1222 · 2022
800 ANNI
UNIVERSITÀ DEGLI STUDI DI PADOVA

Logical Design: ER Schema Transformation

Basi di Dati

Bachelor's Degree in Computer Engineering
Academic Year 2024/2025

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Stefano Marchesin

Intelligent Interactive Information Access (IIIA) Hub
Department of Information Engineering
University of Padua

Outline

- Introduction to logical design
- Transformation of the ER schema

Logical Design

ER vs Relational Model

ER Model	Relational Model
Entity	Relation
Relationship	Relation and referential integrity constraint
Attribute	Attribute (not multi-valued/composite)
Cardinality	Integrity constraints
Generalization	✘
Identifier	Super-key, key, primary key

ER vs Relational Model

ER Model	Relational Model
Entity	Relation
Relationship	Relation and referential integrity constraint
Attribute	Attribute (non multi-valued/composite)
Cardinality	Integrity constraints
Generalization	✗
Identifier	Super-key, key, primary key

There is an "impedance mismatch" between the ER and the Relational models which calls for a transformation of the ER schema to ease its mapping to the relational model

Logical Design Steps

1. Transformation of the ER schema
    - removal of “constructs” that cannot be expressed in the relational model
    - choice of the principal identifiers
2. Mapping of the transformed ER schema into a relational schema
    - the mapping is almost “automatic” since it does not require (almost) any choice by the designer
    - the produced relational schema does not contain redundancies apart from those explicitly added for well motivated reasons

Steps of the ER Schema Transformation

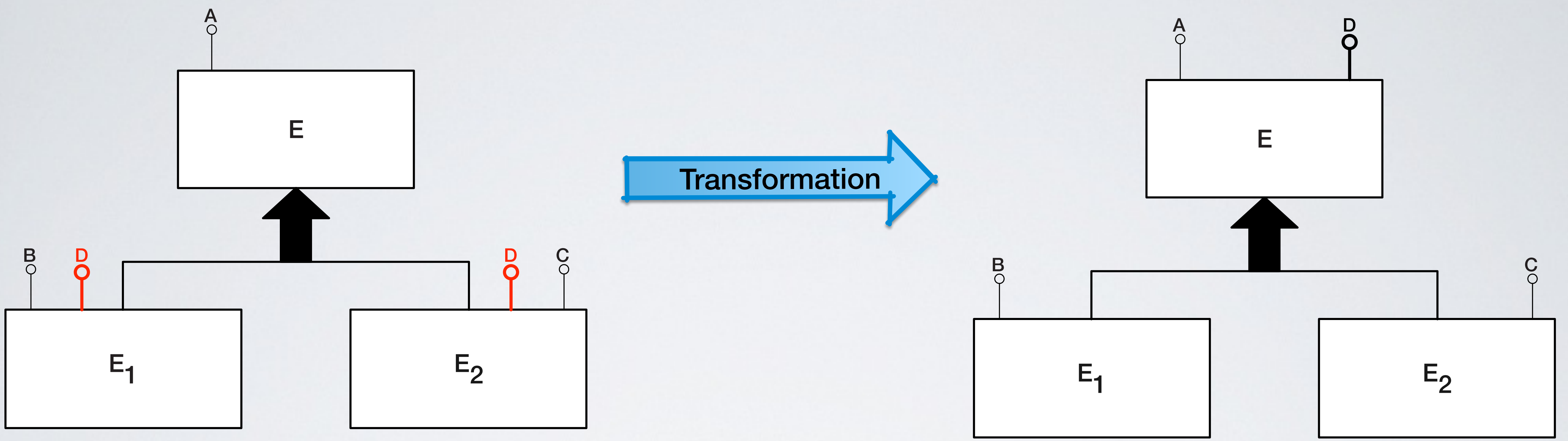
1. Redundancy analysis
2. Removal of multi-valued attributes
3. Removal of composite attributes
4. Removal of IS-A relations and generalizations
5. Choice of principal identifiers
6. Specification of additional external constraints
7. Partitioning and merging of entities

Transformation - Step 1: Redundancy Analysis

Minimality

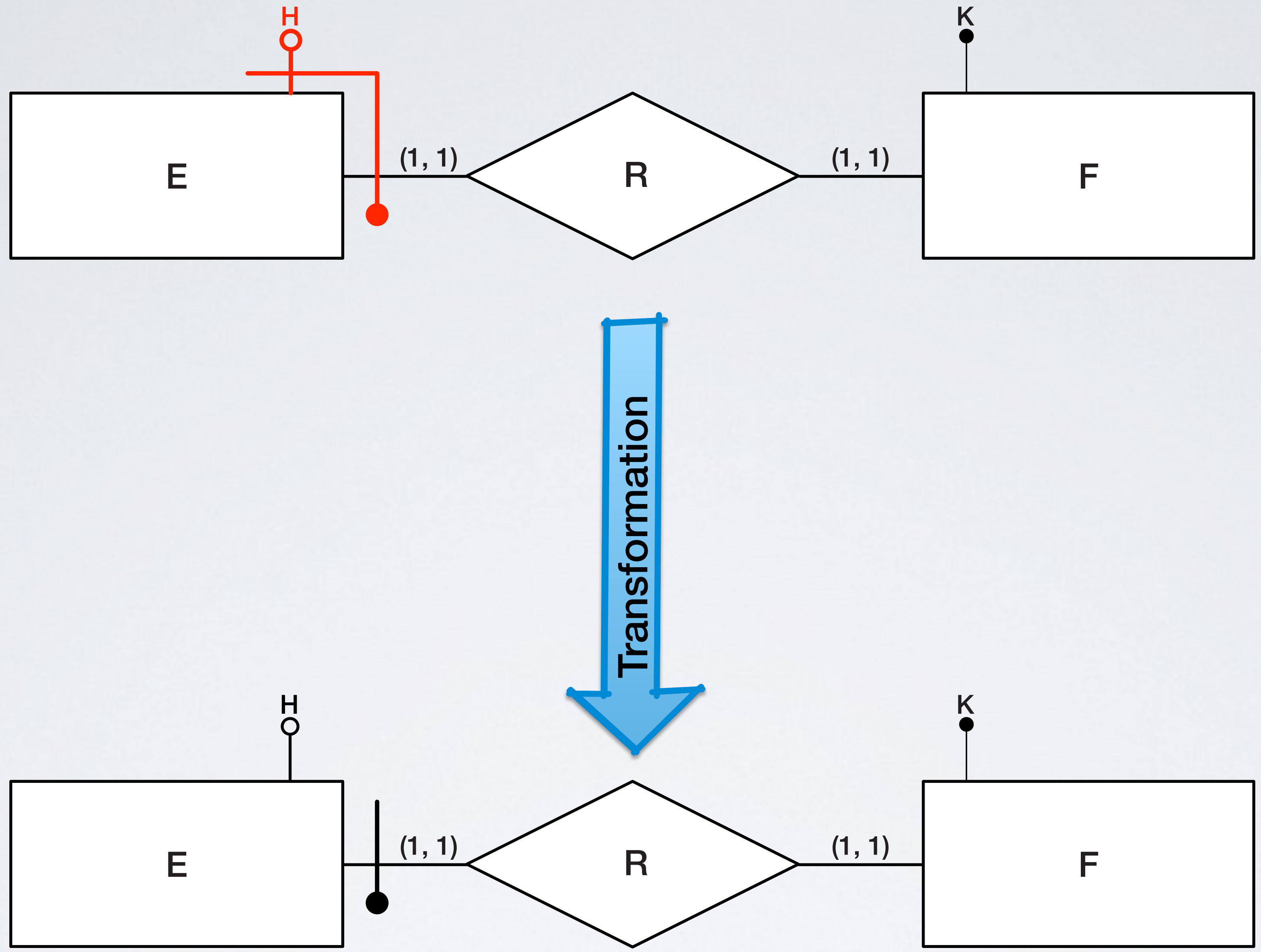
- We should avoid to represent the same property twice
  - **intensional redundancy**: it should be avoided and it is dealt with transformations which preserve the informative content
  - **extensional redundancy**: the same property is represented more than once in the instances of the schema, implicitly or explicitly
- We can keep redundancies only when they are well motivated by design or performance considerations

Intensional Redundancy: Attributes of Generalization



- An attribute **D** common to two entities **E_1** e **E_2** which subclass another entity **E** can be moved to entity **E**

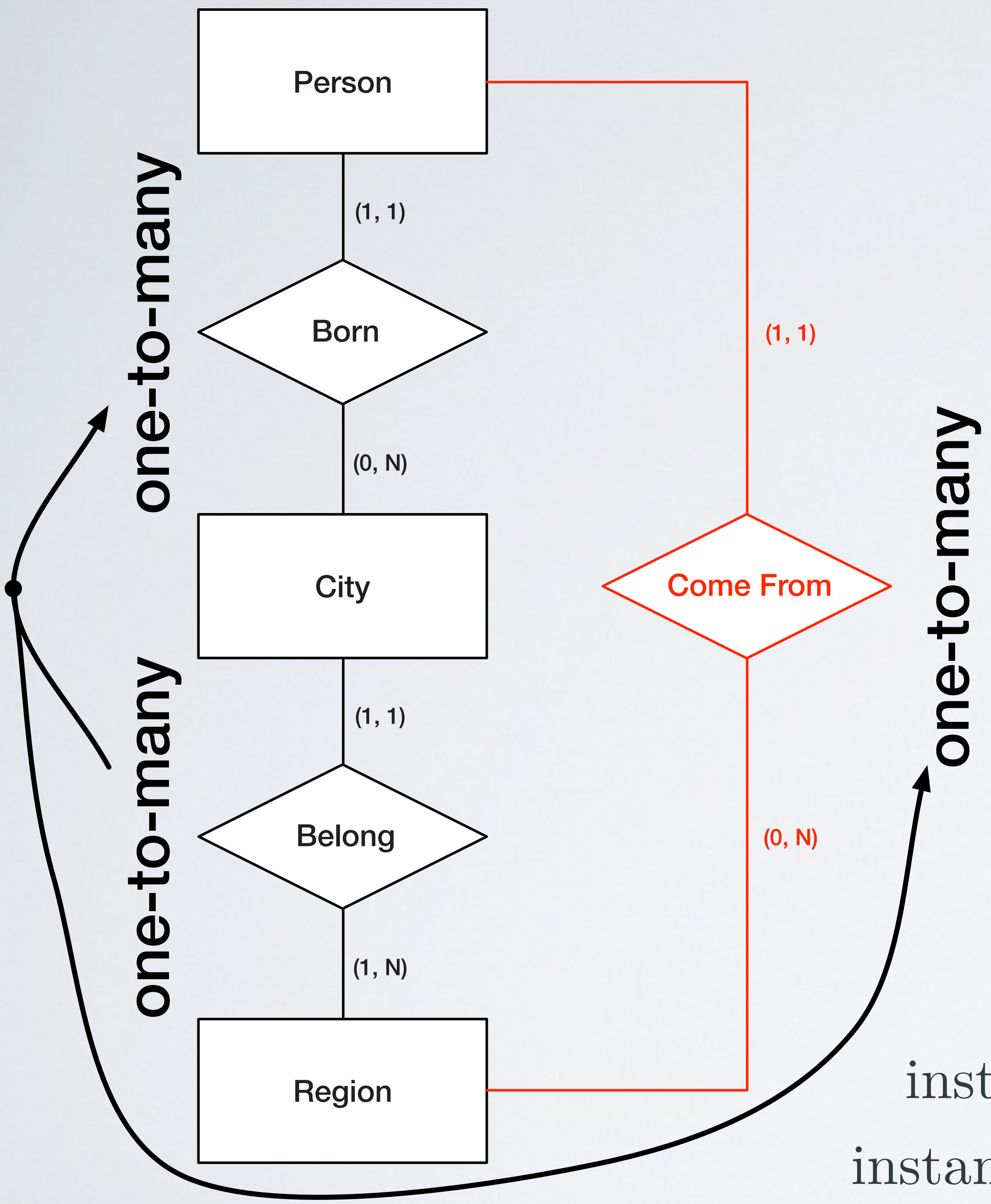
Intensional Redundancy: Identifiers



- An identifier constituted by a super-set of the properties which constitute another identifier can be replaced with the identifier using the minimal set of properties

Extensional Redundancy: Relationship Cycles

- There MAY BE redundancy when a relationship $\mathbf{R_1}$ between two entities has the same informative content of a chain of relationships $\mathbf{R_2}$, $\mathbf{R_3}$, ..., $\mathbf{R_n}$ connecting exactly the same pairs of instances of the involved entities
- Not all the relationship cycles give raise to a **redundancy** but it **depends on the meaning** expressed by the involved relationships
- Some **syntactical checks** may help
  - a **cycle of one-to-one relationships** gives raise to a **one-to-one relationship** which cannot be equivalent to a one-to-many, many-to-one, or many-to-many relationship
  - a **cycle of one-to-many relationships** gives raise to a **one-to-many relationship** which cannot be equivalent to a one-to-one, many-to-one, or many-to-many relationship
  - in the general case, you cannot determine the cardinality ahead; for example, a one-to-many relationship followed by a many-to-one relationship may originate a one-to-one, one-to-many, many-to-one, or many-to-many relationship

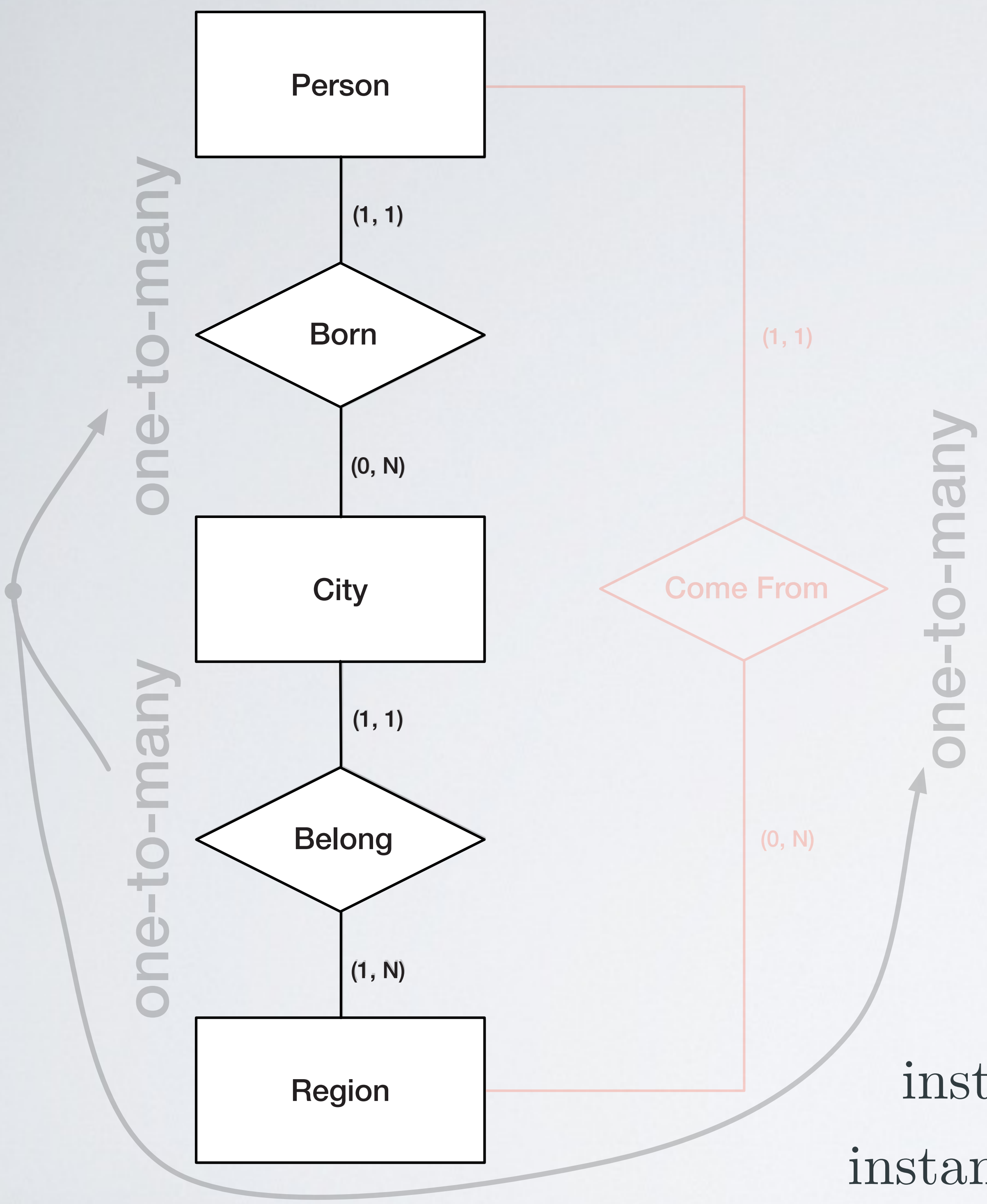


- **External constraint**: every **Person Comes From** the **Region** to which her/his **City** of **Birth Belongs**

$$\text{instance}(\text{i}, \text{Person}) = \{p_1, p_2, p_3, p_4\}$$
$$\text{instance}(\text{i}, \text{City}) = \{c_1, c_2, c_3, c_4\}$$
$$\text{instance}(\text{i}, \text{Region}) = \{r_1, r_2, r_3\}$$

$$\text{instance}(\text{i}, \text{Born}) = \{(p_1, c_1), (p_2, c_2), (p_3, c_2), (p_4, c_3)\}$$
$$\text{instance}(\text{i}, \text{Belong}) = \{(c_1, r_1), (c_2, r_2), (c_3, r_1), (c_4, r_3)\}$$
$$\text{instance}(\text{i}, \text{ComeFrom}) = \{(p_1, r_1), (p_2, r_2), (p_3, r_2), (p_4, r_1)\}$$

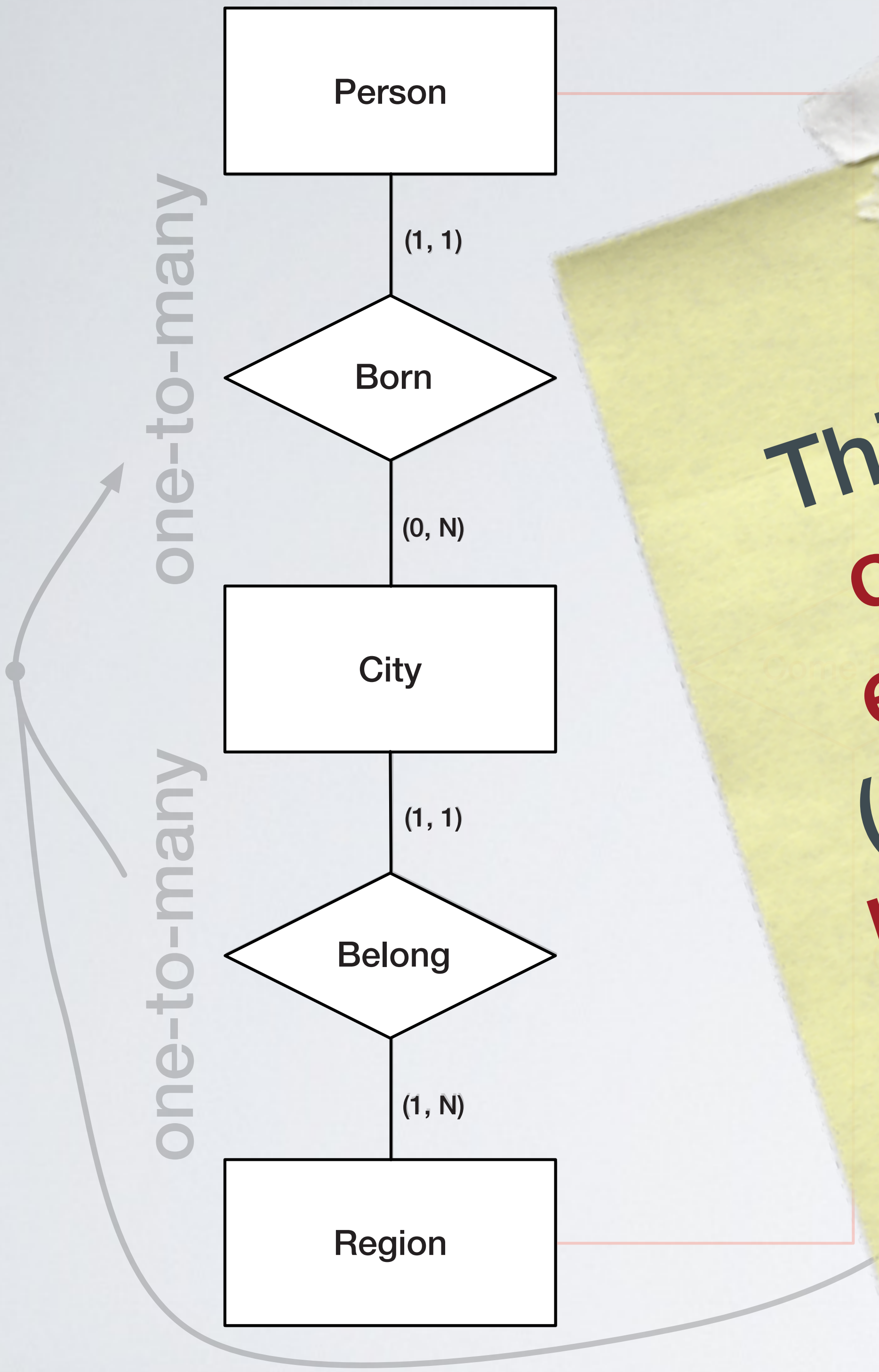
Relationship Cycles: Example (1/2)



- **External constraint**: every **Person Comes From** the **Region** to which her/his **City** of **Birth Belongs**

$$\text{instance}(\text{i}, \text{Person}) = \{p_1, p_2, p_3, p_4\}$$
$$\text{instance}(\text{i}, \text{City}) = \{c_1, c_2, c_3, c_4\}$$
$$\text{instance}(\text{i}, \text{Region}) = \{r_1, r_2, r_3\}$$

$$\text{instance}(\text{i}, \text{Born}) = \{(p_1, c_1), (p_2, c_2), (p_3, c_2), (p_4, c_3)\}$$
$$\text{instance}(\text{i}, \text{Belong}) = \{(c_1, r_1), (c_2, r_2), (c_3, r_1), (c_4, r_3)\}$$
$$\text{instance}(\text{i}, \text{ComeFrom}) = \{(p_1, r_1), (p_2, r_2), (p_3, r_2), (p_4, r_1)\}$$



External constraint: every **Person Comes From** the **Region** to which her/his **City of Birth Belongs**

This transformation **complies with the external constraint** (actually removes it), keeps the **informative content** and it is **self-explaining**

$\text{instance}(i, \text{Person}) = \{p_1, p_2, p_3, p_4\}$

$\text{instance}(i, \text{City}) = \{c_1, c_2, c_3, c_4\}$

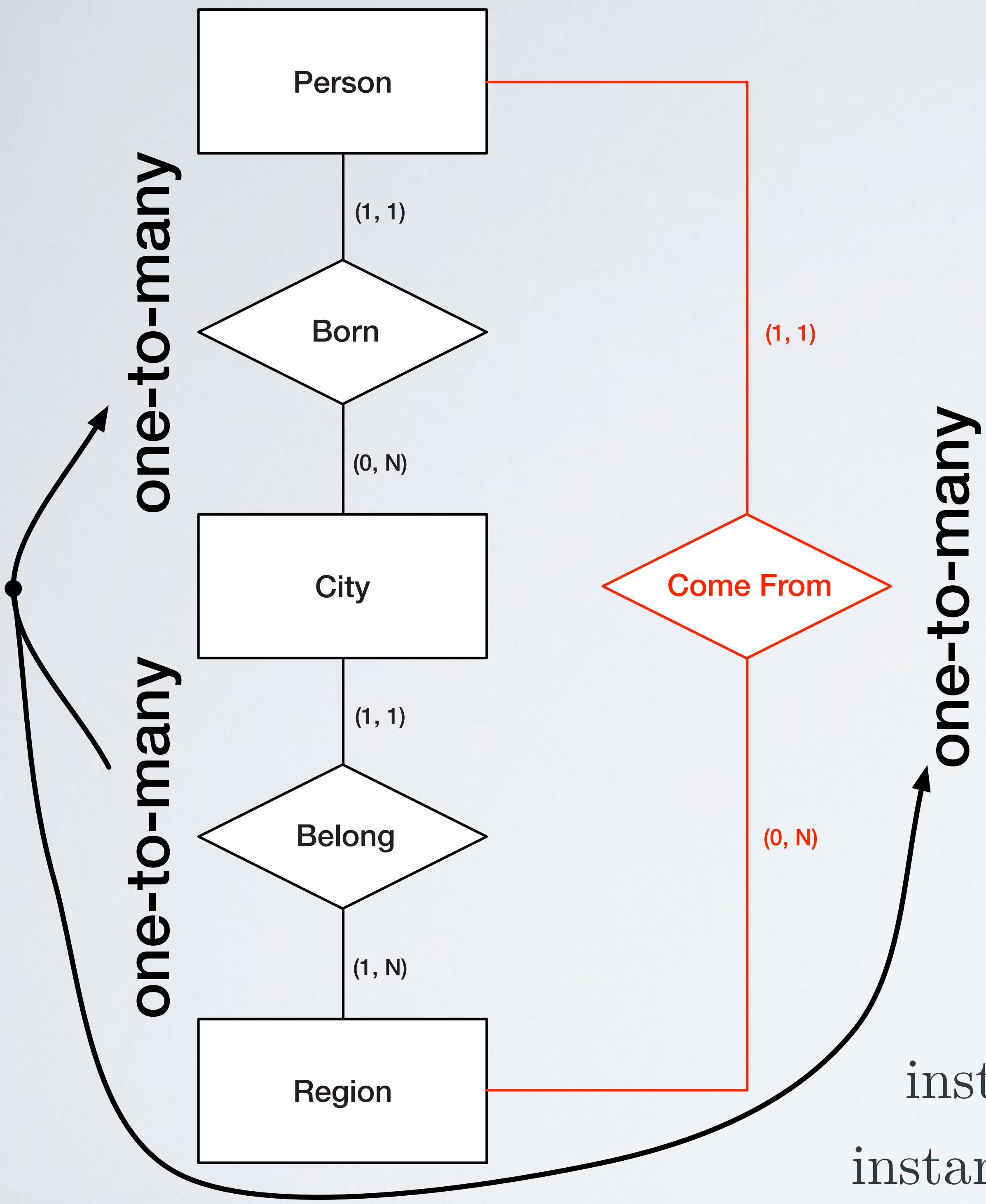
$\text{instance}(i, \text{Region}) = \{r_1, r_2, r_3\}$

$\text{instance}(i, \text{Born}) = \{(p_1, c_1), (p_2, c_2), (p_3, c_2), (p_4, c_3)\}$

$\text{instance}(i, \text{Belong}) = \{(c_1, r_1), (c_2, r_2), (c_3, r_1), (c_4, r_3)\}$

$\text{instance}(i, \text{ComeFrom}) = \{(p_1, r_1), (p_2, r_2), (p_3, r_2), (p_4, r_1)\}$

Relationship Cycles: Example (1/2)

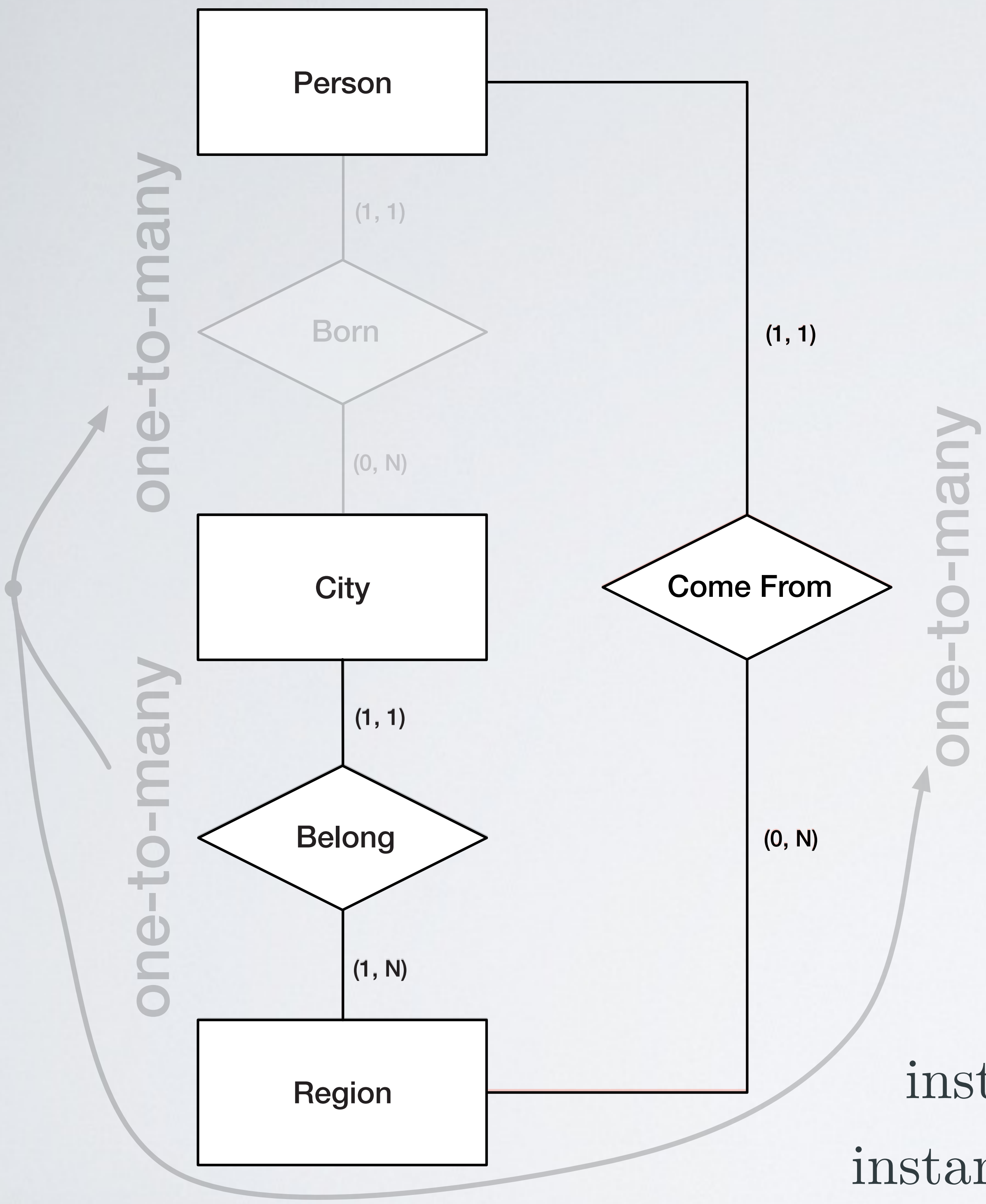


- **External constraint**: every **Person Comes From** the **Region** to which her/his **City** of **Birth Belongs**

$$\text{instance(i, Person)} = \{p_1, p_2, p_3, p_4\}$$
$$\text{instance(i, City)} = \{c_1, c_2, c_3, c_4\}$$
$$\text{instance(i, Region)} = \{r_1, r_2, r_3\}$$

$$\text{instance(i, Born)} = \{(p_1, c_1), (p_2, c_2), (p_3, c_2), (p_4, c_3)\}$$
$$\text{instance(i, Belong)} = \{(c_1, r_1), (c_2, r_2), (c_3, r_1), (c_4, r_3)\}$$
$$\text{instance(i, ComeFrom)} = \{(p_1, r_1), (p_2, r_2), (p_3, r_2), (p_4, r_1)\}$$

Relationship Cycles: Example (1/2)

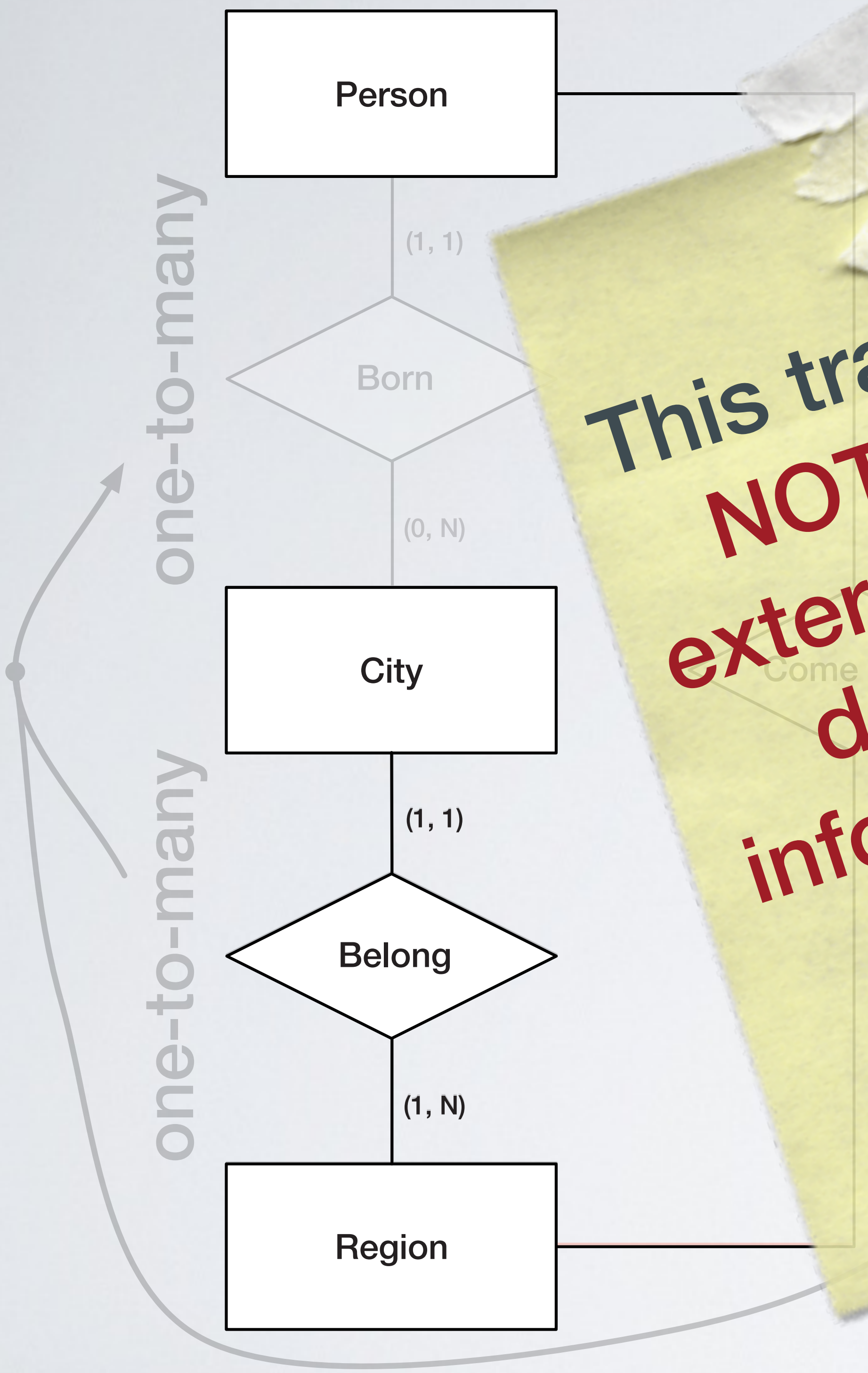


- **External constraint**: every **Person Comes From** the **Region** to which her/his **City** of **Birth Belongs**

$$\text{instance}(\text{i}, \text{Person}) = \{p_1, p_2, p_3, p_4\}$$
$$\text{instance}(\text{i}, \text{City}) = \{c_1, c_2, c_3, c_4\}$$
$$\text{instance}(\text{i}, \text{Region}) = \{r_1, r_2, r_3\}$$

$$\text{instance}(\text{i}, \text{Born}) = \{(p_1, c_1), (p_2, c_2), (p_3, c_2), (p_4, c_3)\}$$
$$\text{instance}(\text{i}, \text{Belong}) = \{(c_1, r_1), (c_2, r_2), (c_3, r_1), (c_4, r_3)\}$$
$$\text{instance}(\text{i}, \text{ComeFrom}) = \{(p_1, r_1), (p_2, r_2), (p_3, r_2), (p_4, r_1)\}$$

Relationship Cycles: Example (1/2)



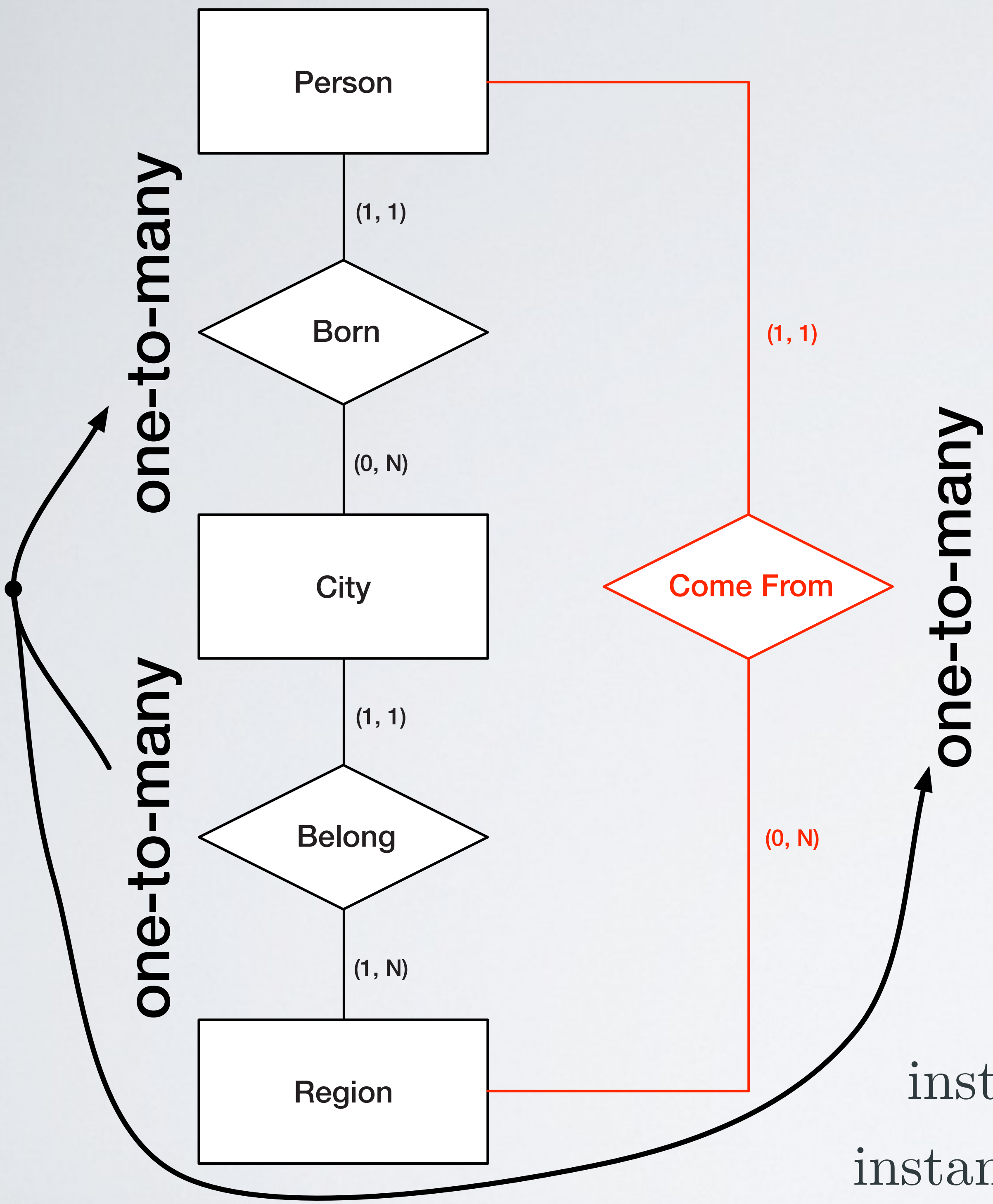
This transformation does NOT comply with the external constraint and it does NOT keep the informative content. You cannot say anymore whether p_1 is born in c_1 or in c_3

External constraint: every **Person Comes From** the **Region** to which her/his **City** of **Birth Belongs**

$$\text{instance}(i, \text{Person}) = \{p_1, p_2, p_3, p_4\}$$
$$\text{instance}(i, \text{City}) = \{c_1, c_2, c_3, c_4\}$$
$$\text{instance}(i, \text{Region}) = \{r_1, r_2, r_3\}$$

$$\text{instance}(i, \text{Born}) = \{(p_1, c_1), (p_2, c_2), (p_3, c_2), (p_4, c_3)\}$$
$$\text{instance}(i, \text{Belong}) = \{(c_1, r_1), (c_2, r_2), (c_3, r_1), (c_4, r_3)\}$$
$$\text{instance}(i, \text{ComeFrom}) = \{(p_1, r_1), (p_2, r_2), (p_3, r_2), (p_4, r_1)\}$$

Relationship Cycles: Example (1/2)

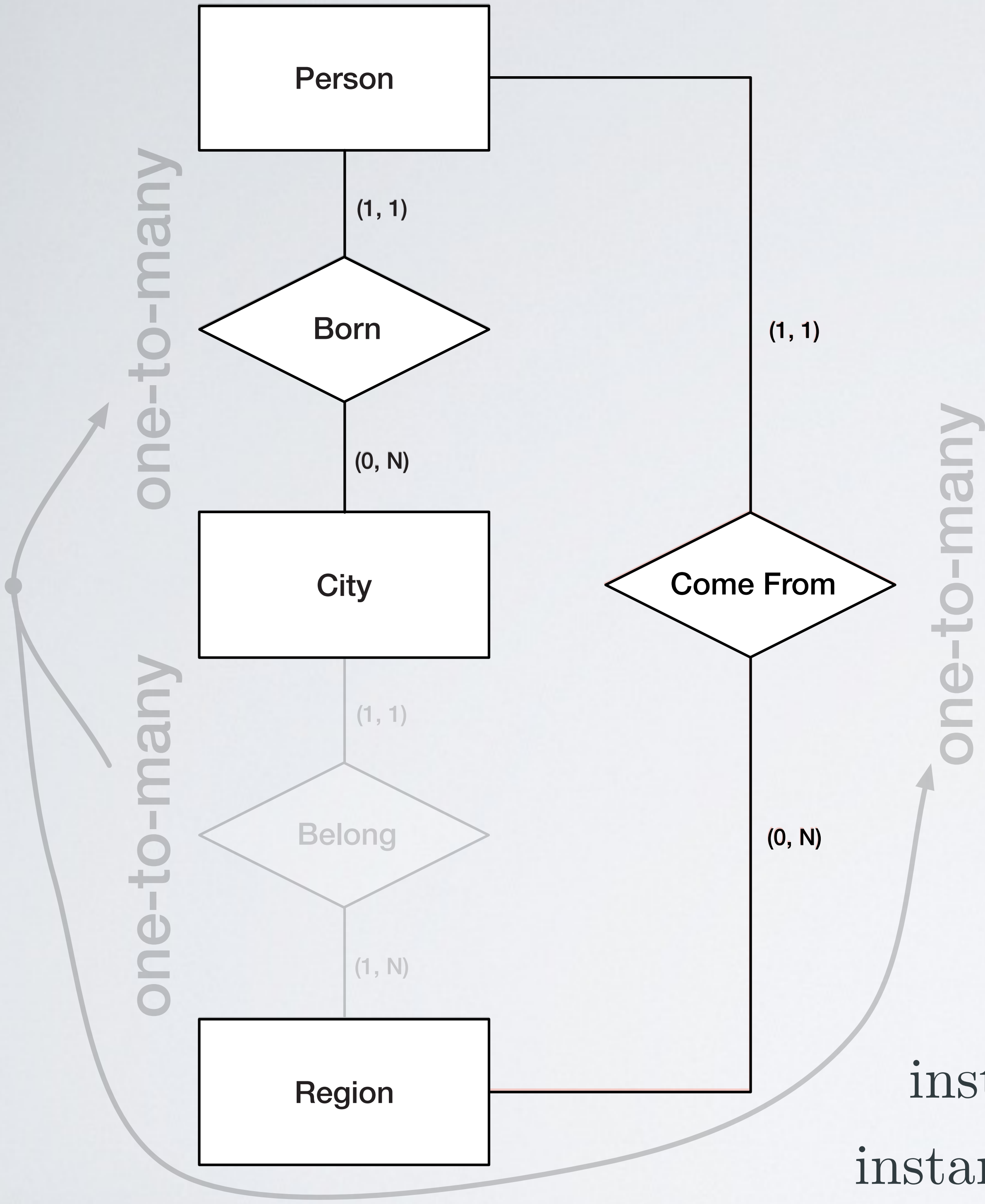


- **External constraint**: every **Person Comes From** the **Region** to which her/his **City** of **Birth Belongs**

$$\text{instance(i, Person)} = \{p_1, p_2, p_3, p_4\}$$
$$\text{instance(i, City)} = \{c_1, c_2, c_3, c_4\}$$
$$\text{instance(i, Region)} = \{r_1, r_2, r_3\}$$

$$\text{instance(i, Born)} = \{(p_1, c_1), (p_2, c_2), (p_3, c_2), (p_4, c_3)\}$$
$$\text{instance(i, Belong)} = \{(c_1, r_1), (c_2, r_2), (c_3, r_1), (c_4, r_3)\}$$
$$\text{instance(i, ComeFrom)} = \{(p_1, r_1), (p_2, r_2), (p_3, r_2), (p_4, r_1)\}$$

Relationship Cycles: Example (1/2)

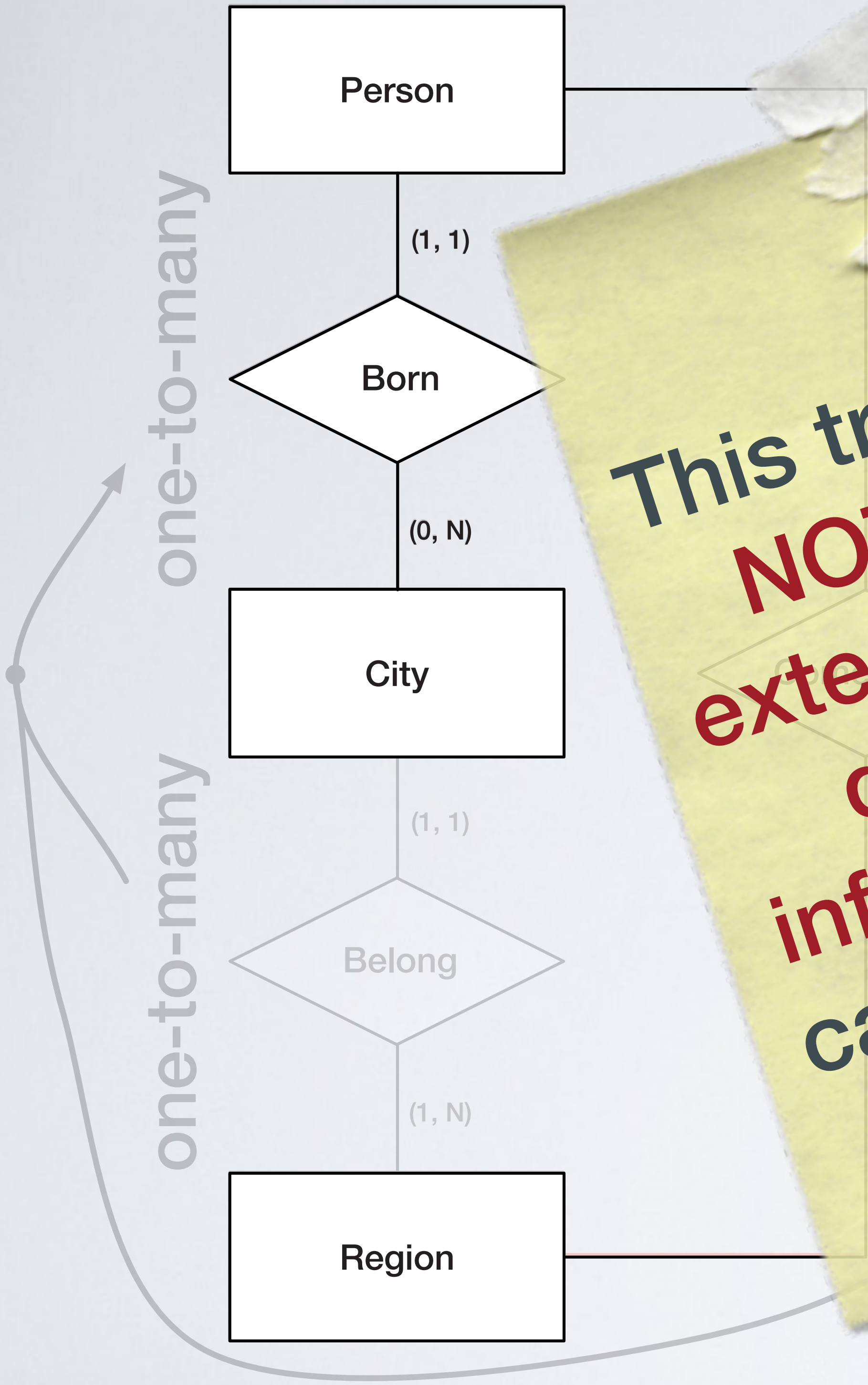


- **External constraint**: every **Person Comes From** the **Region** to which her/his **City** of **Birth Belongs**

$$\text{instance(i, Person)} = \{p_1, p_2, p_3, p_4\}$$
$$\text{instance(i, City)} = \{c_1, c_2, c_3, c_4\}$$
$$\text{instance(i, Region)} = \{r_1, r_2, r_3\}$$

$$\text{instance(i, Born)} = \{(p_1, c_1), (p_2, c_2), (p_3, c_2), (p_4, c_3)\}$$
$$\text{instance(i, Belong)} = \{(c_1, r_1), (c_2, r_2), (c_3, r_1), (c_4, r_3)\}$$
$$\text{instance(i, ComeFrom)} = \{(p_1, r_1), (p_2, r_2), (p_3, r_2), (p_4, r_1)\}$$

Relationship Cycles: Example (1/2)



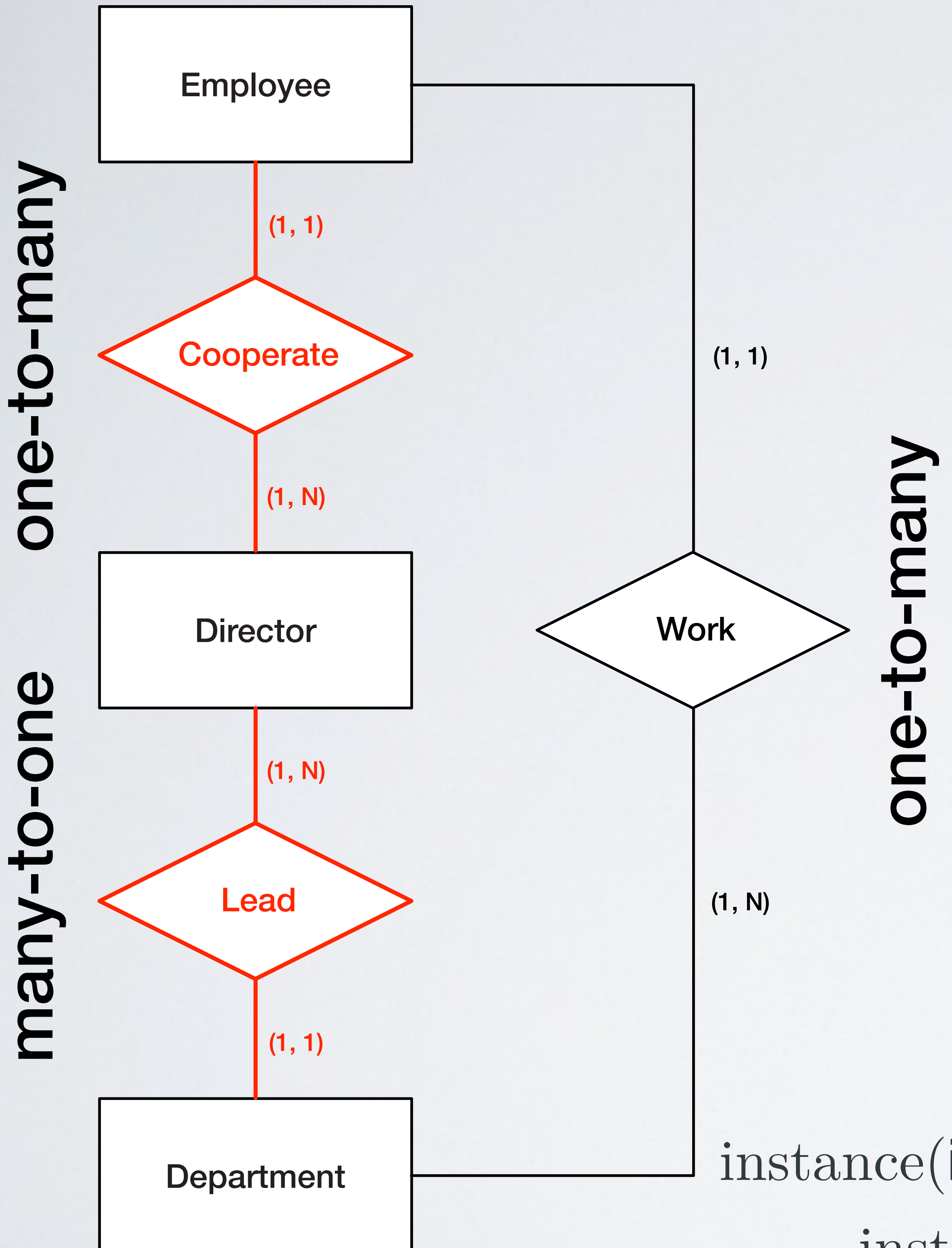
External constraint: every **Person Comes From** the **Region** to which her/his **City** of **Birth Belongs**

This transformation does NOT comply with the external constraint and it does NOT keep the informative content. You cannot say anymore that c_4 belongs to r_3

$$\text{instance(i, Person)} = \{p_1, p_2, p_3, p_4\}$$
$$\text{instance(i, City)} = \{c_1, c_2, c_3, c_4\}$$
$$\text{instance(i, Region)} = \{r_1, r_2, r_3\}$$

$$\text{instance(i, Born)} = \{(p_1, c_1), (p_2, c_2), (p_3, c_2), (p_4, c_3)\}$$
$$\text{instance(i, Belong)} = \{(c_1, r_1), (c_2, r_2), (c_3, r_1), (c_4, r_3)\}$$
$$\text{instance(i, ComeFrom)} = \{(p_1, r_1), (p_2, r_2), (p_3, r_2), (p_4, r_1)\}$$

Relationship Cycles: Example (2/2)

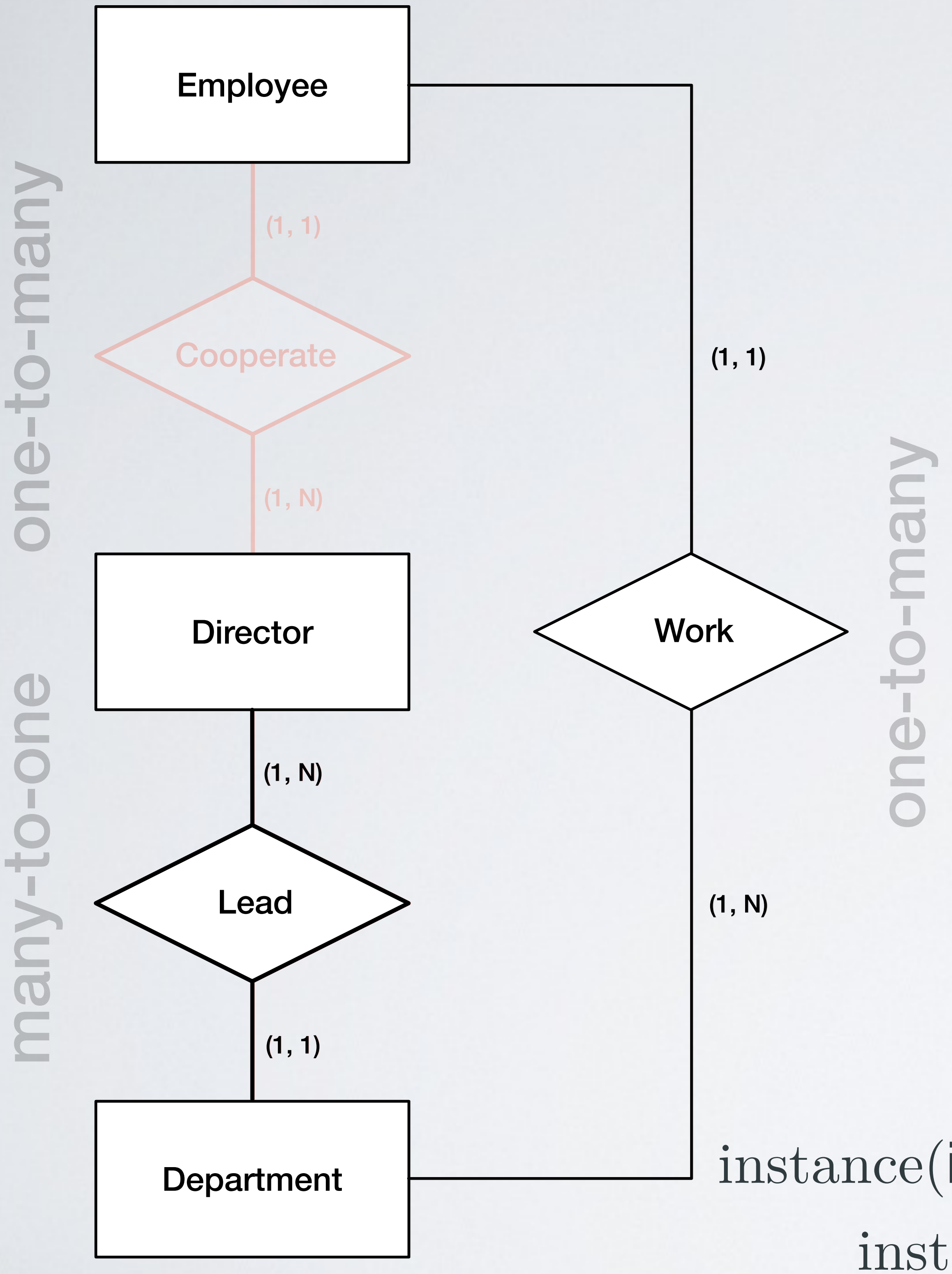


- **External constraint**: every **Employee Cooperates** with the **Director** who **Leads** the **Department** where she/he **Works**

$$\text{instance}(\mathrm{i}, \mathrm{Employee}) = \{e_1, e_2, e_3, e_4\}$$
$$\text{instance}(\mathrm{i}, \mathrm{Director}) = \{d_1, d_2\}$$
$$\text{instance}(\mathrm{i}, \mathrm{Department}) = \{dp_1, dp_2, dp_3\}$$

$$\text{instance}(\mathrm{i}, \mathrm{Cooperate}) = \{(e_1, d_1), (e_2, d_2), (e_3, d_1), (e_4, d_2)\}$$
$$\text{instance}(\mathrm{i}, \mathrm{Lead}) = \{(d_1, dp_1), (d_2, dp_2), (d_1, dp_3)\}$$
$$\text{instance}(\mathrm{i}, \mathrm{Work}) = \{(e_1, dp_1), (e_2, dp_2), (e_3, dp_3), (e_4, dp_2)\}$$

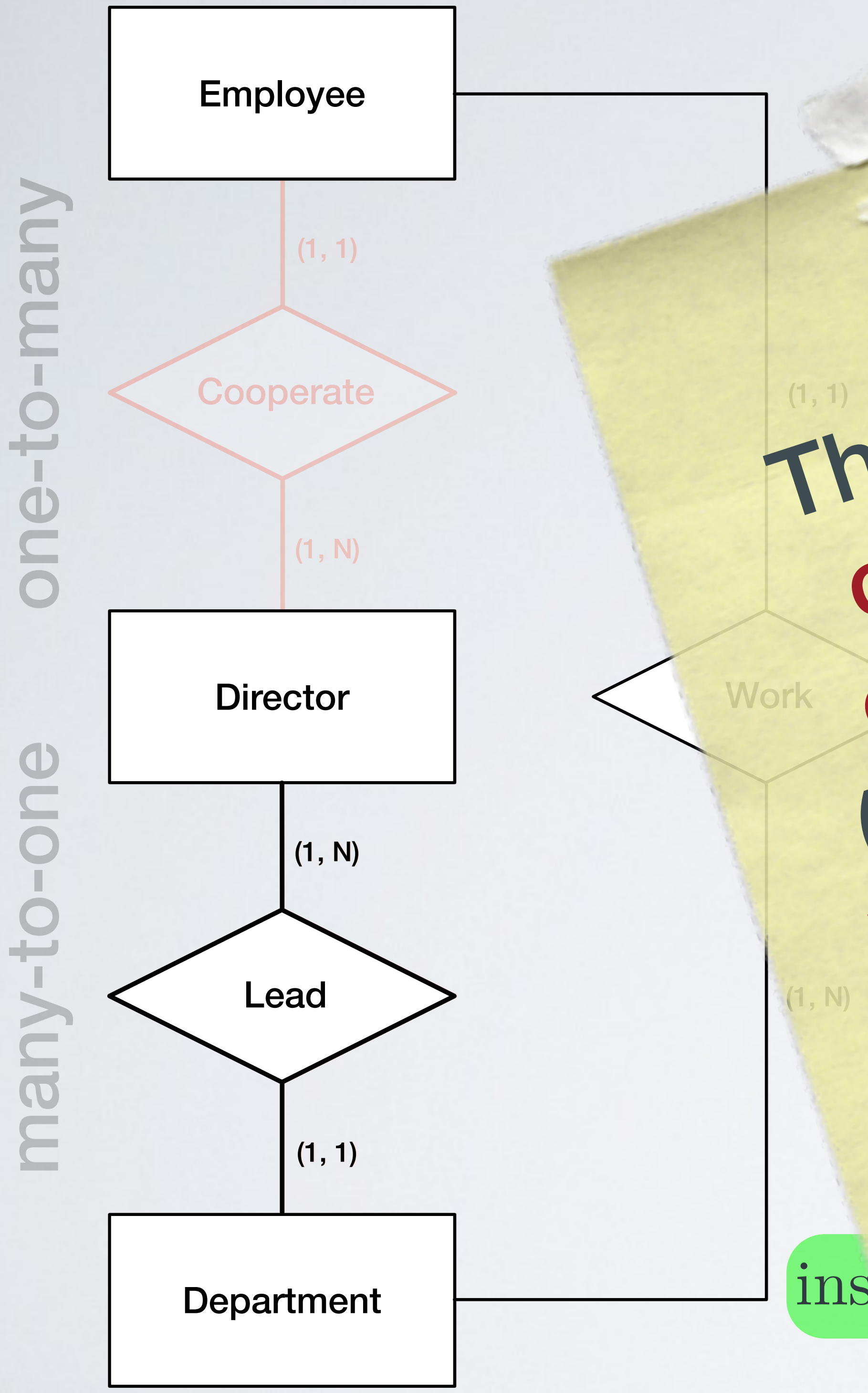
Relationship Cycles: Example (2/2)



- **External constraint**: every **Employee Cooperates** with the **Director** who **Leads** the **Department** where she/he **Works**

$$\text{instance}(\mathsf{i}, \mathsf{Employee}) = \{e_1, e_2, e_3, e_4\}$$
$$\text{instance}(\mathsf{i}, \mathsf{Director}) = \{d_1, d_2\}$$
$$\text{instance}(\mathsf{i}, \mathsf{Department}) = \{dp_1, dp_2, dp_3\}$$

$$\text{instance}(\mathsf{i}, \mathsf{Cooperate}) = \{(e_1, d_1), (e_2, d_2), (e_3, d_1), (e_4, d_2)\}$$
$$\text{instance}(\mathsf{i}, \mathsf{Lead}) = \{(d_1, dp_1), (d_2, dp_2), (d_1, dp_3)\}$$
$$\text{instance}(\mathsf{i}, \mathsf{Work}) = \{(e_1, dp_1), (e_2, dp_2), (e_3, dp_3), (e_4, dp_2)\}$$



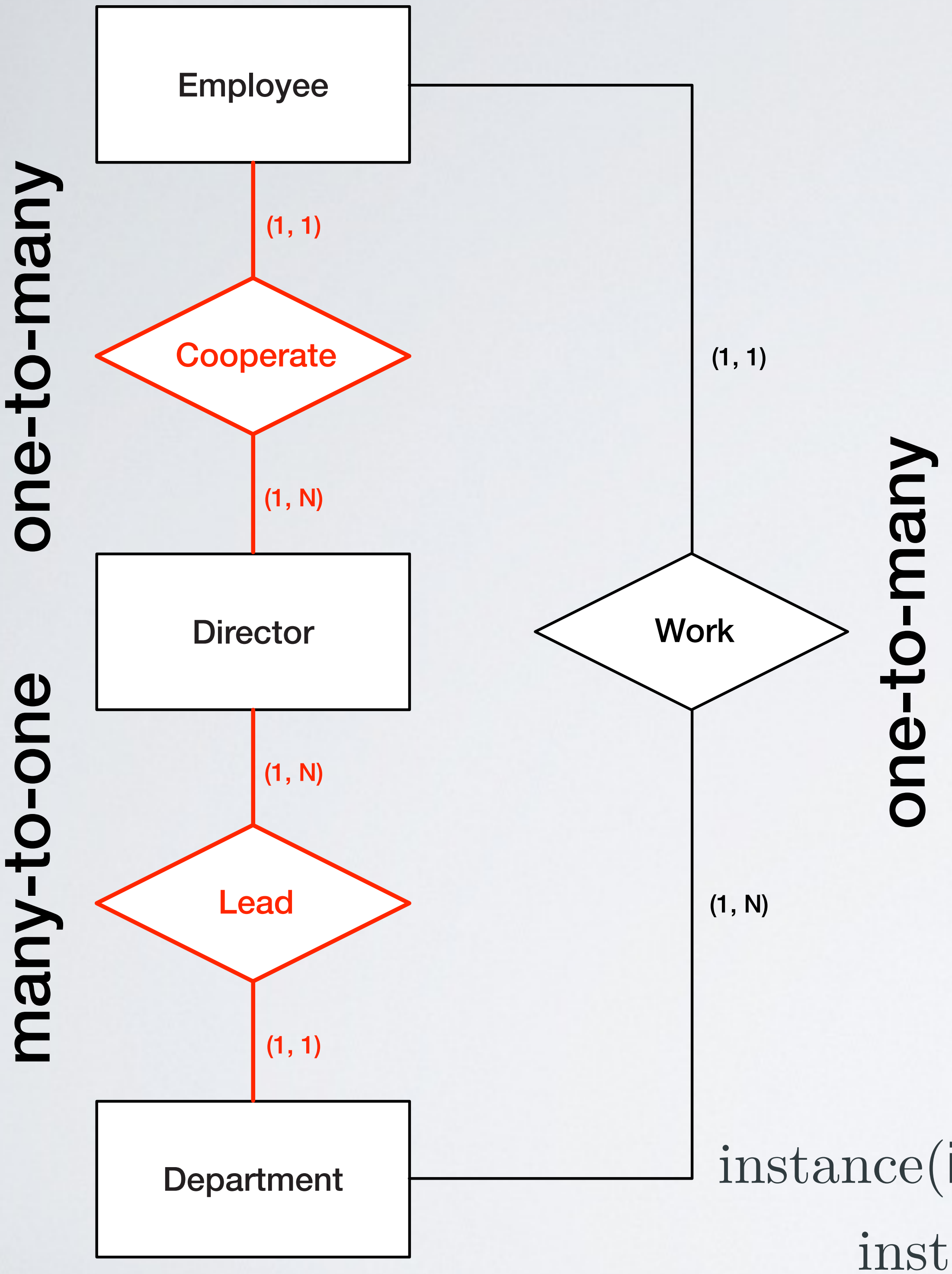
This transformation complies with the external constraint (actually removes it), keeps the informative content and it is self-explaining

External constraint: every **Employee Cooperates** with the **Director** who **Leads** the **Department** where she/he **Works**

$$\text{instance(i, Employee)} = \{e_1, e_2, e_3, e_4\}$$
$$\text{instance(i, Director)} = \{d_1, d_2\}$$
$$\text{instance(i, Department)} = \{dp_1, dp_2, dp_3\}$$

$$\text{instance(i, Cooperate)} = \{(e_1, d_1), (e_2, d_2), (e_3, d_1), (e_4, d_2)\}$$
$$\text{instance(i, Lead)} = \{(d_1, dp_1), (d_2, dp_2), (d_1, dp_3)\}$$
$$\text{instance(i, Work)} = \{(e_1, dp_1), (e_2, dp_2), (e_3, dp_3), (e_4, dp_2)\}$$

Relationship Cycles: Example (2/2)

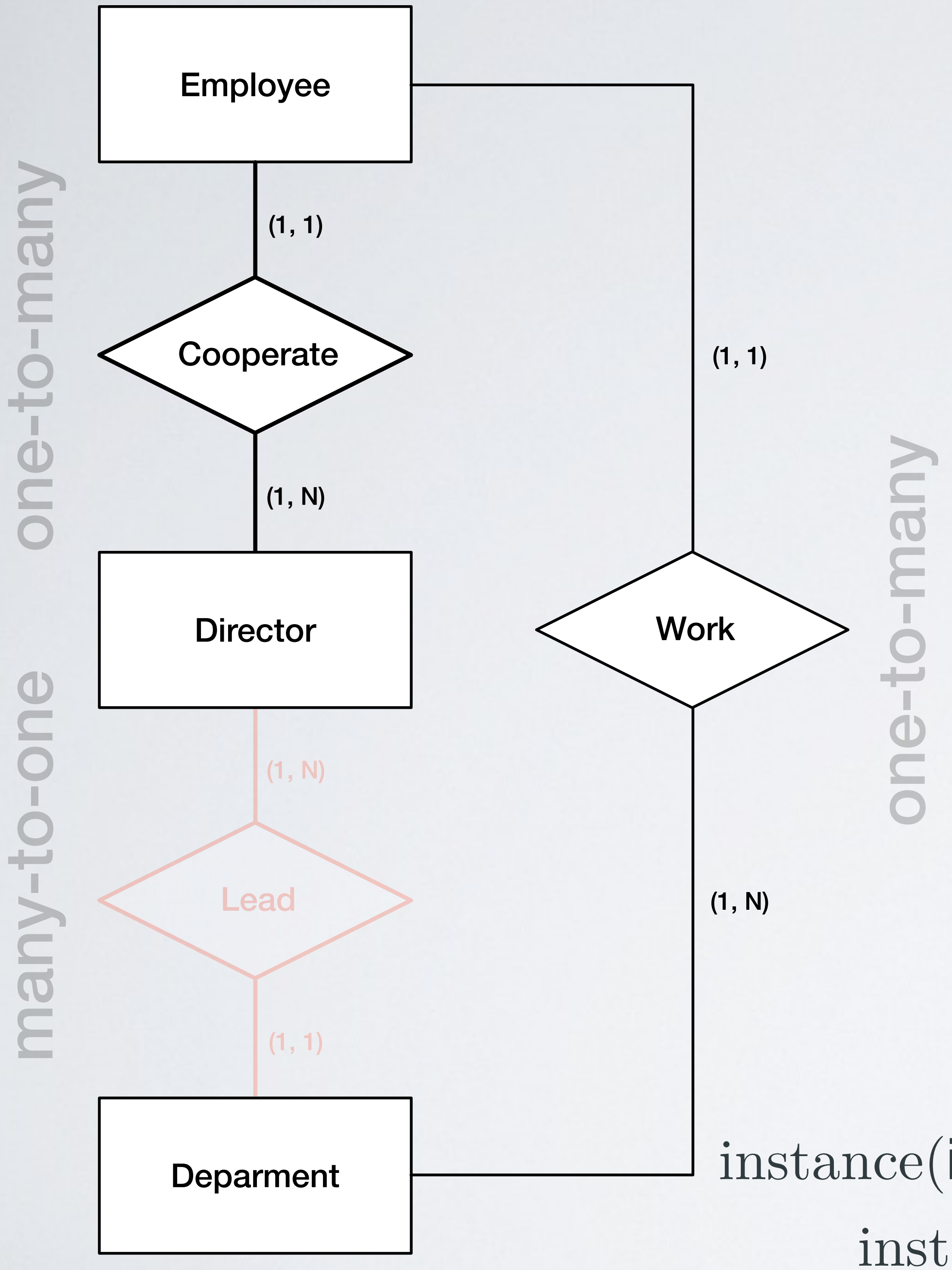


- **External constraint**: every **Employee Cooperates** with the **Director** who **Leads** the **Department** where she/he **Works**

$$\text{instance}(\text{i}, \text{Employee}) = \{e_1, e_2, e_3, e_4\}$$
$$\text{instance}(\text{i}, \text{Director}) = \{d_1, d_2\}$$
$$\text{instance}(\text{i}, \text{Department}) = \{dp_1, dp_2, dp_3\}$$

$$\text{instance}(\text{i}, \text{Cooperate}) = \{(e_1, d_1), (e_2, d_2), (e_3, d_1), (e_4, d_2)\}$$
$$\text{instance}(\text{i}, \text{Lead}) = \{(d_1, dp_1), (d_2, dp_2), (d_1, dp_3)\}$$
$$\text{instance}(\text{i}, \text{Work}) = \{(e_1, dp_1), (e_2, dp_2), (e_3, dp_3), (e_4, dp_2)\}$$

Relationship Cycles: Example (2/2)

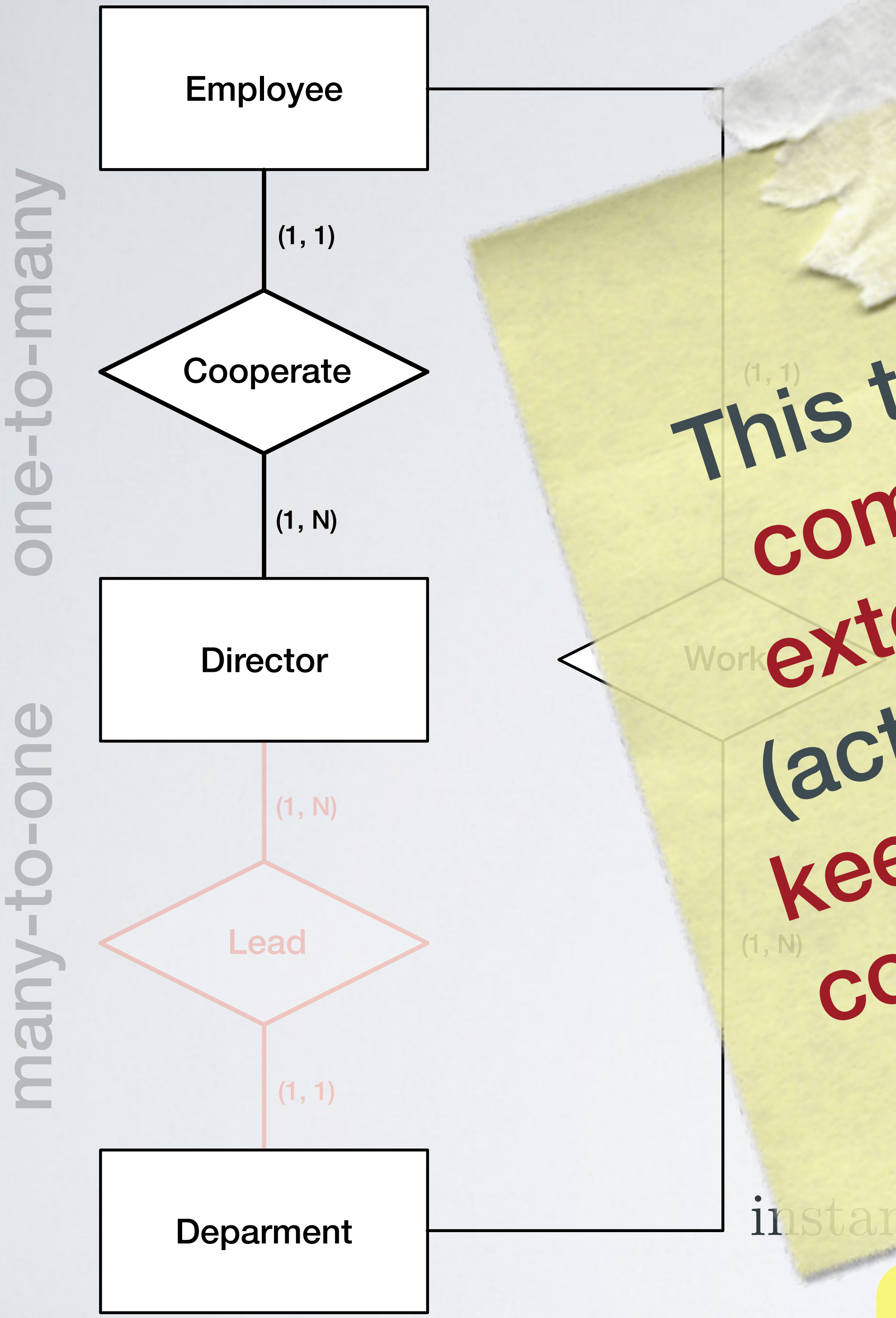


- **External constraint**: every **Employee Cooperates** with the **Director** who **Leads** the **Department** where she/he **Works**

$$\text{instance(i, Employee)} = \{e_1, e_2, e_3, e_4\}$$
$$\text{instance(i, Director)} = \{d_1, d_2\}$$
$$\text{instance(i, Department)} = \{dp_1, dp_2, dp_3\}$$

$$\text{instance(i, Cooperate)} = \{(e_1, d_1), (e_2, d_2), (e_3, d_1), (e_4, d_2)\}$$
$$\text{instance(i, Lead)} = \{(d_1, dp_1), (d_2, dp_2), (d_1, dp_3)\}$$
$$\text{instance(i, Work)} = \{(e_1, dp_1), (e_2, dp_2), (e_3, dp_3), (e_4, dp_2)\}$$

Relationship Cycles: Example (2/2)



This transformation **complies with the external constraint** (actually removes it), keeps the **informative content** and it is **NOT self-explaining**

External constraint: every **Employee Cooperates** with the **Director** who **Leads** the **Department** where she/he **Works**

$\text{instance(i, Employee)} = \{e_1, e_2, e_3, e_4\}$

$\text{instance(i, Director)} = \{d_1, d_2\}$

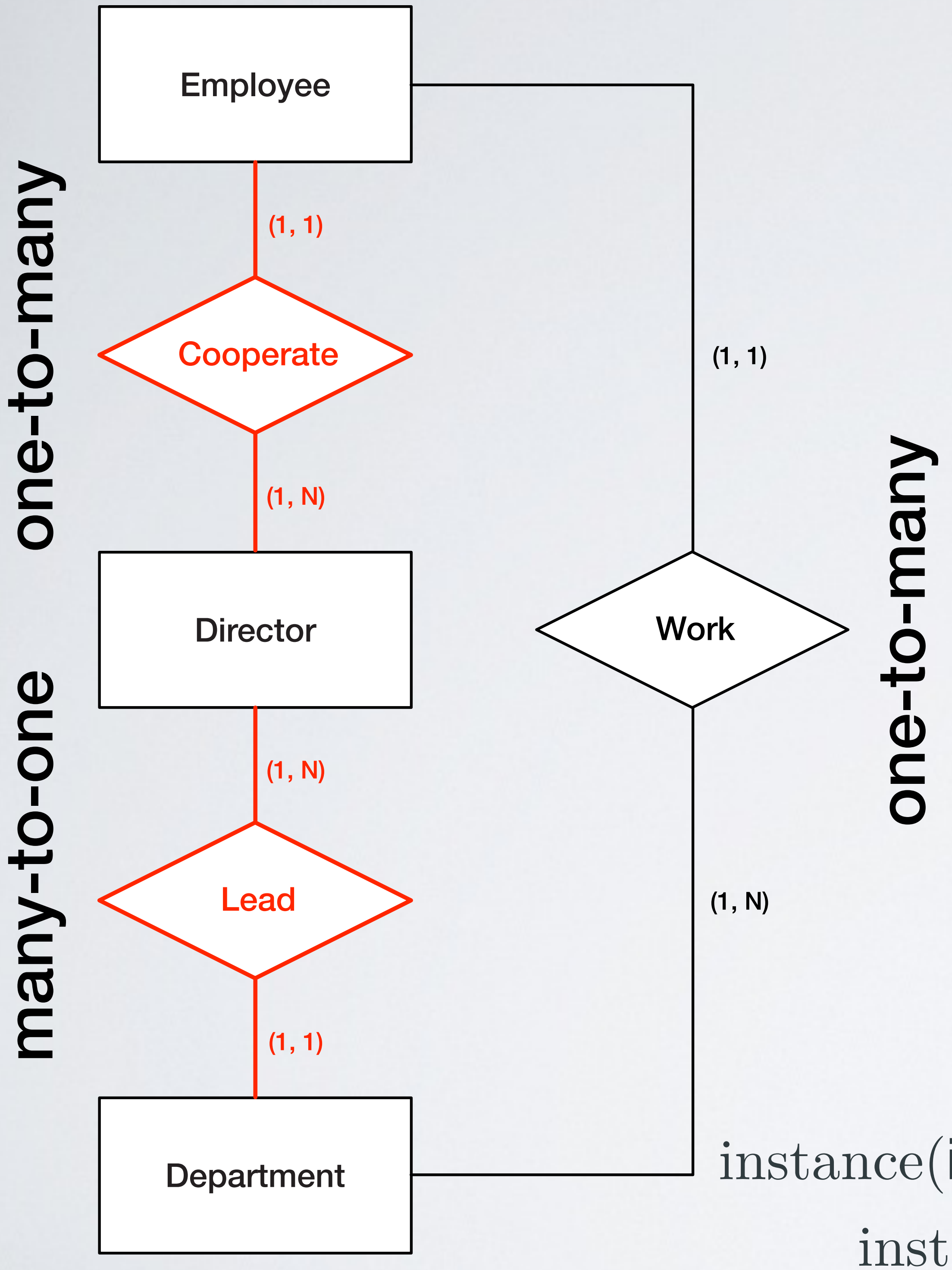
$\text{instance(i, Department)} = \{dp_1, dp_2, dp_3\}$

$\text{instance(i, Cooperate)} = \{(e_1, d_1), (e_2, d_2), (e_3, d_1), (e_4, d_2)\}$

$\text{instance(i, Lead)} = \{(d_1, dp_1), (d_2, dp_2), (d_1, dp_3)\}$

$\text{instance(i, Work)} = \{(e_1, dp_1), (e_2, dp_2), (e_3, dp_3), (e_4, dp_2)\}$

Relationship Cycles: Example (2/2)

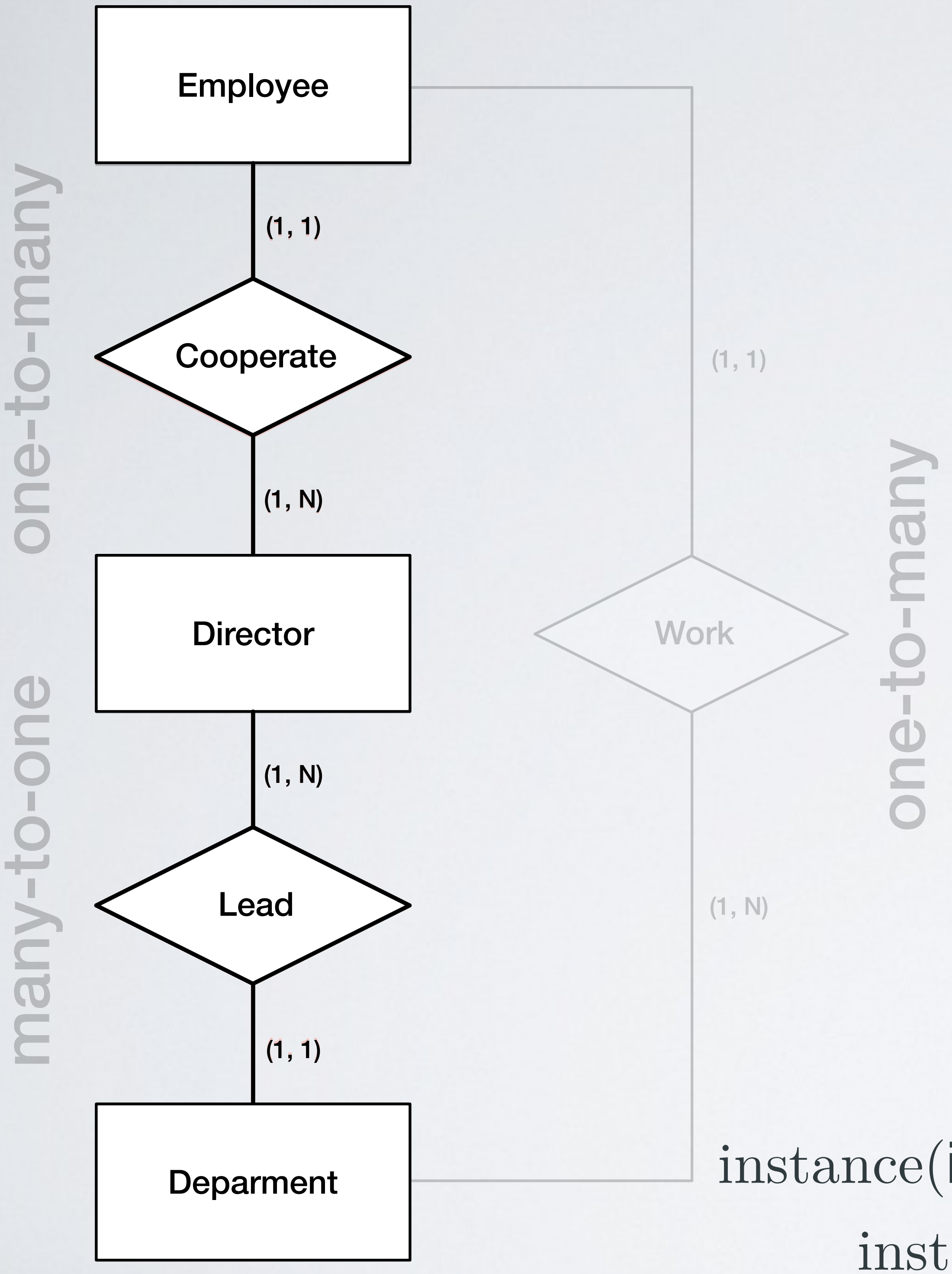


- **External constraint**: every **Employee Cooperates** with the **Director** who **Leads** the **Department** where she/he **Works**

$$\text{instance}(\text{i}, \text{Employee}) = \{e_1, e_2, e_3, e_4\}$$
$$\text{instance}(\text{i}, \text{Director}) = \{d_1, d_2\}$$
$$\text{instance}(\text{i}, \text{Department}) = \{dp_1, dp_2, dp_3\}$$

$$\text{instance}(\text{i}, \text{Cooperate}) = \{(e_1, d_1), (e_2, d_2), (e_3, d_1), (e_4, d_2)\}$$
$$\text{instance}(\text{i}, \text{Lead}) = \{(d_1, dp_1), (d_2, dp_2), (d_1, dp_3)\}$$
$$\text{instance}(\text{i}, \text{Work}) = \{(e_1, dp_1), (e_2, dp_2), (e_3, dp_3), (e_4, dp_2)\}$$

Relationship Cycles: Example (2/2)

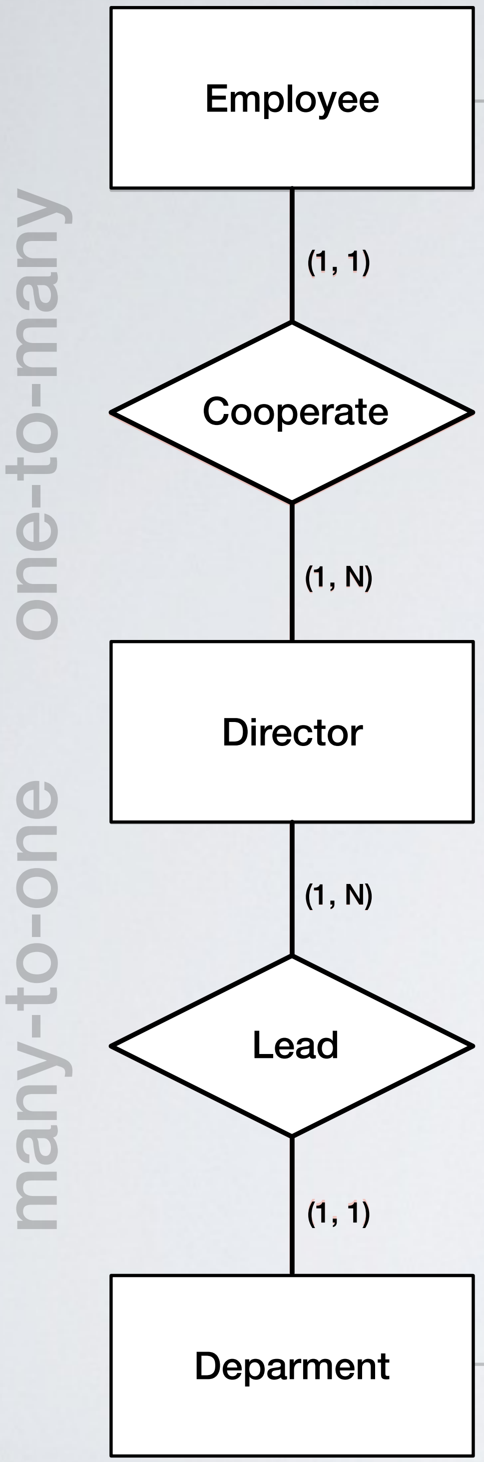


- **External constraint**: every **Employee Cooperates** with the **Director** who **Leads** the **Department** where she/he **Works**

$$\text{instance}(\text{i}, \text{Employee}) = \{e_1, e_2, e_3, e_4\}$$
$$\text{instance}(\text{i}, \text{Director}) = \{d_1, d_2\}$$
$$\text{instance}(\text{i}, \text{Department}) = \{dp_1, dp_2, dp_3\}$$

$$\text{instance}(\text{i}, \text{Cooperate}) = \{(e_1, d_1), (e_2, d_2), (e_3, d_1), (e_4, d_2)\}$$
$$\text{instance}(\text{i}, \text{Lead}) = \{(d_1, dp_1), (d_2, dp_2), (d_1, dp_3)\}$$
$$\text{instance}(\text{i}, \text{Work}) = \{(e_1, dp_1), (e_2, dp_2), (e_3, dp_3), (e_4, dp_2)\}$$

Relationship Cycles: Example (2/2)



External constraint: every **Employee Cooperates** with the **Director** who **Leads** the **Department** where she/he **Works**

This transformation does **NOT comply** with the **external constraint** and it **does NOT keep** the **informative content**. You cannot say anymore whether e_1 works in dp_1 or in dp_3

$$\text{instance(i, Employee)} = \{e_1, e_2, e_3, e_4\}$$

$$\text{instance(i, Director)} = \{d_1, d_2\}$$

$$\text{instance(i, Department)} = \{dp_1, dp_2, dp_3\}$$

$$\text{instance(i, Cooperate)} = \{(e_1, d_1), (e_2, d_2), (e_3, d_1), (e_4, d_2)\}$$

$$\text{instance(i, Lead)} = \{(d_1, dp_1), (d_2, dp_2), (d_1, dp_3)\}$$

$$\text{instance(i, Work)} = \{(e_1, dp_1), (e_2, dp_2), (e_3, dp_3), (e_4, dp_2)\}$$

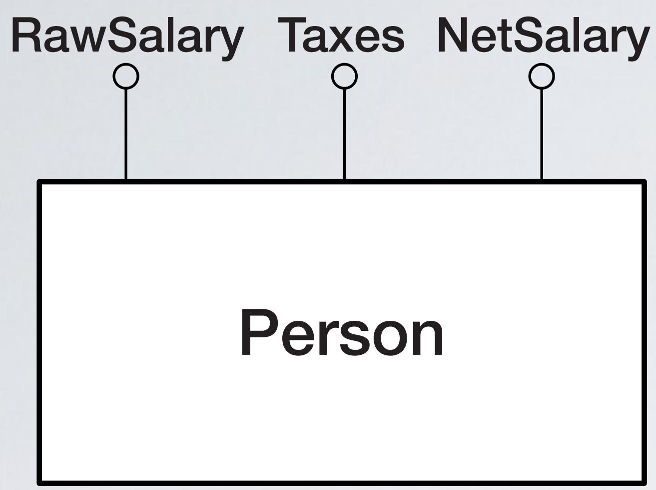
Derived Attribute

- Derived attributes are due to
  1. other attributes in the same entity (relationship)
  2. attributes of other entities (relationships)

- For each redundancy, we should consider the expected **application load** (updates, queries, space occupation) and decide whether it is better to keep or remove it
  - **Pros**: it is not necessary to compute the value of the attribute at running time and we reduce the number of accesses to the databases
  - **Cons**: we need to add further external constraints to ensure consistency, we need additional processing to keep the value of the derivable attribute aligned with the data it is computed from, we use more storage space

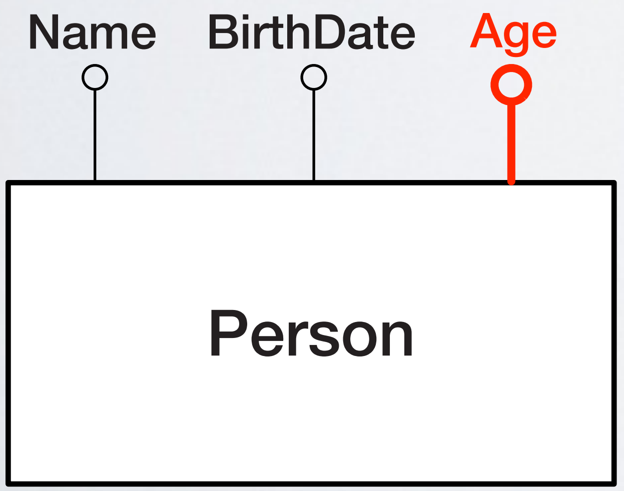
- If we decide to keep the redundancy, it must be properly documented

Derived Attributes: Same Entity



- **External Constraint:**

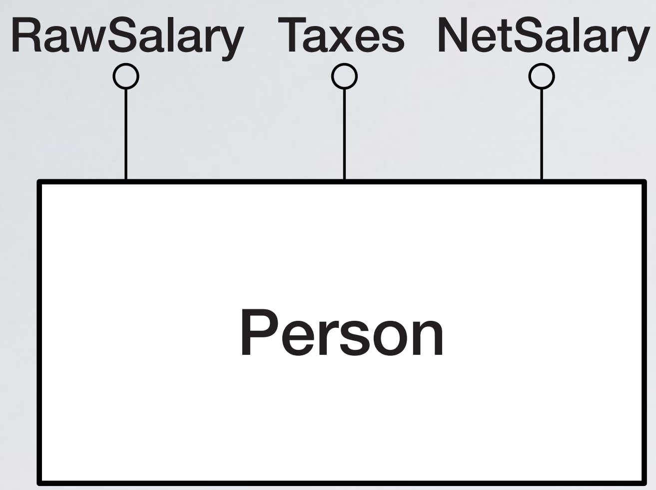
  NetSalary = RawSalary - Taxes



- **External Constraint:**

  Age = Now() - BirthDate

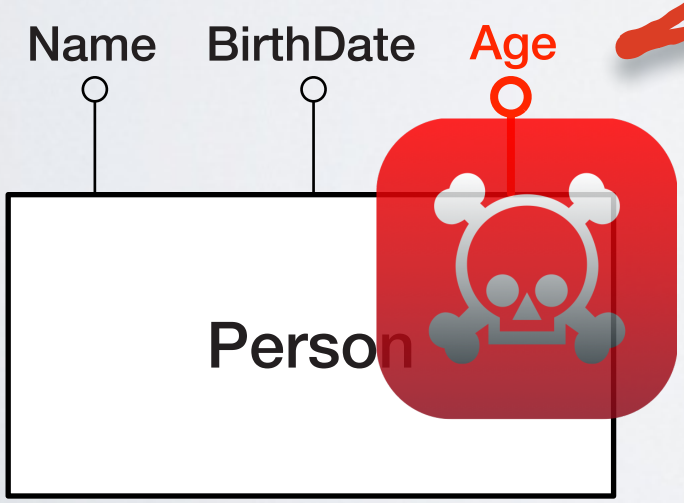
Derived Attributes: Same Entity



- **External Constraint**:

NetSalary = RawSalary - Taxes

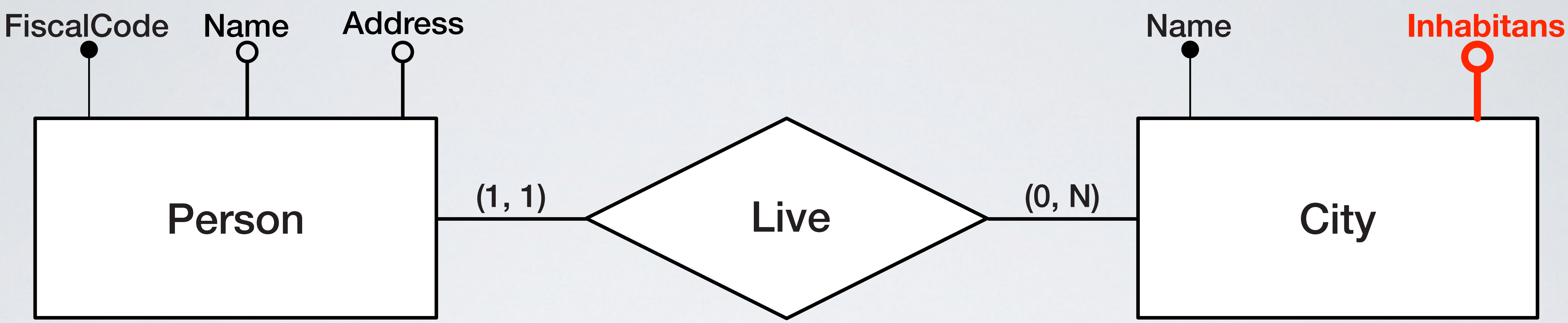
Not only Age can be derived from BirthDate, leading to update disadvantages, but it is also potentially always **inconsistent**, because the age keeps changing over time



- **External Constraint**:

Age = Now() - BirthDate

Derived Attributes: Other Entities



- Inhabitants can be computed by summing, for each instance of City, the number of instances of the Live relationship where that City takes part in
- To decide whether to keep (or not) Inhabitants we need to perform an analysis of the estimate load of the database

Database Load

The load of a database is the site of activities and/or application it has to support

- Notion similar to other areas of engineering, like load on a beam or on an electric circuit
- We need two types of information to characterise the load of a database
  - **data volume**
  - **operations** to be supported

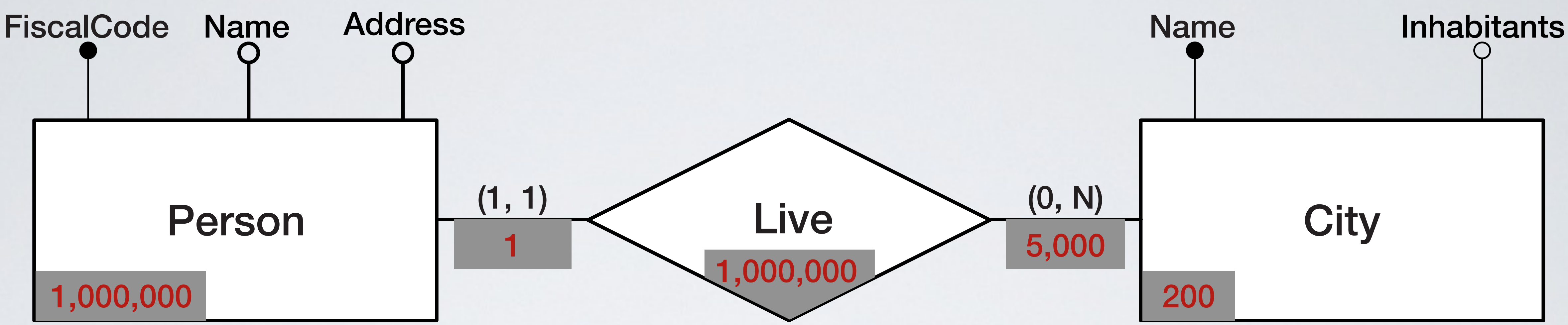
Data Volume

- The data volume is estimated by
  - average number of instances of each entity
  - average number of instances of each relationship
  - average number of instances of an entity participating in a relationship

- We can represent the data volume on the ER schema by indicating the average number of instances within entities and relationships and the average cardinalities next to the cardinality constraints (x, y)

- The data volume is summarised in a table reporting all the entities and relationships, as well as the average number of their instances

Data Volume: Example



Concept	Construct	Volume
Person	Entity	1,000,000
City	Entity	200
Live	Relationship	1,000,000

Operations

An operation is a sequence of elementary interactions (insert, update, delete, read) with the database.

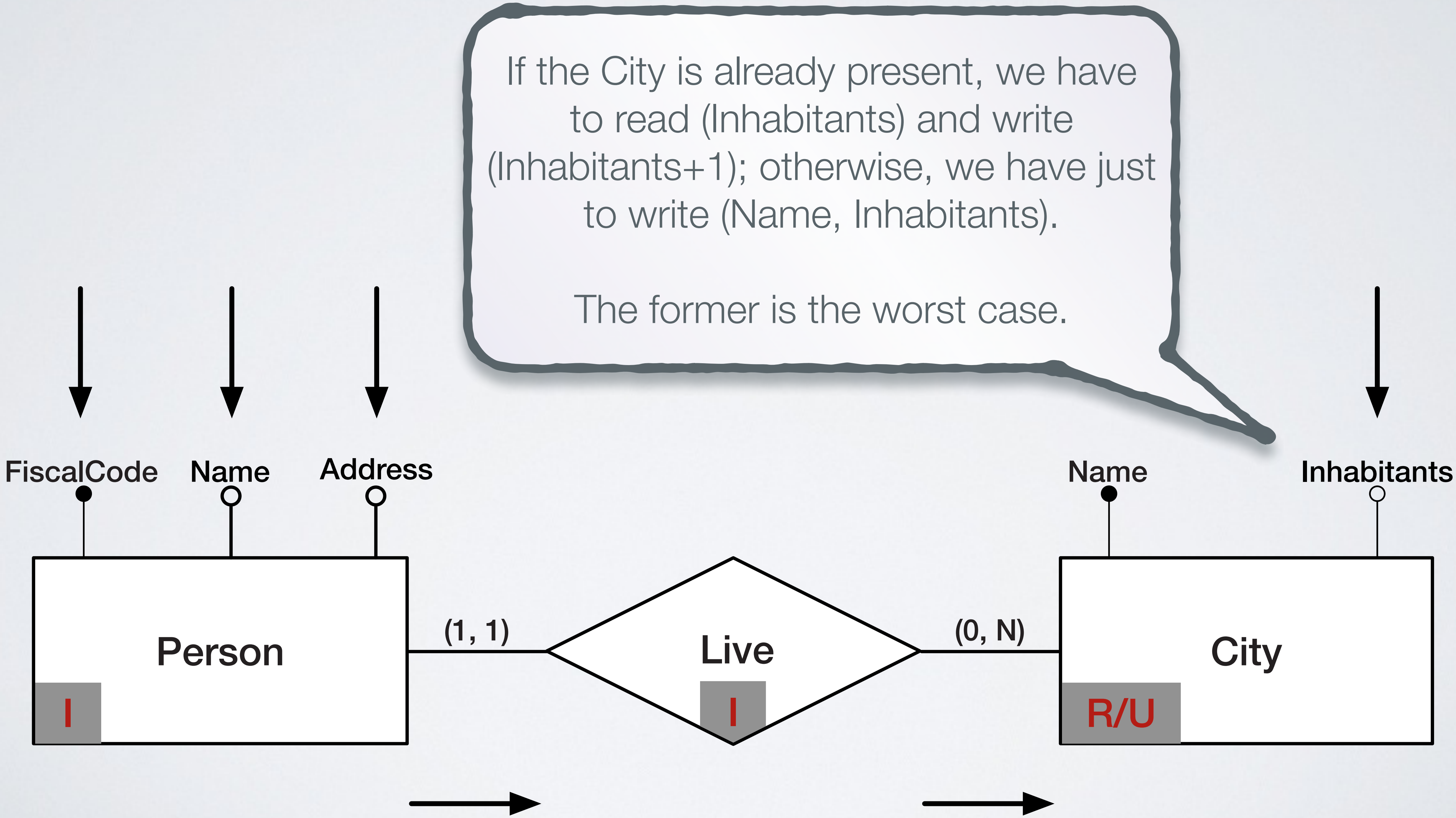
- For each operation we draw the **operation schema**, i.e. a subset of the whole ER schema containing only the entities and relationships involved in an operation
- From the operation schema, we can draw a navigation schema where
  - arrows indicate attributes used in the selection conditions
  - arrows between relationships indicate navigation
  - the symbols **I**, **U**, **D**, and **R** within entities and/or relationships mean insert, update, delete and read, respectively

Operations: Example

- **O_1 - Insert new person**: store a new person together with his/her city
- **O_2 - Print data about a city**: print all the data about a city, including the number of its inhabitants
- **O_3 - Summarise data about all the cities**: summarise all the data about all the cities, including the number of inhabitants

Navigation Schema: Example

- **O_1 - Insert new person**: store a new person together with his/her city



Online vs Batch Operations

- **Online operations** have to be executed interactively to satisfy a request issued by a user
- **Batch operations** are executed independently from the interaction with the user and can be run in background
  - they typically are lengthy operations, run at scheduled times, e.g. during night, when the database load is lighter
- Online operations should be considered more important than batch ones

Operations: Information to be Gathered

- **Frequency table**
  - the name of the operation
  - a description of the operation
  - the frequency of the operation
  - the type of the operation (online vs batch)
- **Access/volume table**
  - the constructs involved in the operation, as described in the navigation schema
  - the type of construct (entity or relationship)
  - the number of accesses for that construct
  - the type of access (read - R - or write - W - for insert, update, and delete)
  - the average total number of accesses for that construct

$$\text{Average Access} = \text{Access} \times \text{Frequency} \times \text{Weight}$$

where **weight** typically is **1** for **read** and **2** for **write**

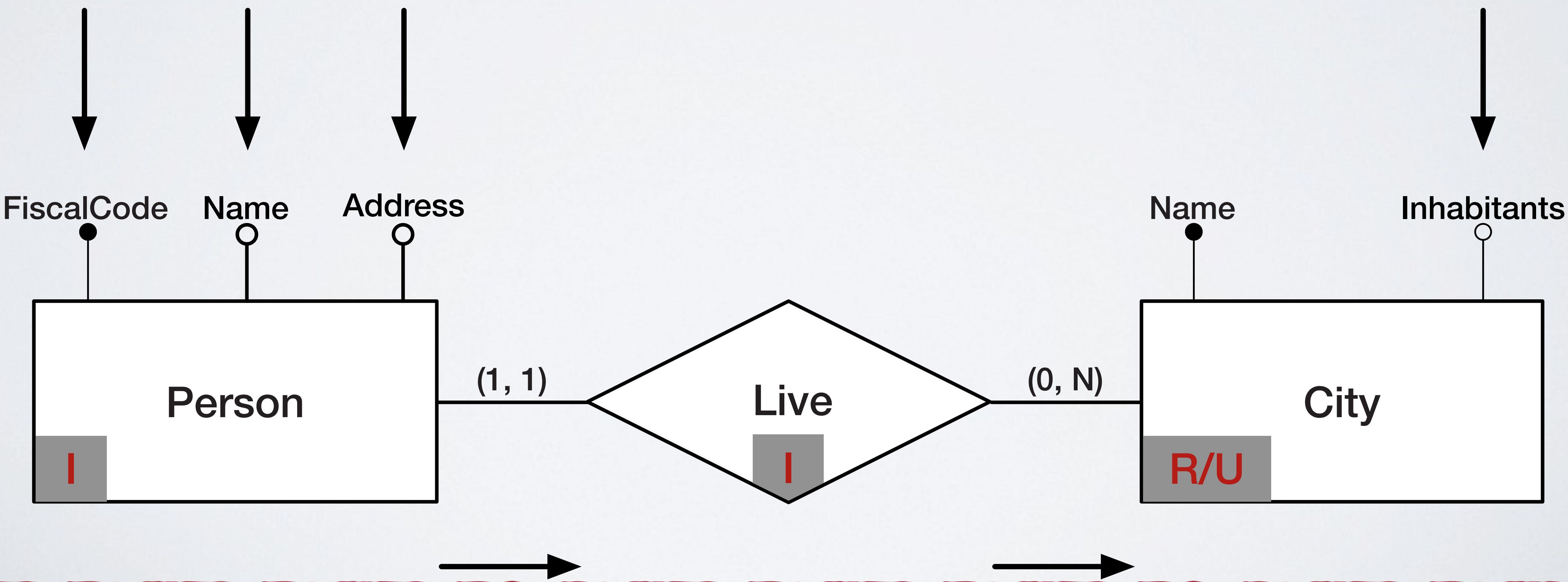
Frequency Table: Example

Operation	Description	Frequency	Type
O_1 Insert new person	store a new person together with his/her city	500/day	Online
O_2 Print data about a city	print all the data about a city, including the number of its inhabitants	2/day	Online
O_3 Summarise data about all the cities	summarise all the data about all the cities, including the number of inhabitants	1/year	Batch

Access/volume Table: Example with Redundancy (1/2)

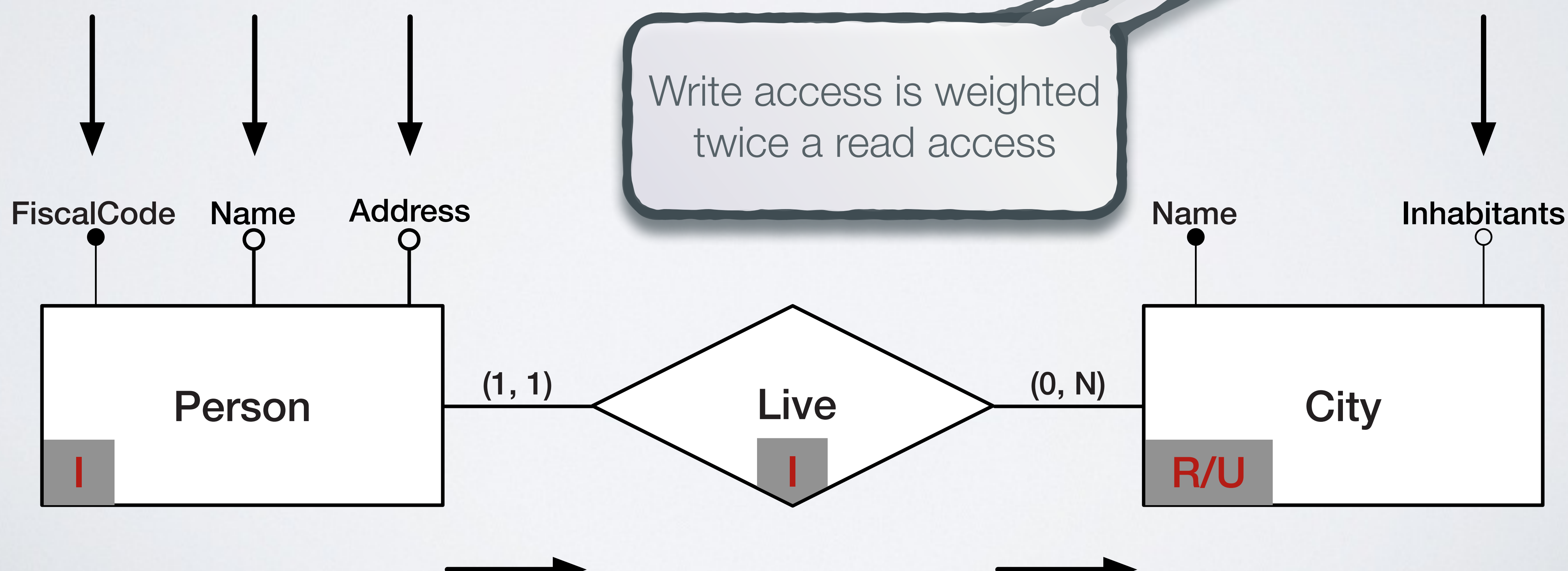
Operation O_1 (500/day)

Concept	Construct	Access	Type	Average Access
Person	Entity	1	W	$1 \times 500 \times 2 = 1{,}000$
Live	Relationship	1	W	$1 \times 500 \times 2 = 1{,}000$
City	Entity	1	R	$1 \times 500 \times 1 = 500$
City	Entity	1	W	$1 \times 500 \times 2 = 1{,}000$
Total Access			**3,500**	



Operation O_1 (500/day)

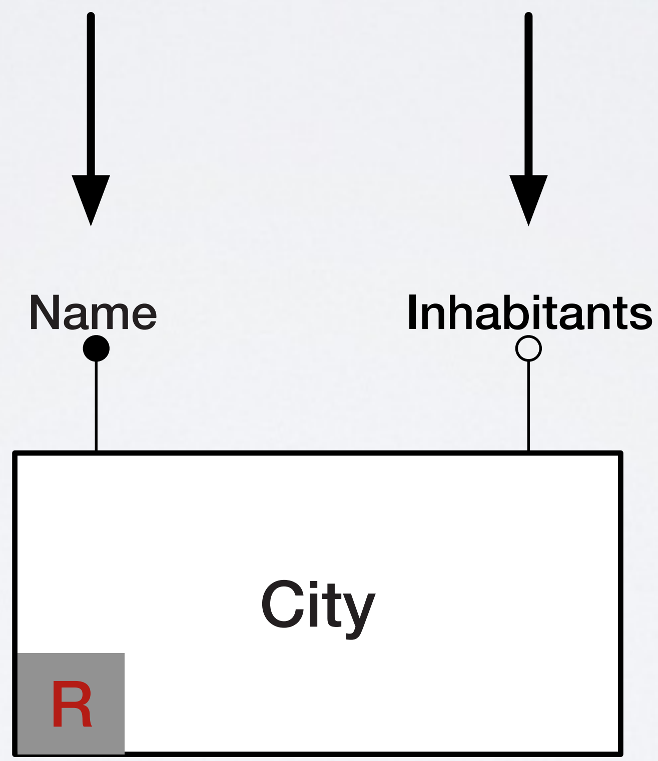
Concept	Construct	Access	Type	Average Access
Person	Entity	1	W	1 × 500 × 2 = 1,000
Live	Relationship	1	W	1 × 500 × 2 = 1,000
City	Entity	1	R	1 × 500 × 1 = 500
City	Entity	1	W	1 × 500 × 2 = 1,000
Total Access				**3,500**



Access/volume Table: Example with Redundancy (2/2)

Operation O_2 (2/day)

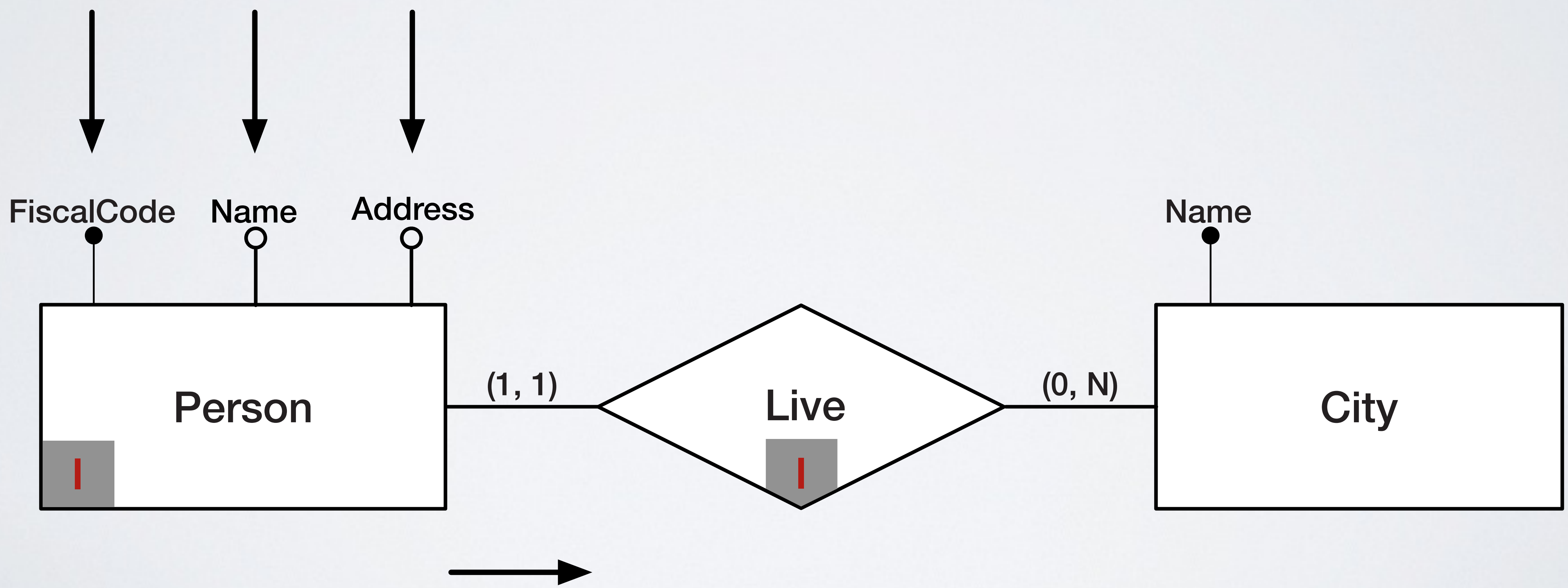
Concept	Construct	Access	Type	Average Access
City	Entity	1	R	$1 \times 2 \times 1 = 2$
Total Access			**2**	



Access/volume Table: Example without Redundancy (1/2)

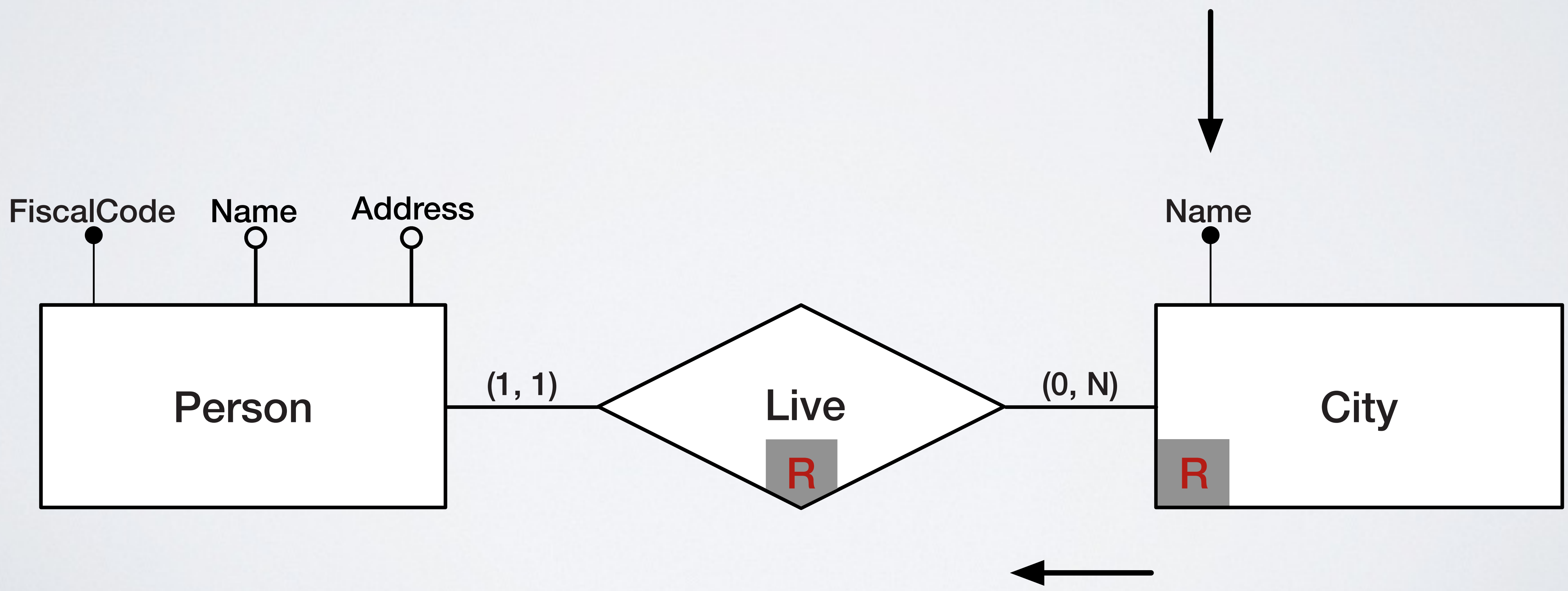
Operation O_1 (500/day)

Concept	Construct	Access	Type	Average Access
Person	Entity	1	W	1 × 500 × 2 = 1,000
Live	Relationship	1	W	1 × 500 × 2 = 1,000
Total Access			**2,000**	



Operation O_2 (2/day)

Concept	Construct	Access	Type	Average Access
City	Entity	1	R	1 × 2 × 1 = 2
Live	Relationship	5,000	R	5,000 × 2 × 1 = 10,000
Total Access			**10,002**	



Redundancy Analysis: Comparison and Choice

Operation	With Redundancy	Without Redundancy
O1	3,500	2,000
O2	2	10,002
Total Access/Day	**3,502**	**12,002**

- If you consider only O_1 it is better to remove the redundancy BUT, if you consider only O_2 or both O_1 and O_2, then it is clearly better to keep the redundancy
- The redundancy must be documented and we need to add an external constraint on the insert/delete of Person and Live to update City accordingly

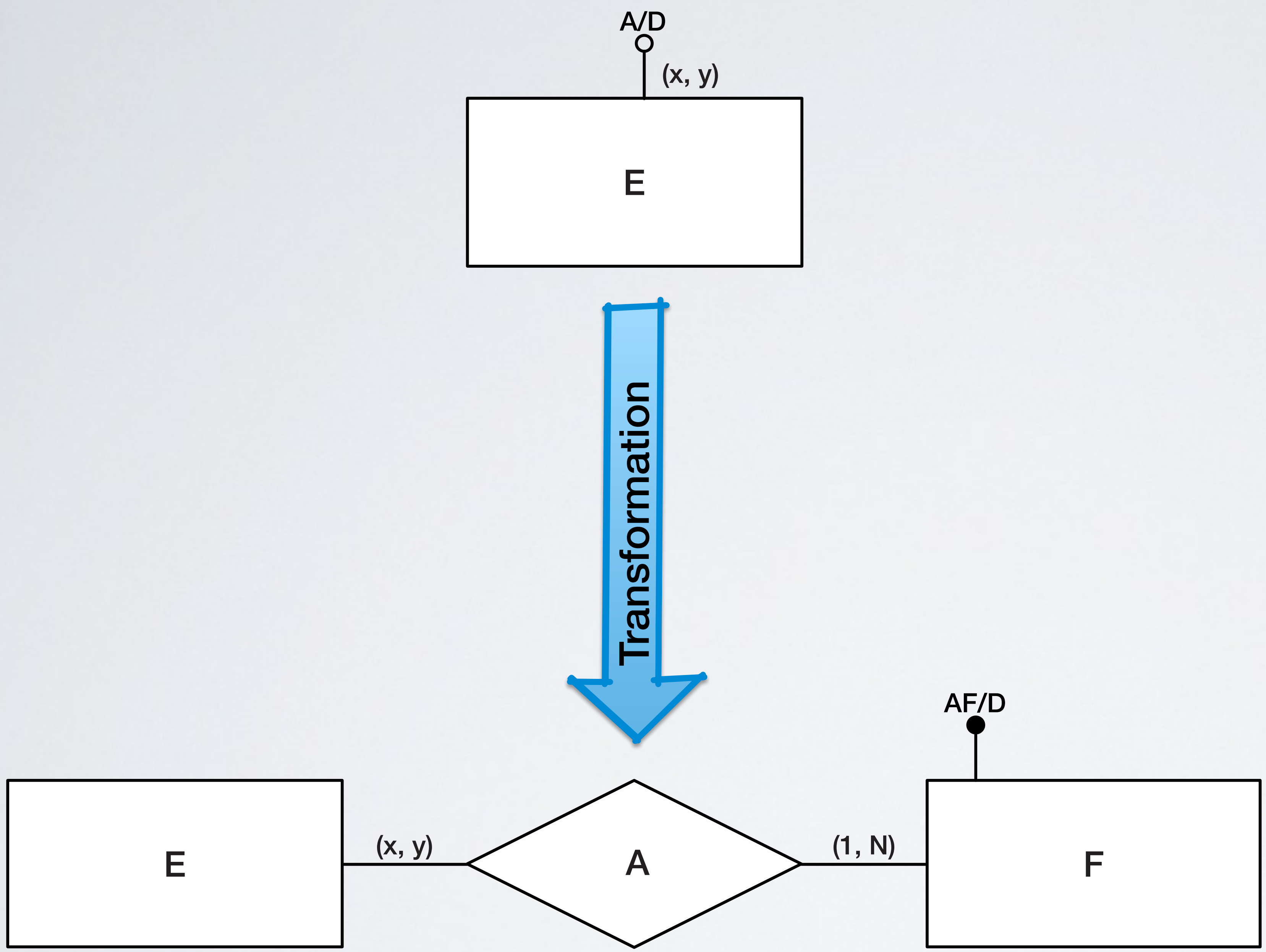
Transformation - Step 2: Removal of multi-valued attributes

Multi-valued Attributed

- A **multi-value attribute** cannot be directly mapped into the relational model

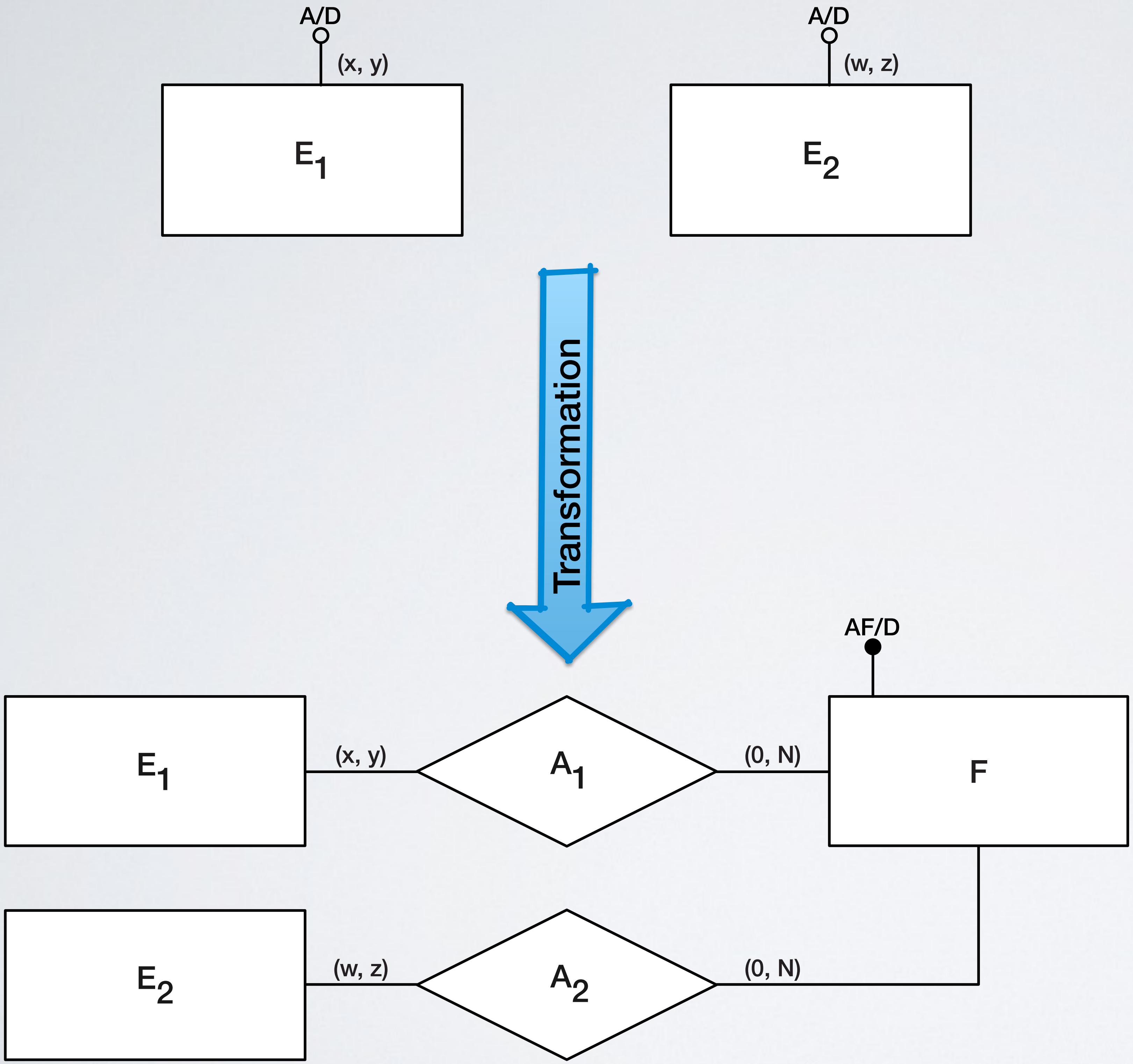
- We need to remove all the multi-valued attributes:
  - In the case of an entity, we transform the attribute into a new entity linked to the original one by a binary relationship
  - in the case of a relationships, we have to transform the relationship into an entity, first; and then we proceed as above

Removal of a multi-valued attribute of an entity (1/2)



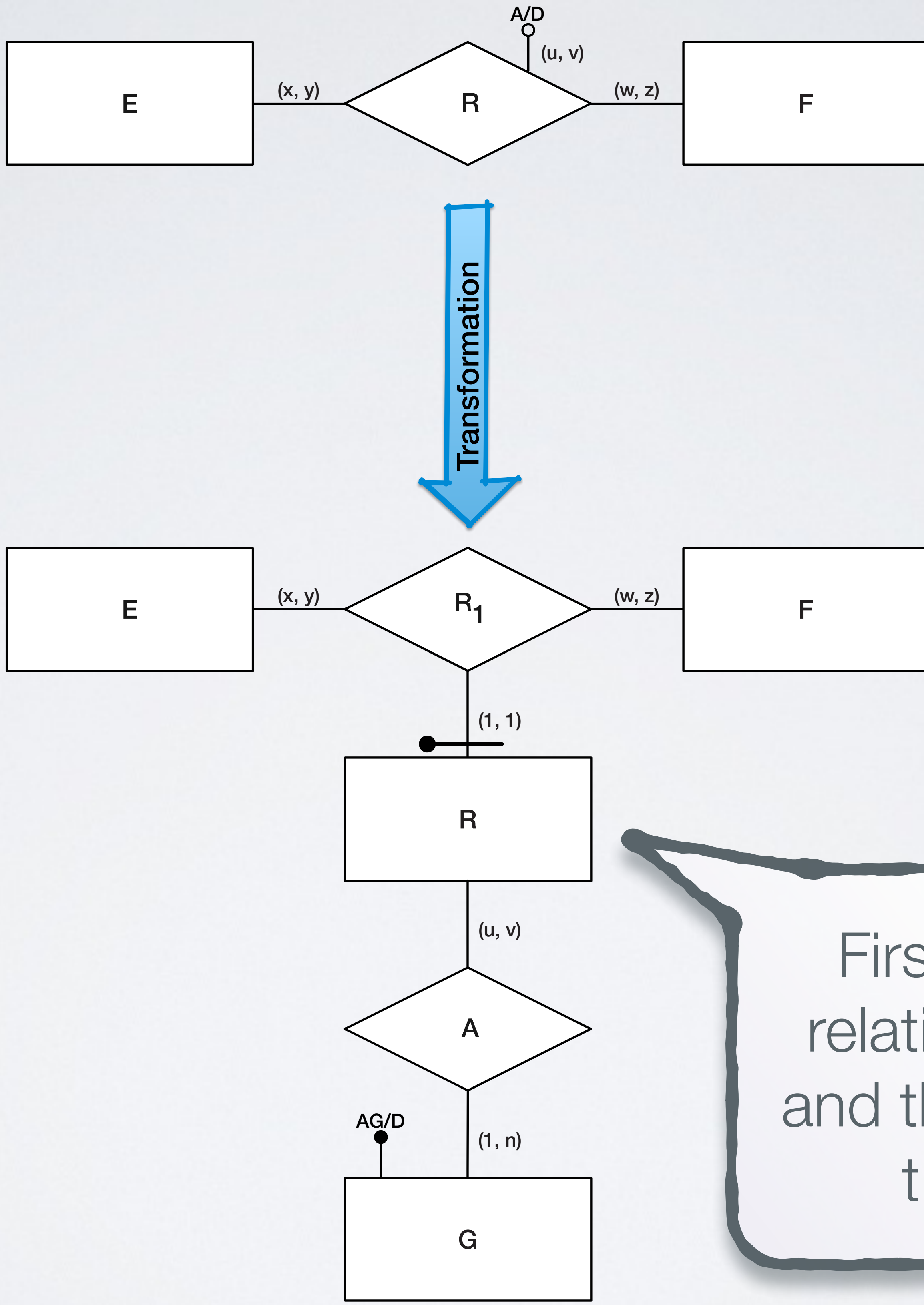
- The cardinality (1, n) means that we are interested only in the instances of **F** which represents values of the attribute **A** actually used by the instances of **E** in the original schema

Removal of a multi-valued attribute of an entity (2/2)



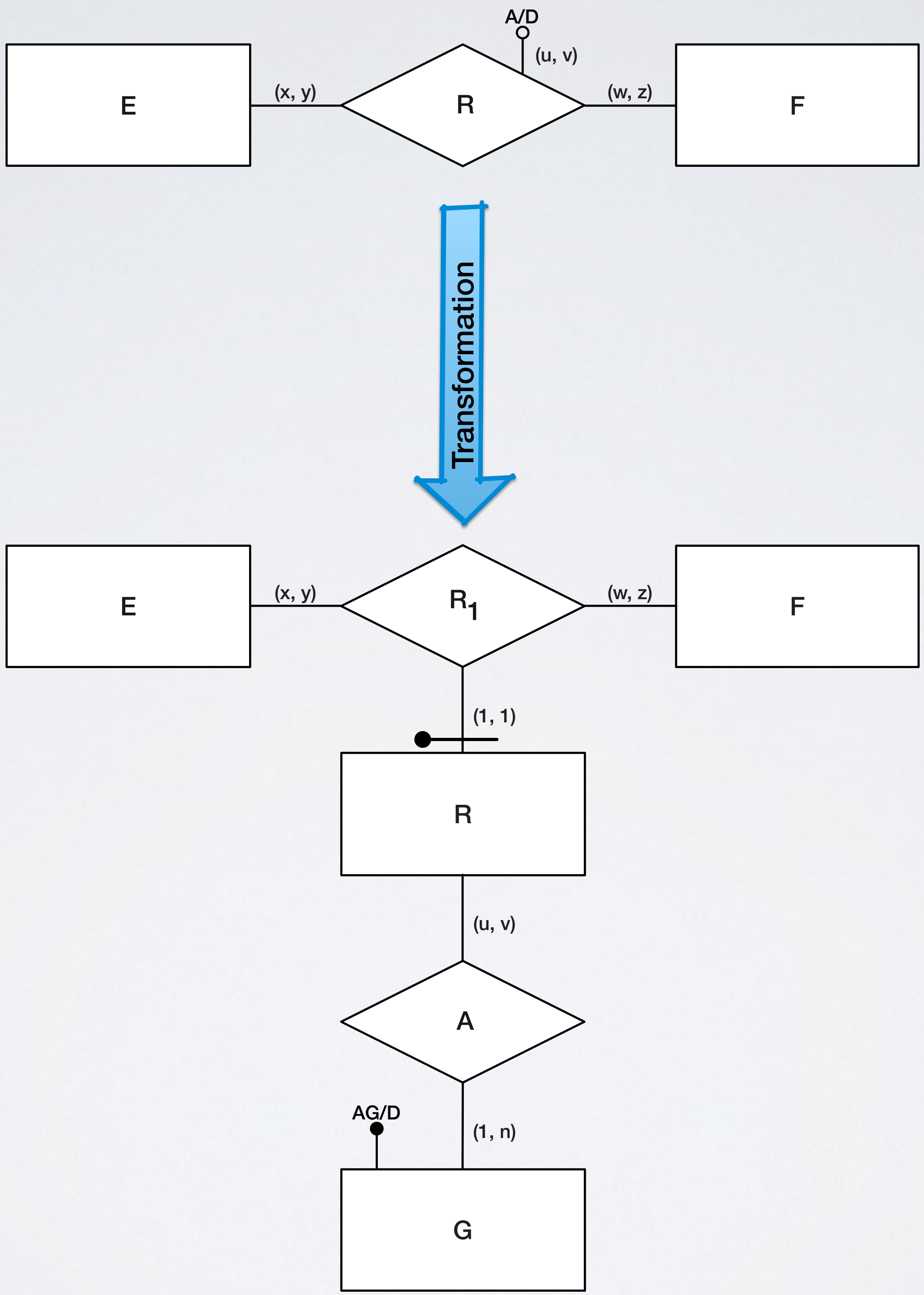
- We need to add an **external constraint** to "mimic" the cardinality (1, n) on the participation of **F** to the union of instances the relationships $\mathbf{A_1}$ and $\mathbf{A_2}$

Removal of a multi-valued attribute of a relationship (1/2)

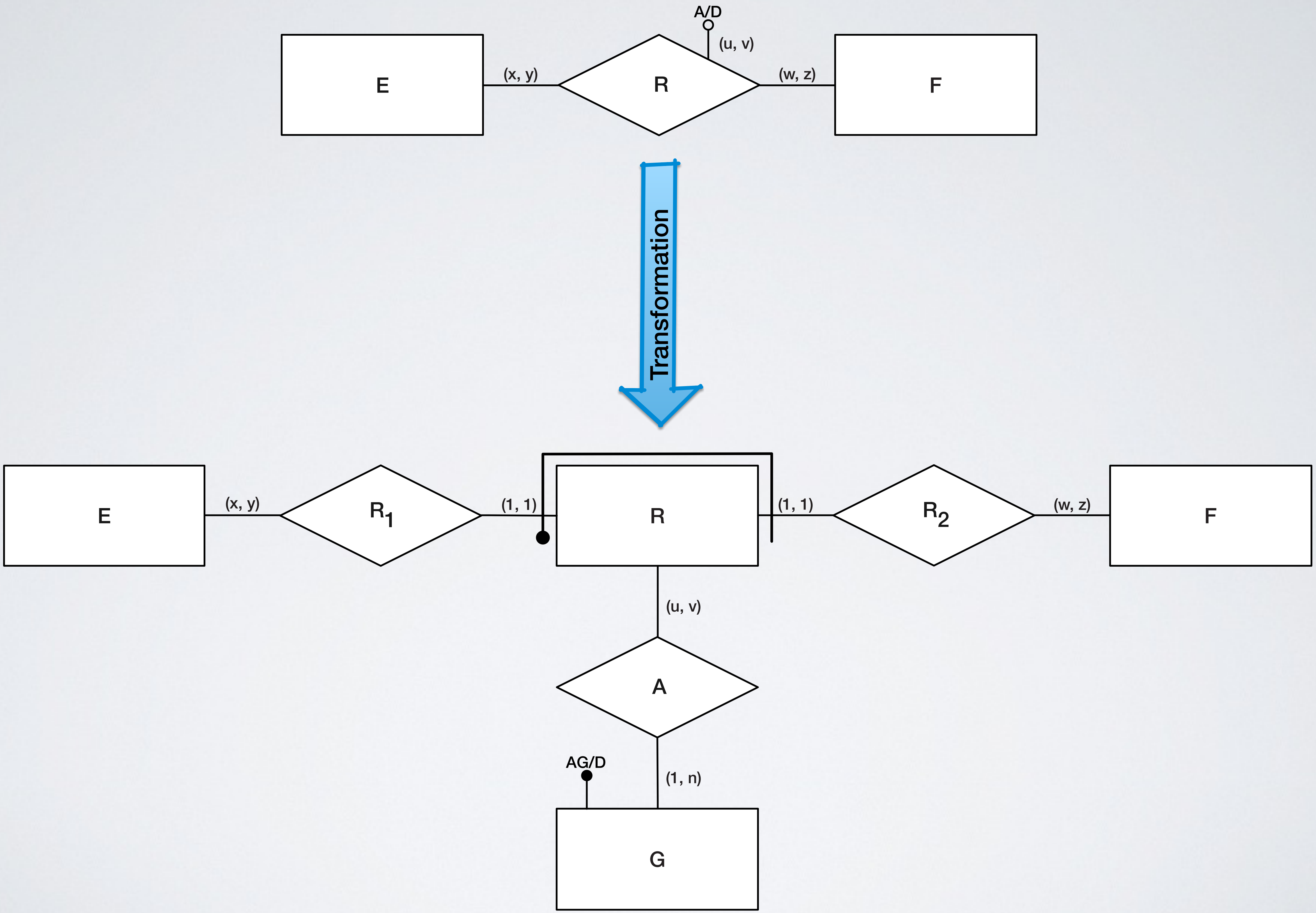


First, we transform the relationship into an entity and then we proceed as in the previous case

Removal of a multi-valued attribute of a relationship (1/2)



Removal of a multi-valued attribute of a relationship (2/2)

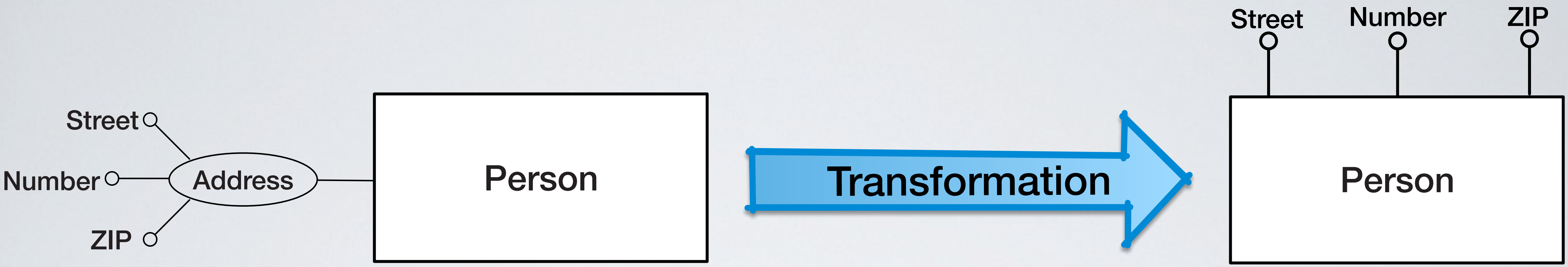


Transformation - Step 3: Removal of composite attributes

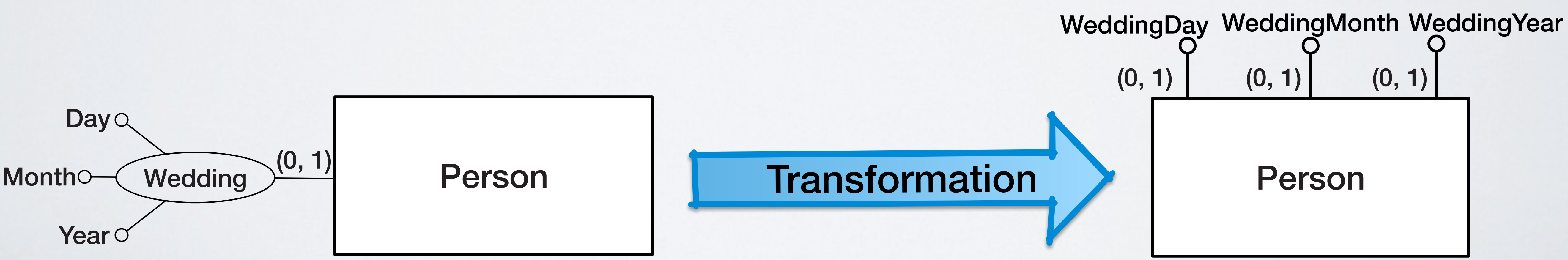
Composite Attributes

- At this point, a **composite attribute** has cardinality either (1, 1) or (0, 1)
- If the attribute has cardinality (1, 1), we can directly associate the component attributes to the entity (relationship)
- If the attribute has cardinality (0, 1)
  - we can proceed as in the case of the attributes with cardinality (1,1) but paying attention to add an external constraint to represent the optional presence
  - we can transform the composite attribute into a new entity, whose attributes at the component attributes, and link it to the existing entity through a binary relationship with cardinality (0, 1)
    - in the case the composite attribute belongs to a relationship, we have to transform the relationship into an entity and the proceed as above

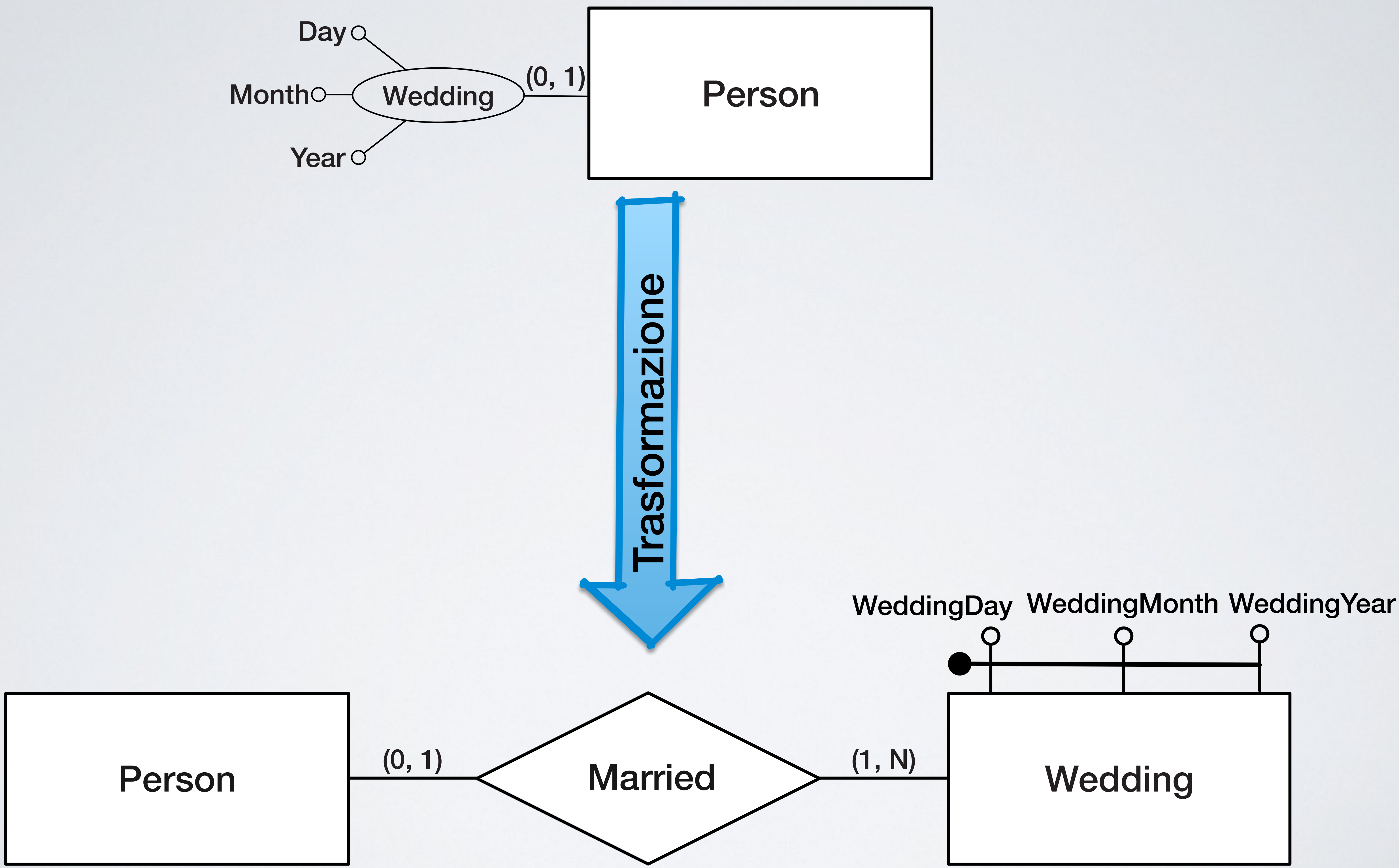
Removal of composite attributes (1/2)



- **External constraint**: for each instance of **Person**, each attribute **WeddingDay**, **WeddingMonth** and **WeddingYear** is defined only if also the other two are define



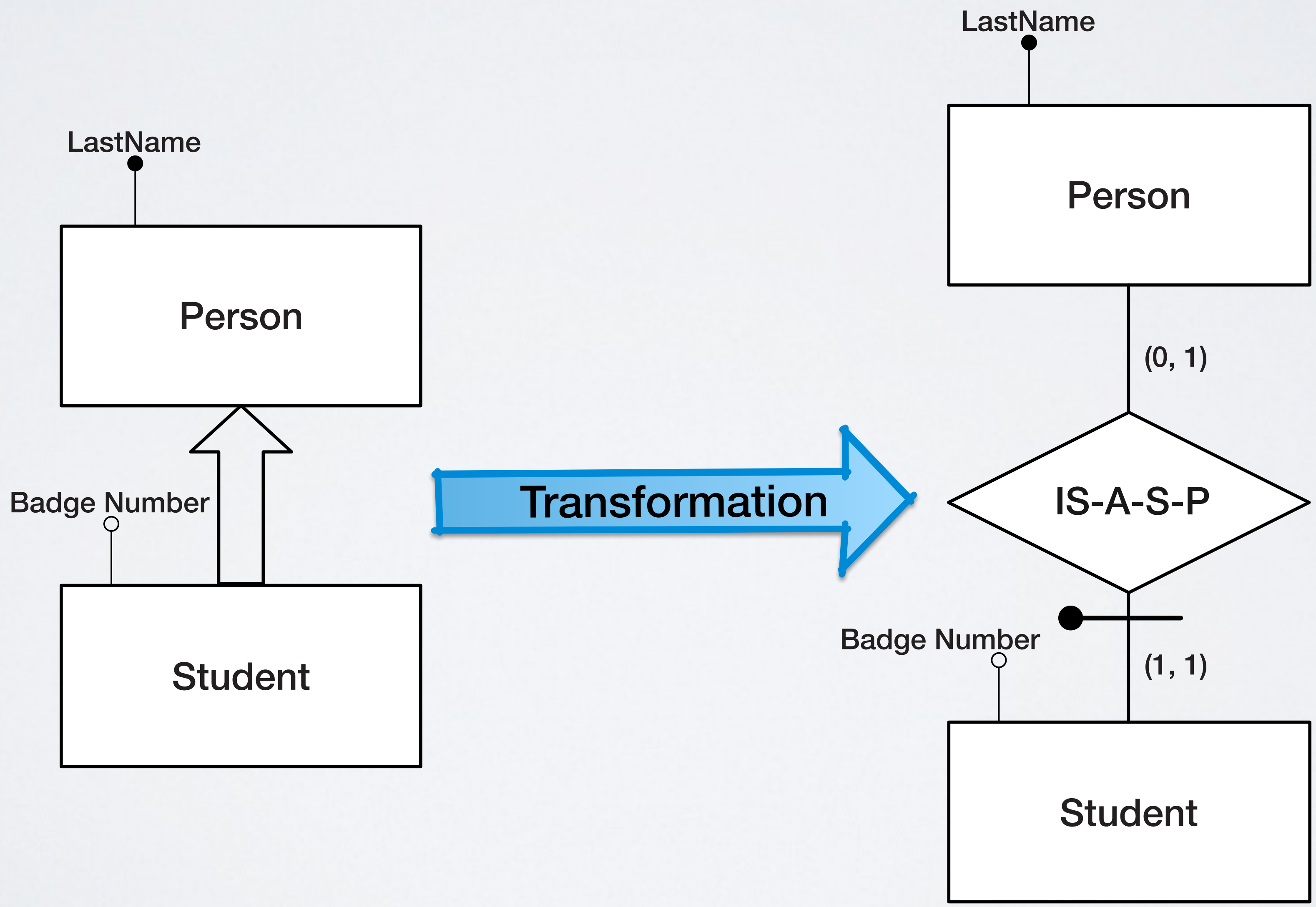
Removal of composite attributes (2/2)



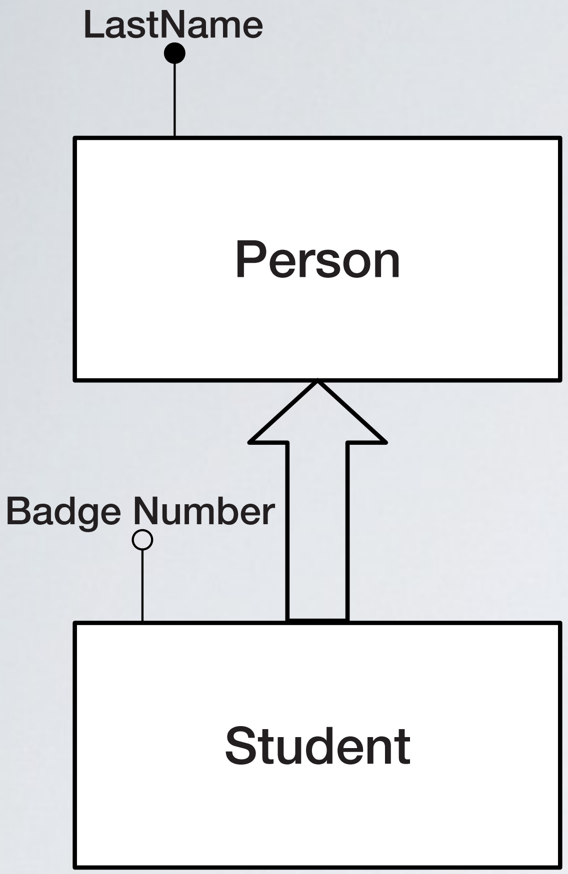
Transformation - Step 4: Removal of IS-A relations and generalizations

Removal of IS-A Relations

- A **E IS-A F** relation between two entities **E** and **F** is transformed into a new binary relationship **ISA-E-F** between **E** and **F** where **E** participates with cardinality (1,1) and **F** with cardinality (0,1)
- We add an external identifier to **E** due to the participation in **ISA-E-F**



Removal of IS-A Relations: Implications for the Extensional Semantics



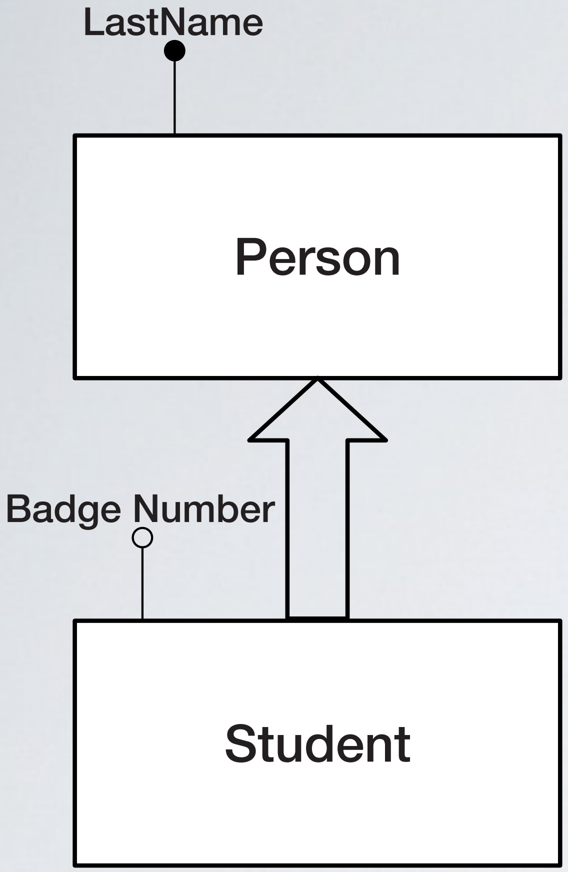
$$\text{instance}(\mathsf{i}, \mathsf{Person}) = \{p_1, p_2, p_3, p_4, p_5\}$$

$$\text{instance}(\mathsf{i}, \mathsf{Student}) = \{p_3, p_4, p_5\}$$

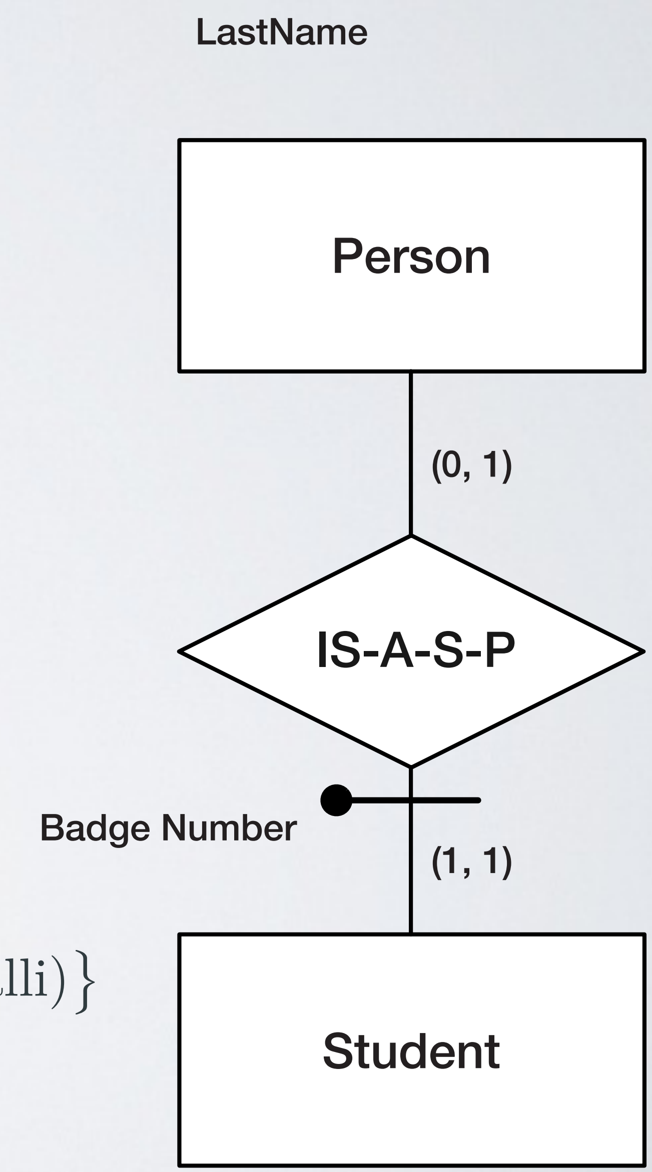
$$\text{instance}(\mathsf{i}, \mathsf{LastName}) = \{(p_1, \text{Rossi}), (p_2, \text{Verdi}), (p_3, \text{Bianchi}), (p_4, \text{Neri}), (p_5, \text{Gialli})\}$$

$$\text{instance}(\mathsf{i}, \mathsf{BadgeNumber}) = \{(p_3, 123456), (p_4, 345678), (p_5, 321654)\}$$

Removal of IS-A Relations: Implications for the Extensional Semantics

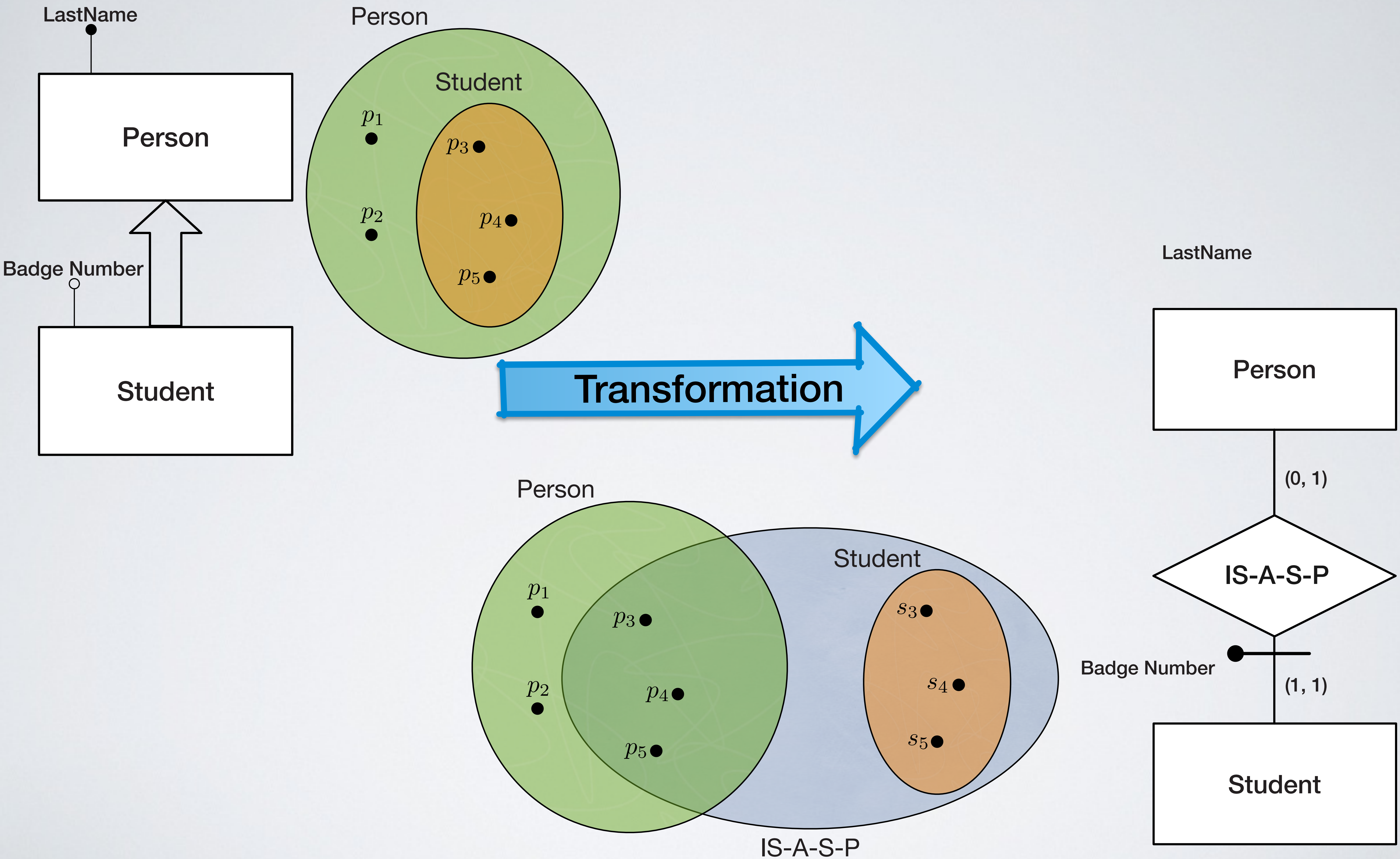


$$\text{instance(i, Person)} = \{p_1, p_2, p_3, p_4, p_5\}$$
$$\text{instance(i, Student)} = \{p_3, p_4, p_5\}$$
$$\text{instance(i, LastName)} = \{(p_1, \text{Rossi}), (p_2, \text{Verdi}), (p_3, \text{Bianchi}), (p_4, \text{Neri}), (p_5, \text{Gialli})\}$$
$$\text{instance(i, BadgeNumber)} = \{(p_3, 123456), (p_4, 345678), (p_5, 321654)\}$$



$$\text{instance(j, Person)} = \{p_1, p_2, p_3, p_4, p_5\}$$
$$\text{instance(j, Student)} = \{s_3, s_4, s_5\}$$
$$\text{instance(j, LastName)} = \{(p_1, \text{Rossi}), (p_2, \text{Verdi}), (p_3, \text{Bianchi}), (p_4, \text{Neri}), (p_5, \text{Gialli})\}$$
$$\text{instance(j, BadgeNumber)} = \{(s_3, 123456), (s_4, 345678), (s_5, 321654)\}$$
$$\text{instance(j, IS-A-S-P)} = \{(s_3, p_3), (s_4, p_4), (s_5, p_5)\}$$

Removal of IS-A Relations: Implications for the Extensional Semantics



Removal of Generalizations: Option 1

- A generalisation among a super-class **p** and the sub-classes **f_1, f_2, …, f_n** is dealt with as n separate **f_1 IS-A p, f_2 IS-A p, …, f_n IS-A p** by introducing n binary relationships **IS-A-f_1-p, IS-A-f_2-p, …, IS-A-f_n-p**
- To account for the generalisation properties, we add external constraints called **generalisation constraints**:
  - **disjointness constraint**: each instance of the superclass can be a member of at most one of the subclasses

$$\text{instance}(\mathsf{i}, \mathsf{f}_i) \cap \text{instance}(\mathsf{i}, \mathsf{f}_j) = \emptyset, \quad 1 \leq i, j \leq n, \; i \neq j$$

    corresponds in the transformed schema to the constraint: each instance of **p** participates **at most at only one IS-A-f_1-p, IS-A-f_2-p, …, IS-A-f_n-p** relationship

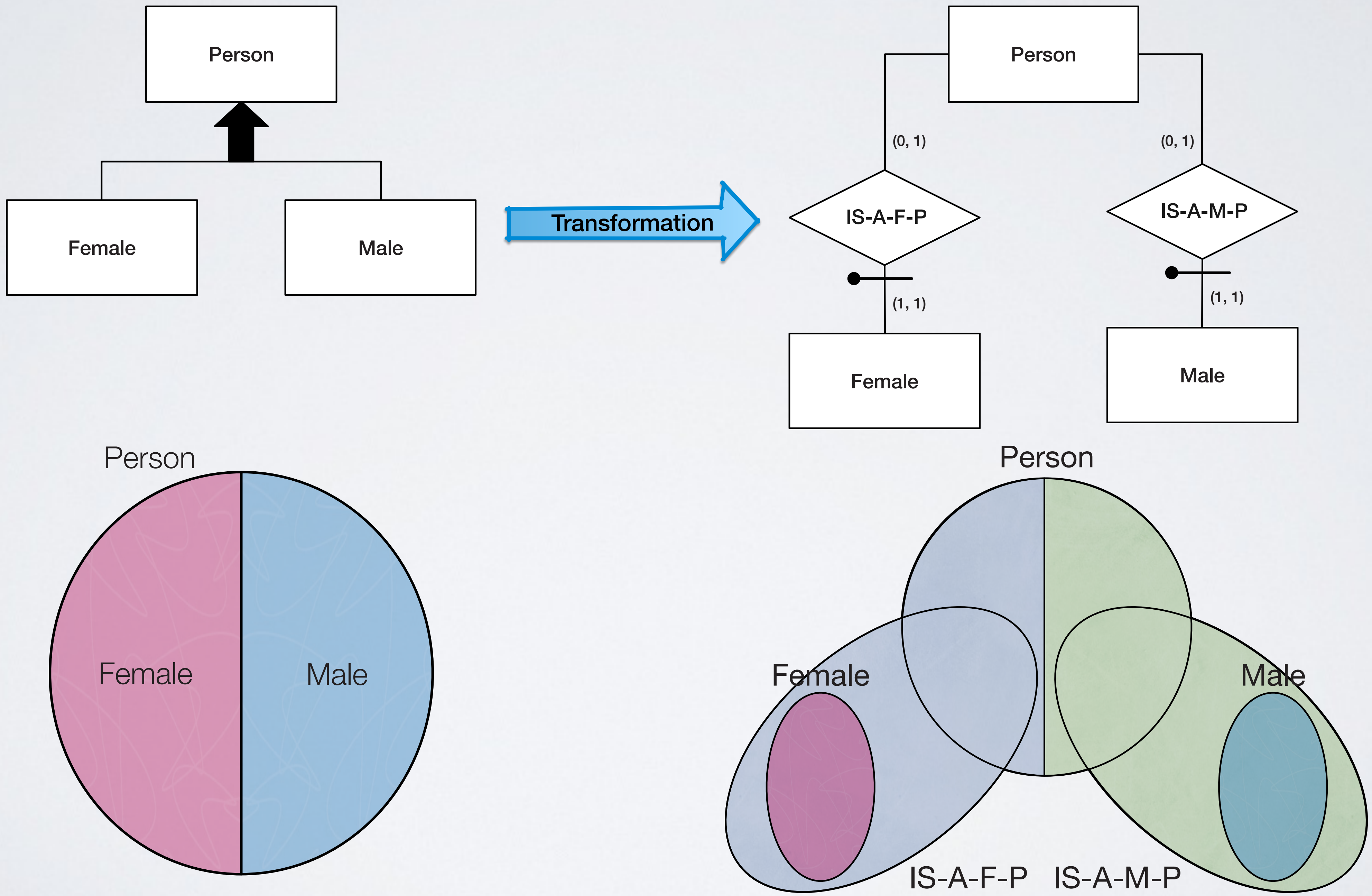
  - **completeness constraint**: each instance of the superclass must be an instance of at least one subclass

$$\text{instance}(\mathsf{i}, \mathsf{f}_1) \cup \text{istanze}(\mathsf{i}, \mathsf{f}_2) \cup \ldots \cup \text{instance}(\mathsf{i}, \mathsf{f}_n) = \text{instance}(\mathsf{i}, \mathsf{p})$$

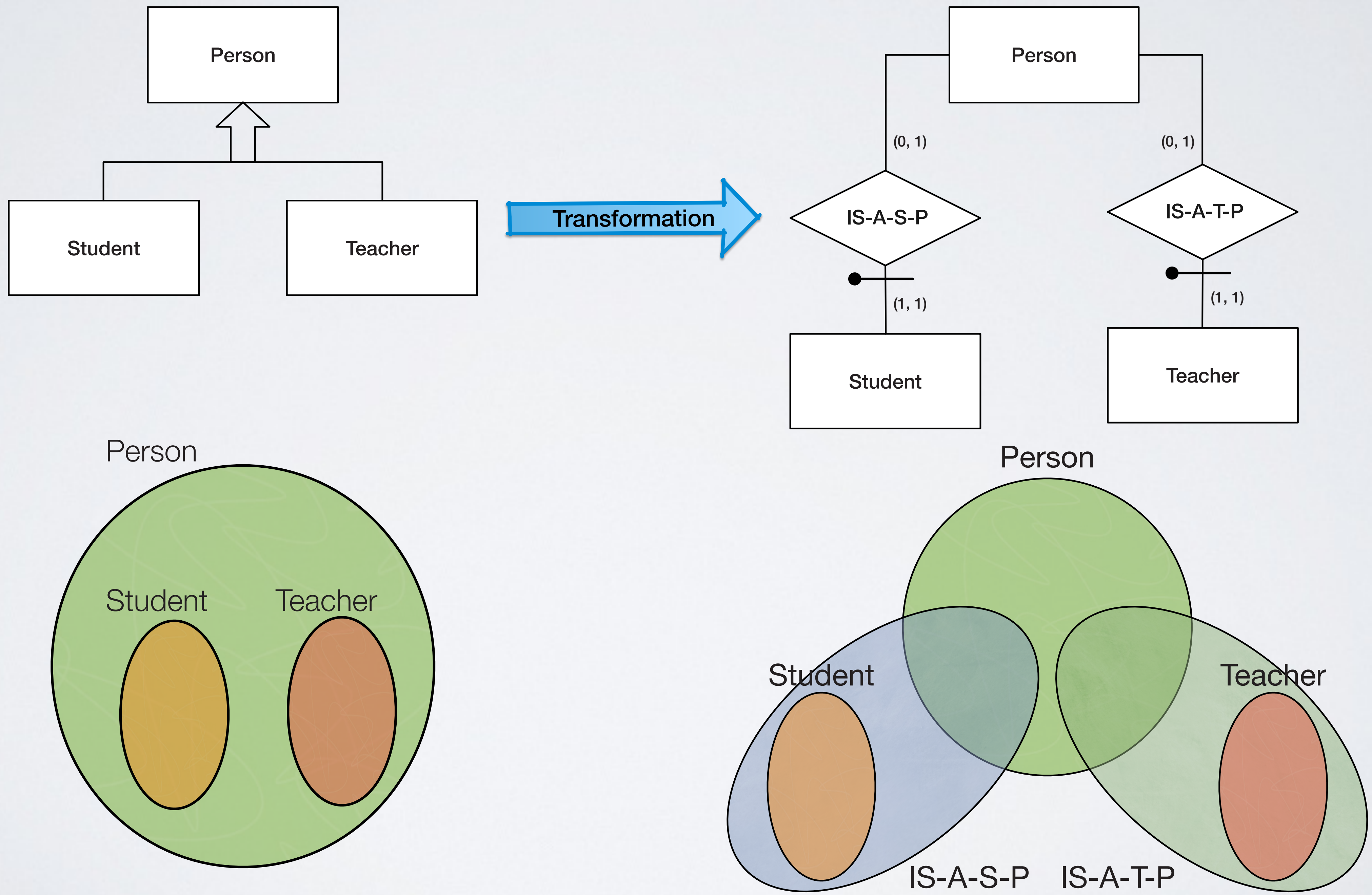
    corresponds in the transformed schema to the constraint: each instance of **p** participates **at least at one IS-A-f_1-p, IS-A-f_2-p, …, IS-A-f_n-p** relationship

Removal of Generalizations: Example of Option 1 (1/4)

- Complete and disjoint generalisation: each instance of **Person** must participate either to **IS-A-F-P** or to **IS-A-M-P** but not to both

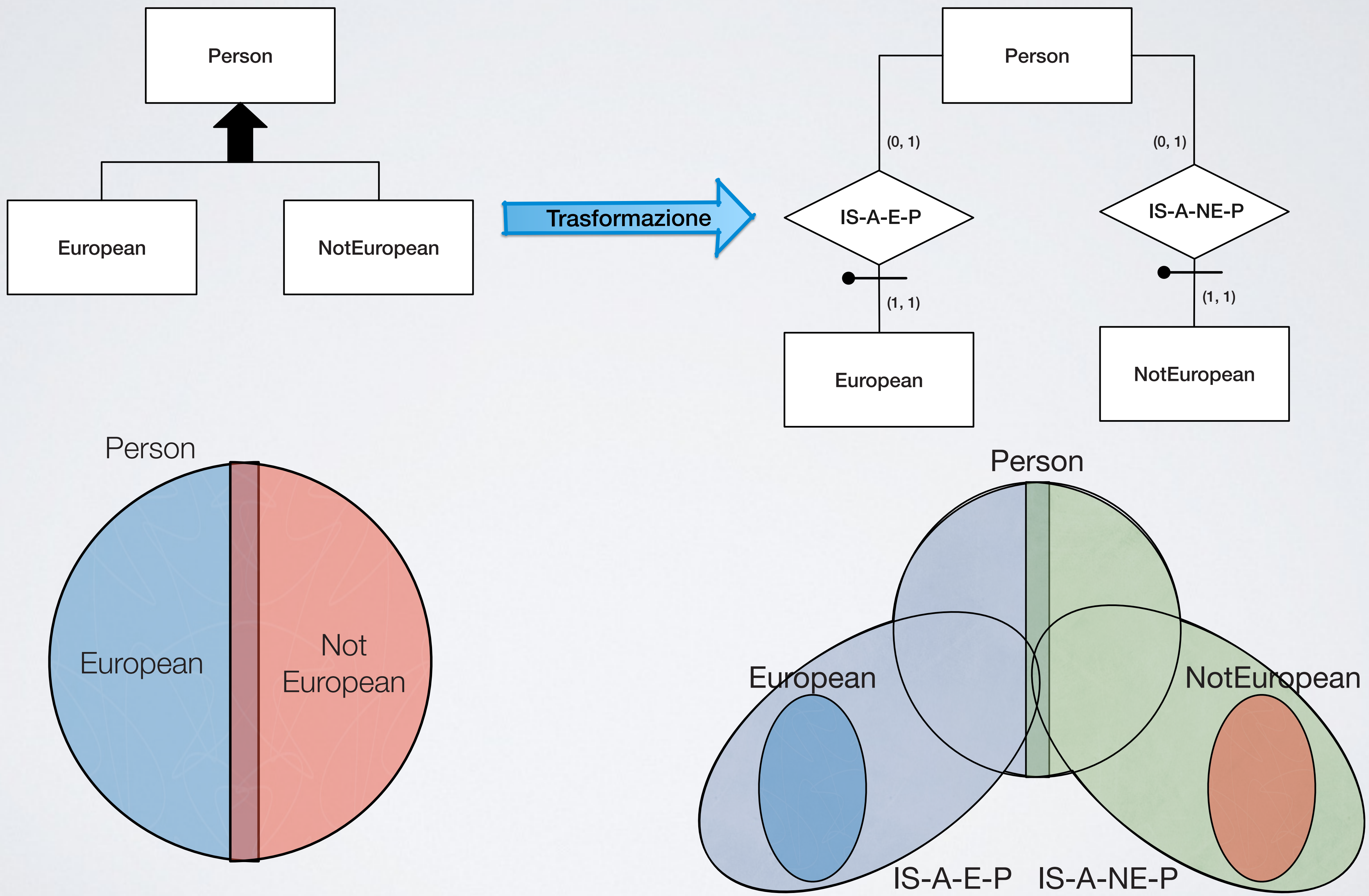


- Not complete and disjoint generalisation: an instance of **Person** may participate either to **IS-A-S-P** or to **IS-A-T-P** but not to both

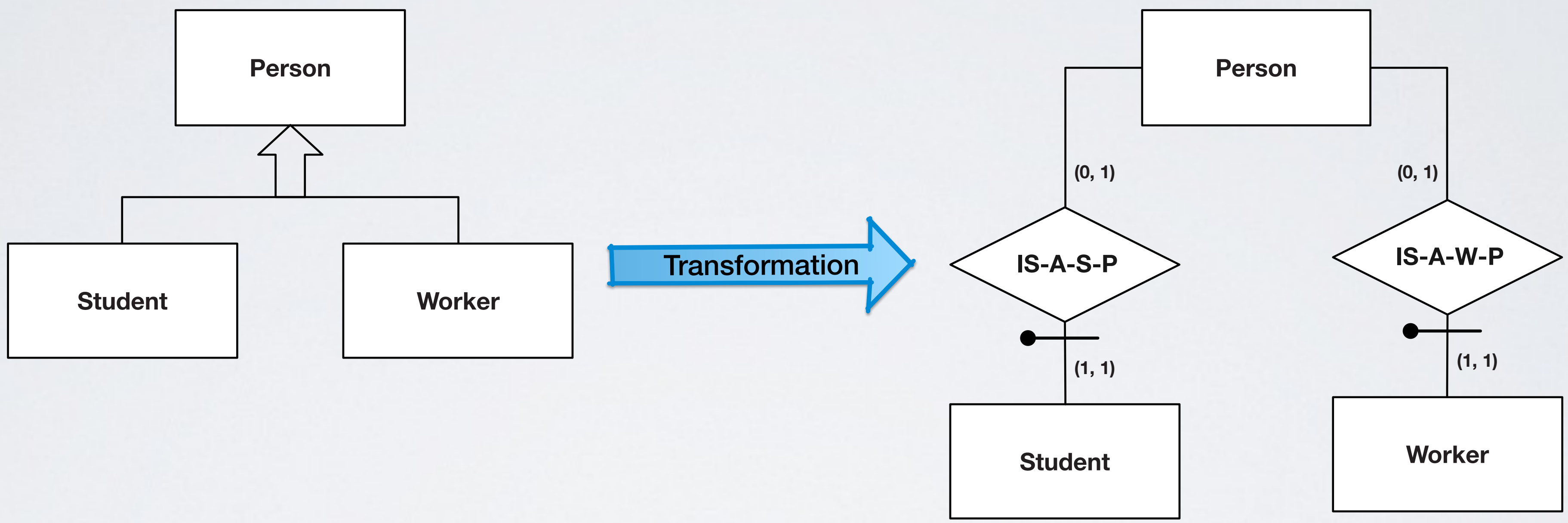


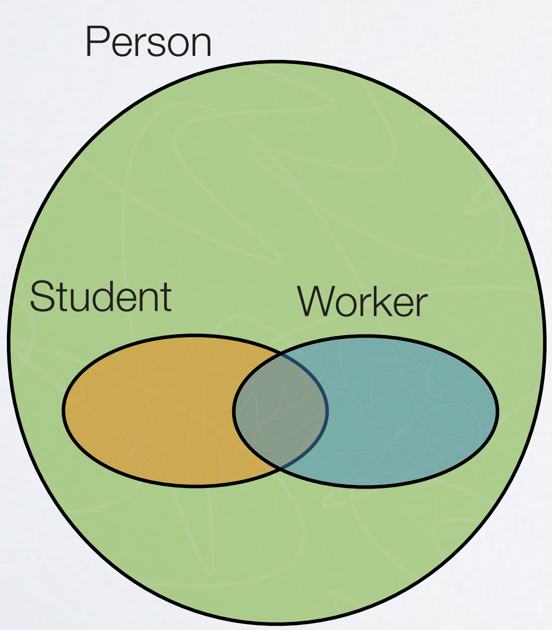
Removal of Generalizations: Example of Option 1 (3/4)

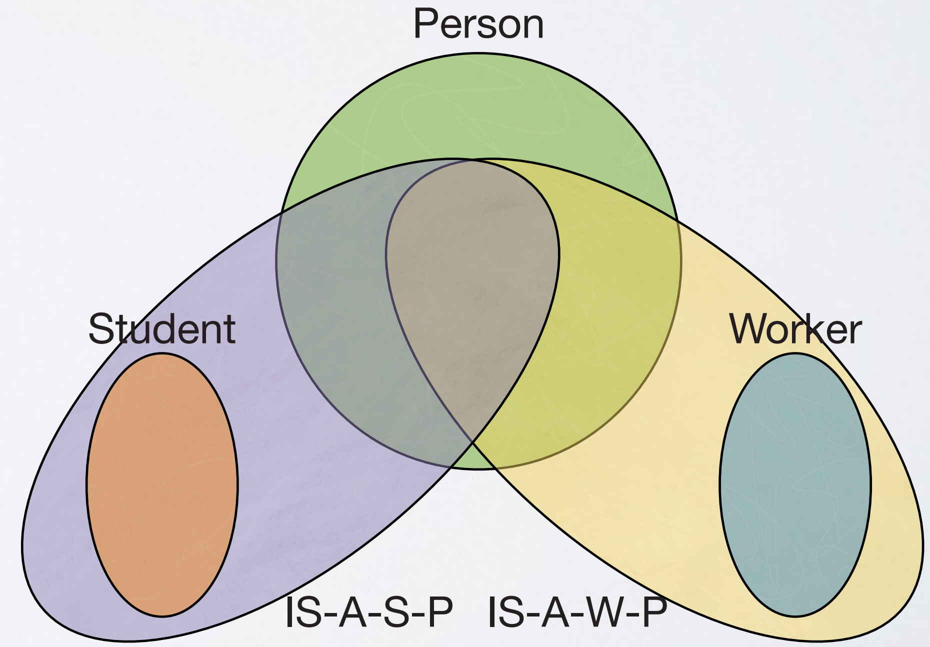
- Complete and not disjoint generalisation: each instance of **Person** must participate either to **IS-A-E-P** or to **IS-A-NE-P** or to both



- Not complete and not disjoint generalisation: each instance of **Person** may participate either to **IS-A-S-P** or to **IS-A-D-P** or to both







Removal of Generalizations: Considerations on Option 1

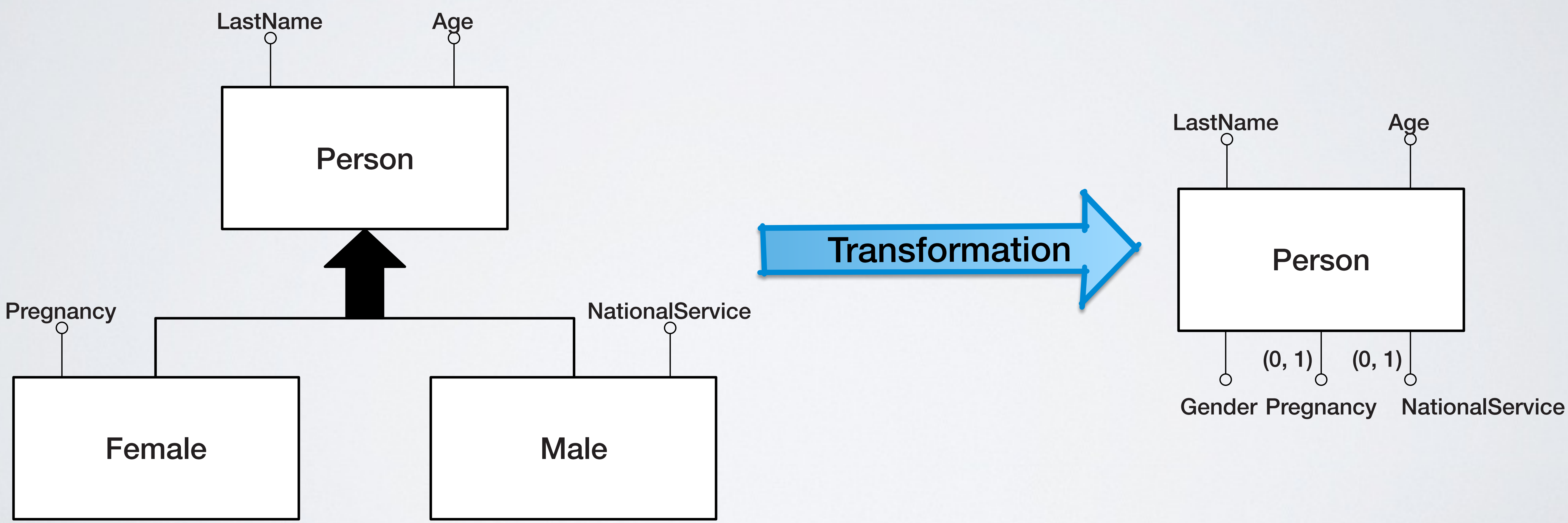
- Pros
  - we can represent all the combinations of completeness and disjointness
  - flexibility if the application requirements change over time
  - efficiency if most of the operations are “local” either to the superclass or to the subclasses with few common to both
- Cons
  - the transformed schema is more complex and there is a proliferation of relationships
  - increased write load: for each new subclass instance, we need to add an instance also in the superclass and in the relationship

Removal of Generalizations: Option 2

- A generalisation among a superclass **p** and the subclasses $\mathbf{f_1}$, $\mathbf{f_2}$, **…**, $\mathbf{f_n}$ is removed by merging all the subclasses into the superclass
- All the attributes from the subclasses are added to the superclass as optional
- We add a **discriminative attribute A** to the superclass to distinguish among the different subclasses
  - the cardinality of A is (1, 1) for complete and disjoint generalisations
  - the minimum cardinality of A is 0 for not complete generalisations
  - the maximum cardinality of A is N for not disjoint generalisations
- The superclass has optional participation to the relationships with the subclasses

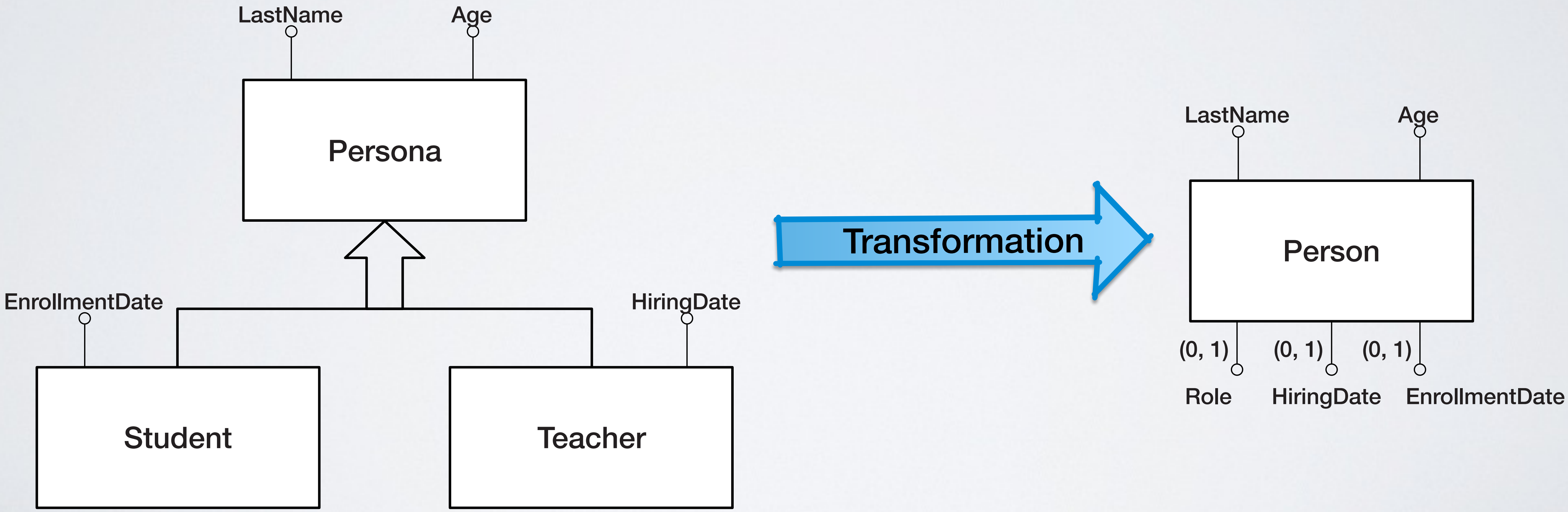
Removal of Generalizations: Example of Option 2 (1/4)

- Complete and disjoint generalisation
- **External constraint**: Pregnancy is used only if Gender = 'F'; NationalService is used only if Gender = 'M'



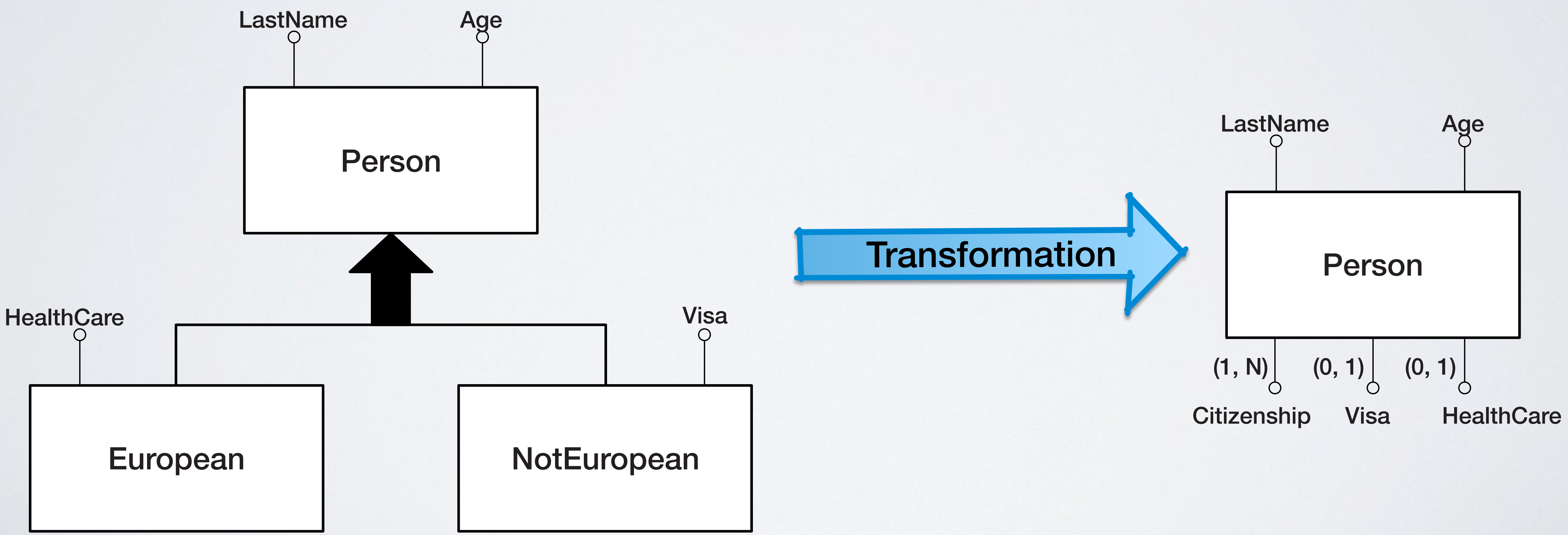
Removal of Generalizations: Example of Option 2 (2/4)

- Not complete and disjoint generalisation
- **External constraint**: Role is used only if an instance of **Person** is either a **Student** or a **Teacher** (but not a generic person); HiringDate is used only if Role = 'Teacher'; EnrollmentDate is used only if Role = 'Student'



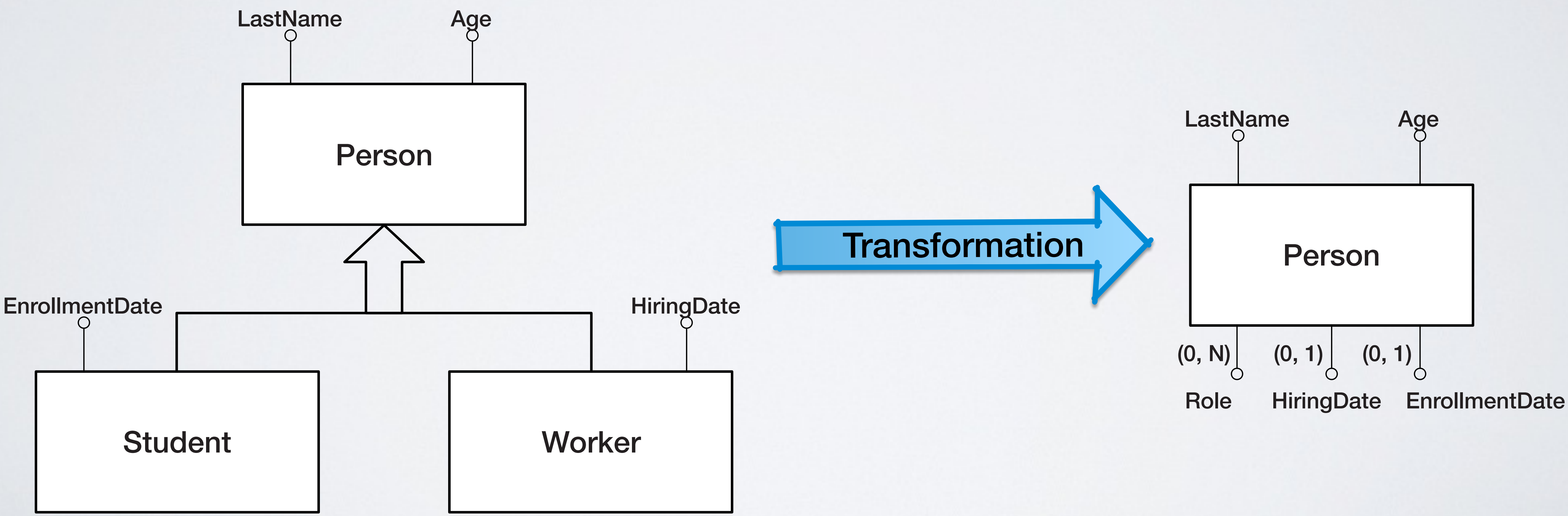
Removal of Generalizations: Example of Option 2 (3/4)

- Complete and not disjoint generalisation
- **External constraint**: HealthCare is used only if Citizenship = ‘European’; Visa is used only if Citizenship = ‘Extra-European’



Removal of Generalizations: Example of Option 2 (4/4)

- Not complete and not disjoint generalisation
- **External constraint**: Role is used only if an instance of **Person** is either a **Student** or a **Teacher** (but not a generic person); note that since the generalisation is not disjoint the maximum cardinality of Role is N; HiringDate is used only if Role = ‘Worker’; EnrollmentDate is used only if Role = ‘Student’



Removal of Generalizations: Considerations on Option 2

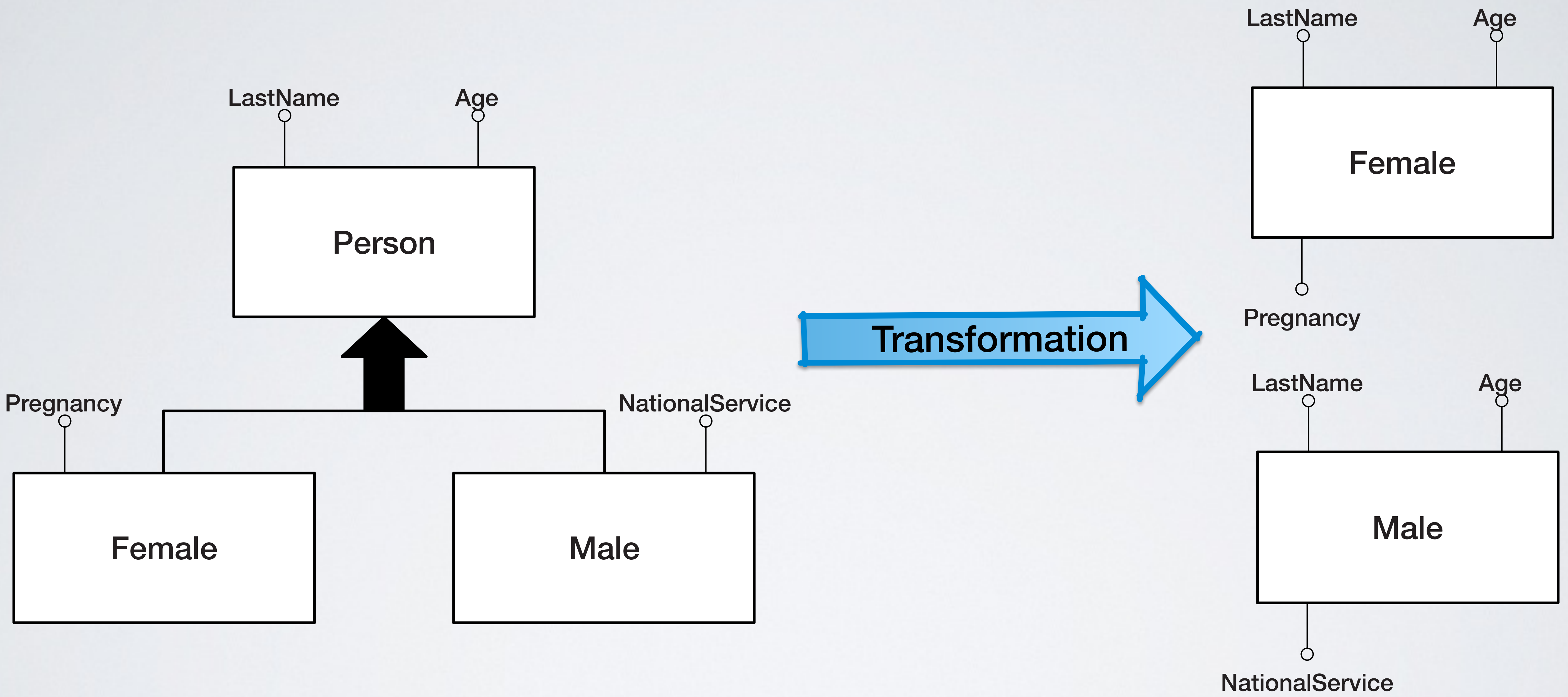
- Pros
  - simplicity
  - we can represent all the combinations of completeness and disjointness
  - flexibility if the application requirements change over time
- Cons
  - presence of systematic NULL values
  - possibile loss of efficiency: the operations that need to access only subclass instances are forced to access all the instances of the superclass

Removal of Generalizations: Option 3

- A generalisation among a superclass **p** and the subclasses $\mathbf{f_1, f_2, \ldots, f_n}$ is removed by merging the superclass into all the subclasses

- The attributes and relationships of the superclass are added to each subclass

Removal of Generalizations: Example of Option 3



Removal of Generalizations: Considerations on Option 3

- Pros
  - ideal if the superclass it is not actually needed by the business logic
  - improved efficiency is most operations are “local” to the subclasses
- Cons
  - it works only for complete and disjoint generalisations
  - duplications of the superclass attributes into the subclasses
  - it may lead to a proliferation of relationships

Removal of Generalisations: Overall Procedure

- We identify two sets of operations
  - **set A**: operations which make use of the attributes of the superclass **p**
    - this set of operations work best with option 1 and 2
  - **set B**: operations which make use of the attributes of the superclass **p** together with those of a subclass $\mathbf{f_i}$ [($\mathbf{p}$, $\mathbf{f_1}$), ($\mathbf{p}$, $\mathbf{f_2}$), ..., ($\mathbf{p}$, $\mathbf{f_n}$)]
    - this set of operations work best with option 3
- If set B is predominant, we choose option 3
  - provided that we have complete and disjoint generalisations
- If set A is predominant
  - we choose option 2 if the attributes of the superclass and the subclass are used together
  - we chose option 1 if the attributes of the superclass and the subclass are used separately

Transformation - Step 5: Choice of the Identifiers

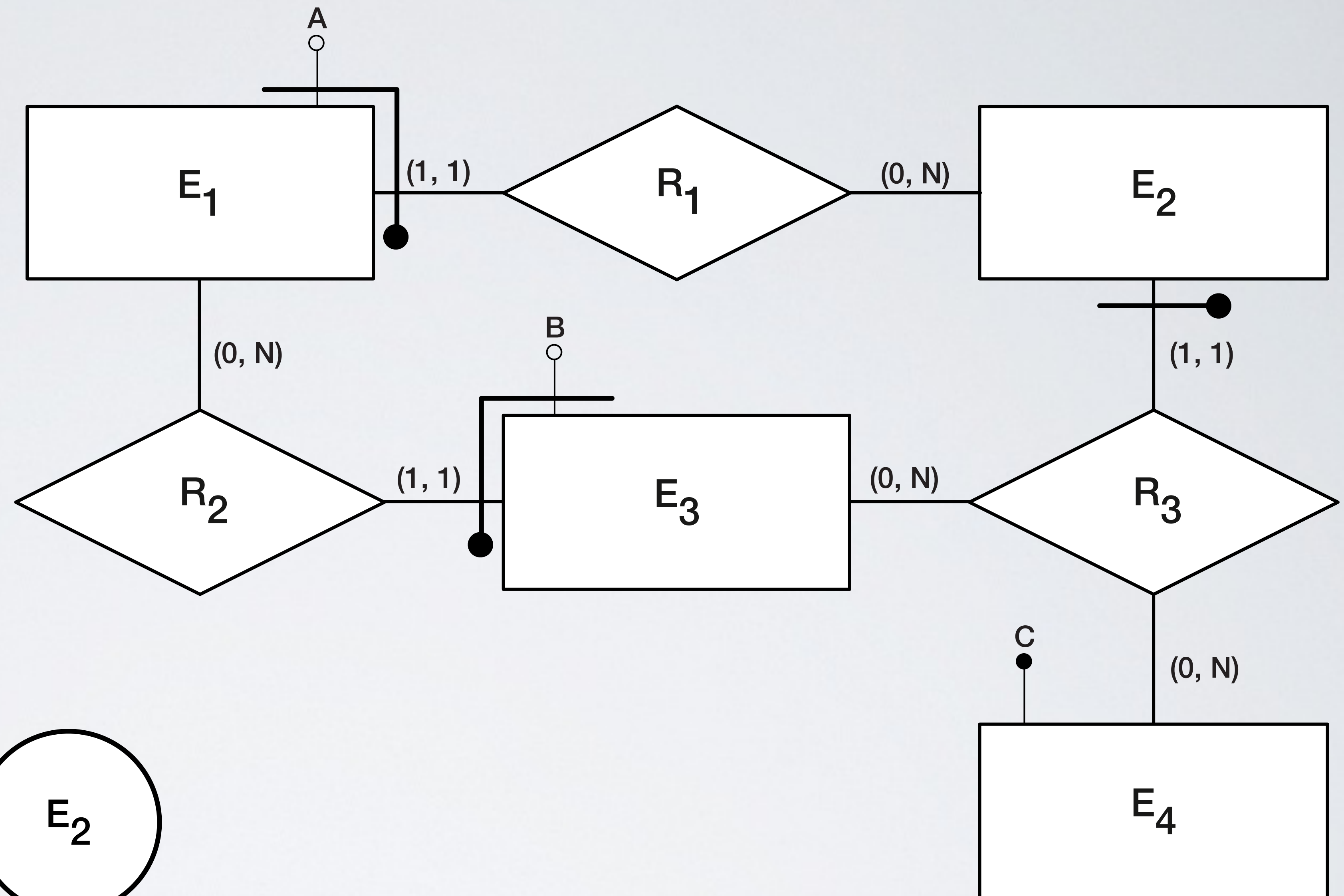
Choice of the Main Identifiers

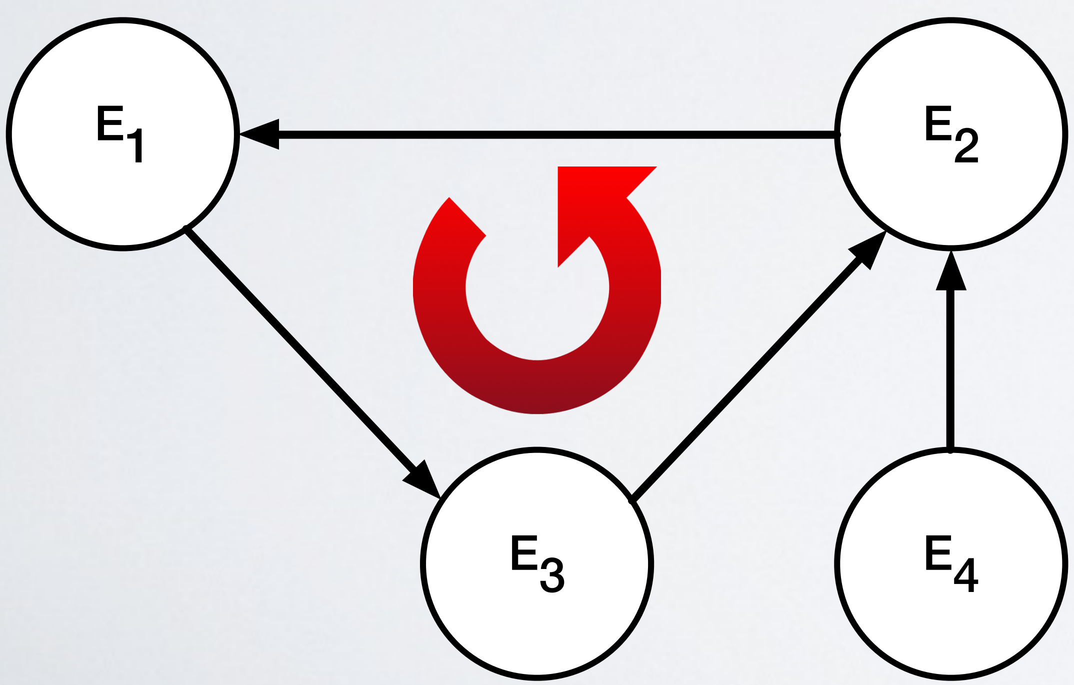
- For each entity, we need to:
  - pick out at least one identifier
  - among the possible identifiers choose a main identifier
- Criteria for choosing a main identifier
  - minimality - the smallest number of attributes possible
  - internal identifiers are to be preferred
  - use in the most important/frequent operations

External Identification Cycles

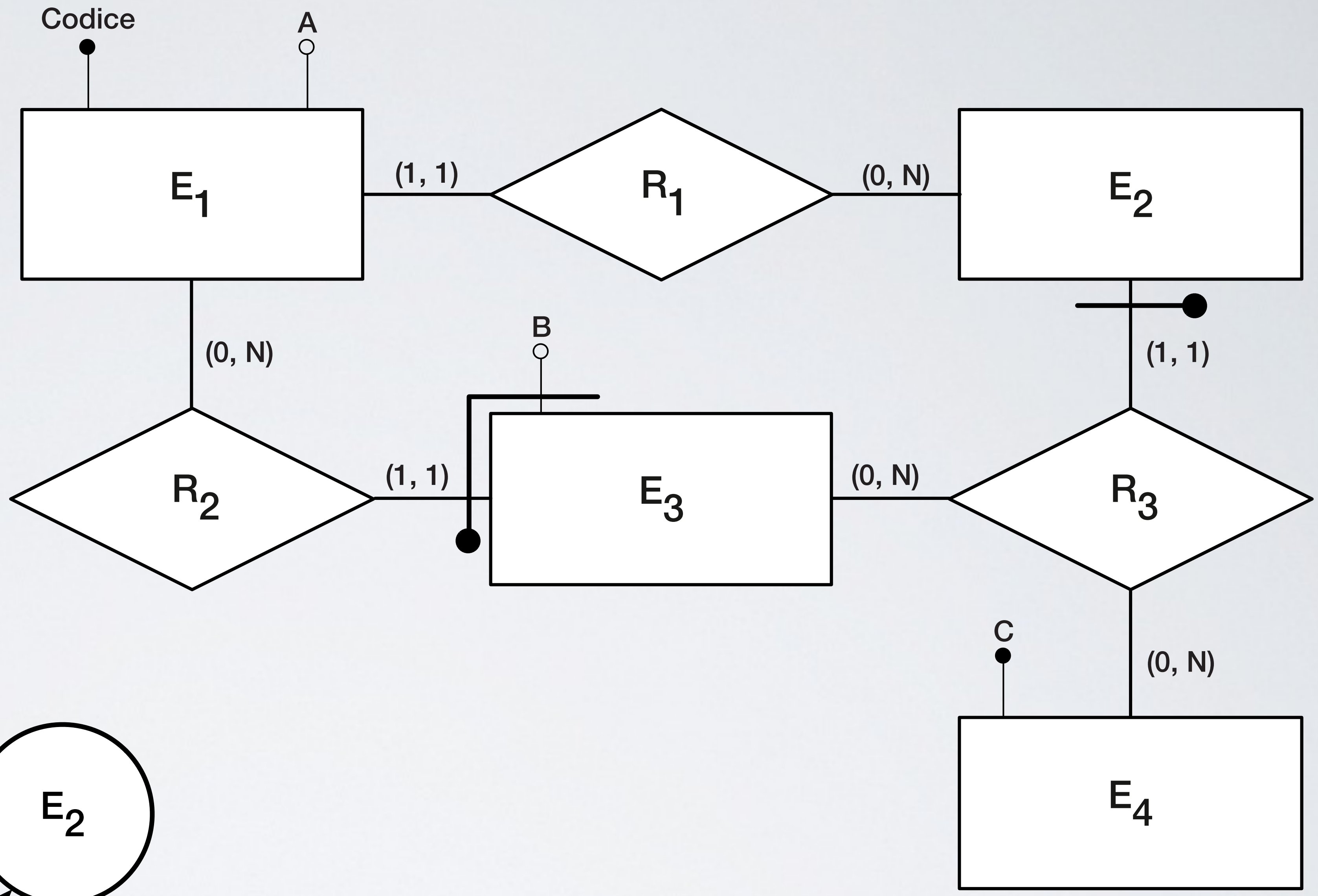
- We create the graph il **graph of the (main) external identifiers** as follows
  - each entity in the ER diagram corresponds to a node in the graph
  - there is an edge between entity E and F if and only if E participates to a relationships which is part of or is the main external identifier of F
- We have an external identification cycle when there is a cycle in this graph
- We need to remove to such external identification cycles by choosing a different identifier or introducing an ad-hoc identifier

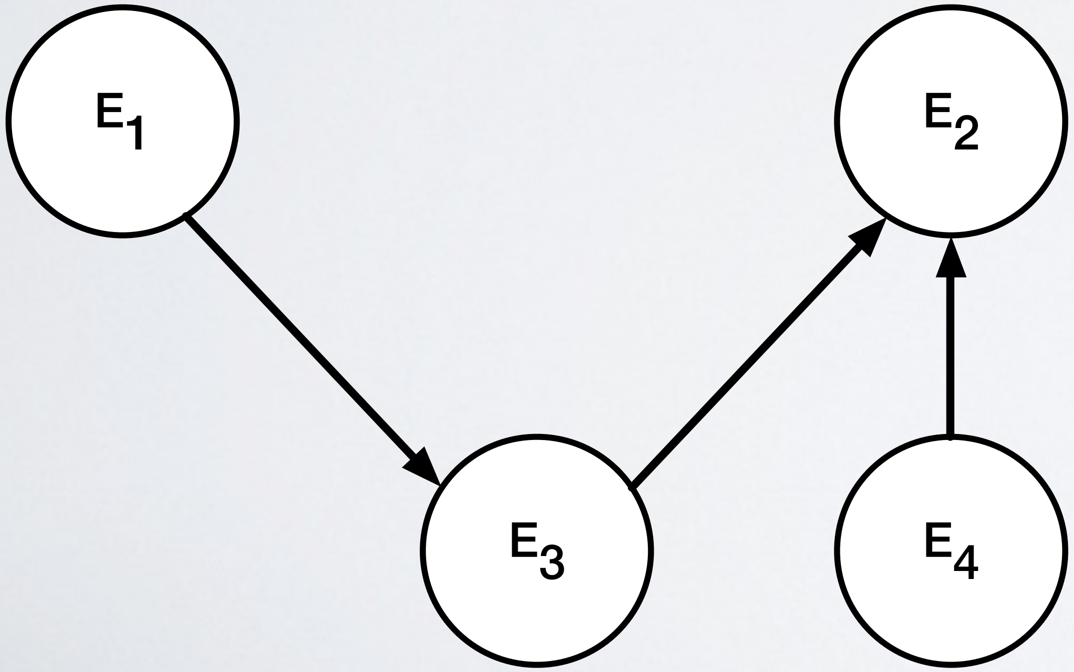
External Identification Cycles: Example





External Identification Cycles: Example





Transformation - Step 6: Definition of additional external constraints

External Constraints

- We need to reformulate all the external constraints on the original schema in terms of the transformed schema

- We need to add the constraints arising from the transformation process
  - constraints for multi-value attributes
  - constraints for optional composite attributes
  - generalisation constraints (disjointness and completeness)
  - constraints for non main identifiers which have been now removed from the schema

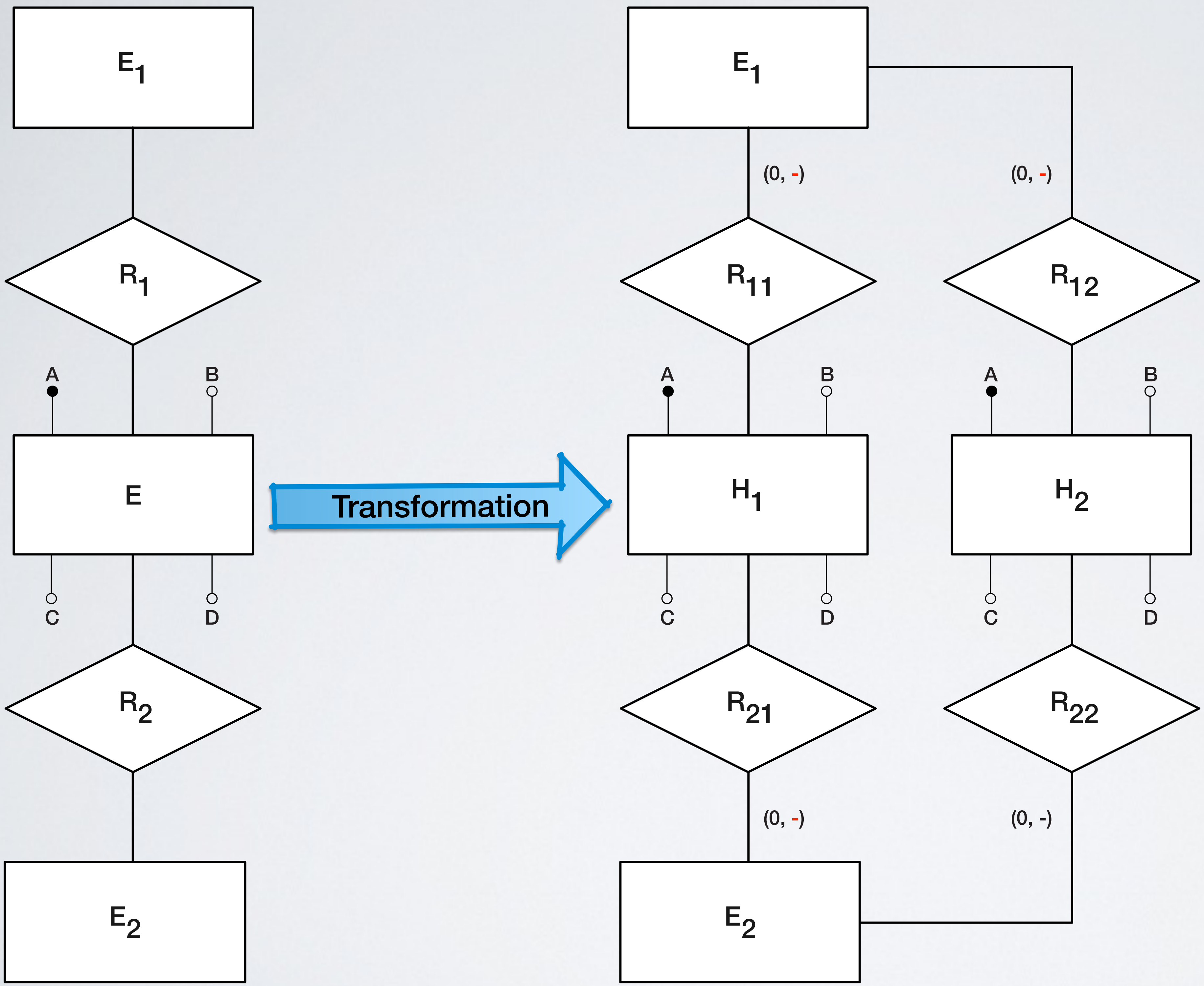
Transformation - Step 7: Partitioning and Merging

Partitioning

Partitioning modifies the distribution of the instances of an entity (horizontal partitioning) or of the attributes of an entity (vertical decomposition) in order to increase performance

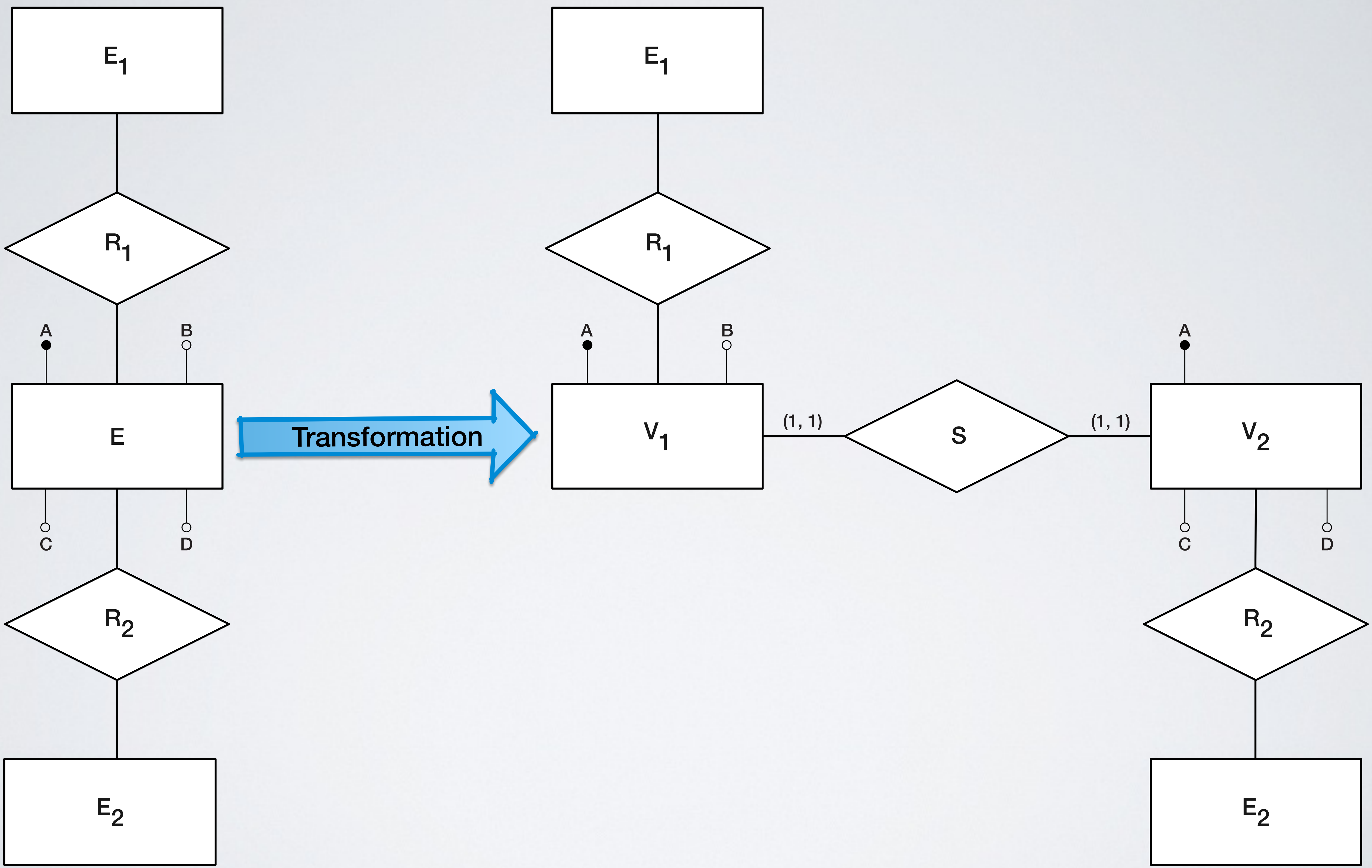
- **Horizontal partitioning**: an entity **E** is split into entities **E_1, E_2, ..., E_n** which have the same attributes of **E** but correspond to different instances of **E** selected on the base of some predicate
- **Vertical partitioning**: an entity **E** is split into entities **E_1, E_2, ..., E_n** which have the same instances of **E** but different groups of attributes of **E**. Each of the **E_1, E_2, ..., E_n** entities has its own identifiers and they are connected by one-to-one relationships

Horizontal Partitioning: Example



- The minimum cardinality of E_1 and E_2 in R_{11}, R_{12}, R_{21} and R_{22} is 0
- The maximum cardinality is
  - 1 if there is no overlap
  - N if there is overlap

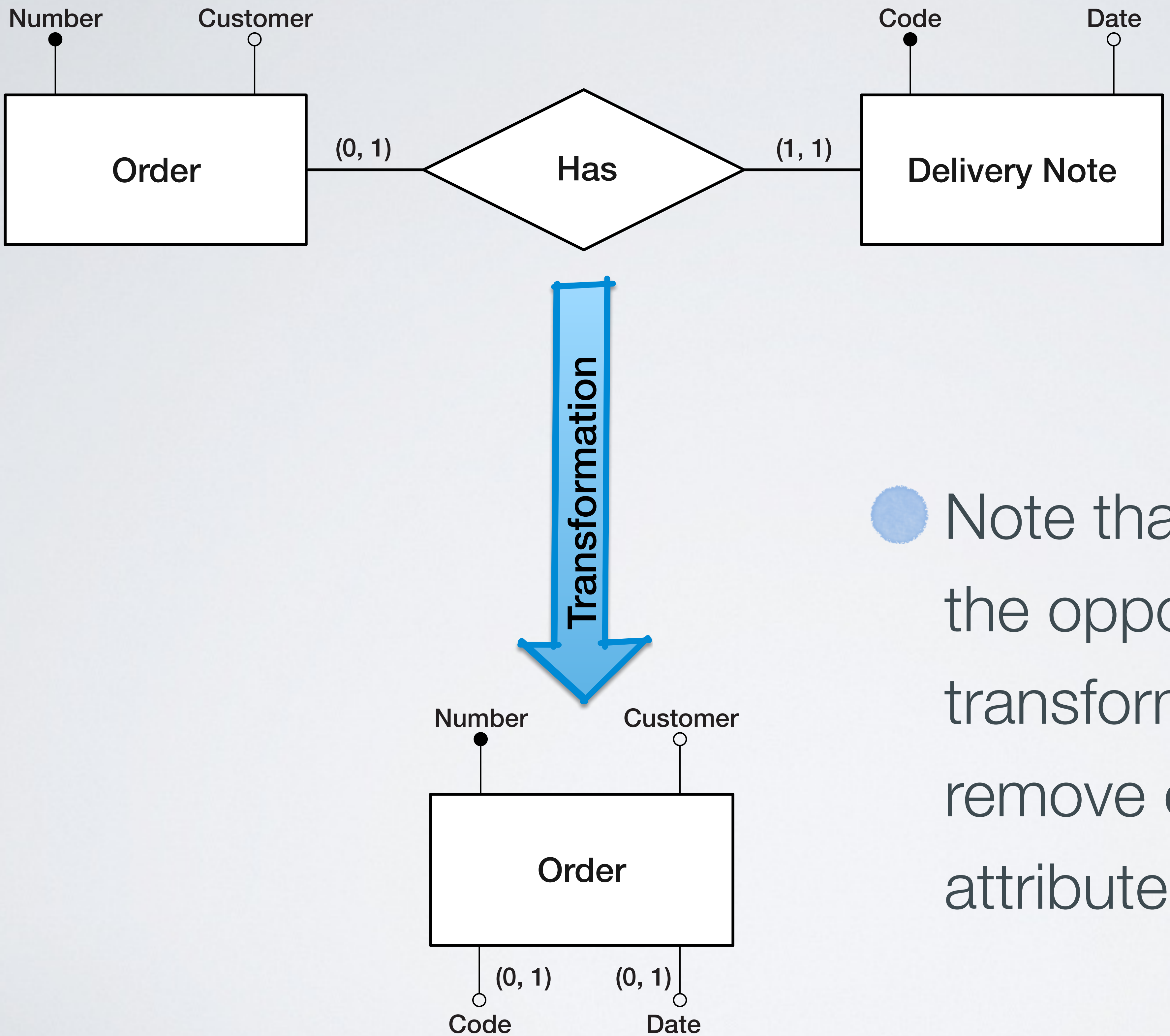
Vertical Partitioning: Example



Merging

Merging groups entities into a single entity which contains all the attributes and relationship of the merged entities with the objective to increase performance

- A side effect of merging is that it may destroy the "de-duplication" in the schema



- Note that this merging is the opposite of the transformation used to remove optional attributes

References

- Batini, C., Ceri, S., and Navathe, S. B. (1992). *Conceptual Database Design. An Entity-Relationship Approach*. The Benjamin/Cummings Publishing Company, Inc., Redwood City (CA), USA.
- Chen, P. P. (2002). Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned. In Broy, M. and Denert, E., editors, *Software Pioneers: Contributions to Software Engineering*, pages 296–310. Springer-Verlag, New York, USA.
- Teorey, T. J. and Fry, J. (1980). The Logical Record Access Approach to Database Design. *ACM Computing Surveys (CSUR)*, 12(2):179–211.
- Teorey, T. J., Yang, D., and Fry, J. (1986). A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model. *ACM Computing Surveys (CSUR*), 18(2):197–222.


Questions?


OUR DIFFERENTIATING VALUE-ADDED STRATEGY IS TRANSFORMATIONAL CHANGE.
www.dilbert.com
scottadams@aol.com
HOW WAS THAT? DOES ANYONE FEEL DIFFERENT?
© 2005 Scott Adams, Inc./Dist. by UFS, Inc.
MY URGE TO HURL HAS INCREASED A LITTLE BIT.
THAT'S WHAT CHANGE FEELS LIKE.