Faculty of Information, Computer Science and Statistics

Bachelor in Applied Computer Science and Artificial Intelligence

Data Management and Analysis, Unit 2

2021-2022

Prof. Giuseppe Pirrò
Department of Computer Science

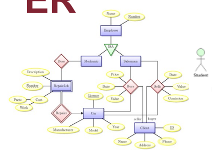
Restructuring of the ER model

SAPIENZA
UNIVERSITÀ DI ROMA

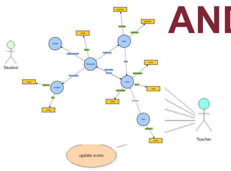
Design of a database

Conceptual schema

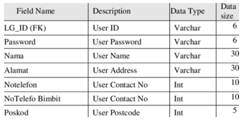
ER



ONTOLOGY



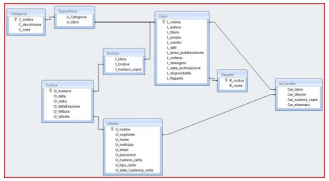
DATA DICTIONARY AND CONSTRAINTS

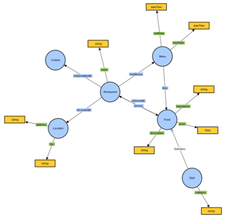


Field Name	Description	Data Type	Data size
LG_ID (FK)	User ID	Varchar	6
Password	User Password	Varchar	6
Nama	User Name	Varchar	30
Alamat	User Address	Varchar	30
Notelefon	User Contact No	Int	10
NoTelefo Bimbit	User Contact No	Int	10
Poskod	User Postcode	Int	5

Logical schema

Depends on the data model





Physical schema

Depends on the DBMS

DBMS

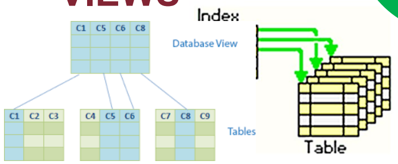
MySQL

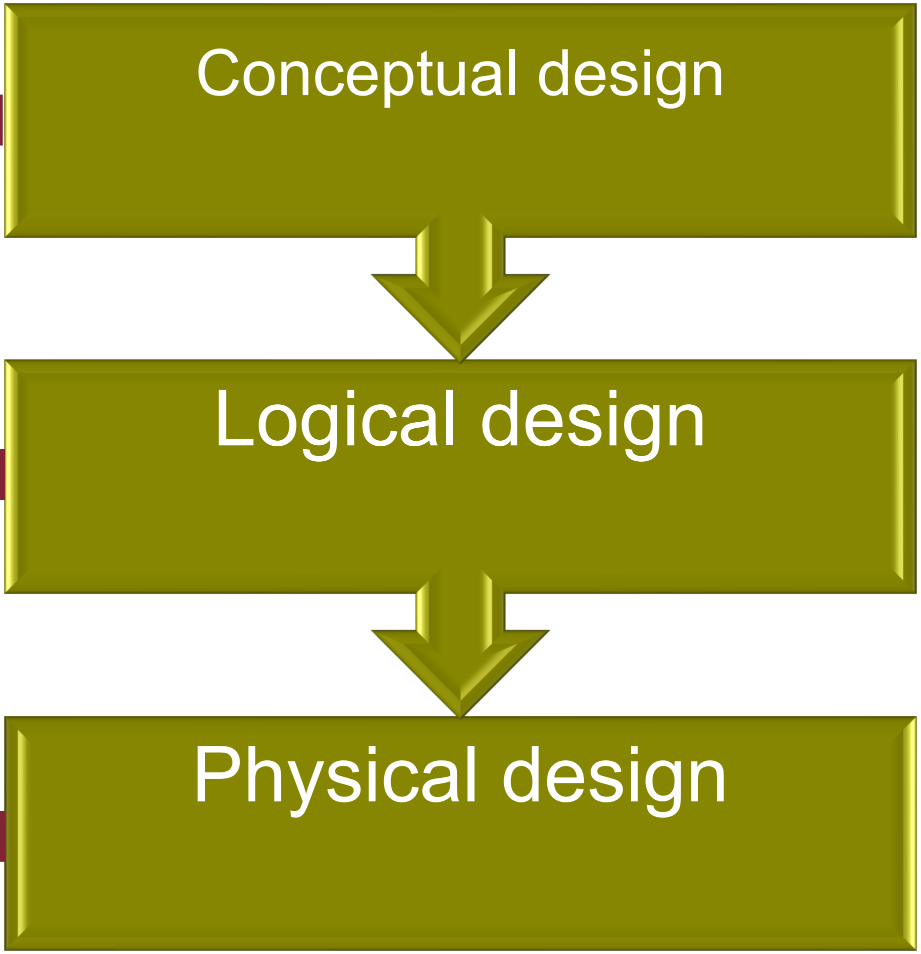
Amazon DynamoDB

cassandra

VIEWS

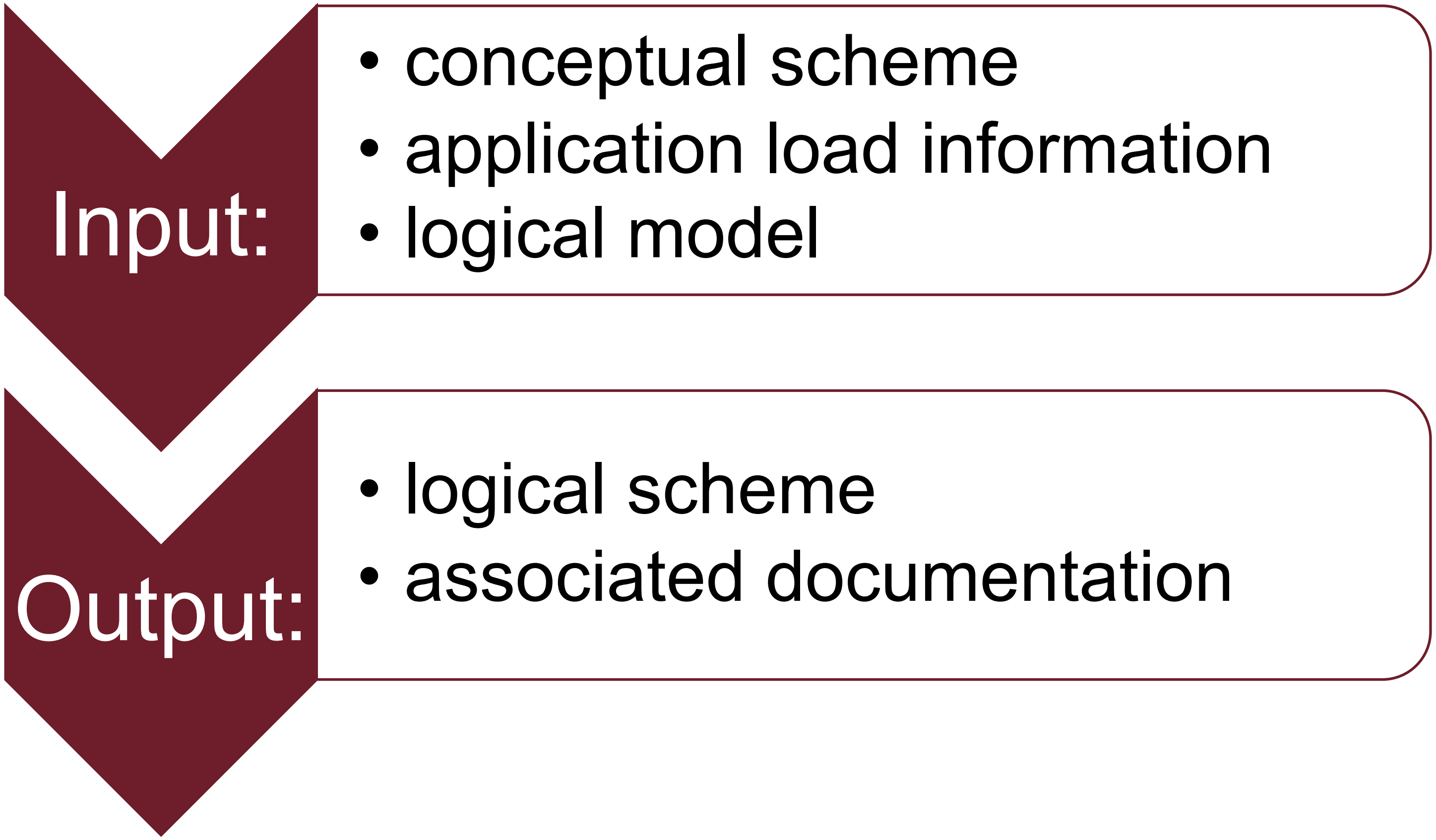
INDEXES





Objective

- Translate the conceptual schema into a logical schema that represents the same data correctly and efficiently



Input:

- conceptual scheme
- application load information
- logical model

Output:

- logical scheme
- associated documentation

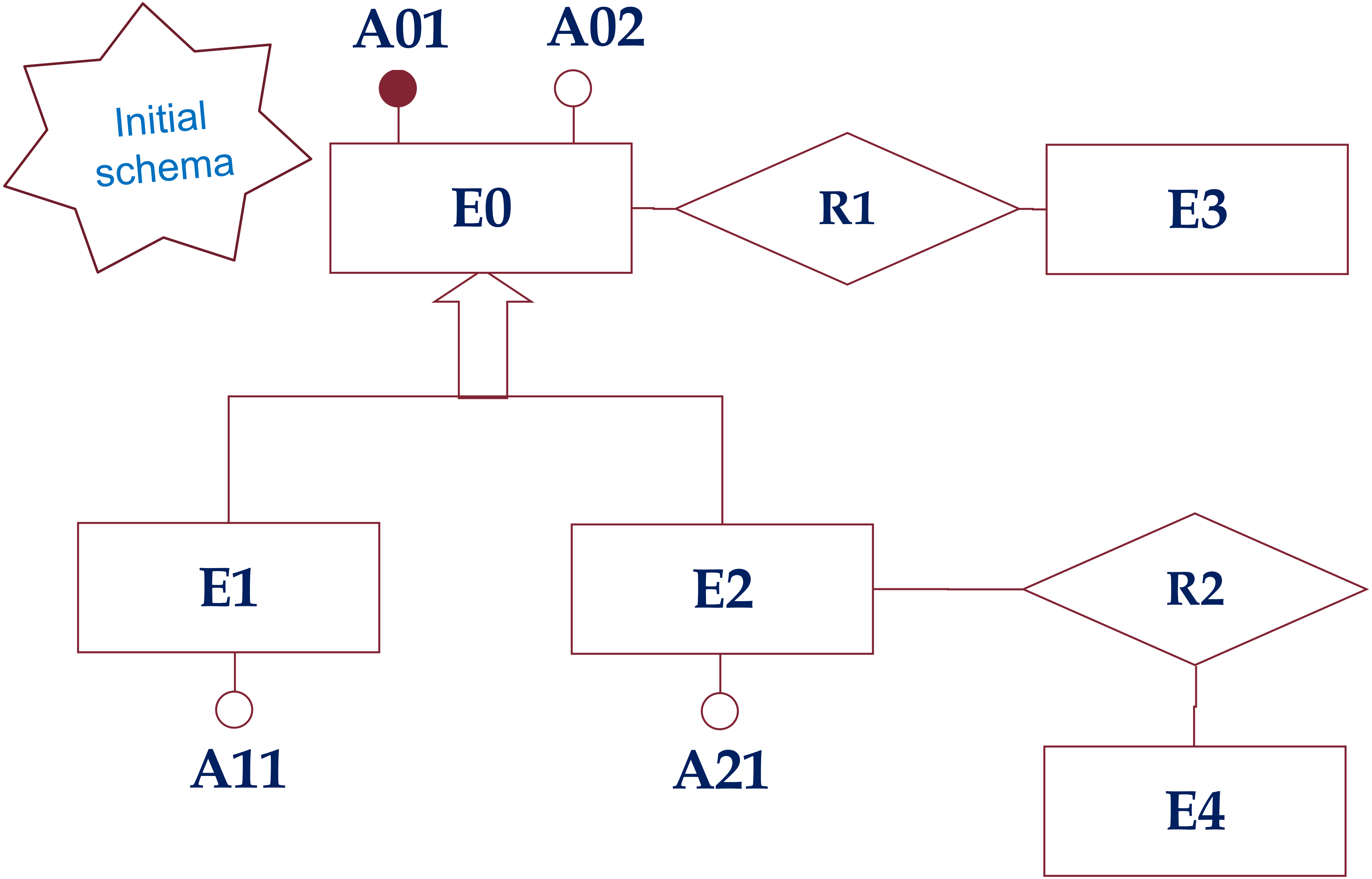
Elimination of generalizations

Elimination of generalizations

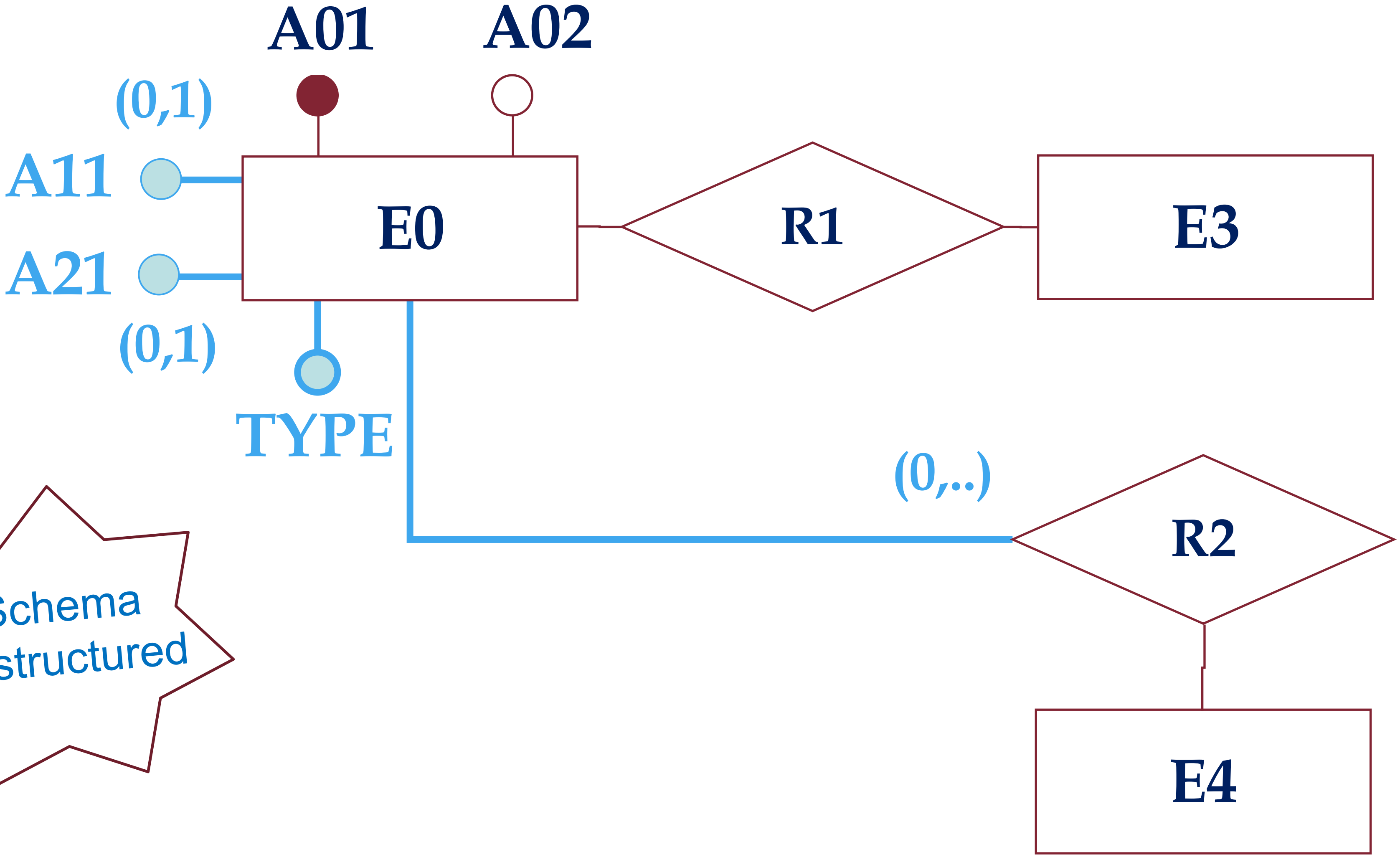
- The relational model cannot directly represent generalizations
  - entities and relationships are directly representable.
- Henece, hierarchies are eliminated and replaced with entities and relationships.

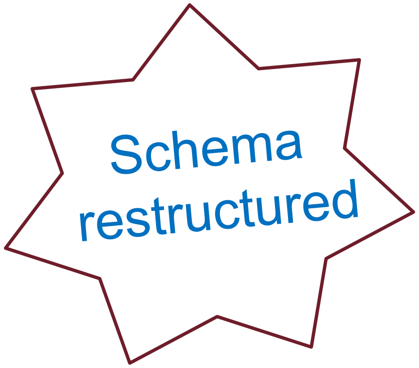
Elimination of generalizations

- There are three possible ways:
  - Method 1: Merging children of the generalization into the parent
  - Method 2: Merging the parent into the child entities
  - Method 3: Replacing generalization with relationships

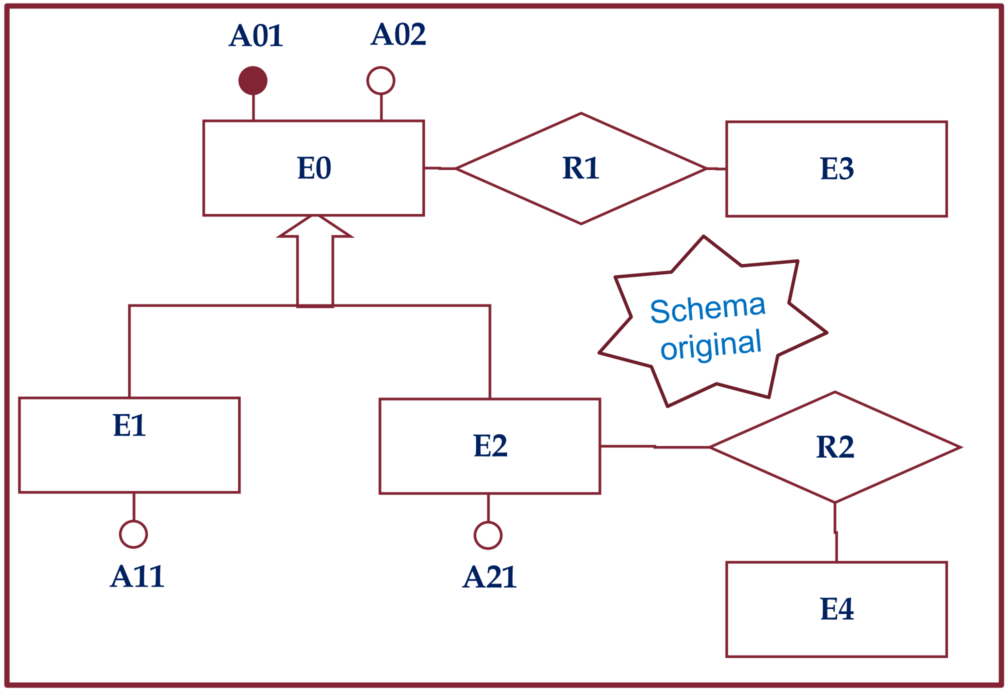

Initial schema
A01
A02
E0
R1
E3
E1
E2
R2
A11
A21
E4

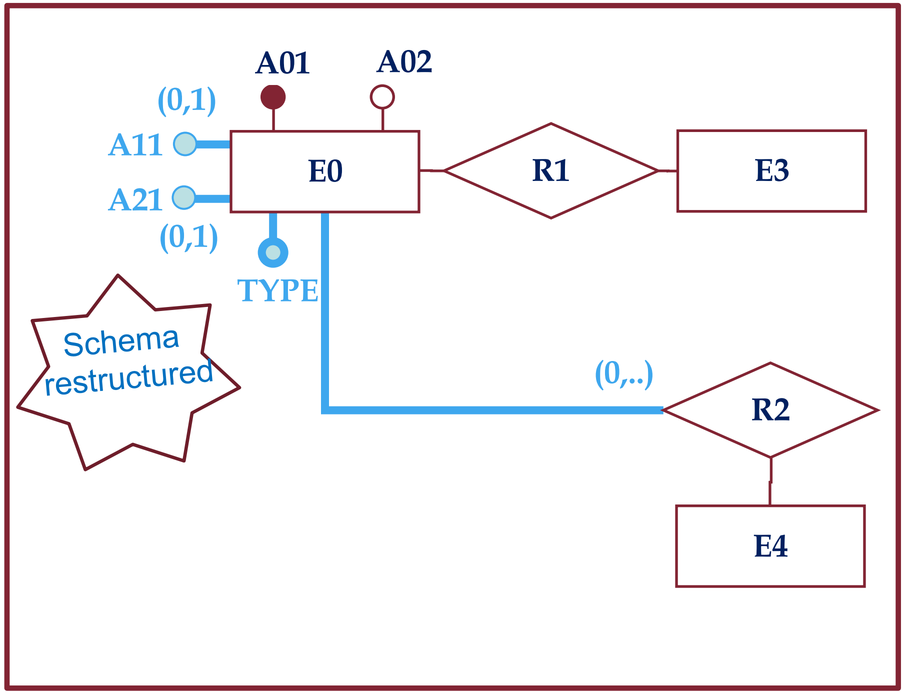
Method 1: Merging Child Entities into the Parent



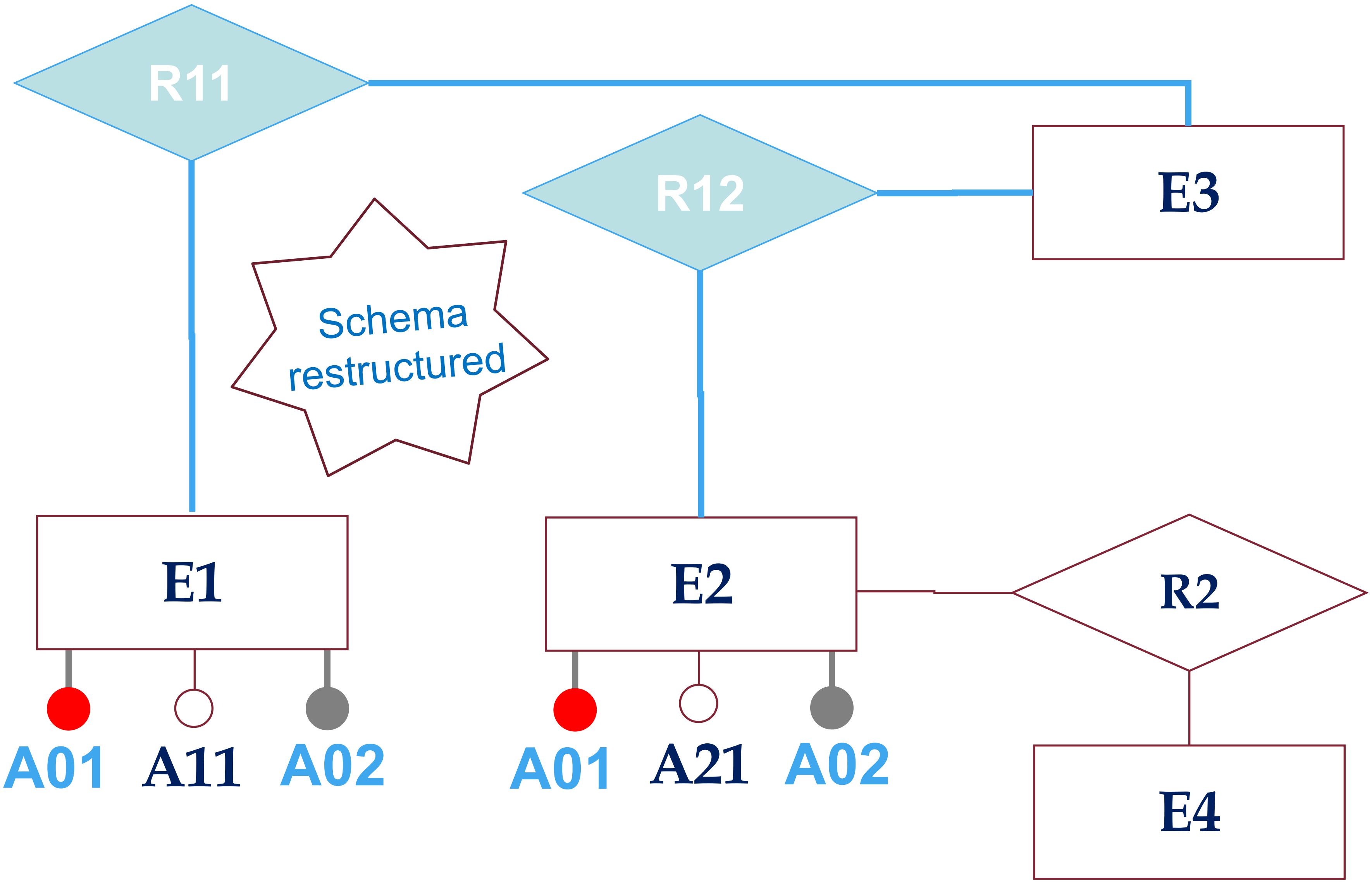


Method 1: Merging Child Entities into the Parent

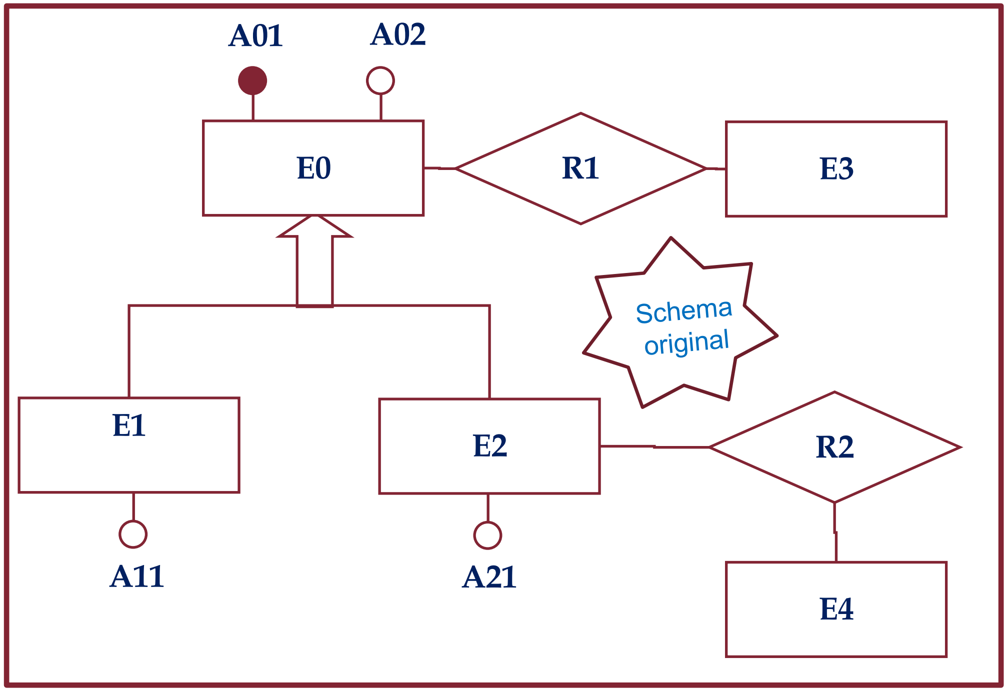


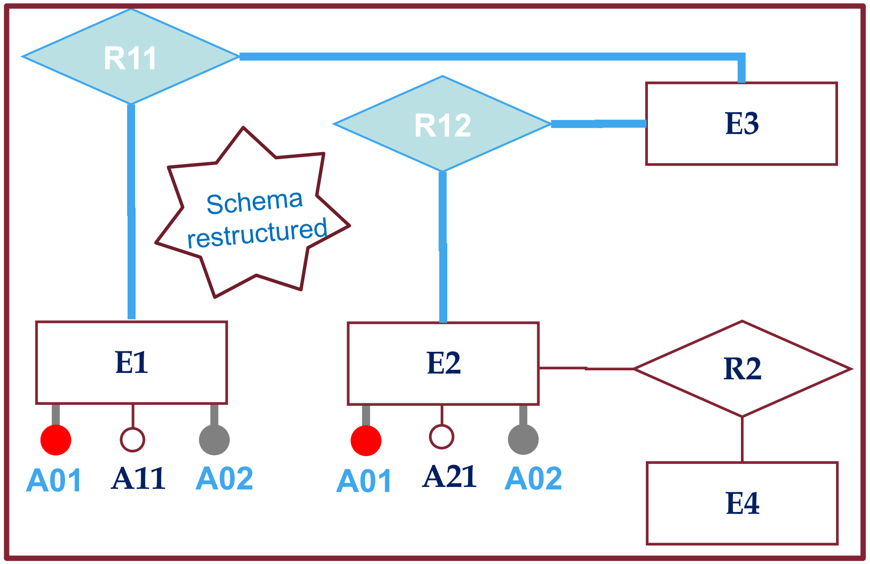


Method 2: Merging the parent into the child entities

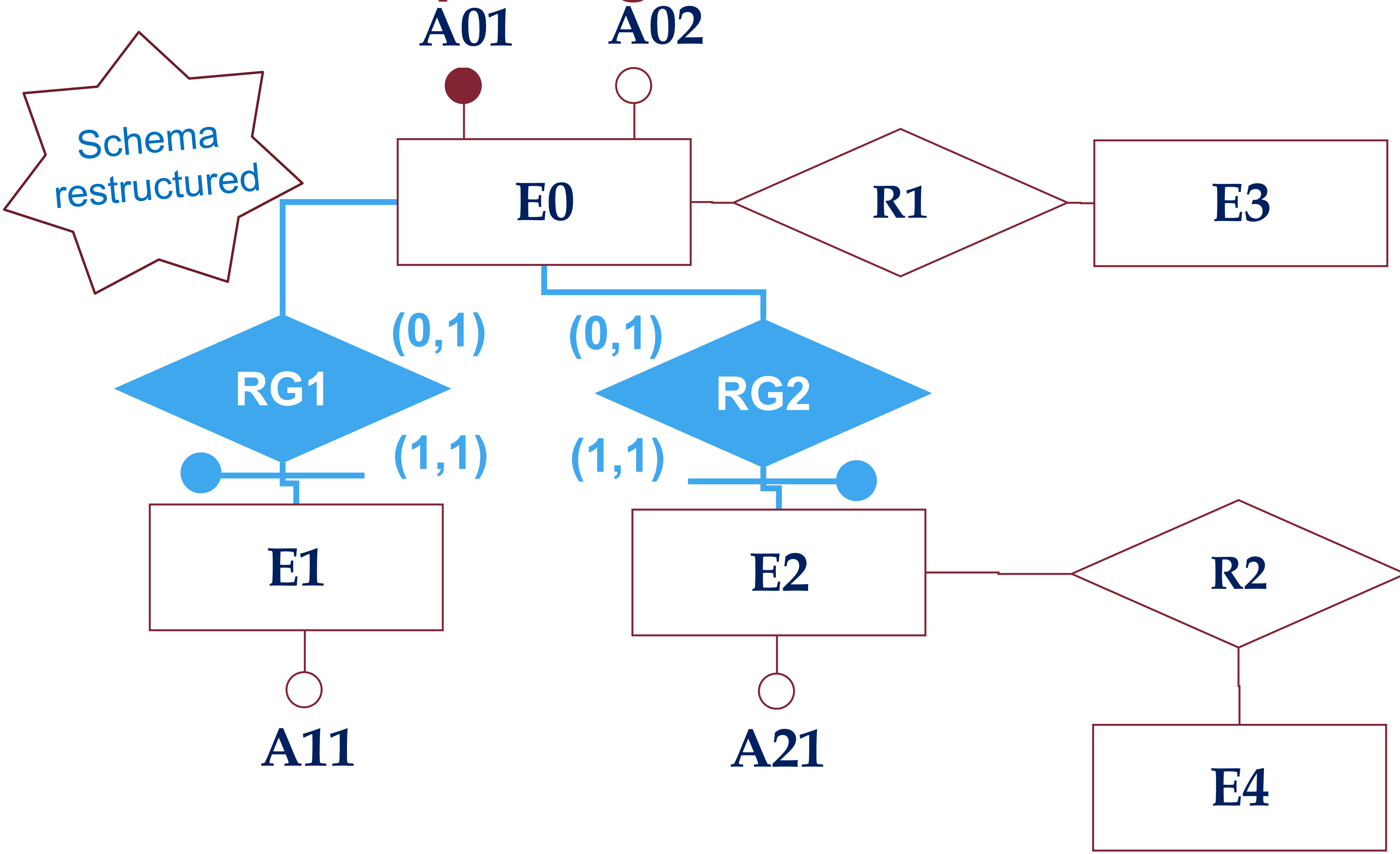


Method 2: Merging the parent into the child entities

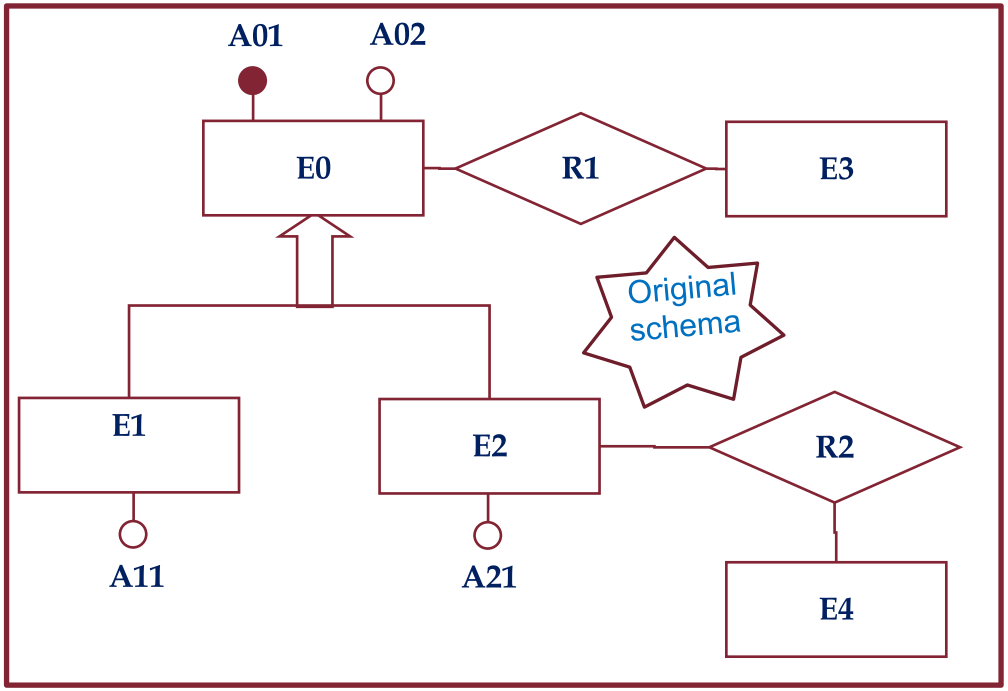


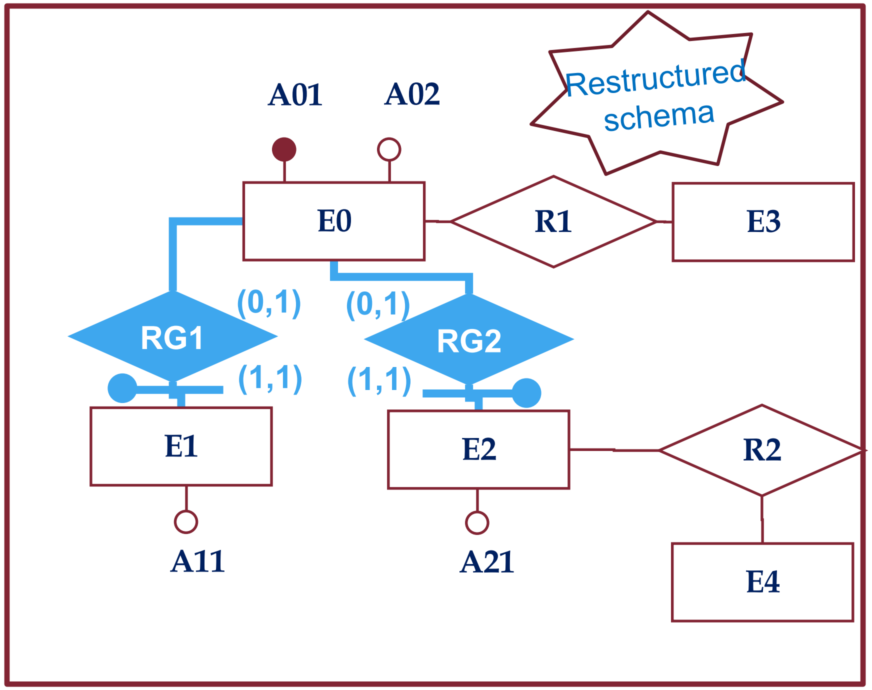


Method 3: Replacing Generalizations



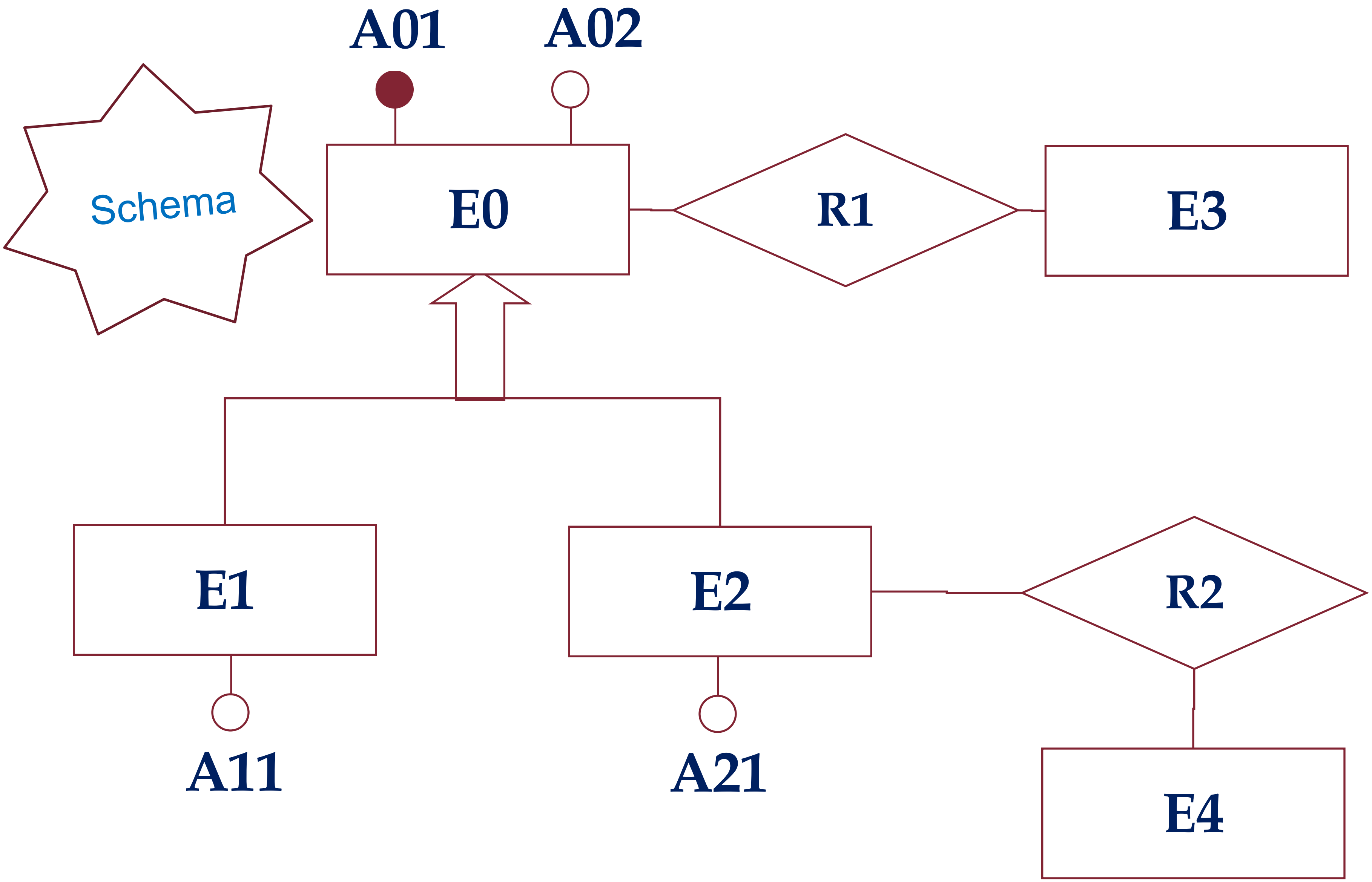
Method 3: Replacing Generalizations

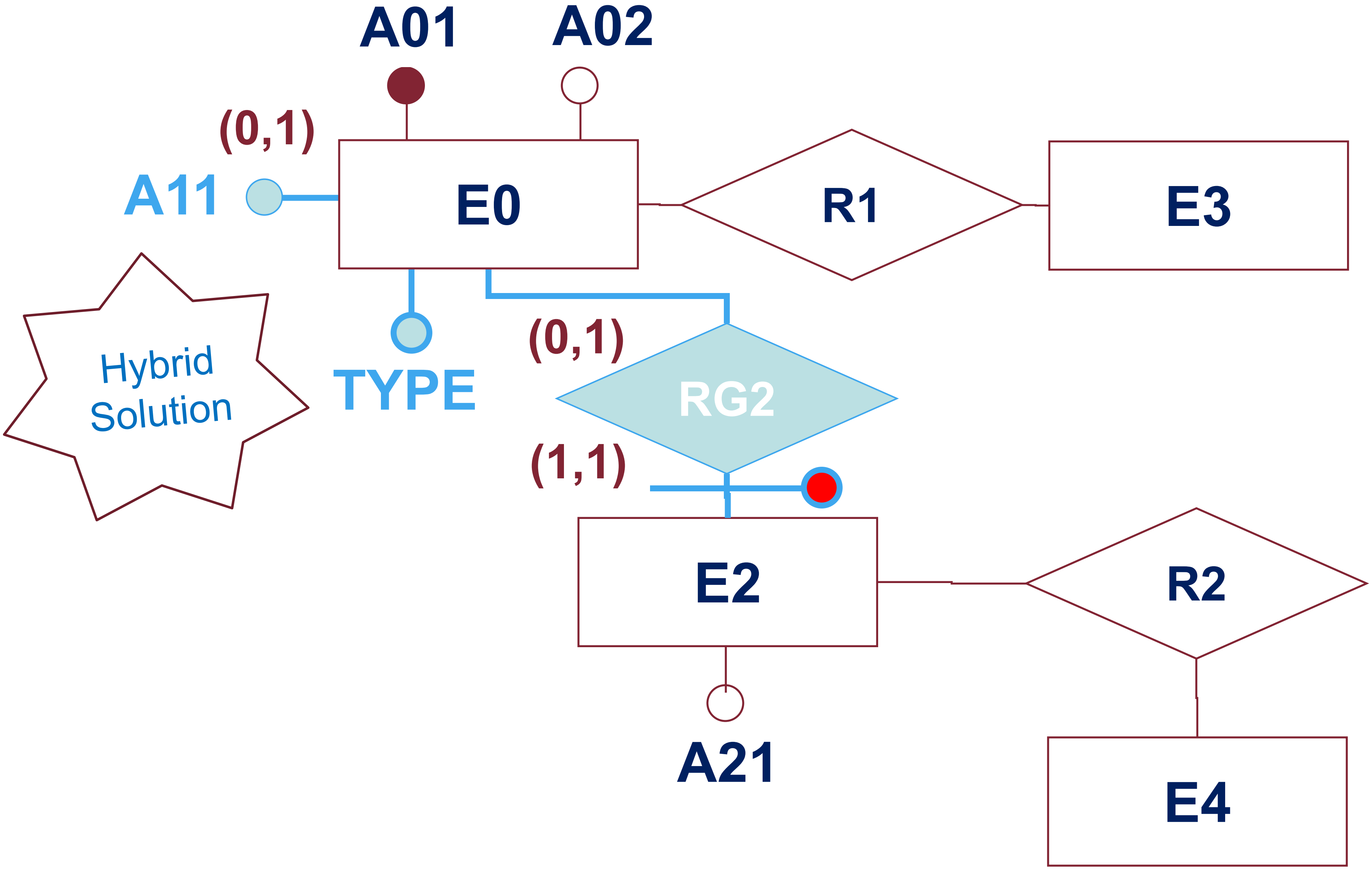




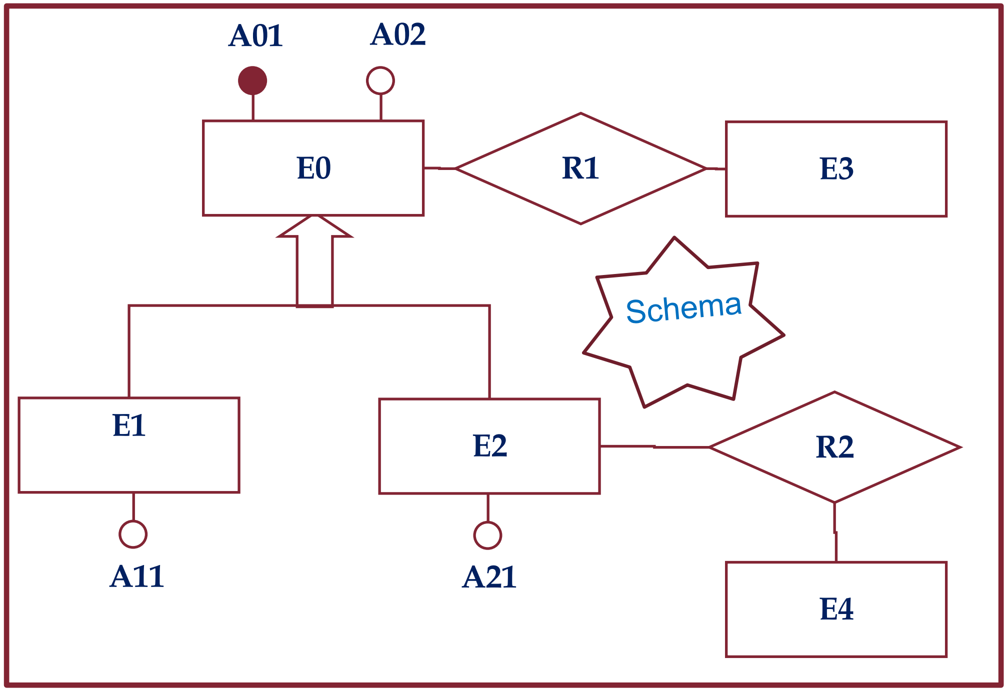
Elimination of generalizations

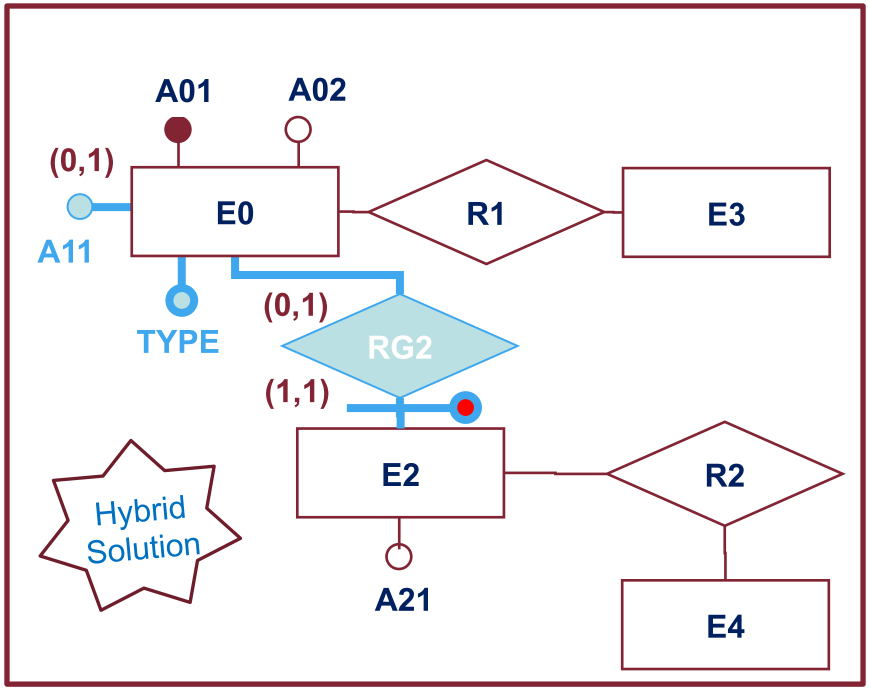
- To choose the following simple rules:
- Method 1 (merging child entities into the parent) is convenient if access to the father and children is contextual
- Method 2 (merging the parent into the child entities) is appropriate if the accesses to the children are distinct
- Method 3 (substitution with relationships) is appropriate if accesses to child entities are separated from accesses to the parent
- For multi-level hierarchies, "hybrid" solutions are possible


A01
A02
Schema
E0
R1
E3
E1
E2
R2
A11
A21
E4


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

Hybrid solution





Partitioning/merging entities and relationships

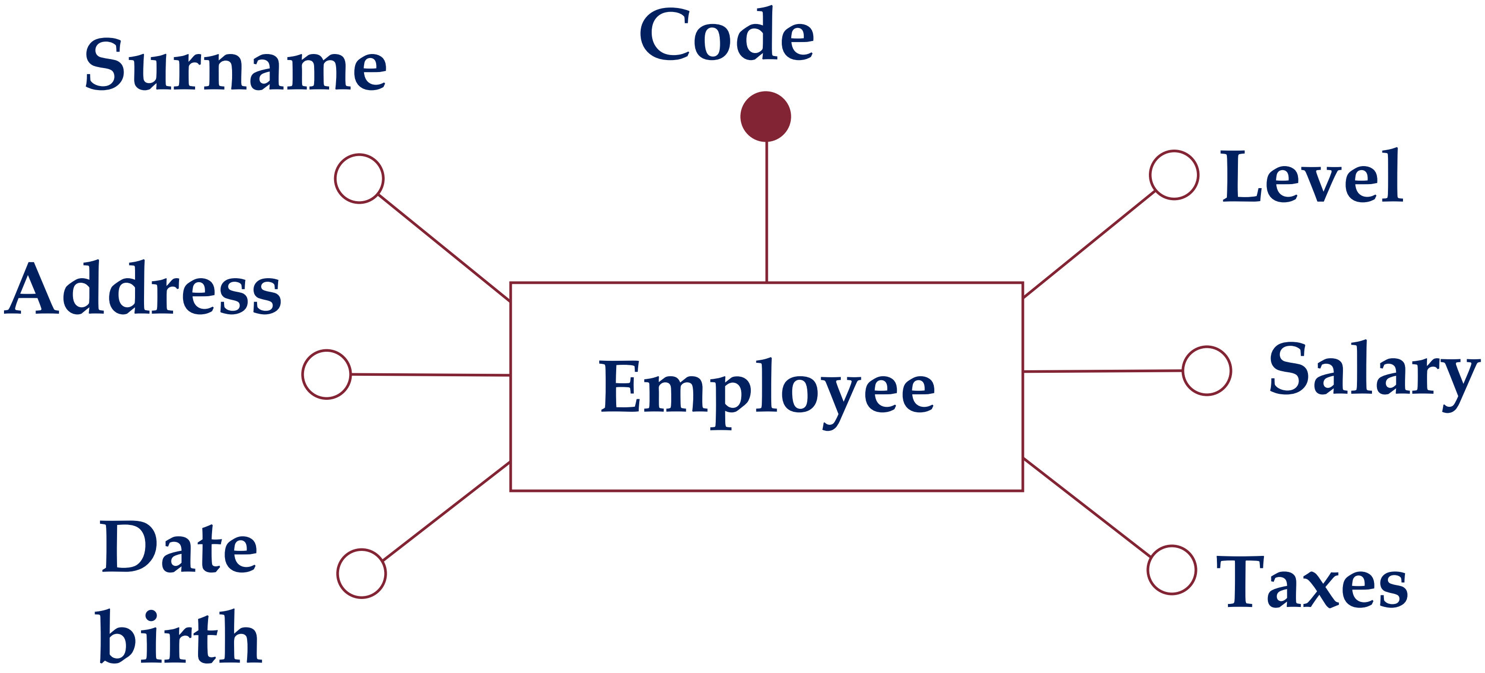
Partitioning/merging entities and relationships

- Restructurings carried out to make operations more efficient based on the simple principle that access is reduced:
  - separating attributes of a concept that are accessed separately
  - grouping attributes of different concepts accessed together

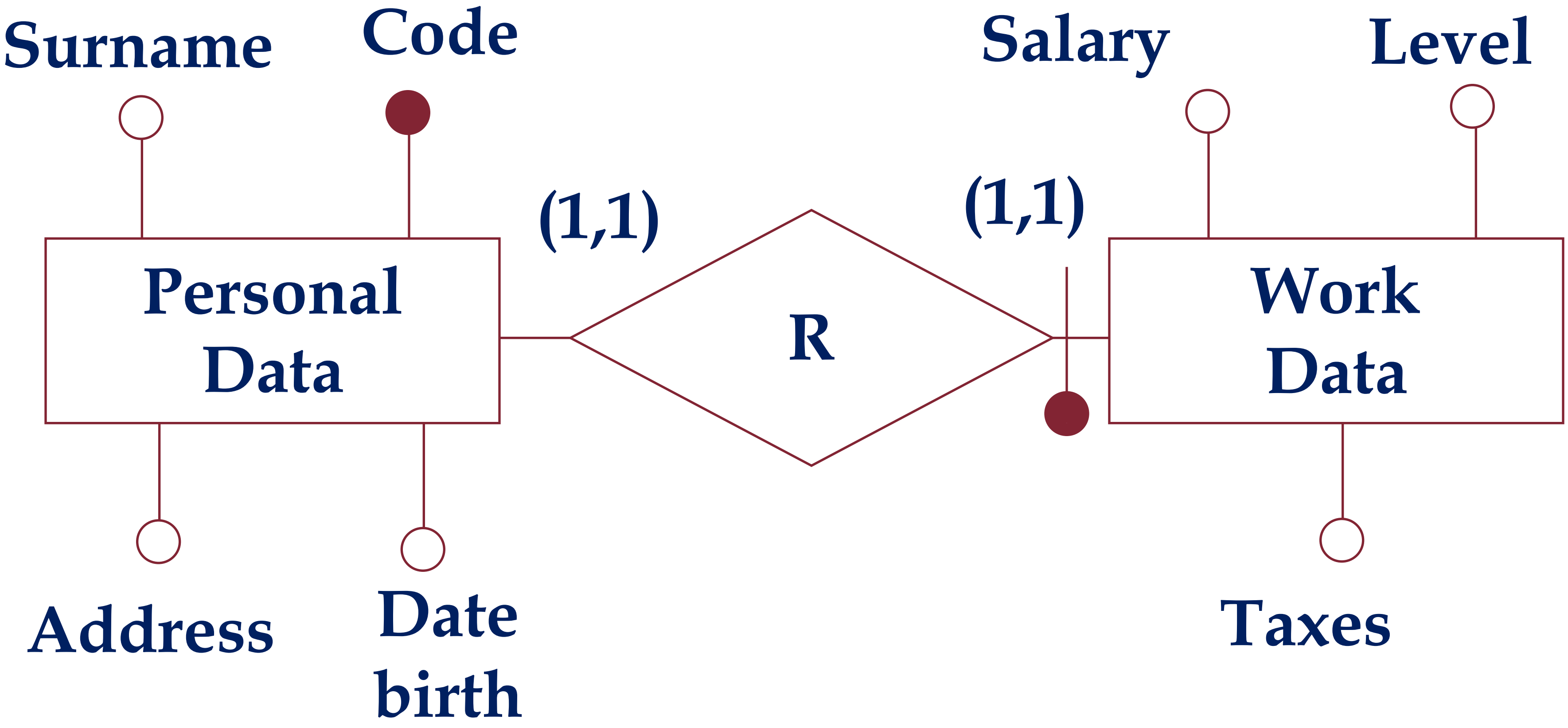
Main cases

- Vertical partitioning of entities
- Horizontal partitioning of relationships
- Deleting Multivalued Attributes
- Merging of entities/relationships

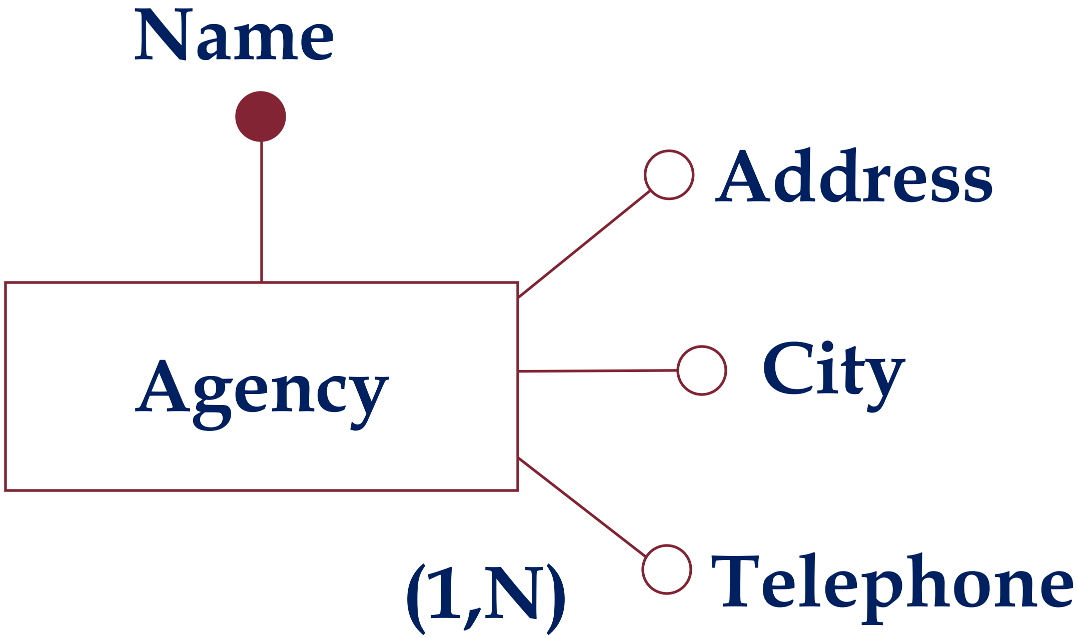
Example: Vertical Partitioning of Entities



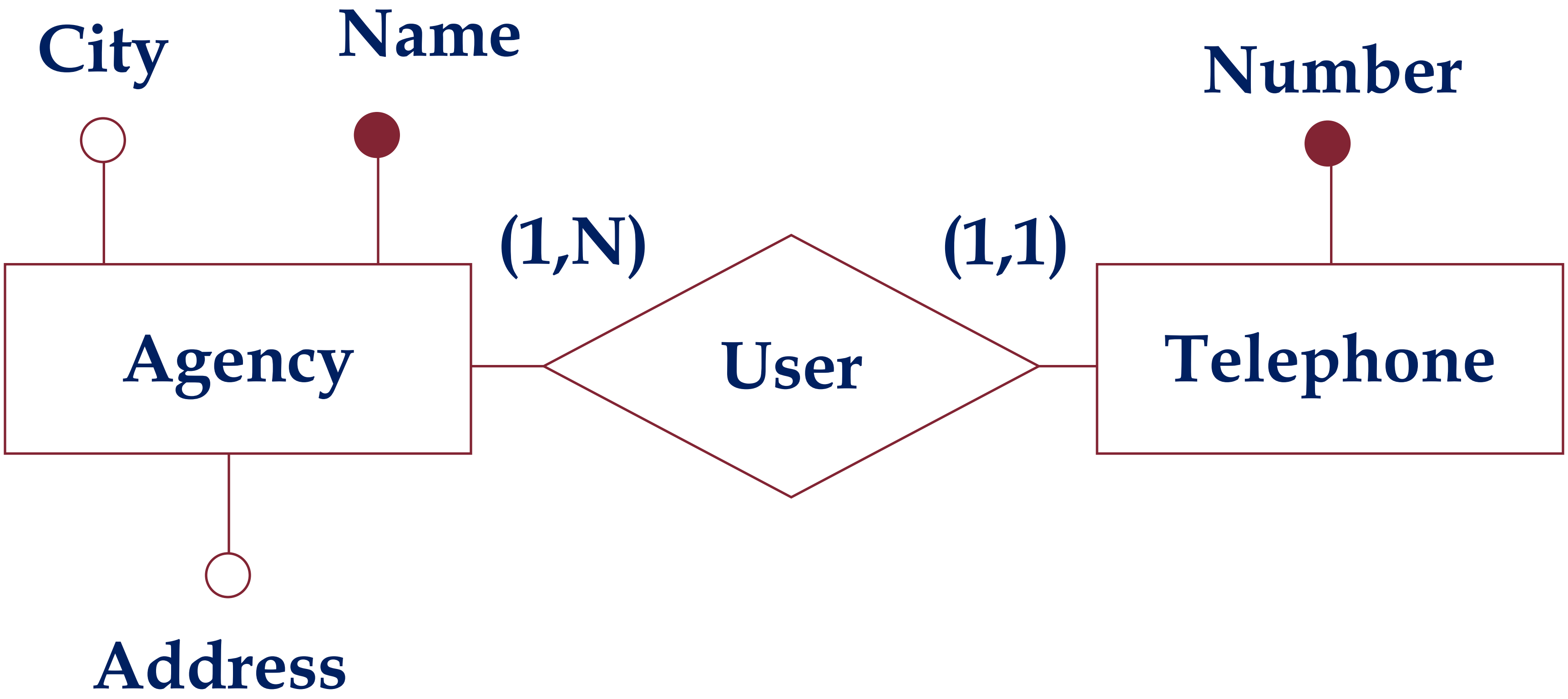
Example: Vertical Partitioning of Entities



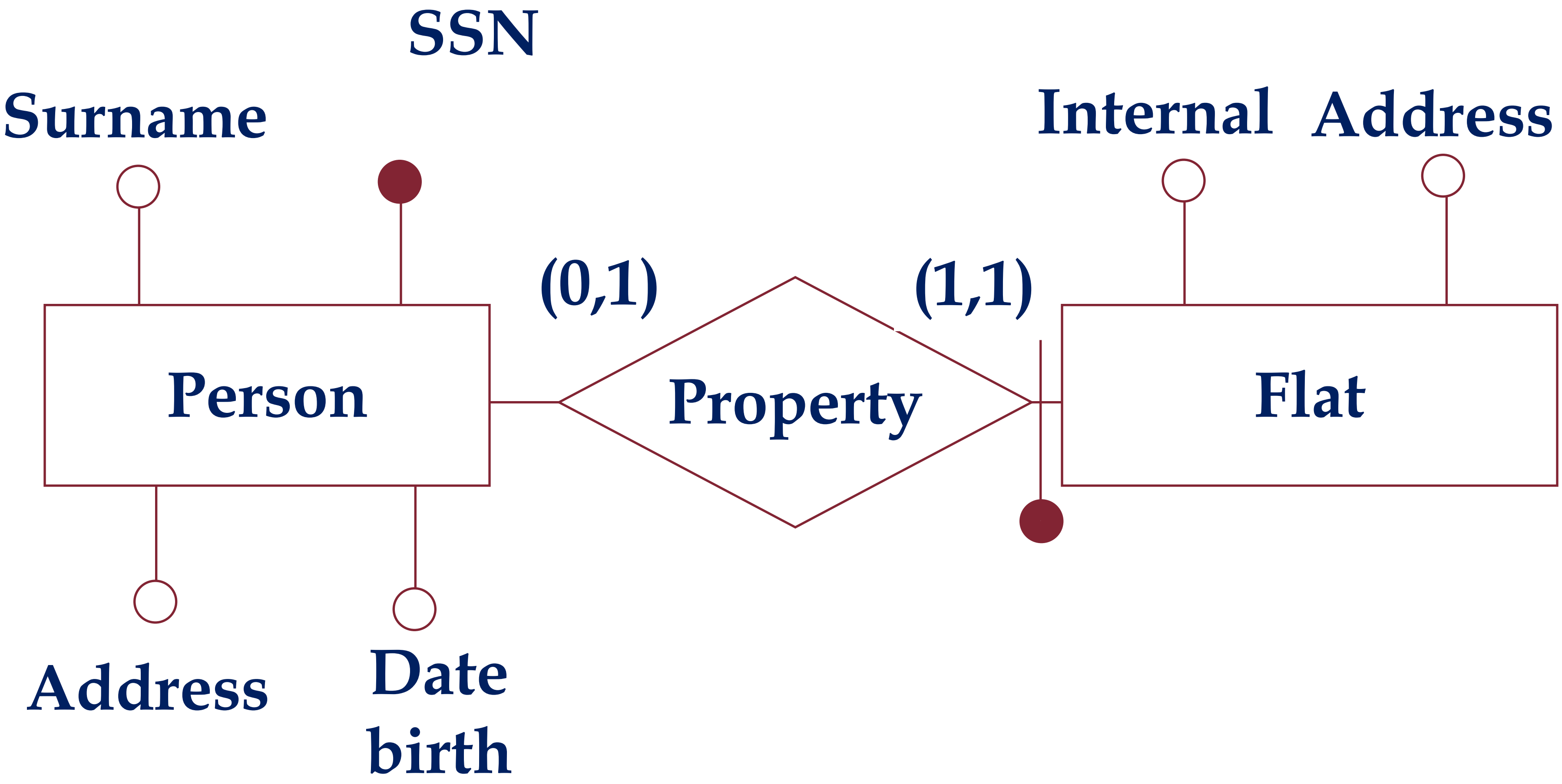
Example: Deleting Multivalued Attribute



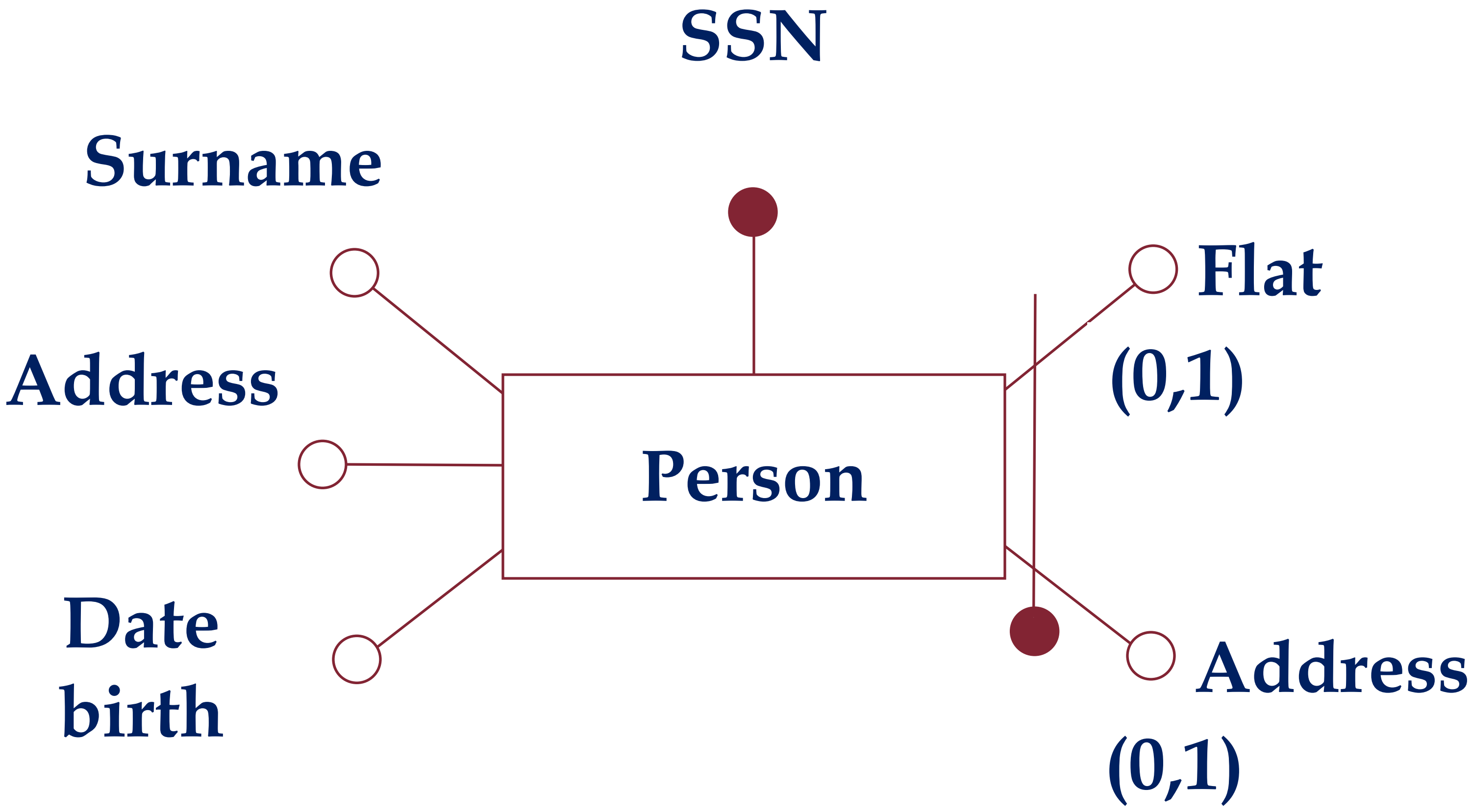
Example: Deleting Multivalued Attribute



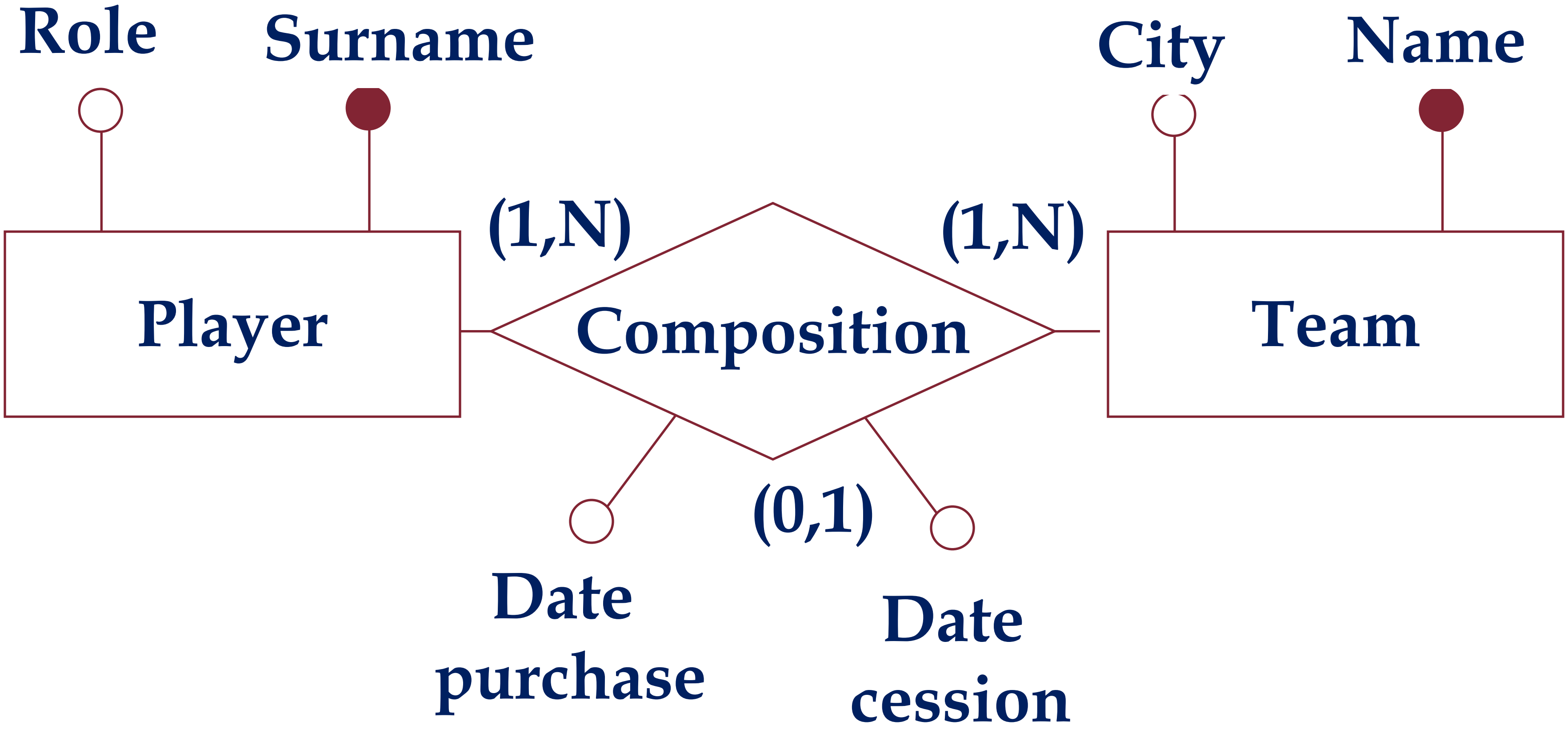
Example


SSN
Surname
Internal
Address
(0,1)
(1,1)
Person
Property
Flat
Address
Date
birth

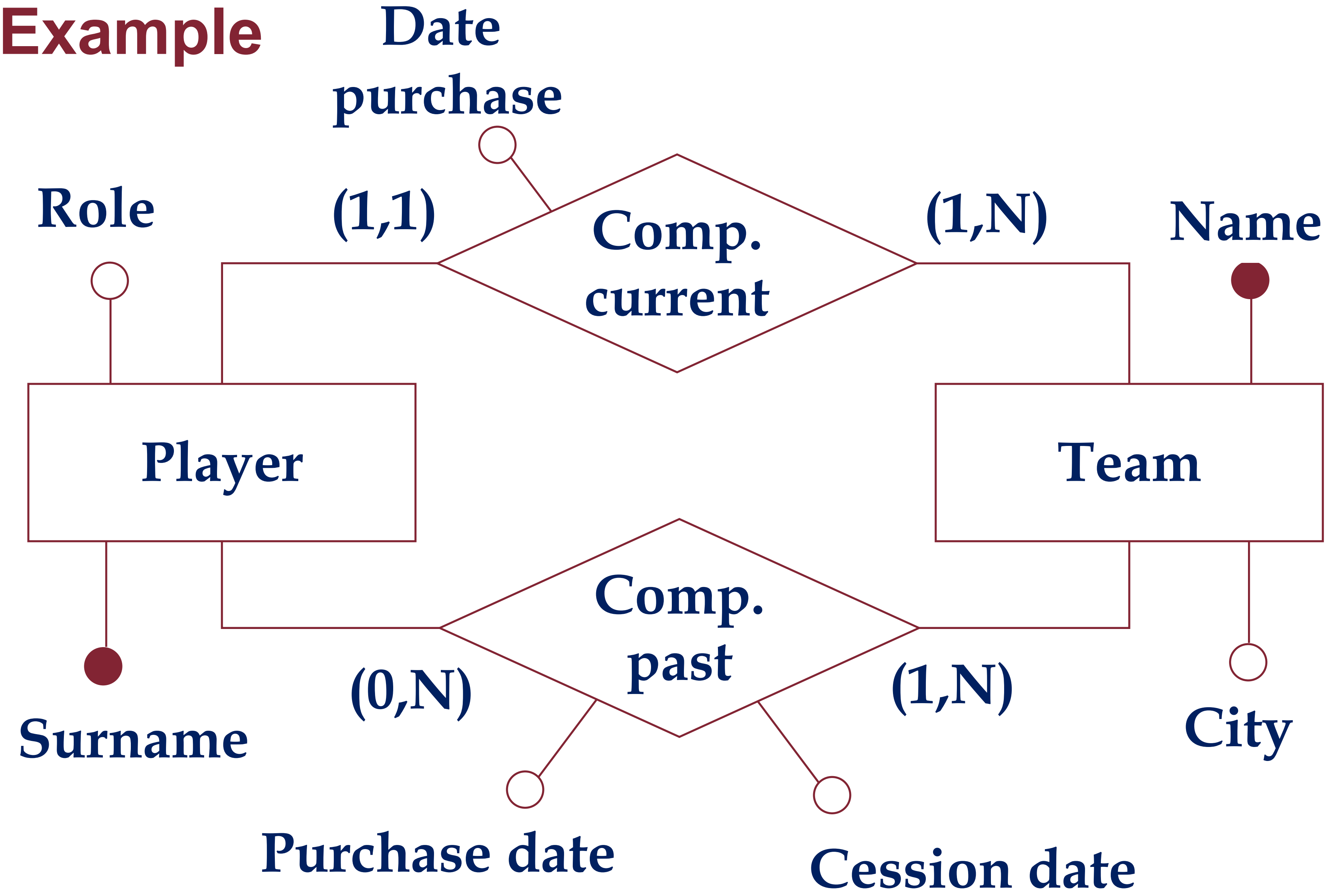
Example



Example


Role
Surname
City
Name
(1,N)
(1,N)
Player
Composition
Team
(0,1)
Date purchase
Date cession

Example


Date purchase
Role
(1,1)
Comp. current
(1,N)
Name
Player
Team
Comp. past
(0,N)
(1,N)
Surname
City
Purchase date
Cession date

Choosing Key Identifiers

Choosing key identifiers

- Essential operation for translation into the relational model
- **Policy:**
  - **absence of optionality**
  - **simplicity**
  - **use in the most frequent or important operations**

Choosing key identifiers

- What happens if none of the identifiers meet the requirements?
- New attributes (codes) are introduced
- Contain specially generated values to act as identifiers

TRANSLATION TOWARDS THE RELATIONAL MODEL

Translation to the relational model

- Basic idea:
  - entities become relationships on the same attributes
  - associations (i.e. ER relationships) become relationships on the identifiers of the entities involved (plus own attributes)

DOMAINS

Domain mappings

- Each attribute is defined on a value domain
- An attribute associates each entity instance or association with a value from the corresponding domain
- Domains are basically the data types
- The ER schema must contain only attributes supported by the DBMS

Domains in the conceptual schema

- Base domains: integer, string, real, date, time, Boolean, etc.
- Specialized domains: integer>0 [x, y] range of integers
- Enumerative domains: {M,F} {BWM, Mercedes, Audi}, etc.
- Record domains: Address, etc.

How to represent the values belonging to these domains in the database?

Domains in DBMS

- Basic domains have a direct match with DBMS domains

Dominio	Dominio DBMS (SQL)
integer	integer, smallint, etc.
real	real, decimal, float, numeric,
date	date, etc.
time	time, etc
datetime	timestamp, etc.

- **Goal: Replace conceptual domains in the ER schema with SQL domains**

Specialized domains

- SQL allows you to define arbitrary user domains
  - create domain command
  - enumerative domains are represented by creating type

Domain	SQL Domain
integer >0	create domain IntPos as integer check (value >0)
real >3	create domain RealGTh as real check (value >3)
[1,20]	create domain Interval_domain as integer check (value >=1 and value <=20)
{M,F}	create type Sex as enum('M', 'F')
datetime	timestamp, etc.

Compound domains

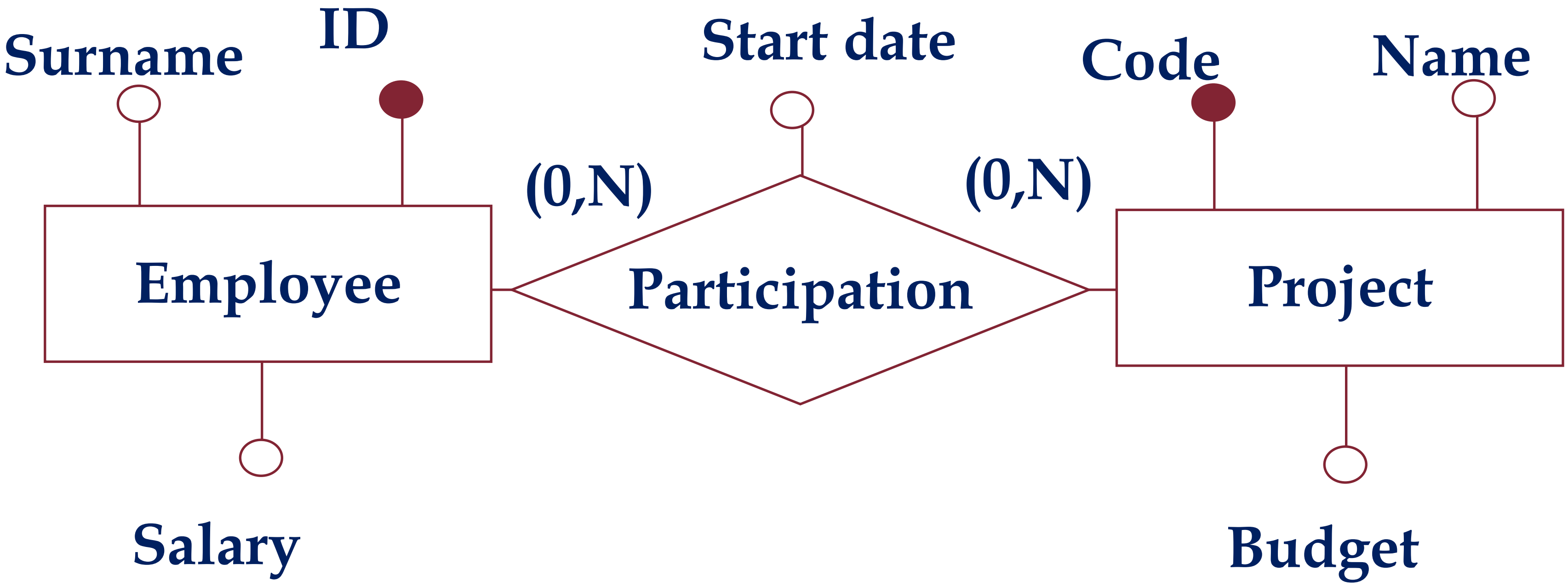
- SQL 1999 allows the user to define structured types with the create type construct, but this feature is not currently supported by all commercial DBMS
- Example domain Address (street: string, number: integer, city: string)

```
create type Address as
(street varchar (100), number integer,
 city varchar (100)
)
```

- Many DBMS offer non-standard constructs that in some cases have limitations

TRANSLATION OF RELATIONS

Entities and relationships many to many



Employee(ID, Surname, Salary)

Project(Code, Name, Budget)

Participation(ID, Code, StartDate)

Entities and relationships many to many

Employee(ID, Surname, Salary)

Project(Code, Name, Budget)

Participation(ID, Code, StartDate)

- Referential integrity constraints between:
  - ID in Participation and (the key of) Employee
  - Code in Participation and (the key of) Project

Entities and relationships many to many

Employee(ID, Surname, Salary)

Project(Code, Name, Budget)

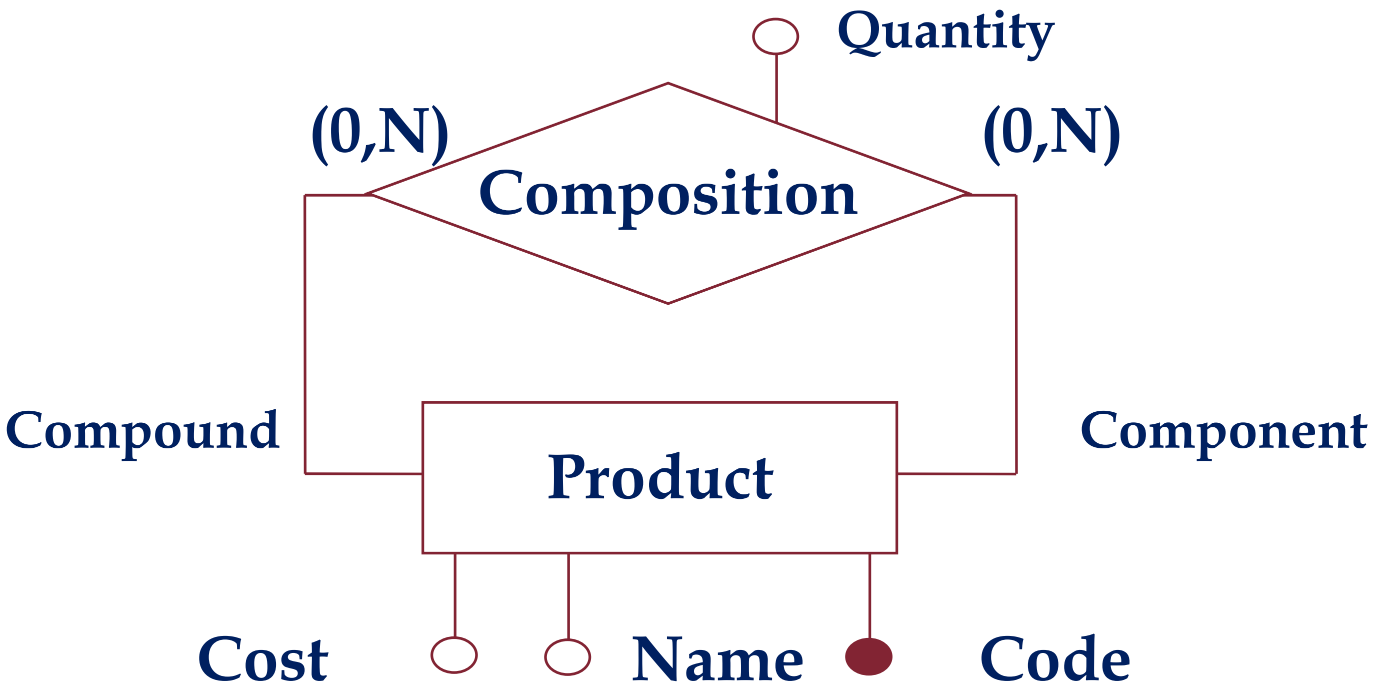
Participation(ID, Code, StartDate)

- Referential integrity constraints between:
  - ID in Participation and (the key of) Employee
  - Code in Participation and (the key of) Project

It is better to use more expressive names that make the constraints more visible

Participation(Employee, Project, StartDate)

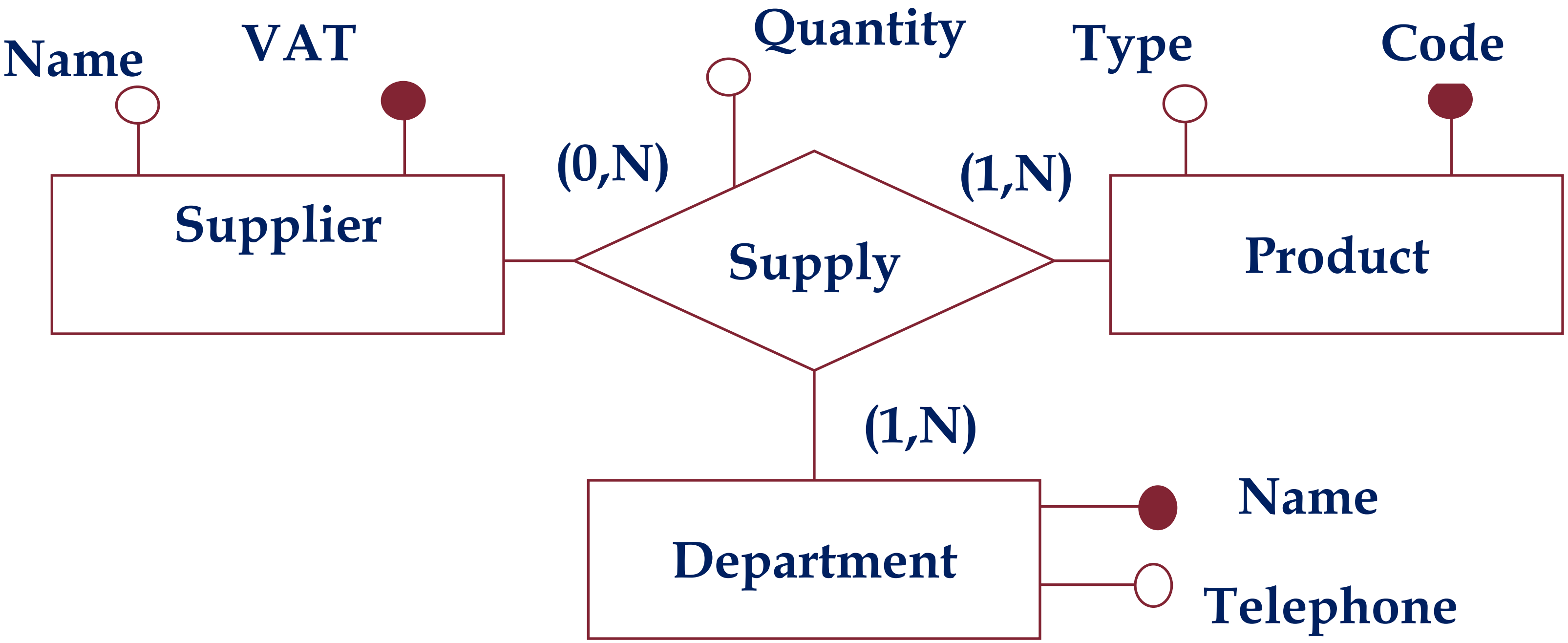
Recursive relationships



Product(Code, Name, Cost)

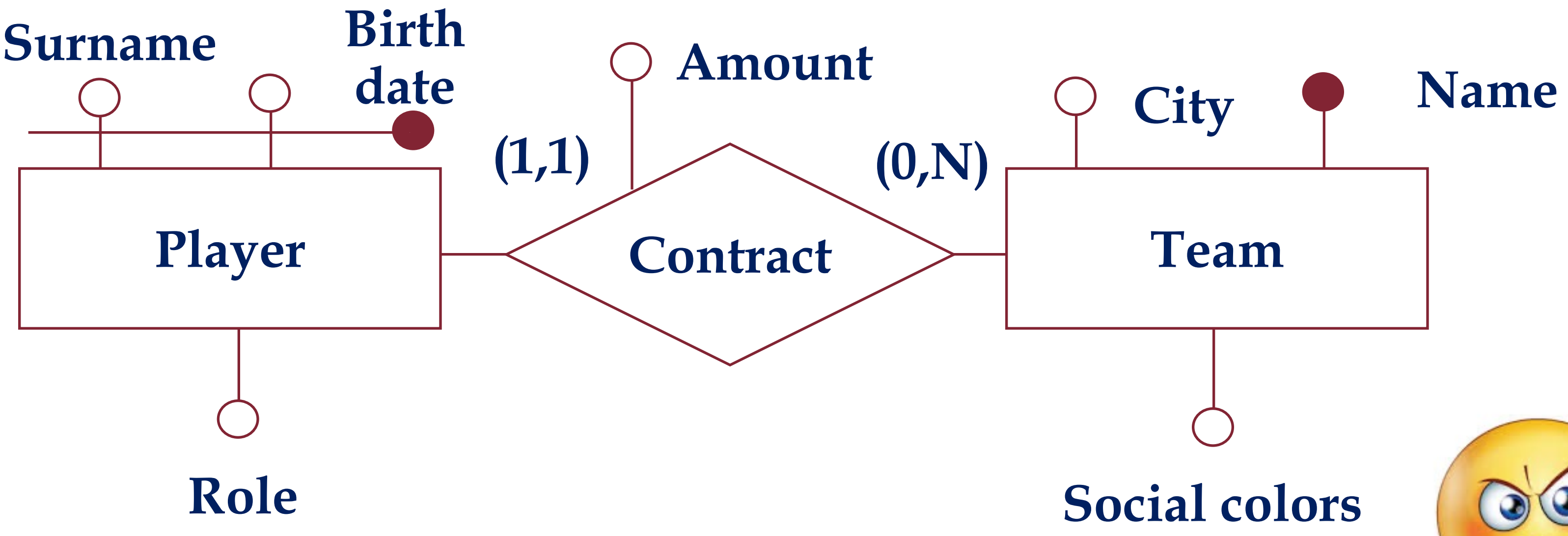
Composition(Compound, Component, Quantity)

N-ary relations



Supplier(VAT, Name)
Product(Code, Type)
Department(Name, Telephone)
Supply(Supplier, Product, Department, Quantity)

One-to-many relations



Player(Surname, BirthDate, Role)

Contract(SurPlayer, BirthDateP, Team, Amount)

Team(Name, City, SocialColors)

Relazioni uno a molti: soluzione più compatta

Player(Surname, BirthDate, Role)

Contract(SurPlayer, BirthDateP, Team, Amount)

Team(Name, City, SocialColors)

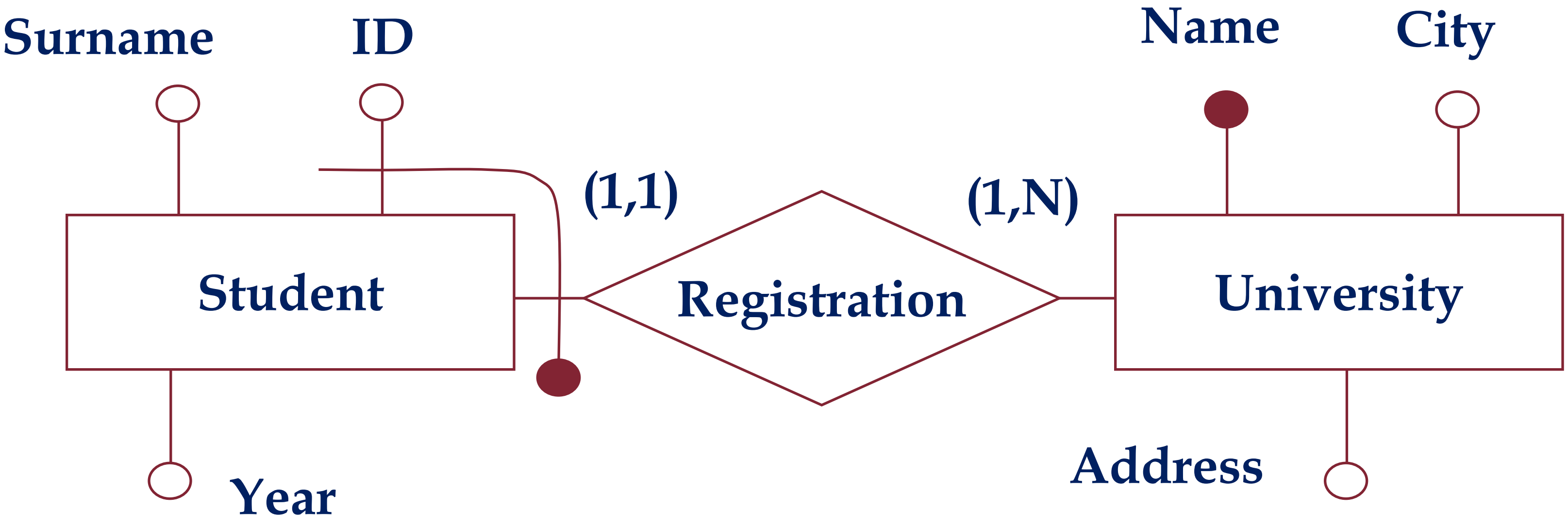
Player(Surname, BirthDate, Role, Team, Amount)

Team(Name, City, SocialColors)

Referential integrity constraints between:
Team in Player and (the key of) Team

- If the minimum cardinality of the relation is 0:
  - Team in Player must admit null value

Entities with external identification

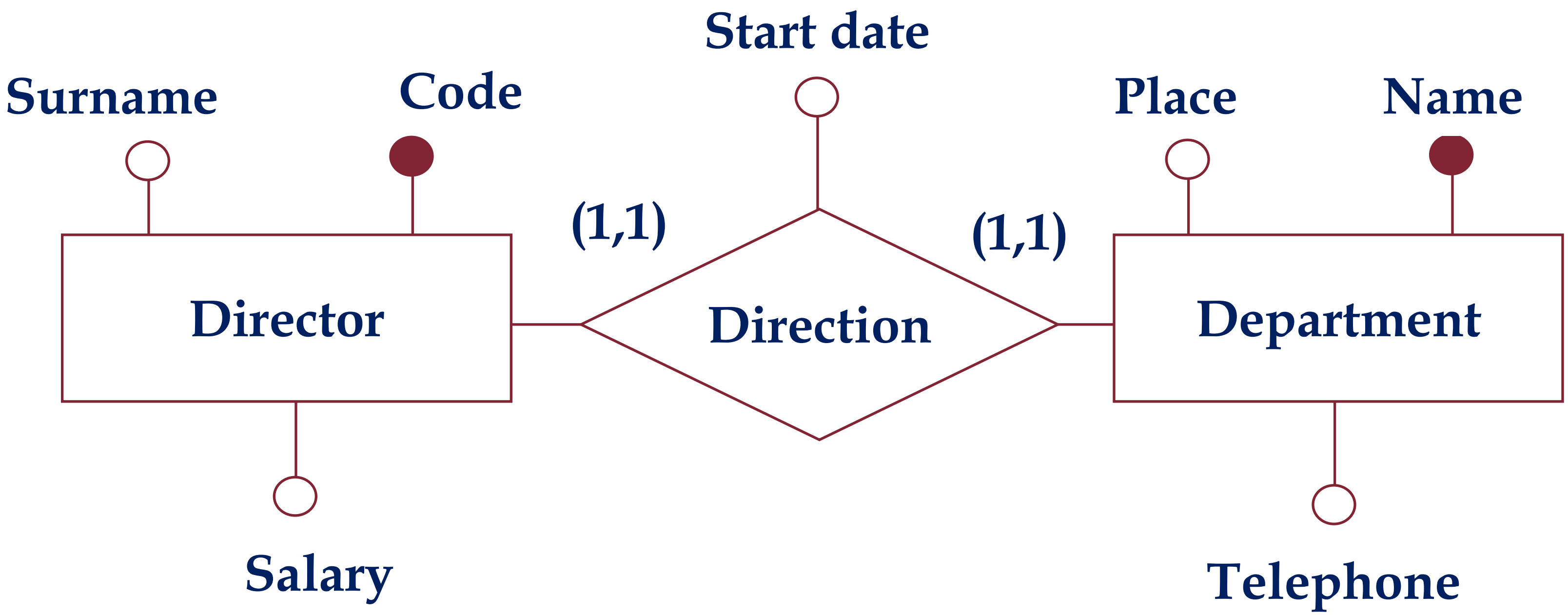


Student(<u>ID</u>, <u>University</u>, Cognome, Year)

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

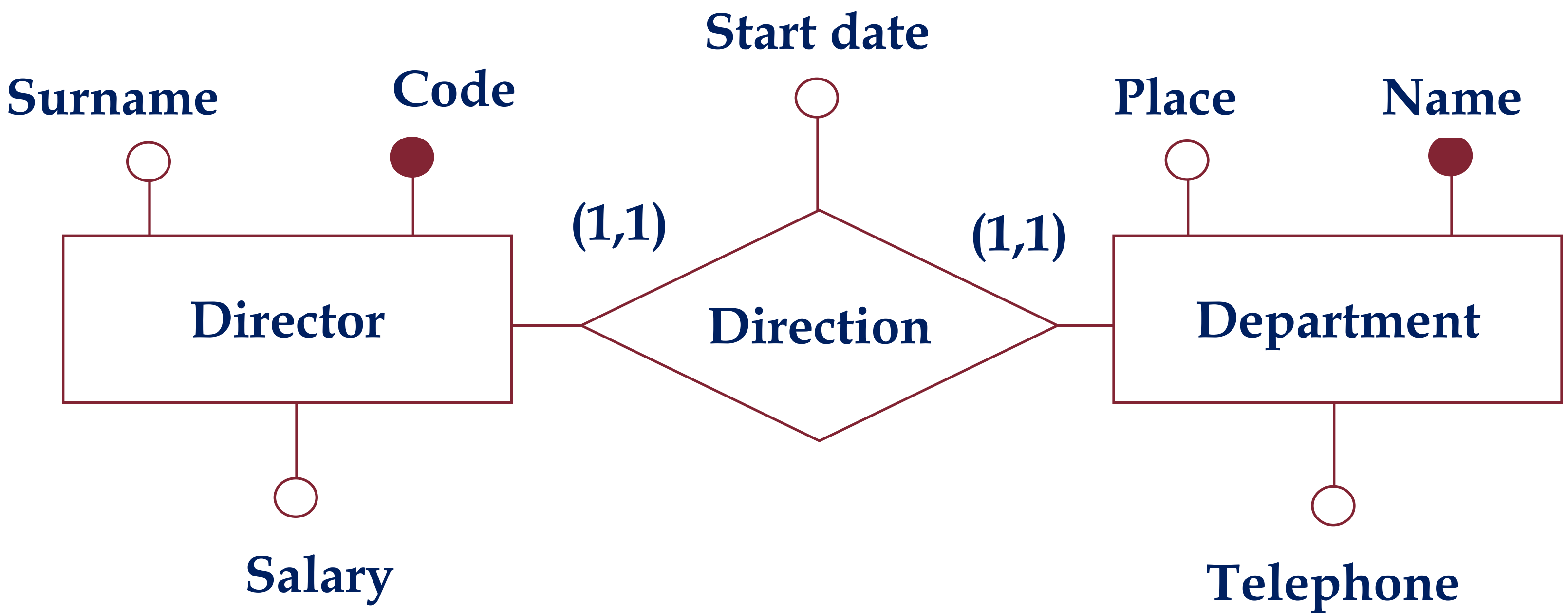
- **with constraint ...**

One-to-one relationship



- Various possibilities:
  - Merge on one side or the other
  - Merge everything?

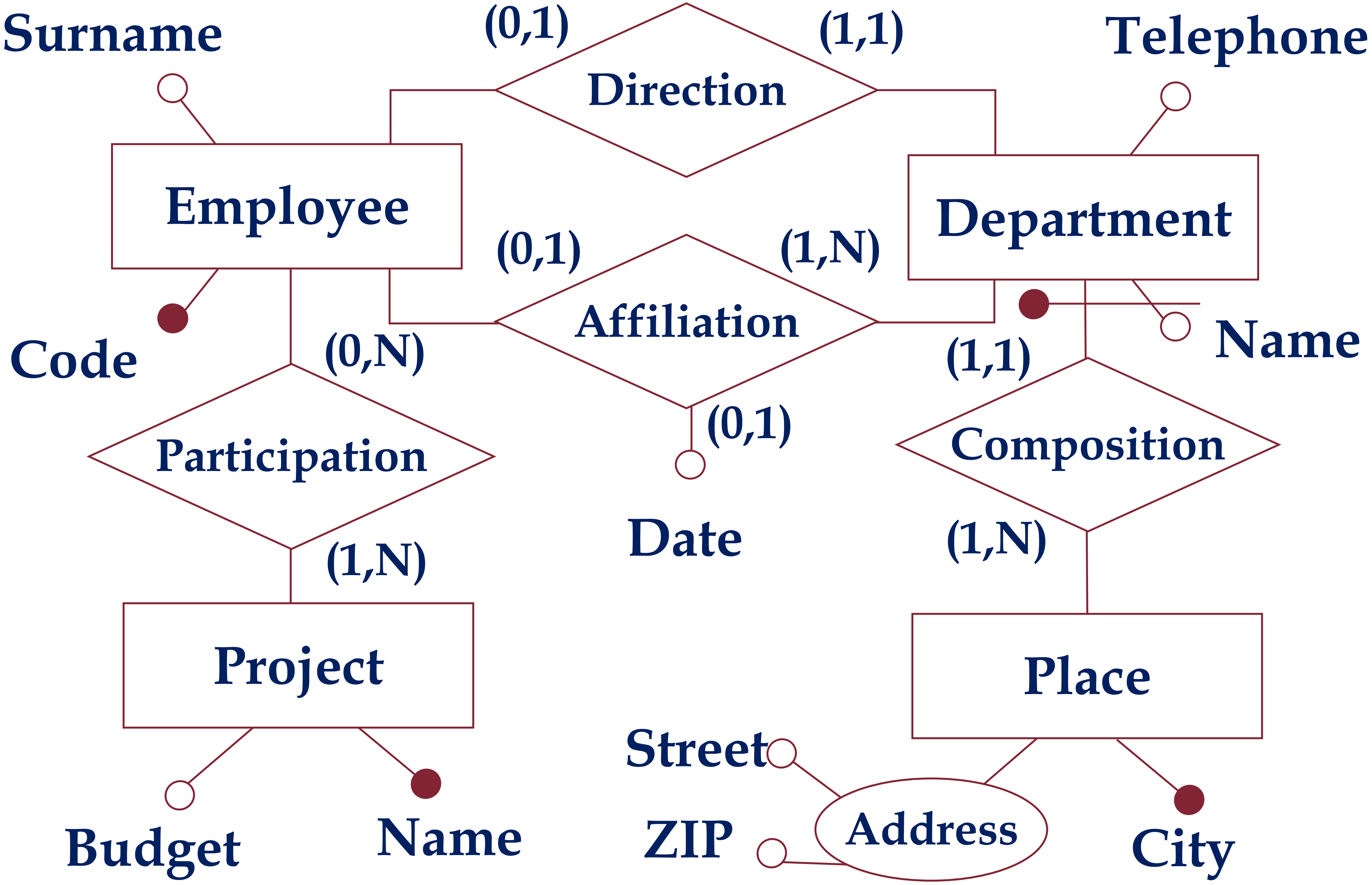
A privileged possibility



Director(Code, Surname, Salary)

Department(Name, Place, Telephone, Director, startD)

With referential integrity constraint, without null values


Surname
(0,1)
Direction
(1,1)
Telephone
Employee
Department
(0,1)
(1,N)
Affiliation
Code
(0,N)
(1,1)
Name
(0,1)
Participation
Composition
Date
(1,N)
(1,N)
Project
Place
Street
Budget
Name
ZIP
Address
City

Final schema

Employee(Code, Surname, Department*, Date*)

Department(Name, City, Telephone, Director)

Place(City, Street, ZIP)

Project(Name, Budget)

Participation(Employee, Project)

* Non-null values

Conclusions

- Logic Design
- Performance analysis
- Restructuring of the ER scheme
- References:
  - Book: Chapter 8