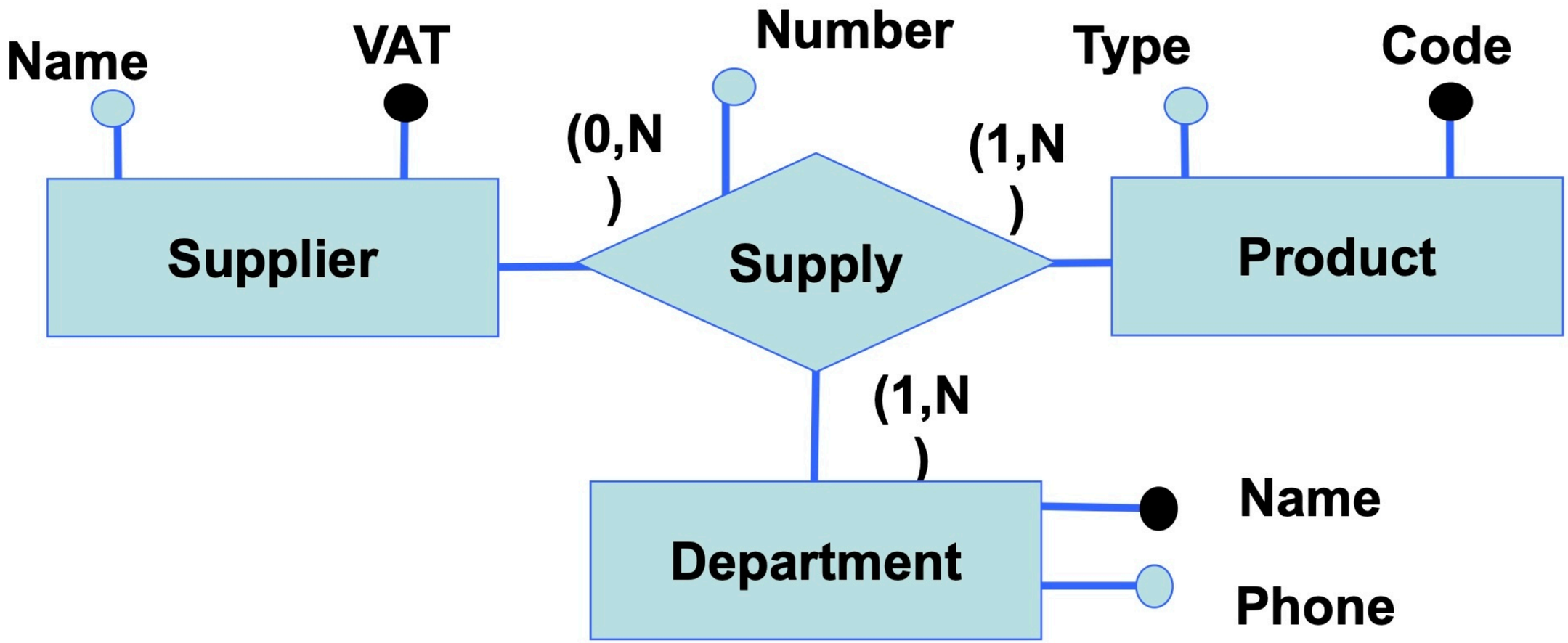
Recursive relationships



Product(<u>Code</u>, Name, Cost)

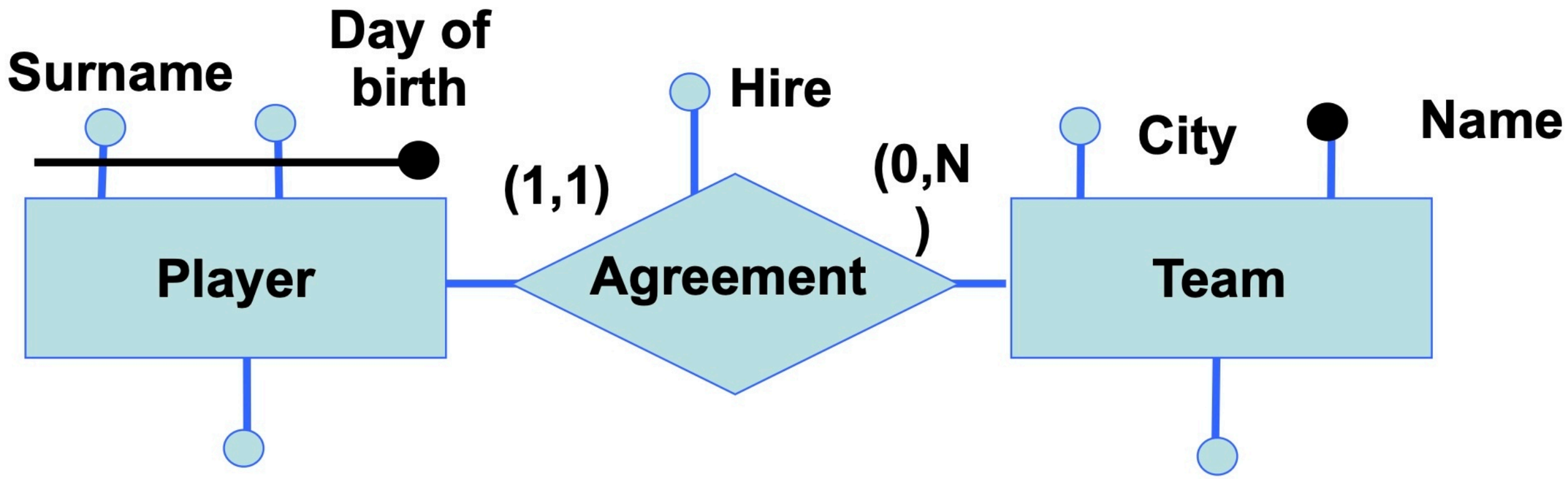
MadeOf(<u>MadeUp</u>, <u>Element</u>, Number)

N-ary Relationships



Supplier(<u>VAT</u>, Name)
Product(<u>Code</u>, Type)
Department(<u>Name</u>, Phone)
Supply(<u>Supplier</u>, <u>Product</u>, <u>Department</u>, Number)

One-to-Many Relationship



Player(Surname, DayOfBirth, Role)
Agreement(SurnameP, BDayP, Team, Hire)
Team(Name, City, TeamColors)

- **correct?**

A less redundant solution

Player(Surname, DayOfBirth, Role)
Agreement(SurnameP, BDayP, Team, Hire)
Team(Name, City, TeamColors)

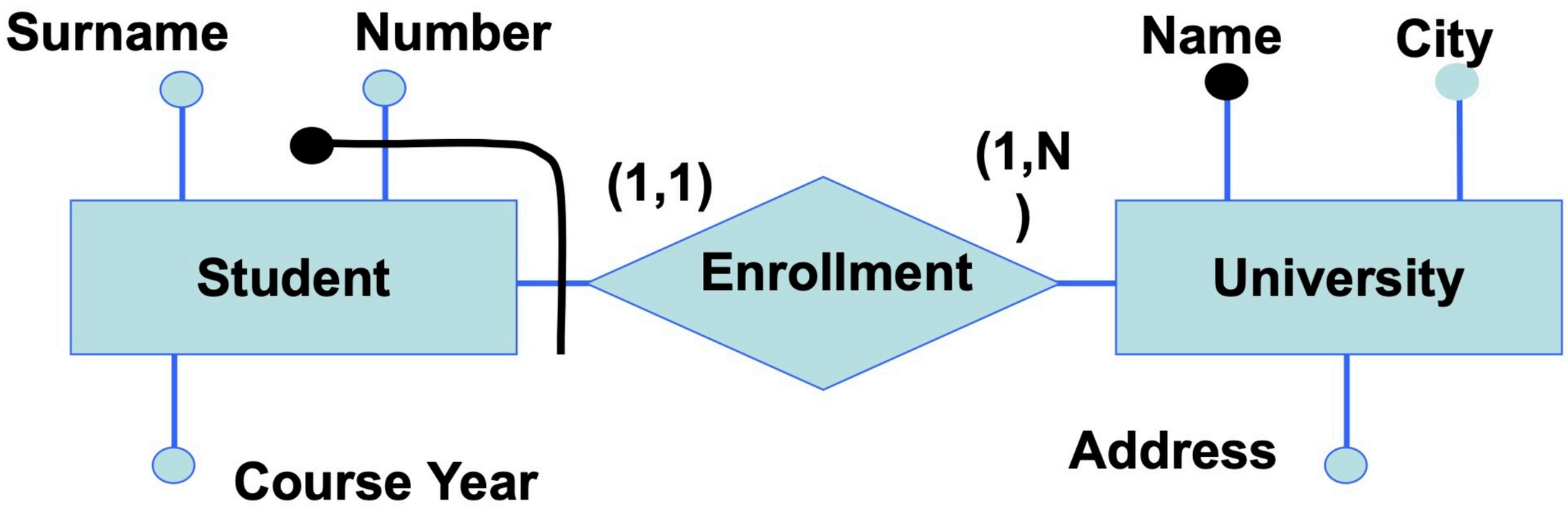
Player(Surname, DayOfBirth, Team, Role, Hire)
Team(Name, City, TeamColors)

- **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 : NULLs allowed
  - 1 : NULLs **not** allowed

Entity with external identifier

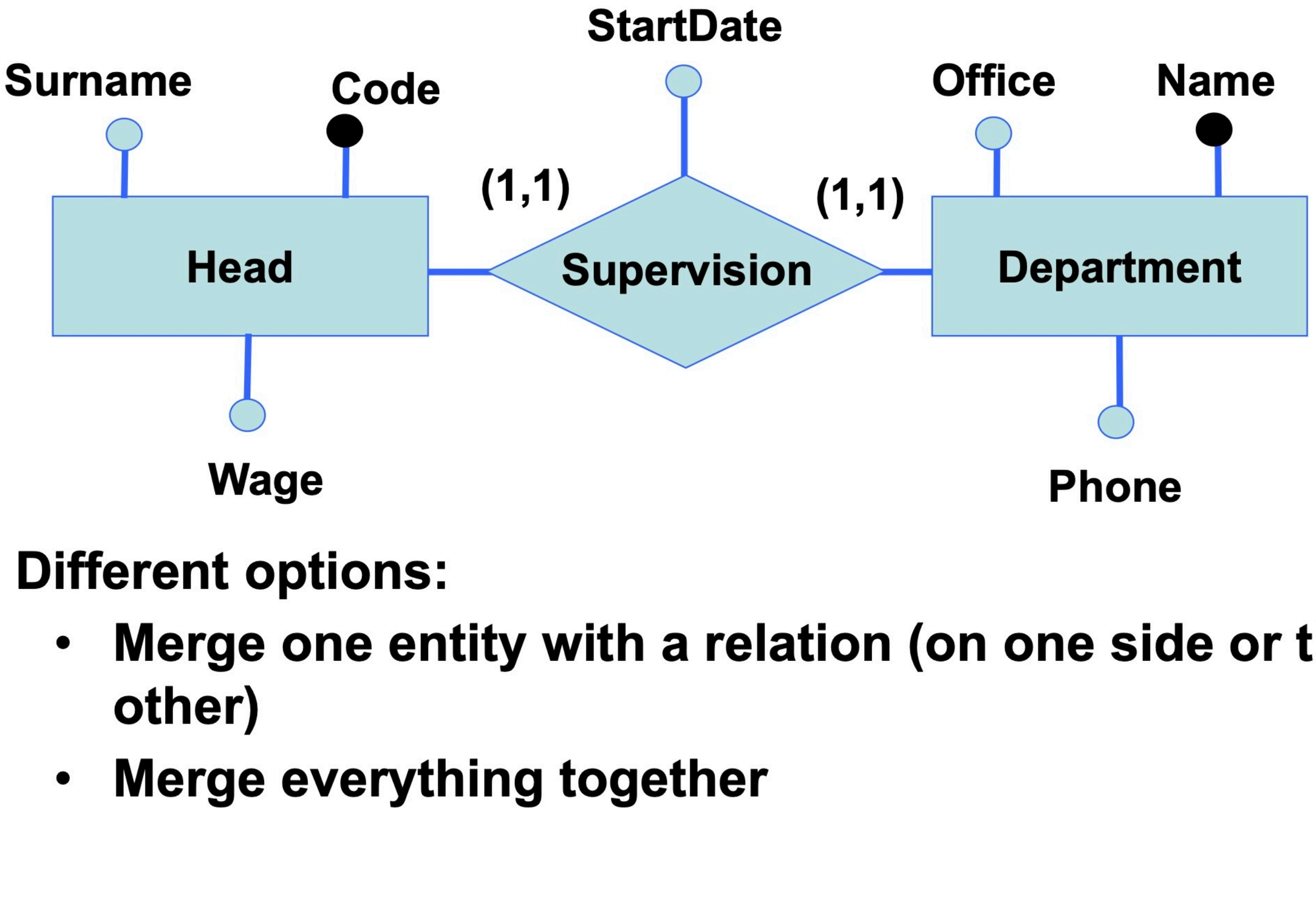


Student(<u>Number, University</u>, Surname, CourseYear)

University(<u>Name</u>, City, Addr)

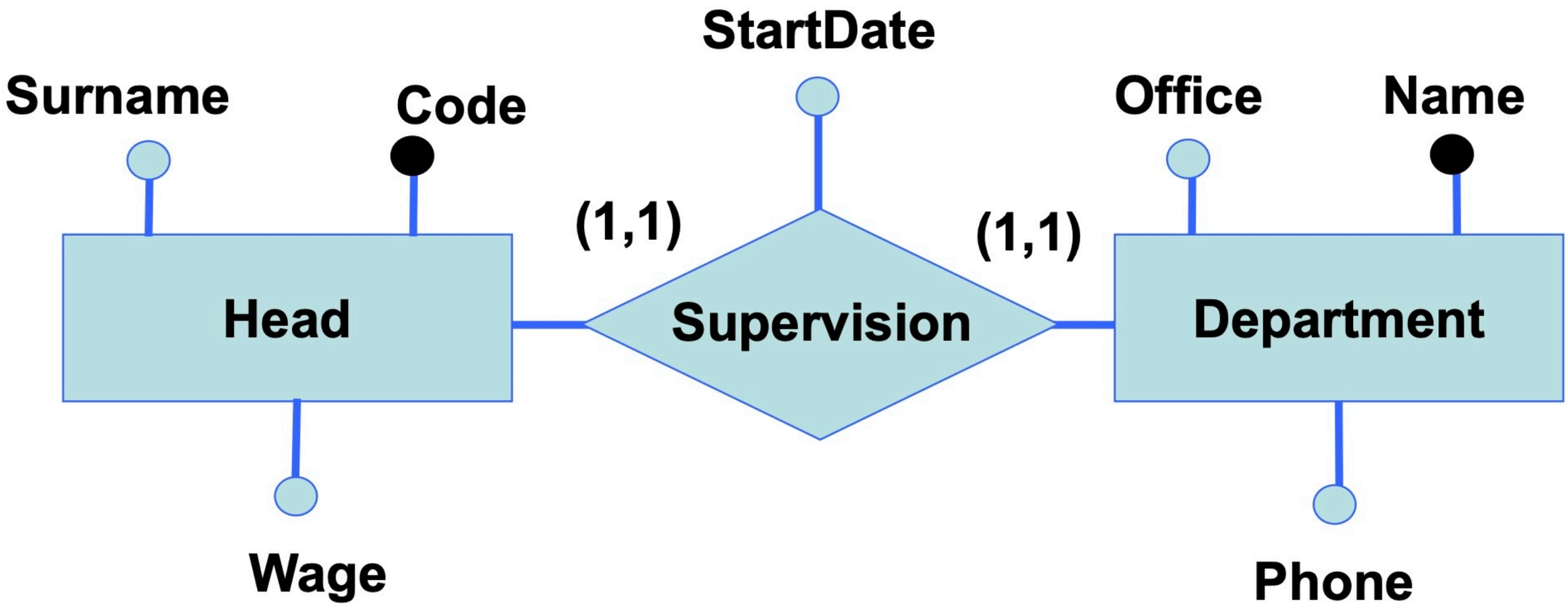
- **Constraint: each student is enrolled to only one university**

One-to-One Relationship



- **Different options:**
  - **Merge one entity with a relation (on one side or the other)**
  - **Merge everything together**

A preferred solution

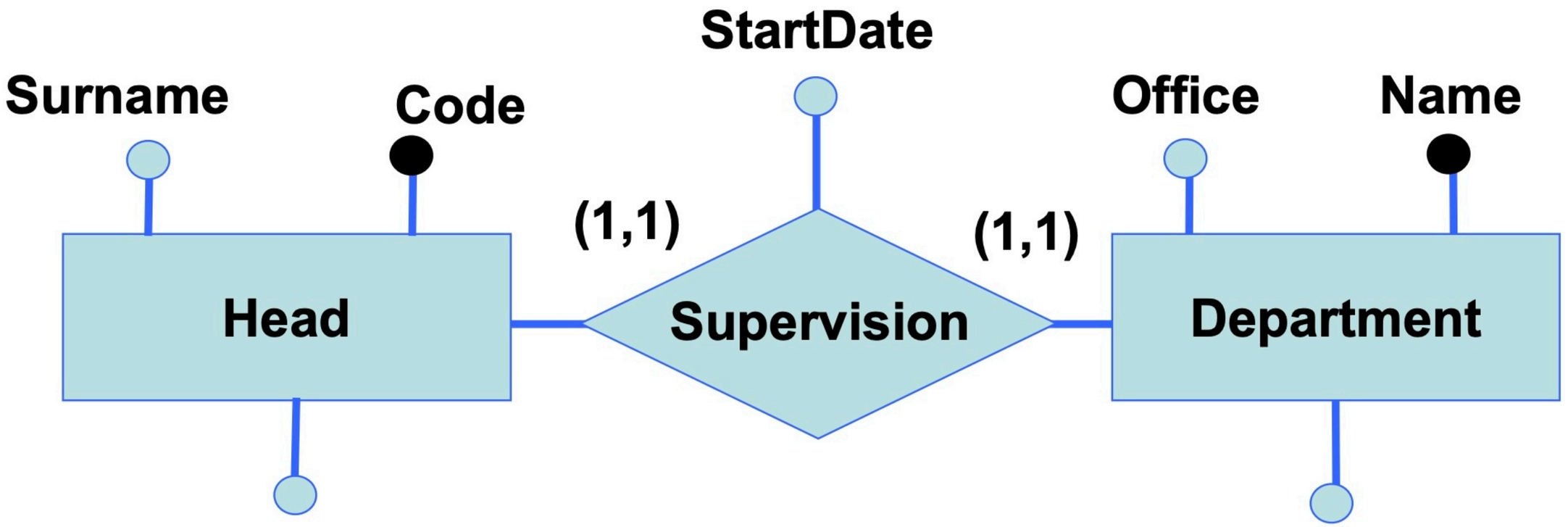


Head(Code, Surname, Wage)

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

- **Referential Integrity Constraint**
- **No NULLs allowed**

Another solution

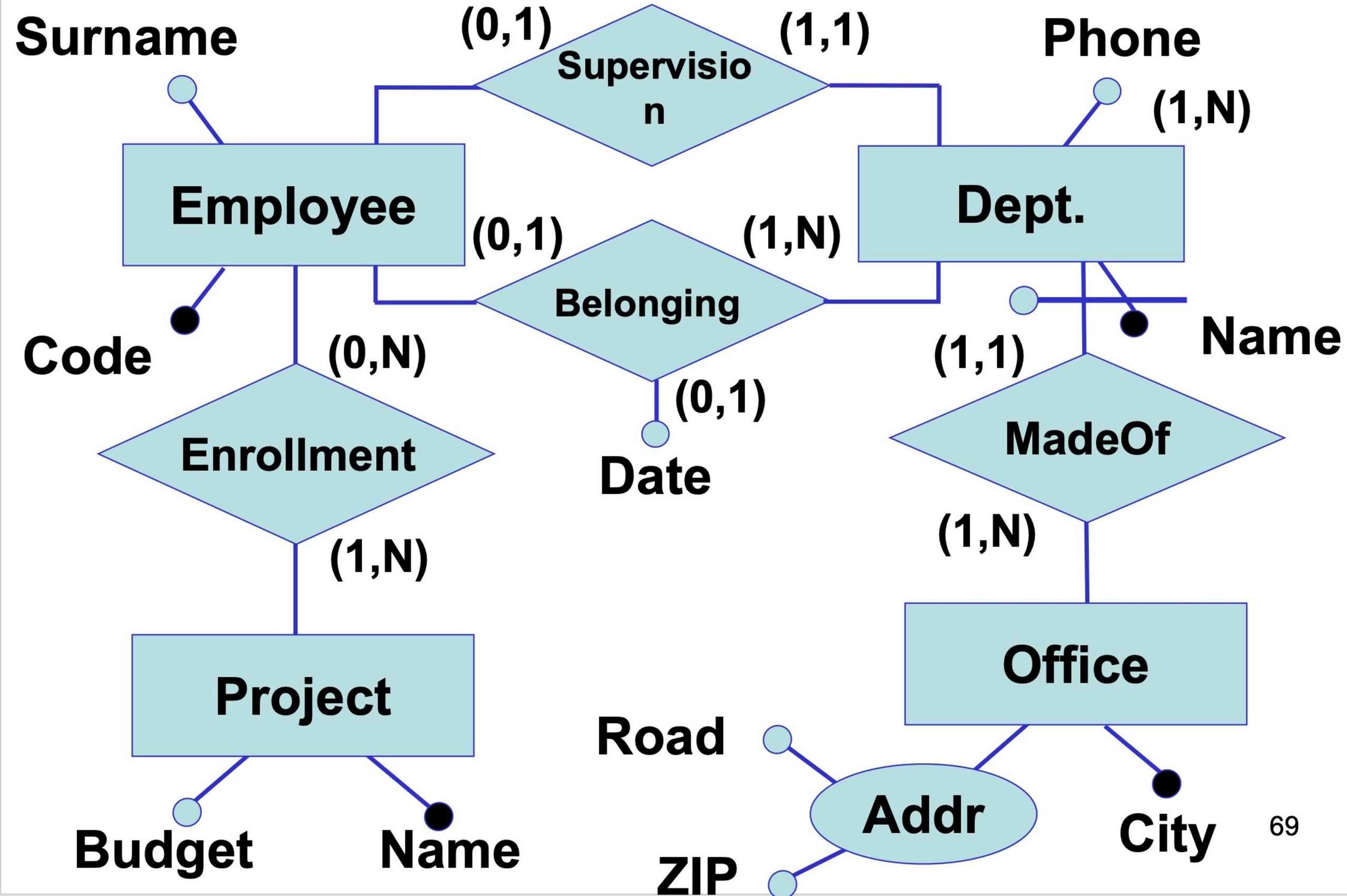


Head(<u>Code</u>, Surname, Wage)

Department(<u>Name</u>, Office, Phone)

Supervision(<u>Head</u>, <u>Department</u>, StartDate)

- Two Referential Integrity Constraints
- No NULLs allowed


Surname
(0,1)
Supervisio
n
(1,1)
Phone
(1,N)
Employee
(0,1)
(1,N)
Dept.
Belonging
Code
(0,N)
(1,1)
Name
(0,1)
Date
Enrollment
MadeOf
(1,N)
(1,N)
Office
Project
Road
Addr
City
ZIP
Budget
Name

Final Schema

Employee(<u>Code</u>, Surname, Department,Office, Date*)

Dept.(<u>Name, City</u>, Phone, Head)

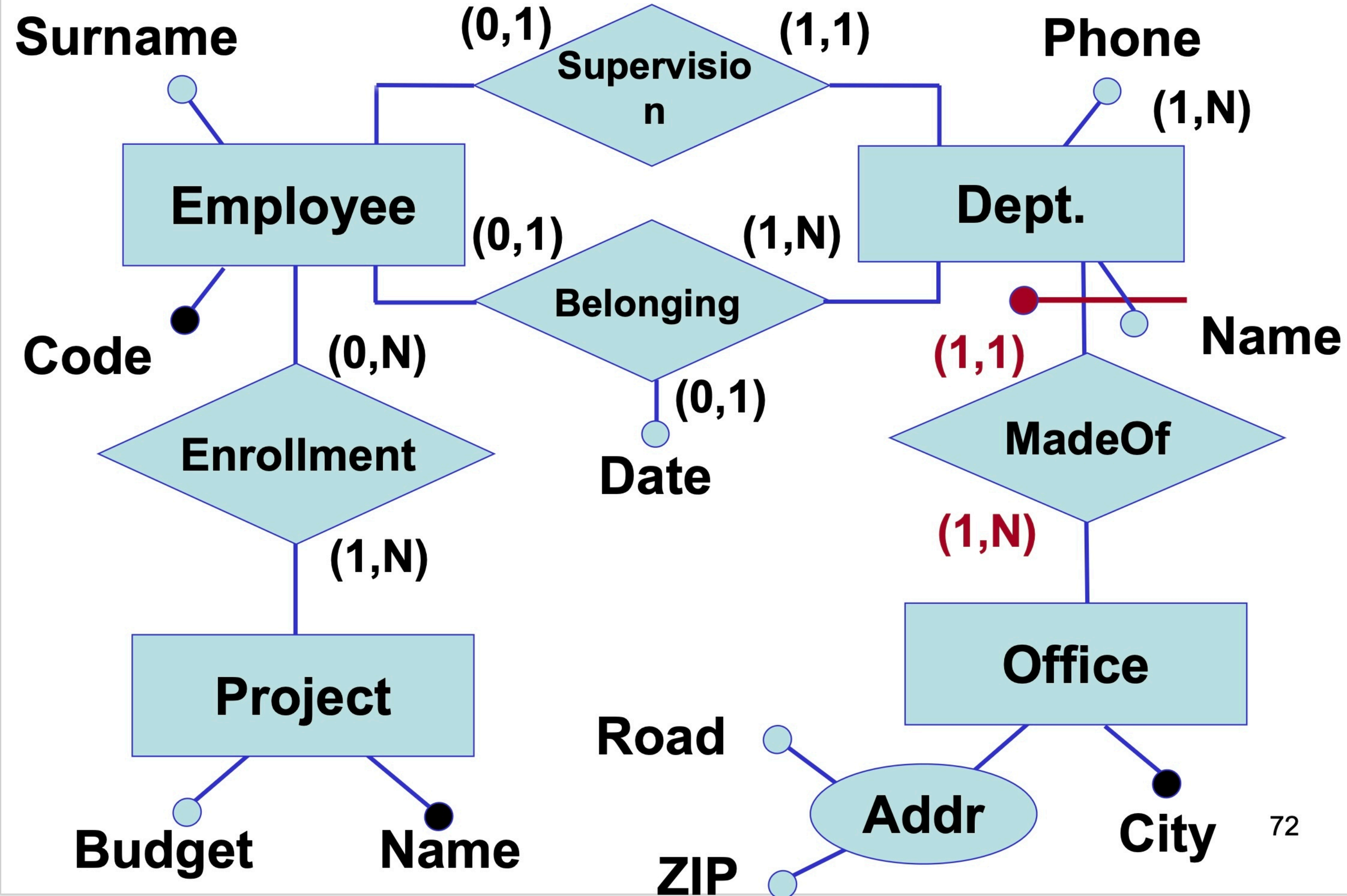
Office(<u>City</u>, Road, Zip)

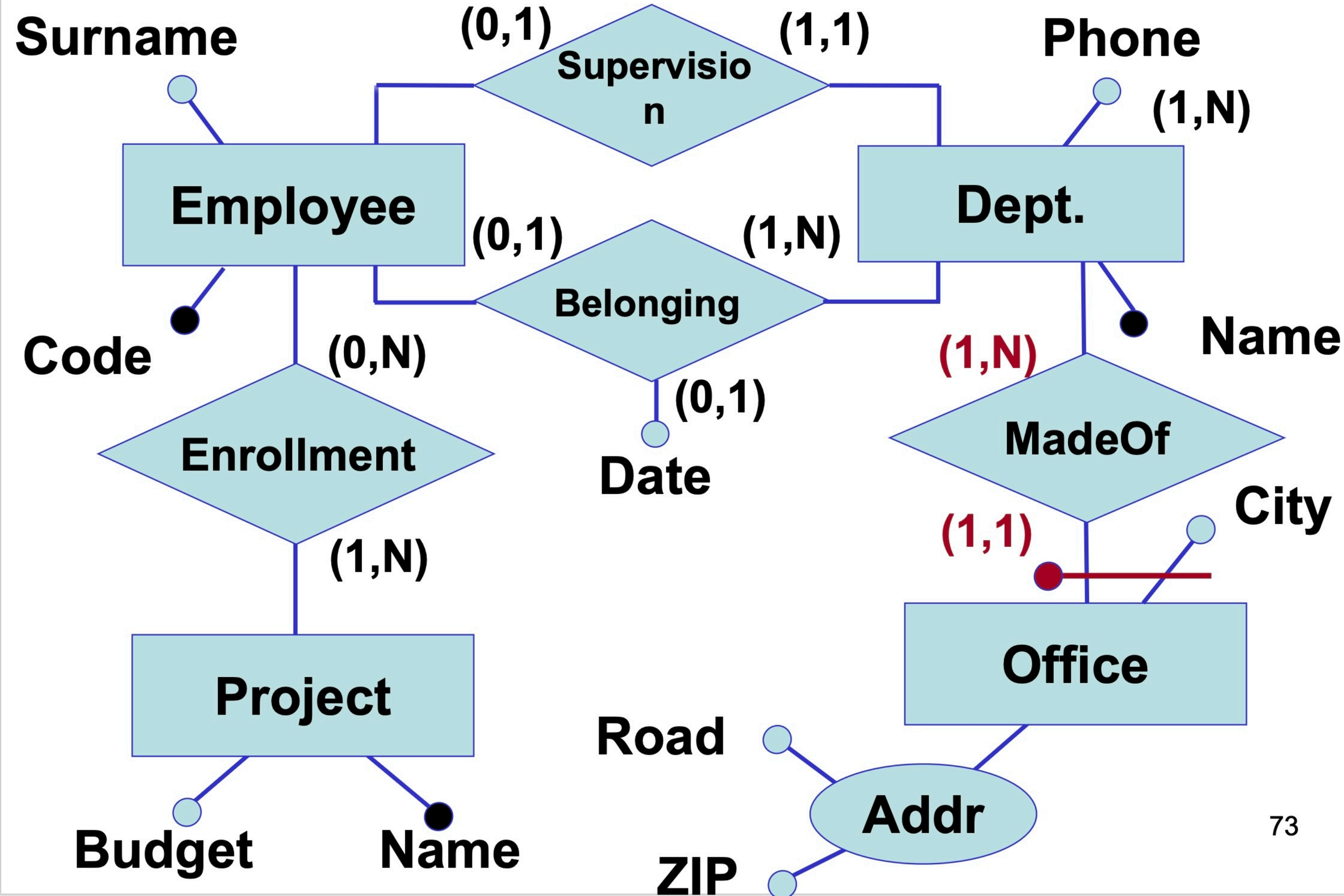
Project(<u>Name</u>, Budget)

Enrollment(<u>Employee</u>, <u>Project</u>)

Warning

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


Surname
(0,1)
Supervisio
n
(1,1)
Phone
(1,N)
Employee
(0,1)
(1,N)
Dept.
Belonging
Code
(0,N)
(1,1)
Name
Enrollment
(0,1)
Date
MadeOf
(1,N)
(1,N)
Office
Project
Road
Addr
City
Budget
Name
ZIP


Surname
(0,1)
Supervisio
n
(1,1)
Phone
(1,N)
Employee
(0,1)
(1,N)
Dept.
Belonging
Code
(0,N)
(1,N)
Name
(0,1)
Date
Enrollment
MadeOf
City
(1,N)
(1,1)
Office
Project
Road
Addr
Budget
Name
ZIP

Second final schema

Employee(Code, Surname, Dept.,Office, Date*)

Dept.(Name, City, Phone, Head)

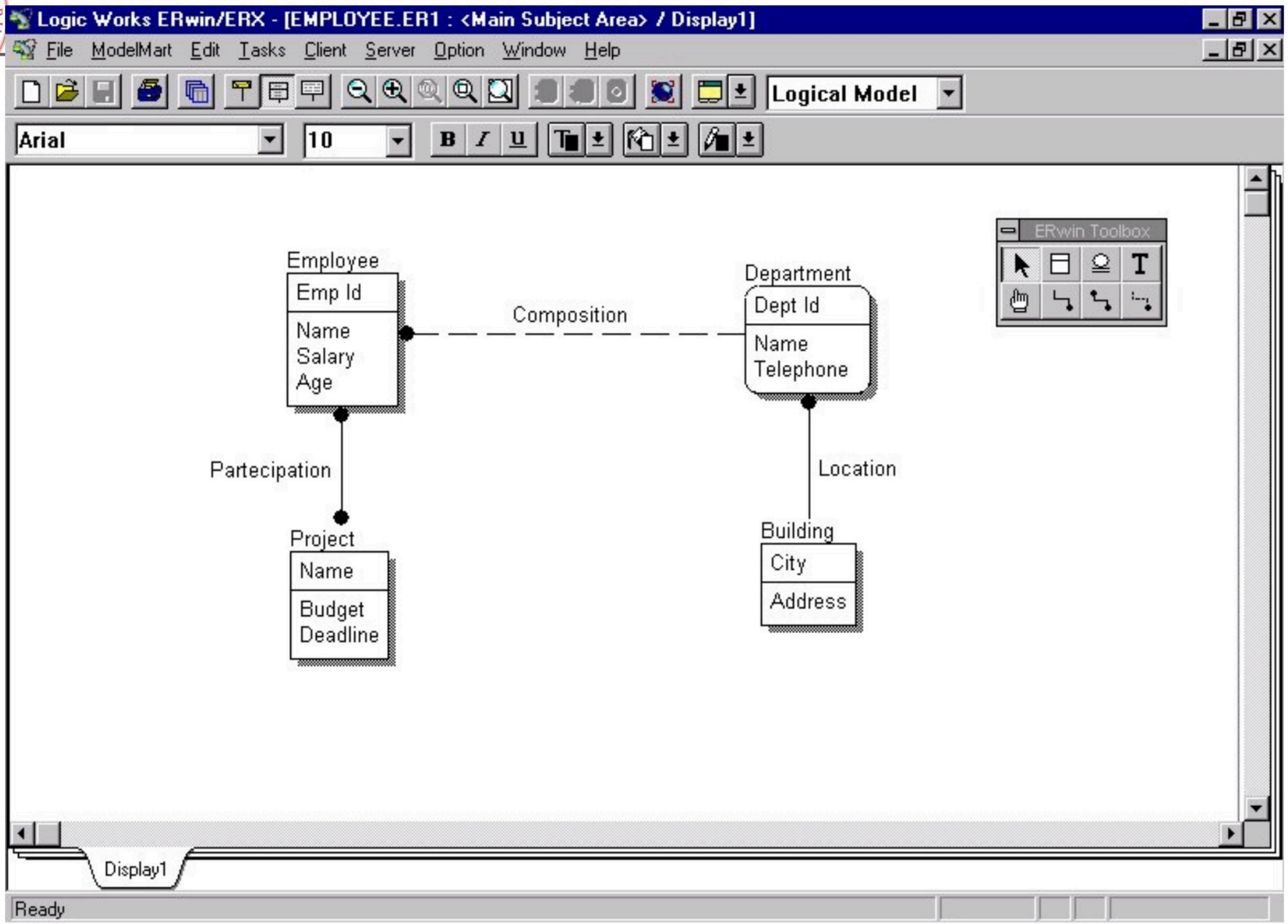
Office(City, Dept., Road, Zip)

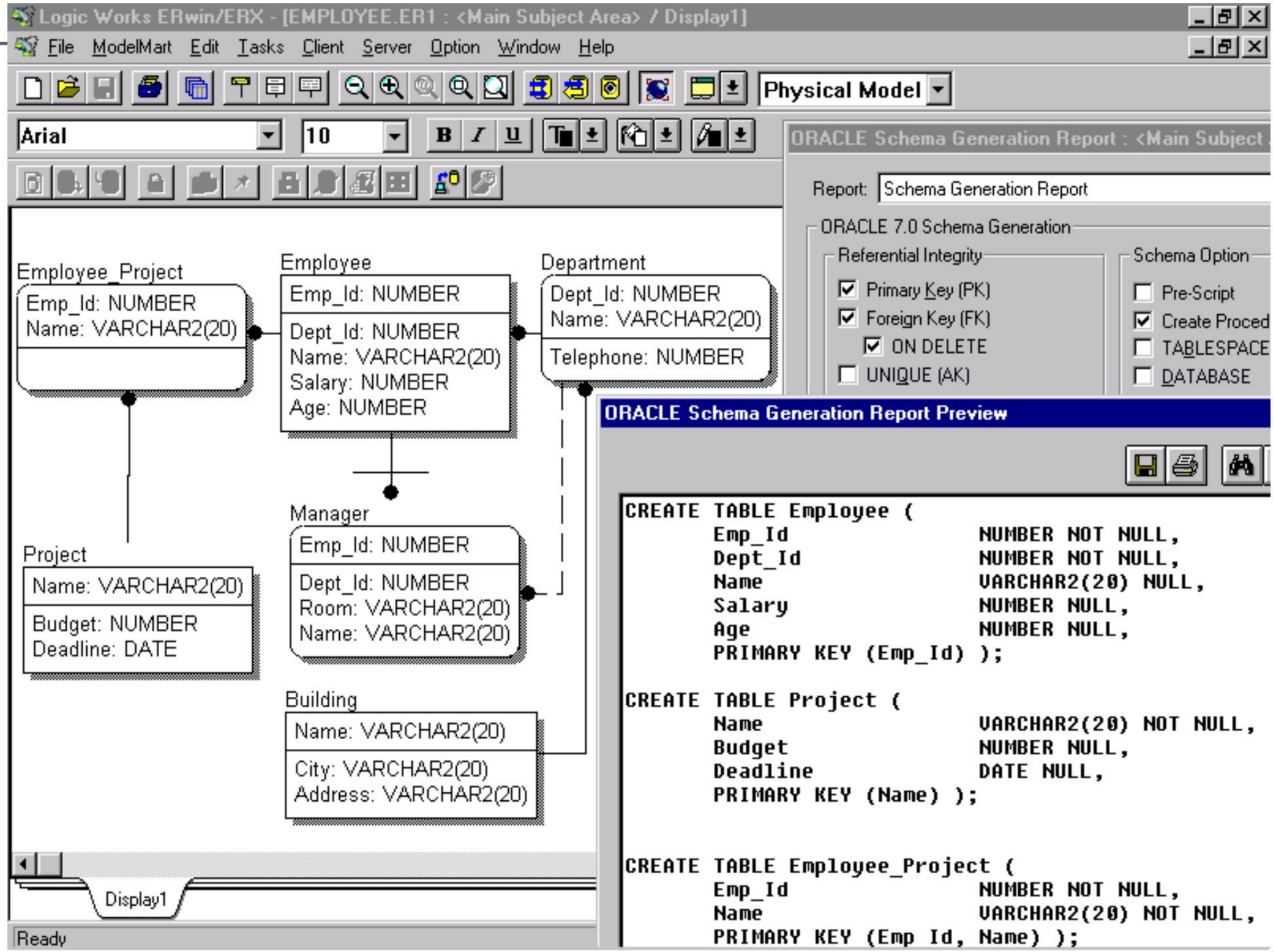
Project(Name, Budget)

Enrollment(Employee, Project)

Tools

- There are some CASE off-the-shelf software that provide support throughout all the modelling phases of designing a database


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


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));