Database Logical Design

Alessandro Bozzon
TI1506: Web and Database Technology
ti1506-ewi@tudelft.nl

TU Delft

Course overview [DB]

1. Introduction to Database Systems
2. The Relational Model
3. Introduction to SQL
4. Advanced SQL Topics
5. Introduction to NoSQL database systems
6. **Conceptual Data Modelling with ER Diagrams**
7. **Database Conceptual Design**
8. **Database Logical Design**

At the end of this lecture, you should be able to …

- **Develop** and **refine** conceptual data models, using appropriate design notation

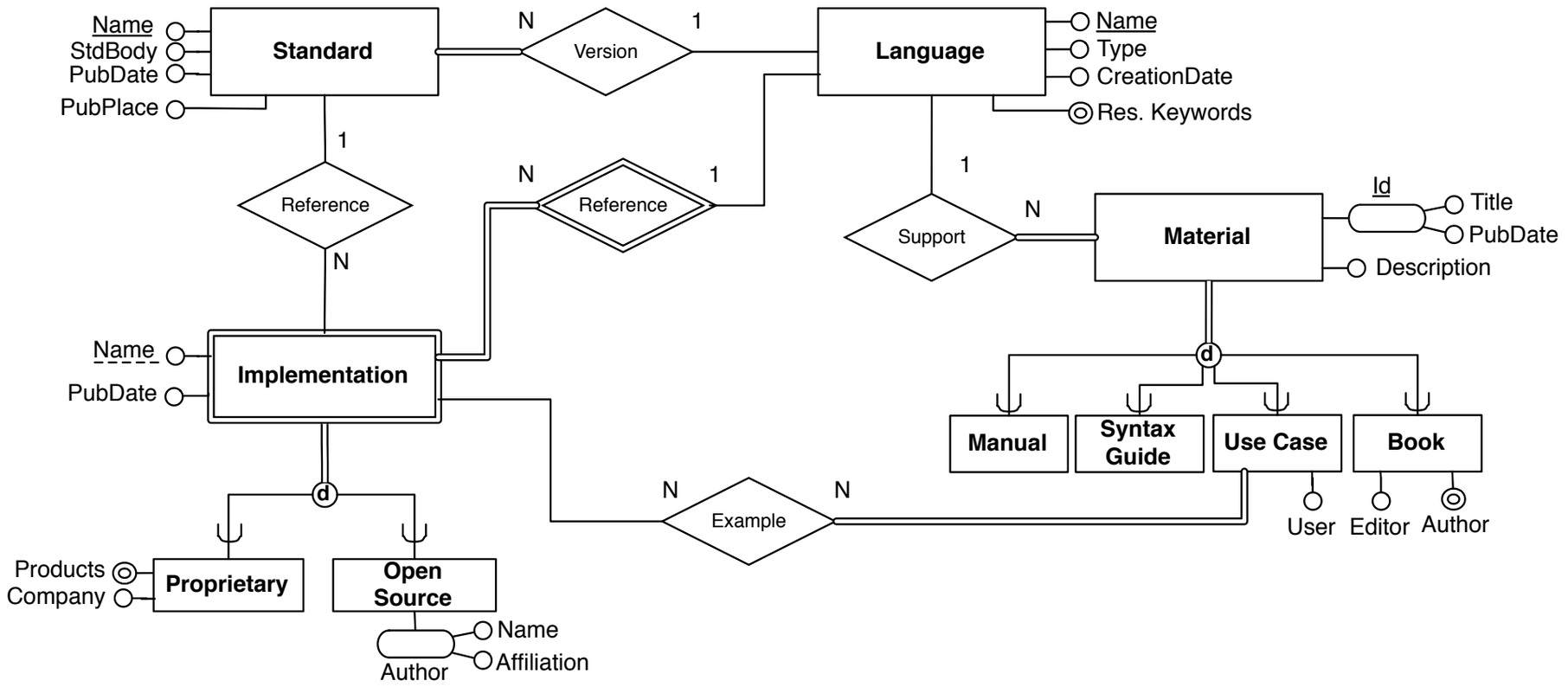
- **Derive** a logical design from a conceptual design, taking into account the requirements of both the adopted database technology, and the designed application

- **Develop** logical database schema, with principled design that enforce data integrity

ASQ Exercise (1 min)

Consider the EER diagram in figure.

How many internal identifiers are defined?



1 2 3 4 5

EER Model

The Enhanced Entity-Relationship (EER) Model

- Created to design more accurate database schemas Reflect the data properties and constraints more precisely

- More complex requirements than traditional applications

- EER model includes all modelling concepts of the ER model.

- In addition, EER includes:
  - Subclasses and superclasses
  - Specialisation and generalisation
  - Category or union type (not covered)
  - Attribute and relationship inheritance (not covered)

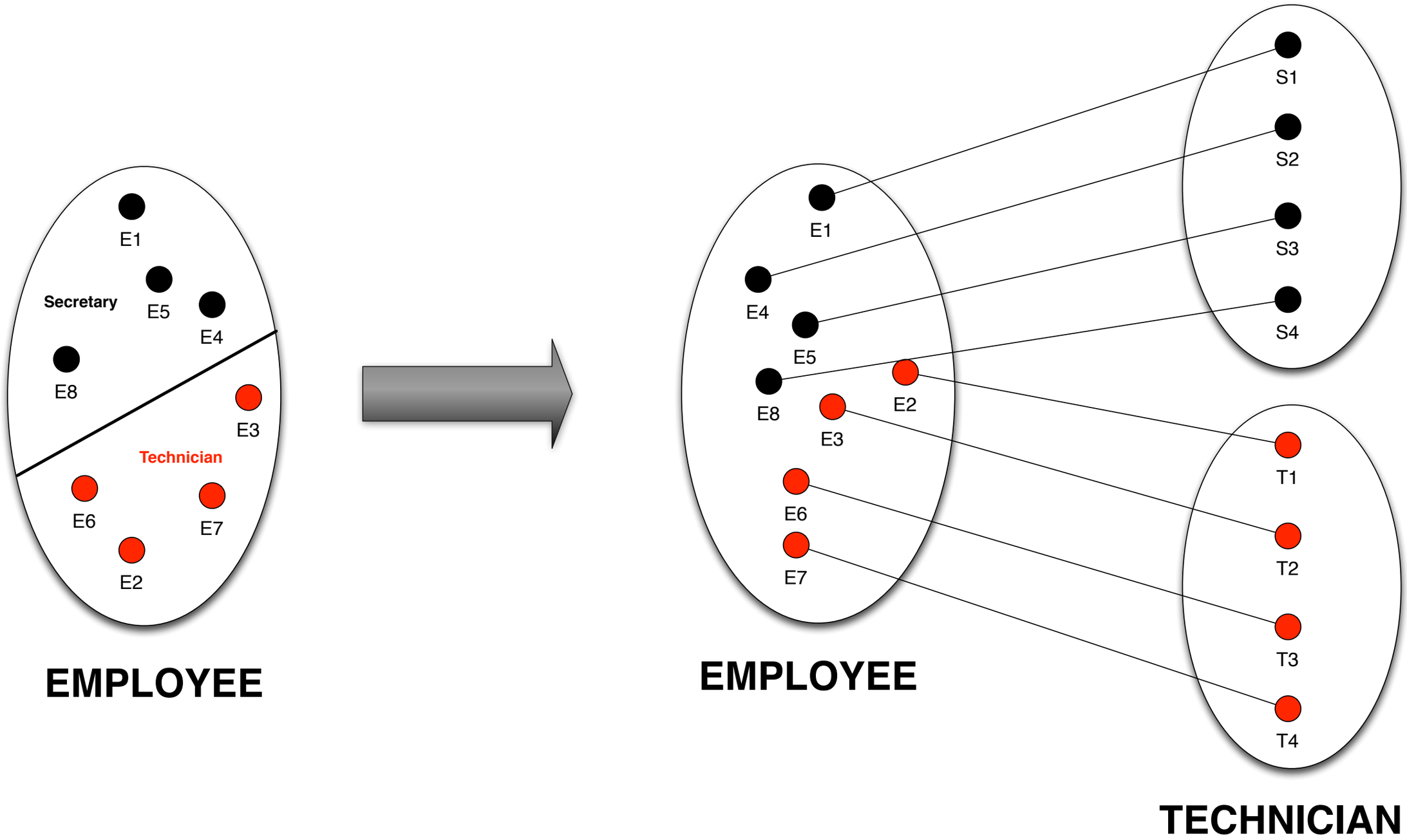
Class / Subclass (Type / Subtype)

- An entity type is used to represent a type of instance (attribute and relationships)
  - **`EMPLOYEE`** (name, BSN, BirthDate, etc)

- But also the *collection of entities (entity set) of that type* that exist in the database
  - E.g. entities that represent specific types of employees
  - `SECRETARY`, `TECHNICIANs`, `MANAGERSs`, etc.

- The relationship that exists between a type of entity and its entity set is called a **class/ subclass** relationship
  - `EMPLOYEE` is the **super type** or **super class**
  - `SECRETARY`, `TECHNICIAN` etc. are **subtypes** or **subclasses** of `EMPLOYEE`

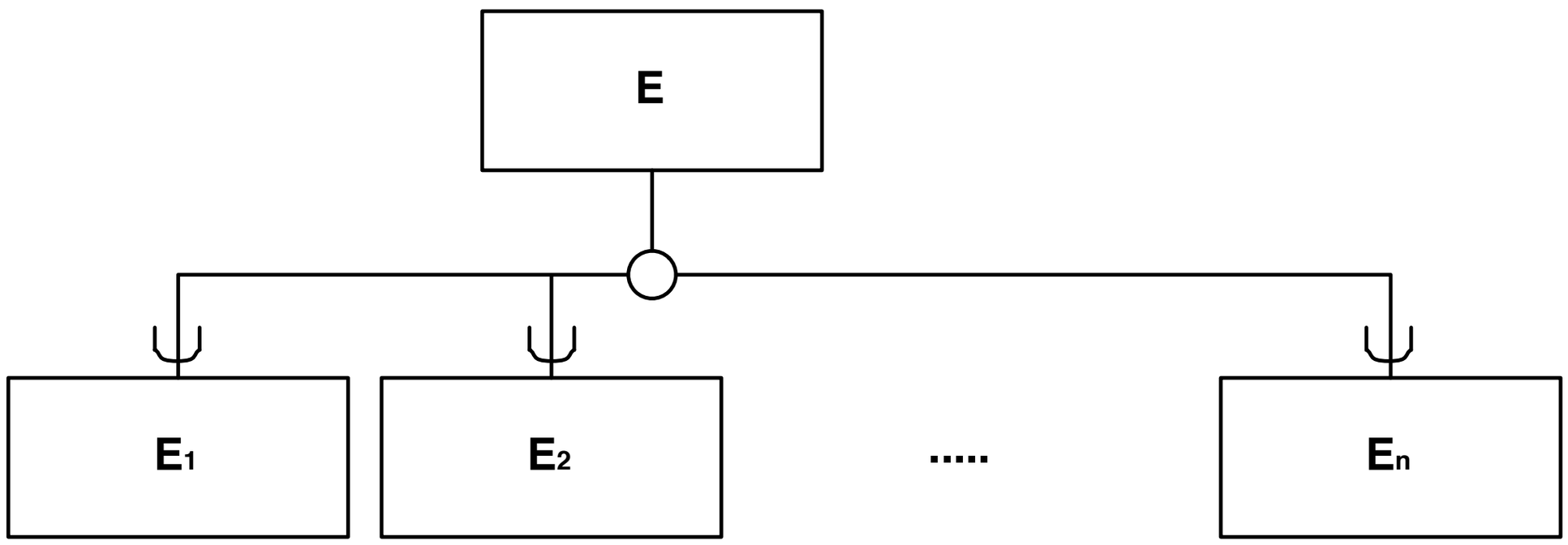
Example Class-Subclass /1



NOT a 1:1 relationship type, but a relationship between OCCURRENCES

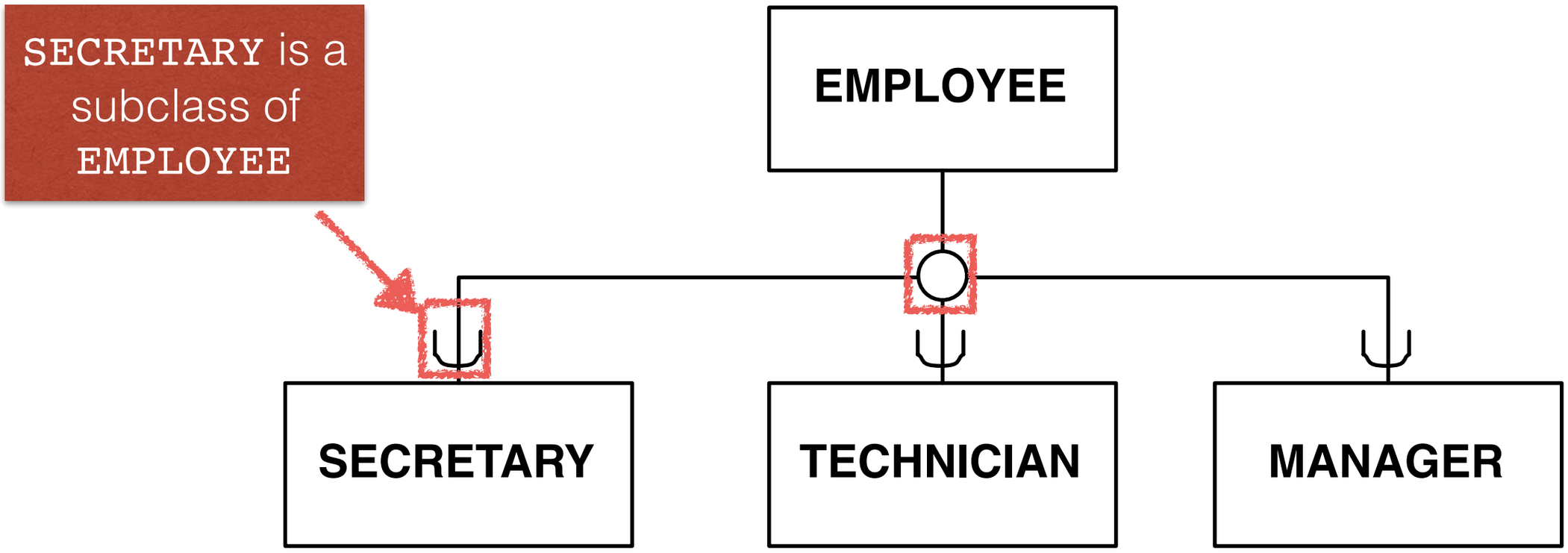
Class/ Subclass relationship

- Also called **generalisation** or **specialisation** relationship
- Represent logical links between an entity $\mathtt{E}$, known as parent entity, and one or more entities $\mathtt{E}_1, \ldots, \mathtt{E}_n$ called child entities, of which $\mathtt{E}$ is more general, in the sense that it comprises them as a particular case.
- In this situation we say that $\mathtt{E}$ is a **generalisation** of $\mathtt{E}_1, \ldots, \mathtt{E}_n$ and that the entities $\mathtt{E}_1, \ldots, \mathtt{E}_n$ are specialisation of the $\mathtt{E}$ entity



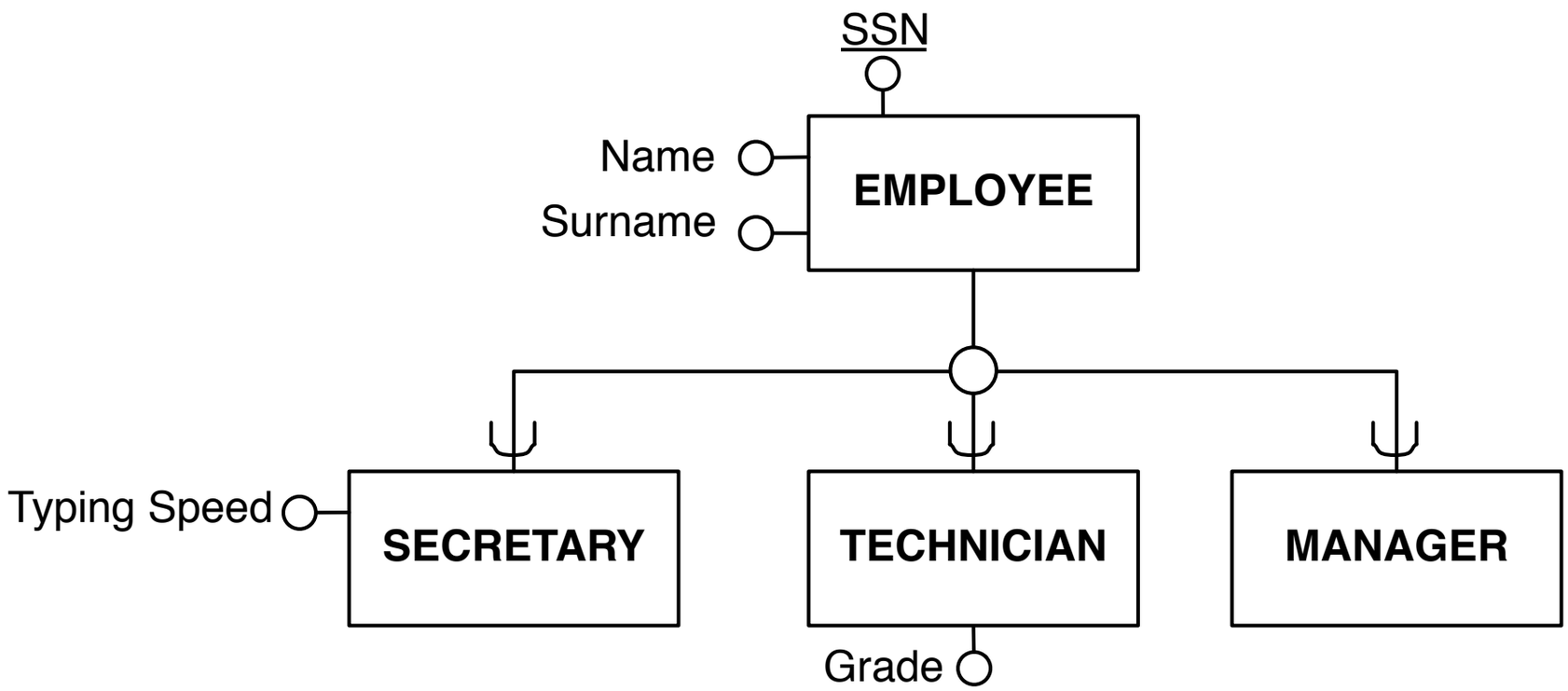
Example Class-Subclass /2

- The subset symbol indicates the *direction* of the **specialisation**
- The circle defines the entities involved in the specialisation



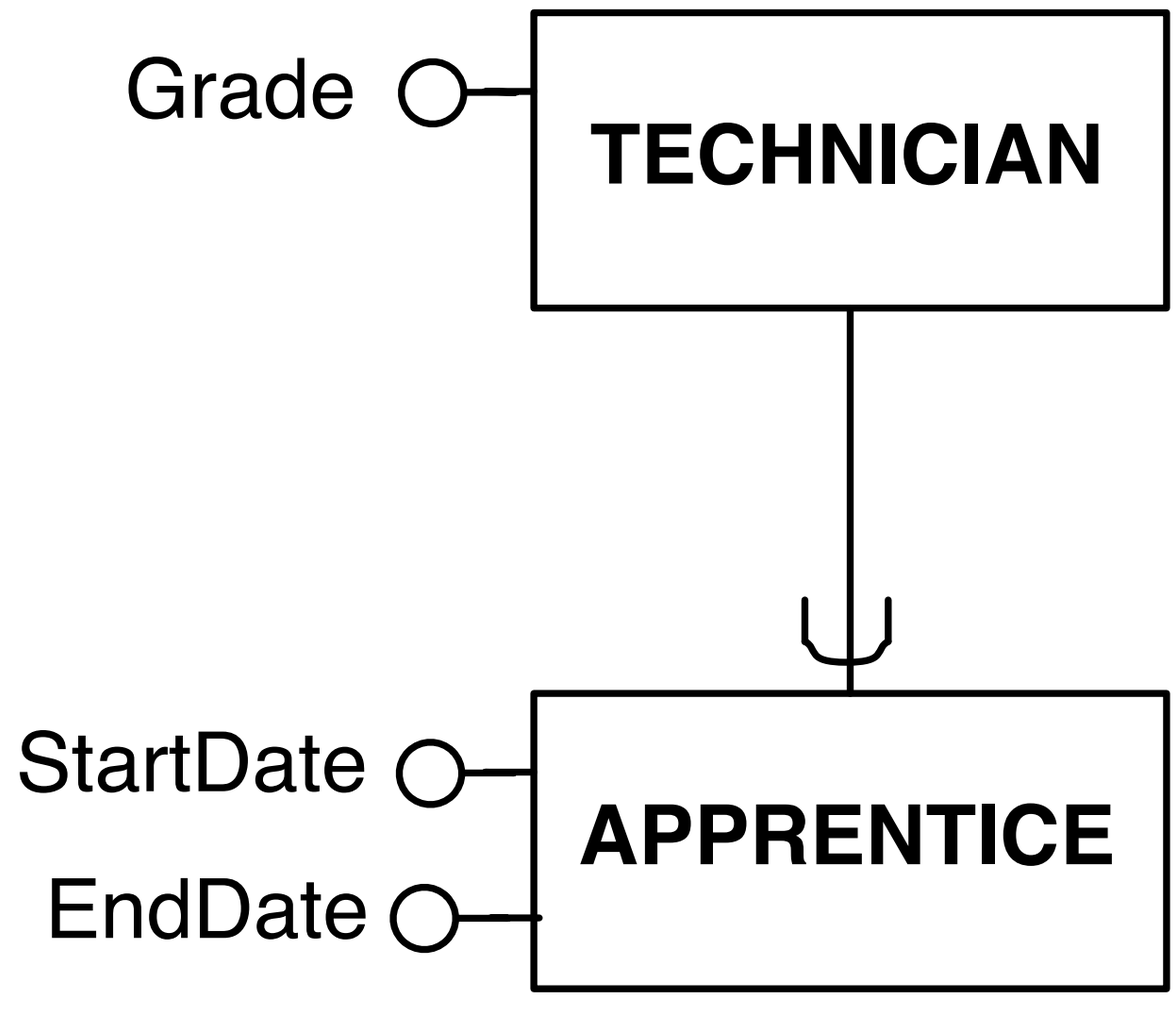
Properties of Specialisation

- Every occurrence of a child entity is also an occurrence of the parent entity
- **Inheritance**
  - Every property of the parent entity (attributes, identifiers, relationships and other generalisations) is also a property of a child entity.
- Specialisation can define **specific *attributes*** and/or specific ***relationship types***



Subset

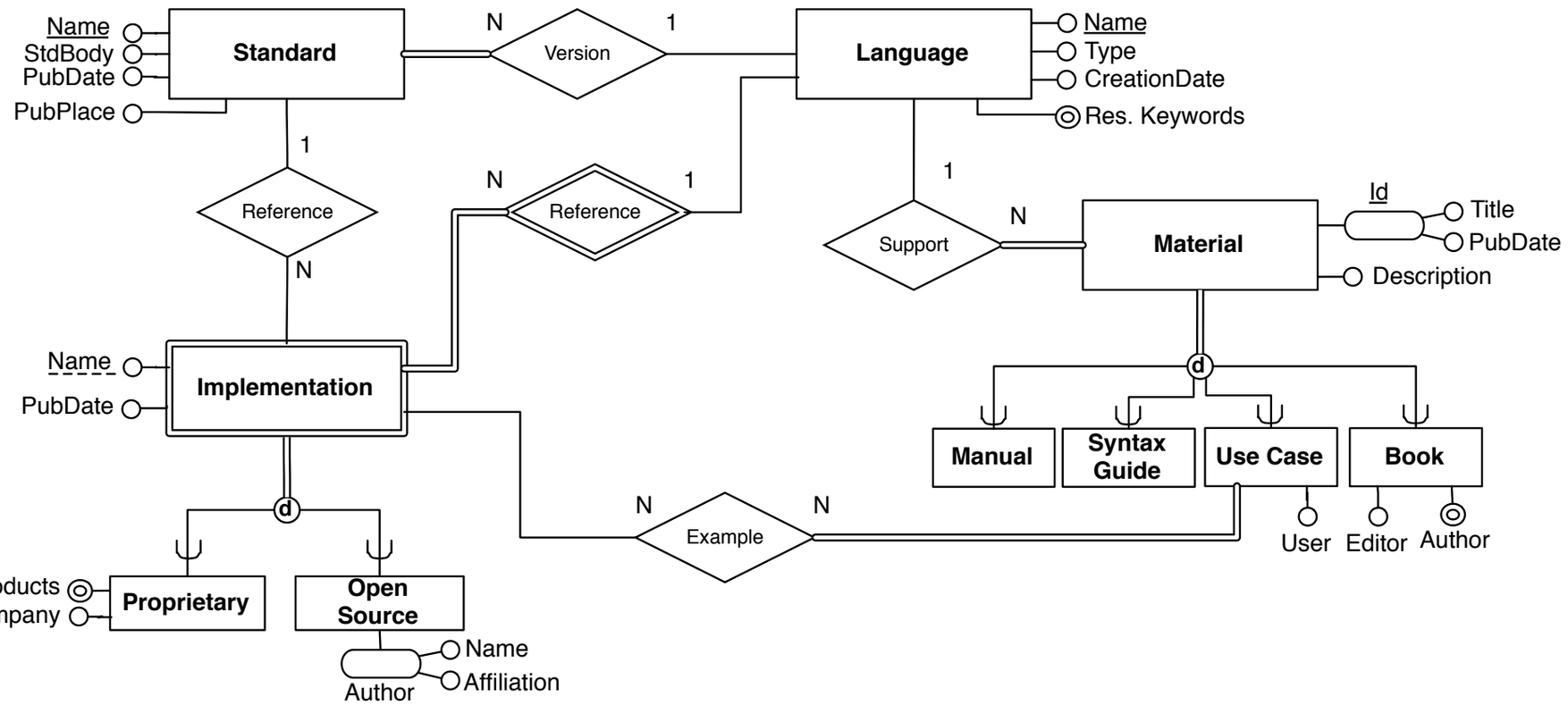
- Special class of specialisation with a single child entity



ASQ Exercise (1 min)

Consider the EER diagram in figure.

Which of the following statements is correct?



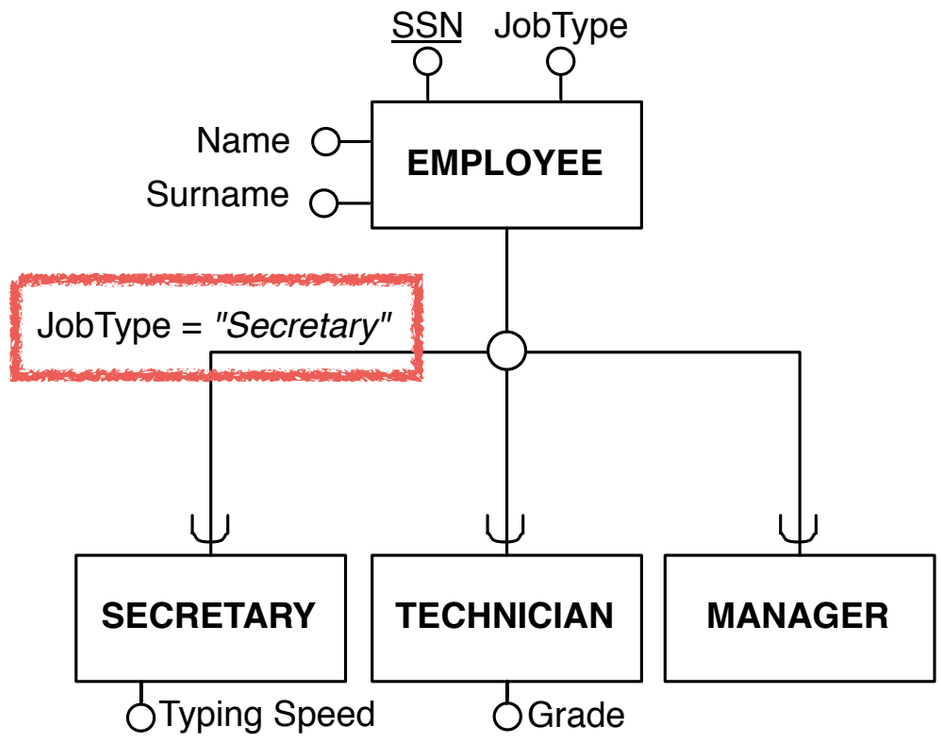
1 A *Use Case* is an example of *Implementation*

2 A *Use Case* is a type of *Material*

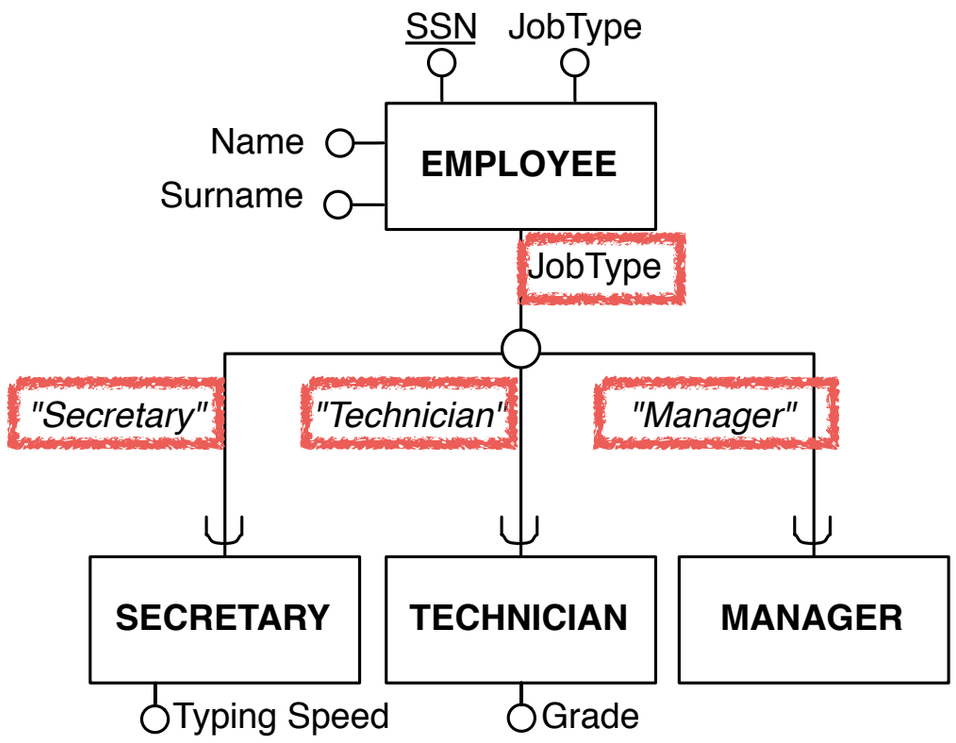
3 An *Implementation* generalises into *Proprietary Implementation*

Constraints: Membership

- In some specialisation we can determine exactly the entities that will become members of each subclass
  - *Predicate-defined* (or condition-defined) subclasses



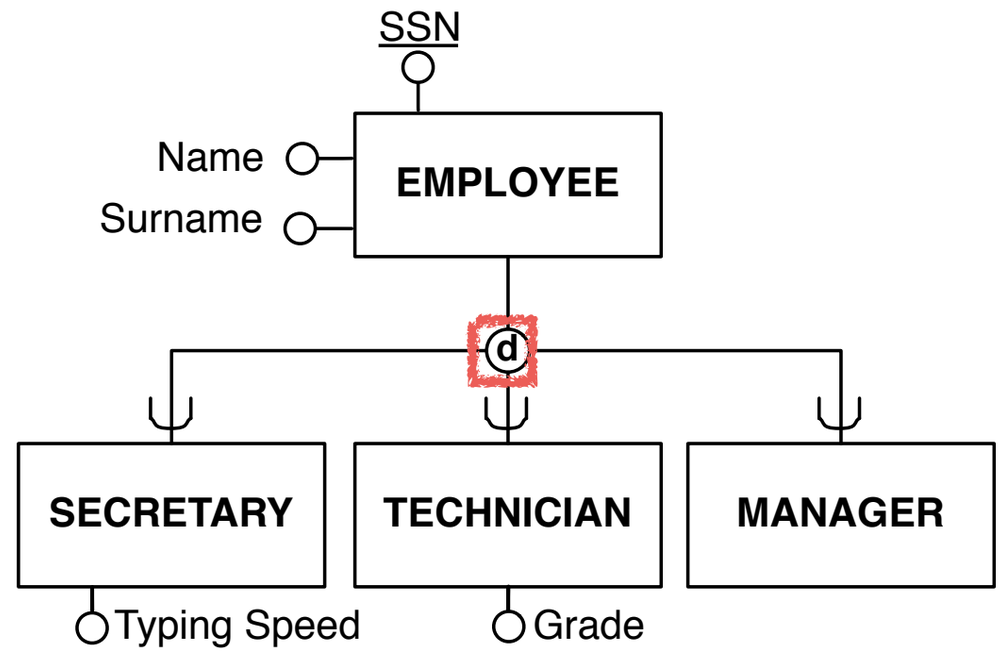
- All the subclasses in a specialisation have their membership condition on the same attribute
  - *Attribute-defined specialisation* (**Defining** attribute of the specialisation)



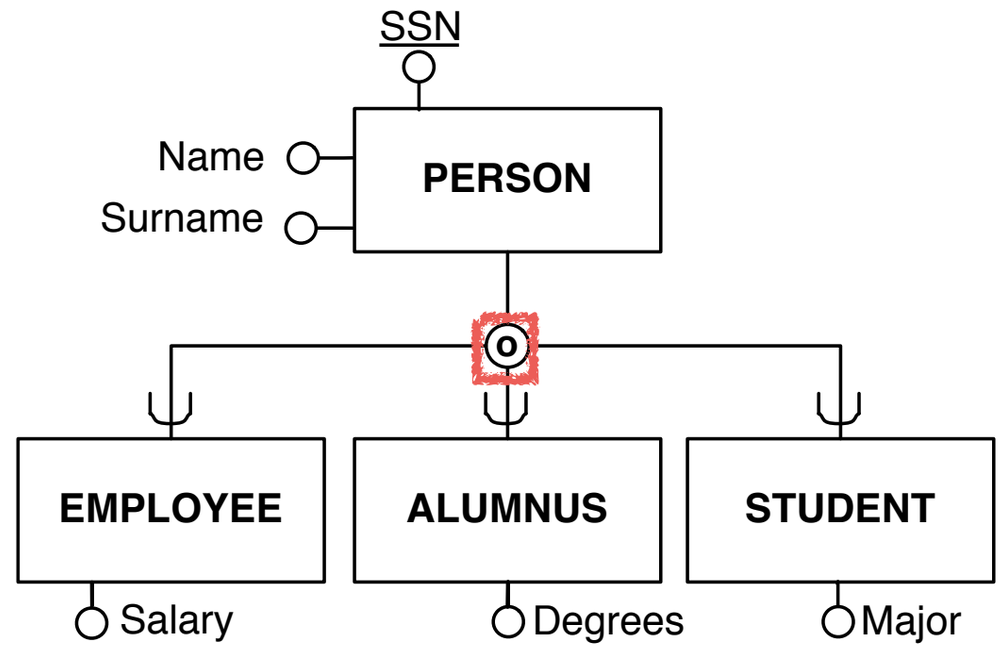
- No specific condition
  - **User-defined** subclass
  - specialisation defined by the database user, instance by instance

Constraints: Disjointness

- An instance can be a member of **at most** one of the subclasses of the specialisation
  - defined by a **`d`** in the graphical notations
  - An attribute-defined specialisation implies the **disjointness** constraint

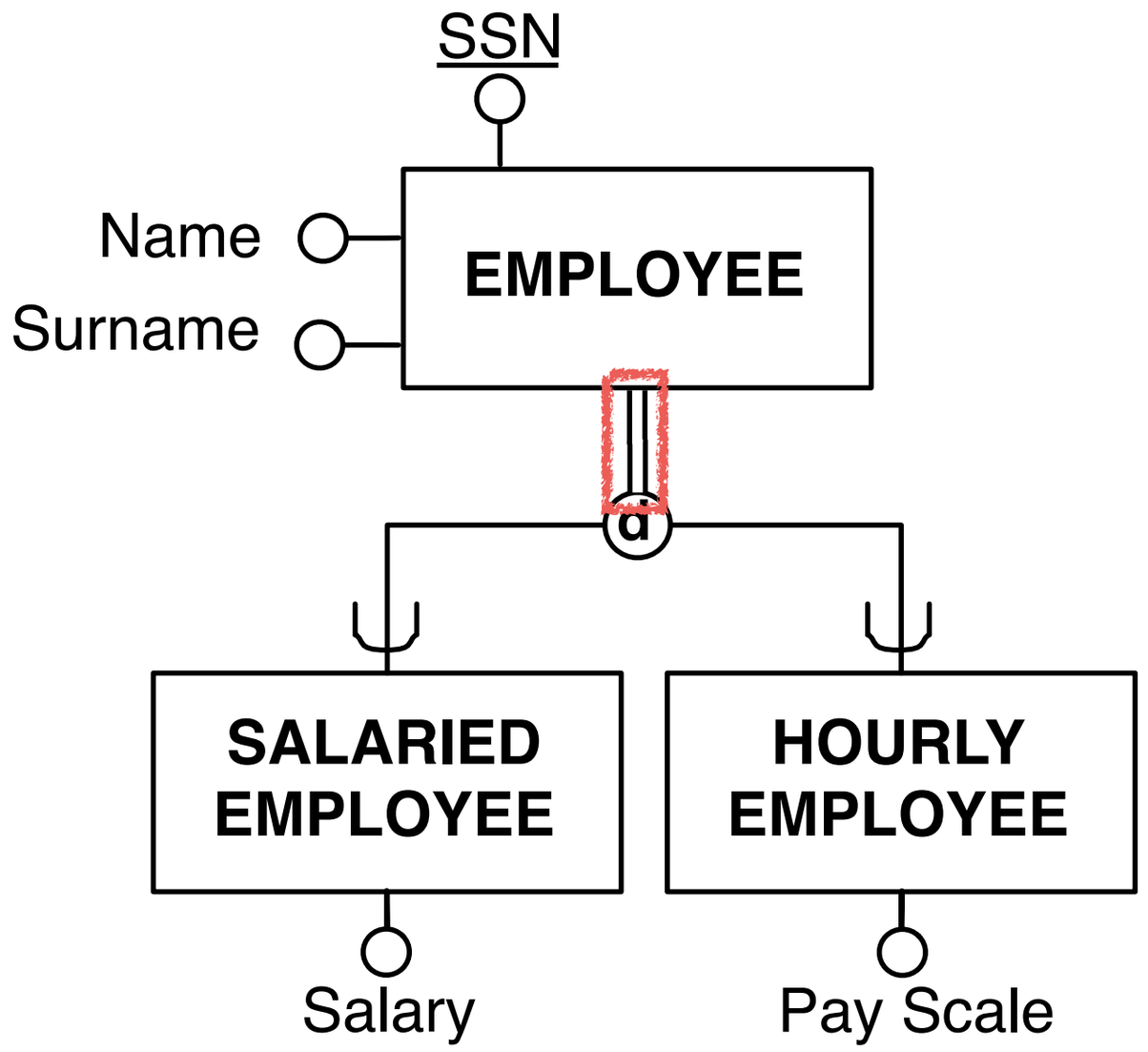


- If the subclasses are not constrained to be disjoint, then their set of instances might be overlapping
  - defined by a **o** in the graphical notations



Constraints: Totalness

- A **Total** specialisation constraint specified that every instance of the superclass must be a member of **at least one** subclass in the specialisation
  - Defined by a **double line** in the graphical notations



- A **Partial** specialisation allows an instance not to belong to any of the subclasses

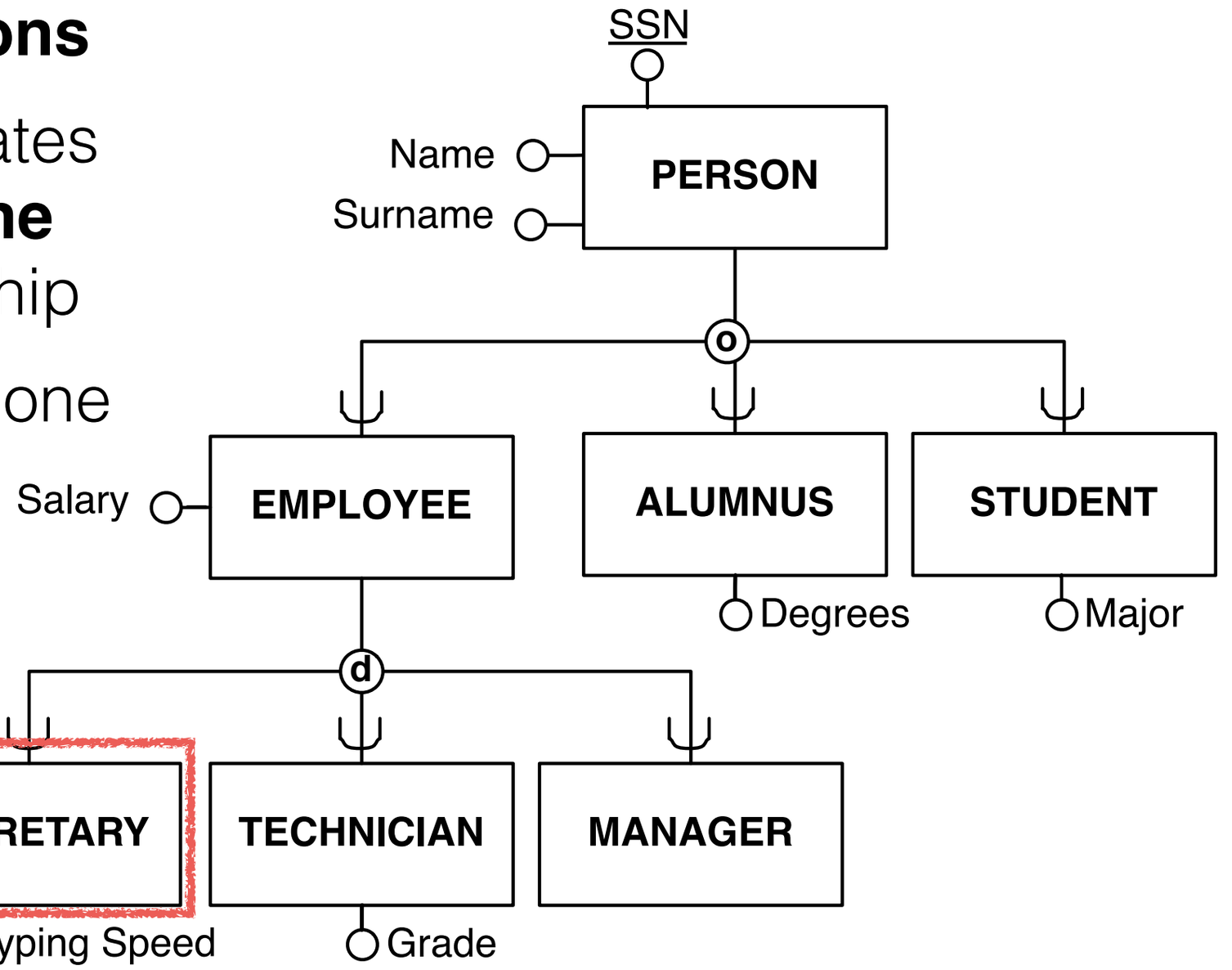
On Disjointness and Totalness

- **Disjointness** and **Totalness** are independent constraints
  - Disjoint, Total
  - Disjoint, Partial
  - Overlapping, Total
  - Overlapping, Partial

- Constraints define rules for instance insertion and deletion. For instance:
  - Deleting an entity from a superclass: deletion from all the subclasses (general)
  - Inserting in a superclass: **mandatory insertion** in a sub-class for predicate-defined or attribute-defined subclasses where the predicate evaluates to `TRUE`
  - Inserting in a superclass of a total specialisation: insertion in **at least one** of the subclasses is mandatory
  - … (other rules left as exercise)

Specialisation Hierarchies

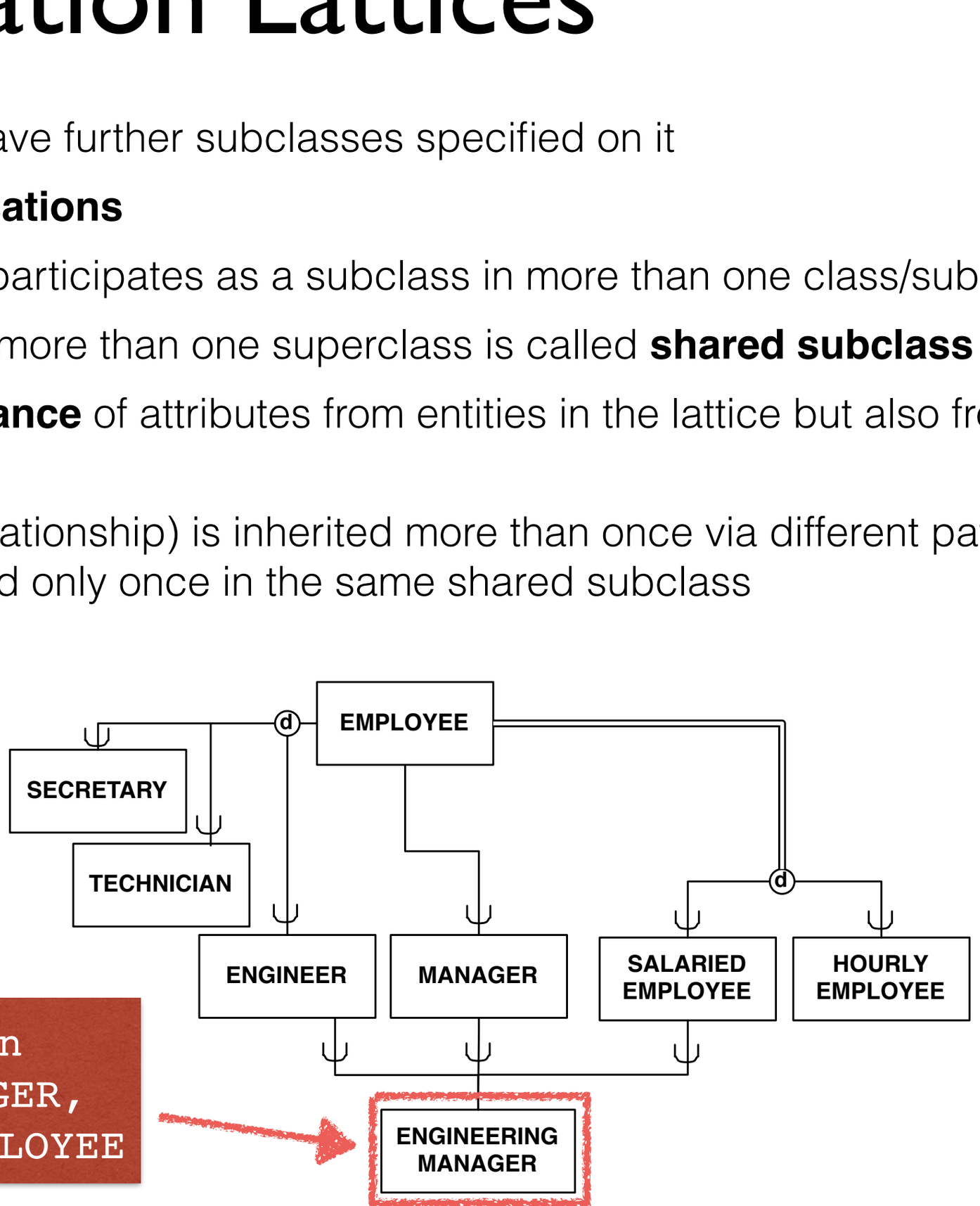
- A subclass might have further subclasses specified on it
- **Hierarchy of specialisations**
  - every subclass participates as a subclass in **only one** class/subclass relationship
- Each subclass has only one parent



A PERSON and an EMPLOYEE

Specialisation Lattices

- A subclass might have further subclasses specified on it
- **Lattice of specialisations**
  - a subclass can participates as a subclass in more than one class/subclass relationship
  - A subclass with more than one superclass is called **shared subclass**
  - **Multiple inheritance** of attributes from entities in the lattice but also from multiple subclasses
  - If attribute (or relationship) is inherited more than once via different paths in lattice, then it is included only once in the same shared subclass



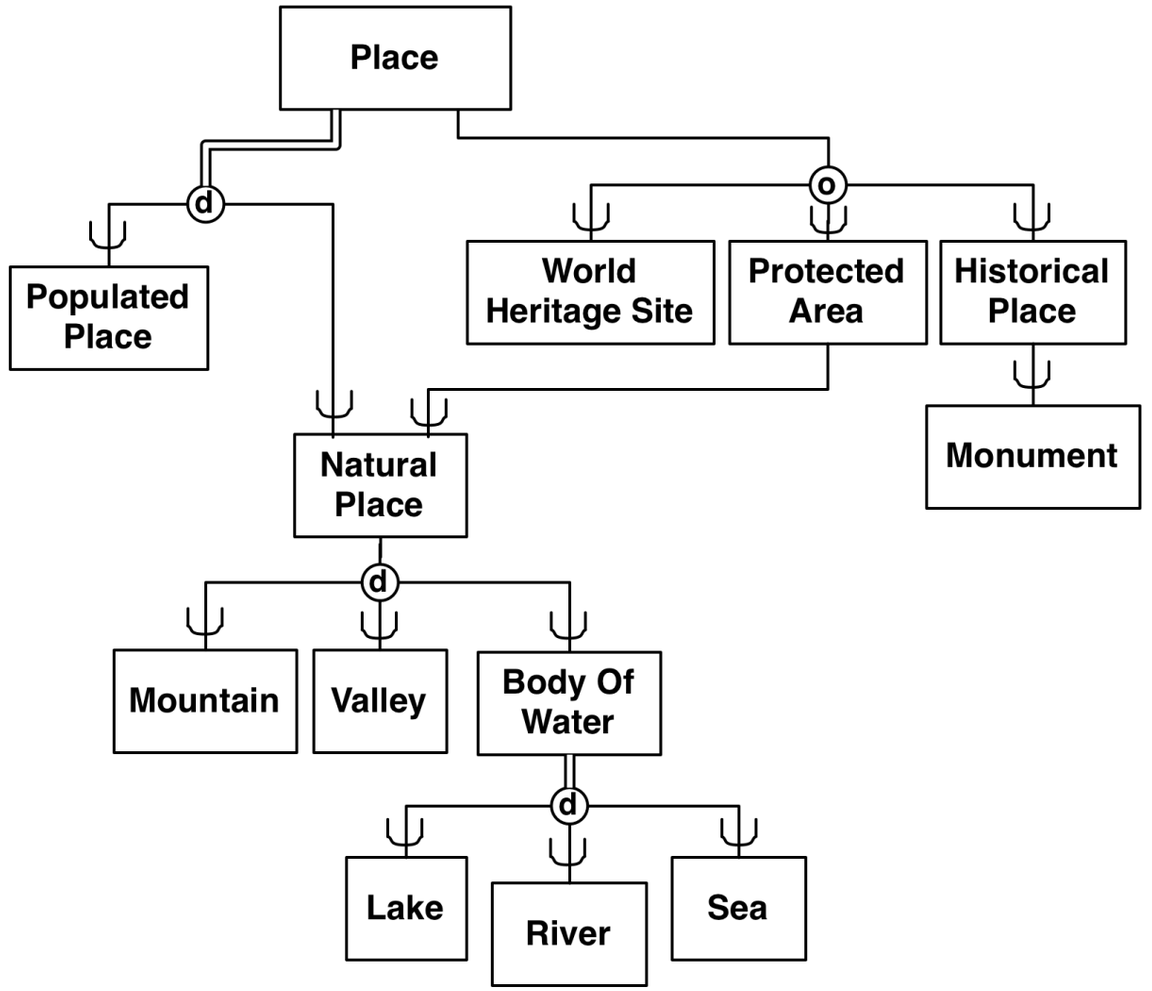
An EMPLOYEE, an ENGINEER, a MANAGER, and a SALARIED EMPLOYEE

ASQ Exercise (1 min)

Consider the EER diagram in figure.

Which of the following statements is correct?

1. An occurrence of the *Monument* entity cannot be an occurrence of the Place entity
2. An occurrence of the *Valley* entity can be an occurrence of the *World Heritage Site* entity
3. An occurrence of the *Valley* entity must be an occurrence of the *Protected Area* entity
4. An occurrence of the *River* entity cannot be an occurrence of the *Populated Place* entity



Database Logical Design

Logical Design

GOAL: construct a relational schema that represents all of the information described in an Entity-Relationship schema

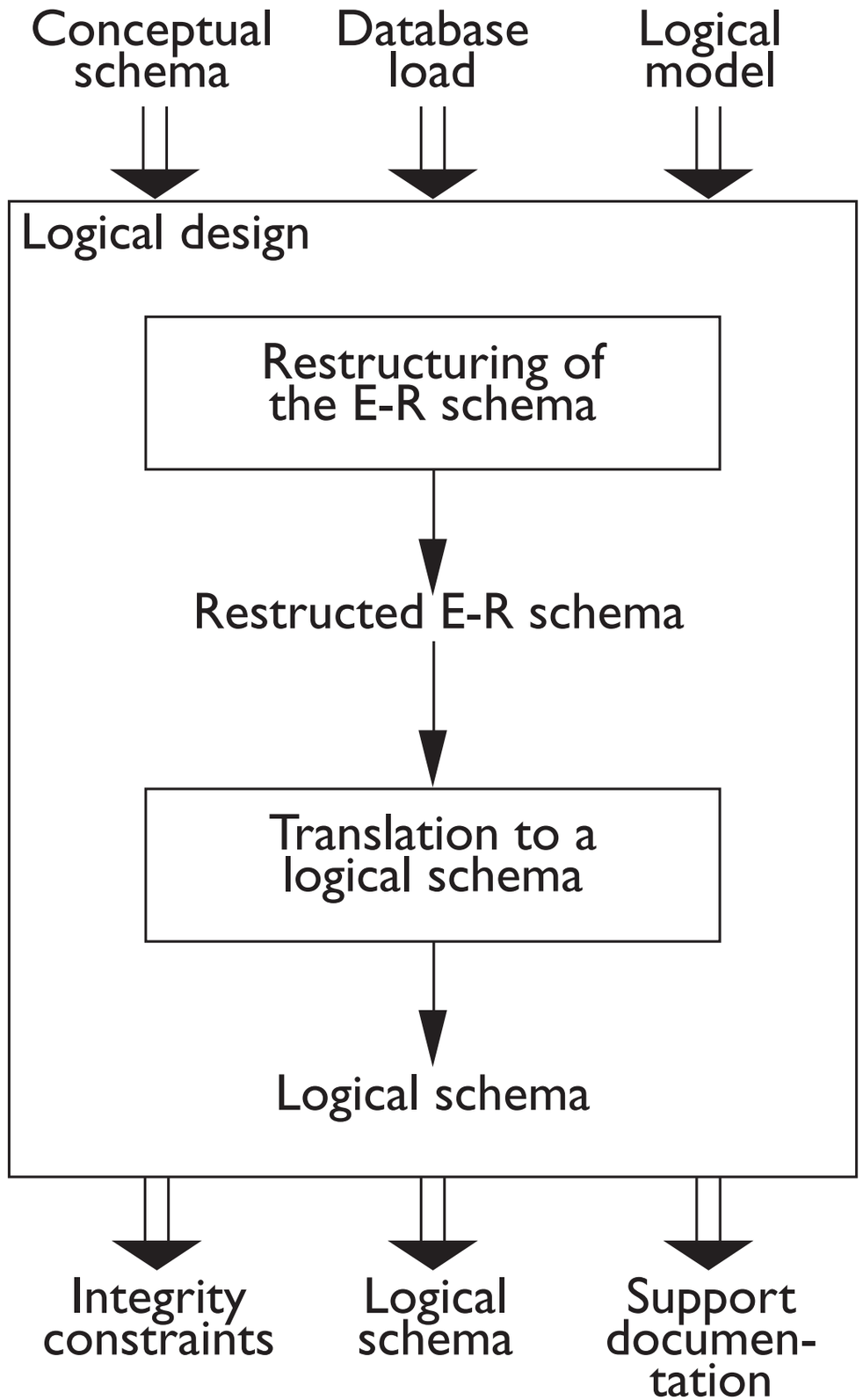
- **correctly**
- **efficiently**

- Not just a simple model translation:
  - **not all** the constructs of the Entity-Relationship model can be translated naturally into the relational model
  - the ER schema **must be restructured** to make the execution of the projected operations as **efficient** as possible

Logical Design Steps

1. **Restructure** the Entity-Relationship schema
   - schema optimisation
   - simplification of following step

2. **Translate** into the logical model
   - based on the features of the logical model
   - in our case, the relational model



Restructuring the ER Schema

Restructuring of an E-R schema

- The restructured ER schema takes into account some <u>implementation</u> aspects (and constraints) coming from the logical model of choice
  - **Not a conceptual model anymore**

- **Goals**
  - Remove constructs for which there is no direct mapping in the relational model
  - Transformations aimed at increasing data access operations

Restructuring Activities

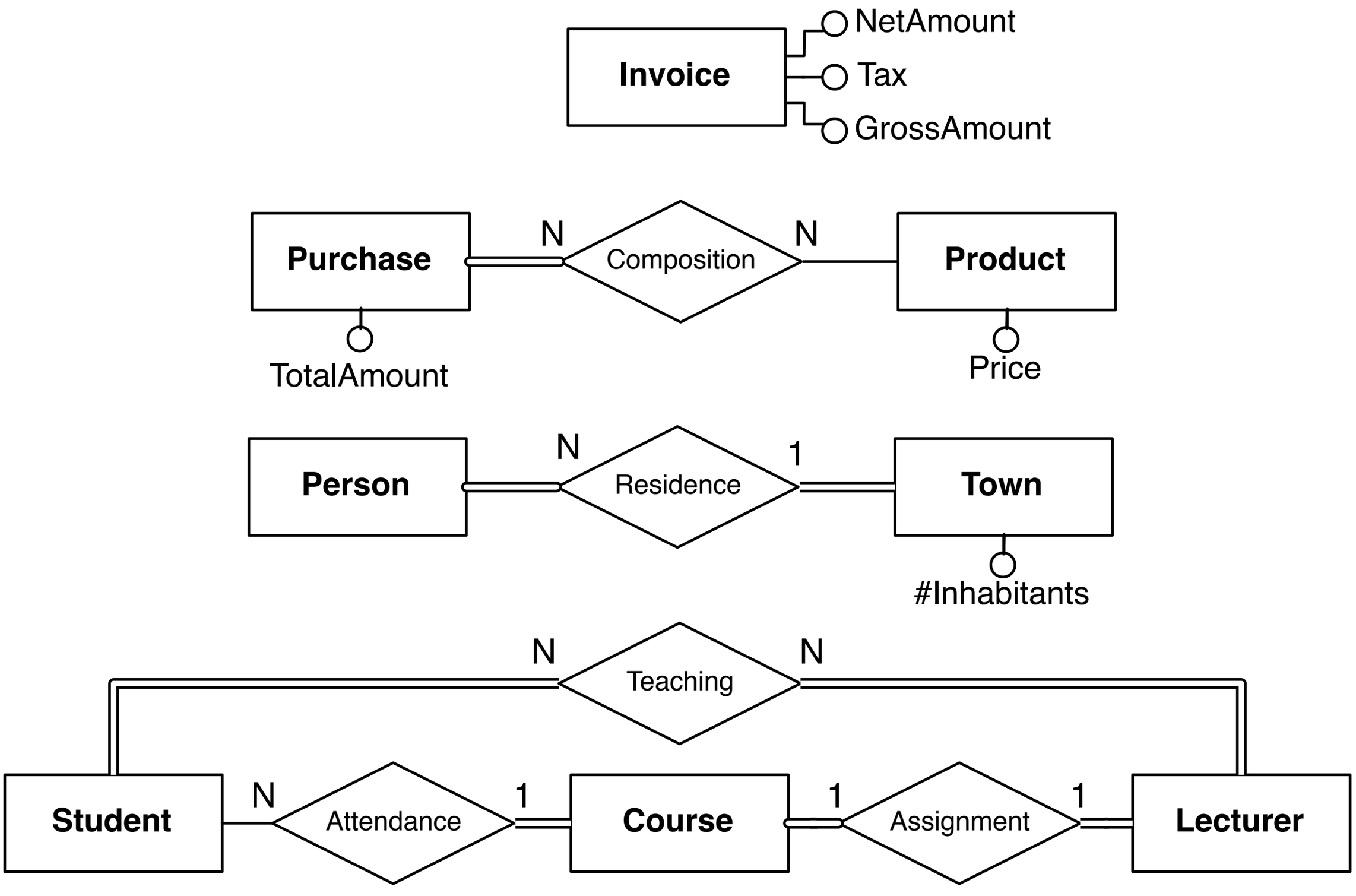
- Analysis of redundancies
- Removal of generalisations
- Partitioning / aggregation of entities and relationships
- Selection of primary keys

Analysis of Redundancies

- Redundancies: relevant information, but derivable from other concepts
  - An Entity-Relationship schema can contain various forms of redundancy
    - Keep, remove, add?

- Effect of redundancies on the logical schema
  - Simpler or faster queries
  - More complex or slower updates
  - Disk space

Example Of Schemas With Redundancies



Taking A Decision About Redundancies

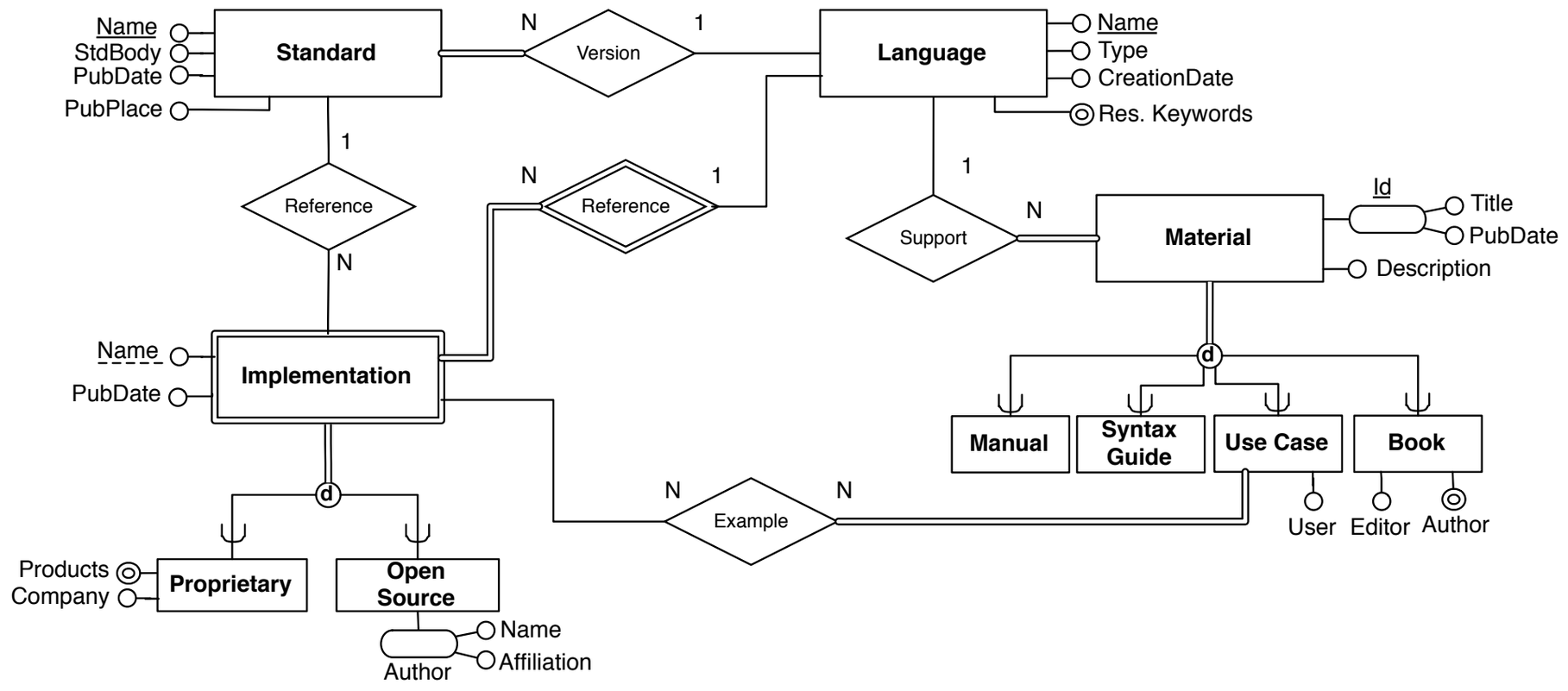
- Redundant information in a database can be
  - **An advantage**: a reduction in the number of accesses necessary to obtain the derived information
  - **A disadvantage**: a larger storage requirement (which is often a negligible cost) and the necessity for carrying out additional operations in order to keep the derived data up to date

- Final decision: cost of operations Vs. needed storage

ASQ Exercise (1 min)

Consider the restructuring of the EER diagram in figure.

Which of the following relationships is redundant?



1 *Reference, between Standard and Implementation*

2 *Reference, between Language and Implementation*

3 *Example*

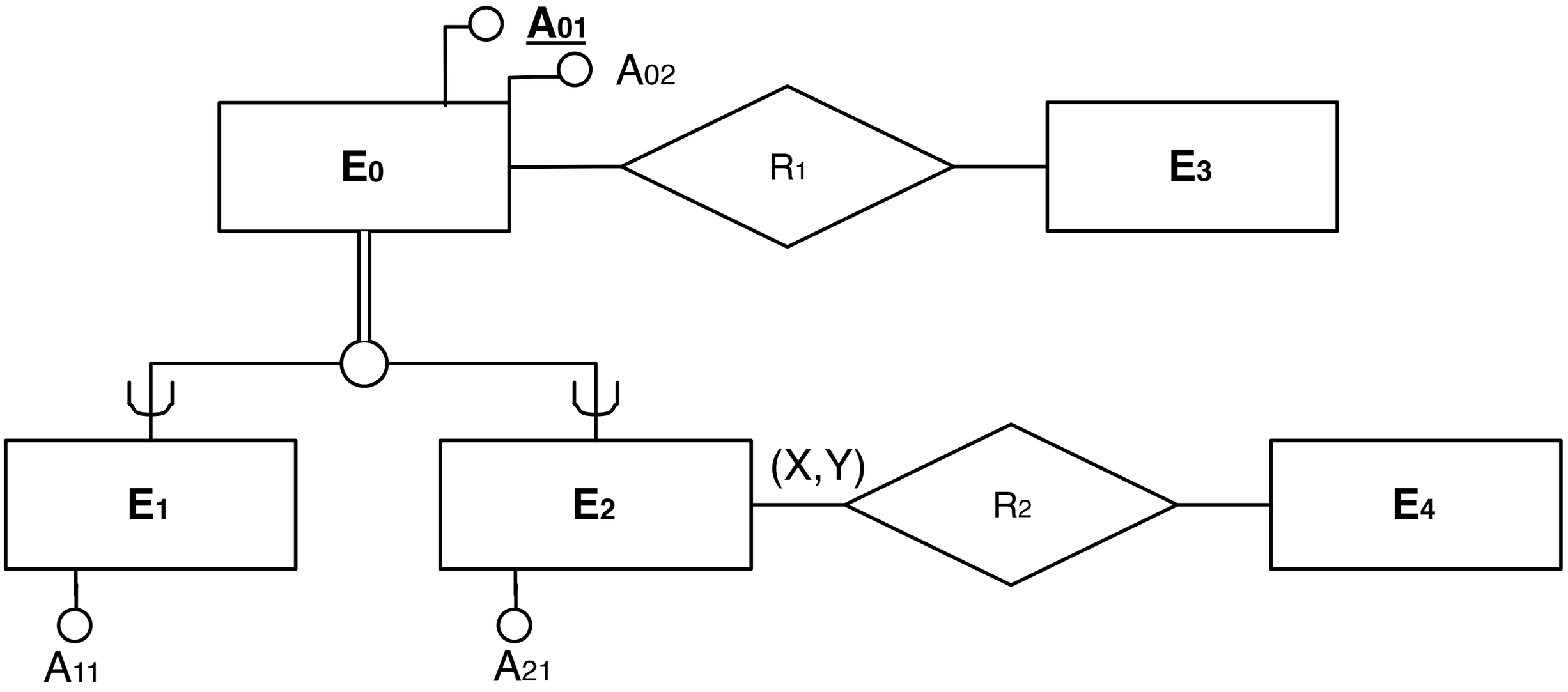
Removing Specialisations

- Specialisation/Generalisation **cannot be directly represented** in the relational model
- Substituted by entities and relationships
- Methods

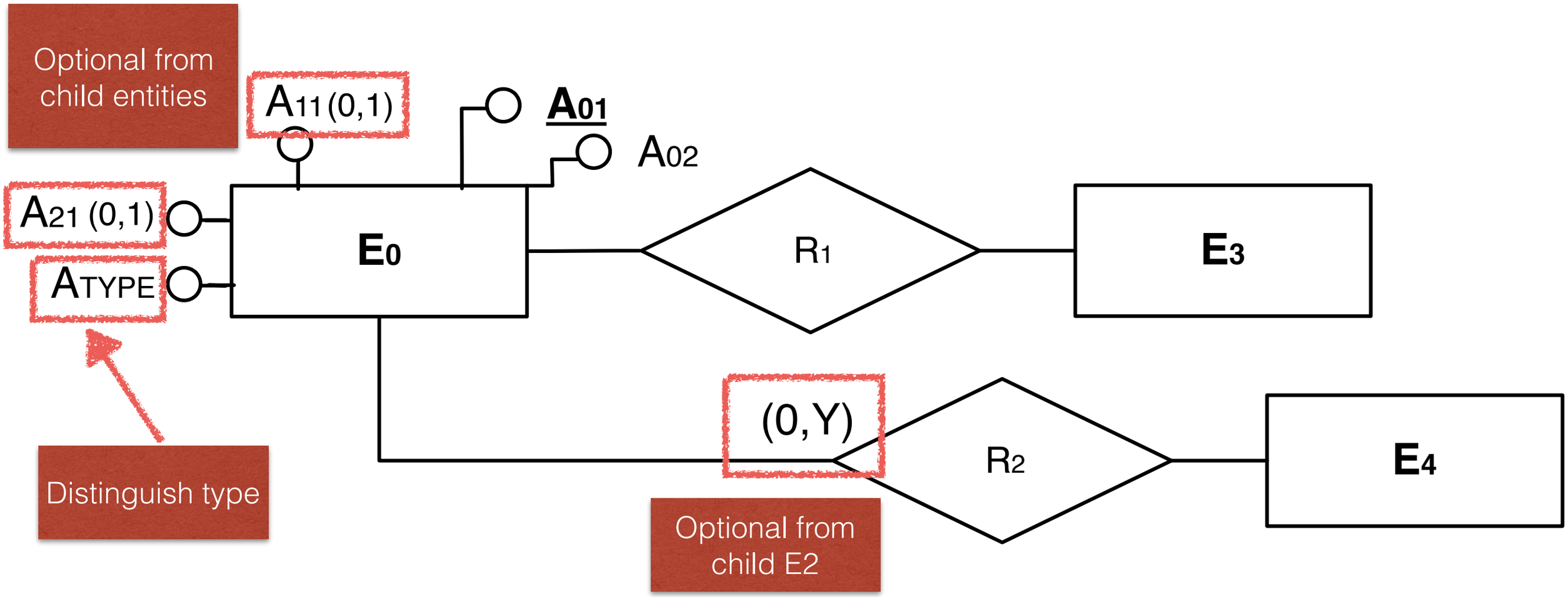
  1. collapse child entities in the parent entity
  2. collapse parent entity on child entities
  3. **substitute the specialisation/generalisation with relationships**
     - **<u>This is our standard way</u>**

- Hierarchies: apply approach, **starting from the deeper levels**

Reference Generalisation Schema

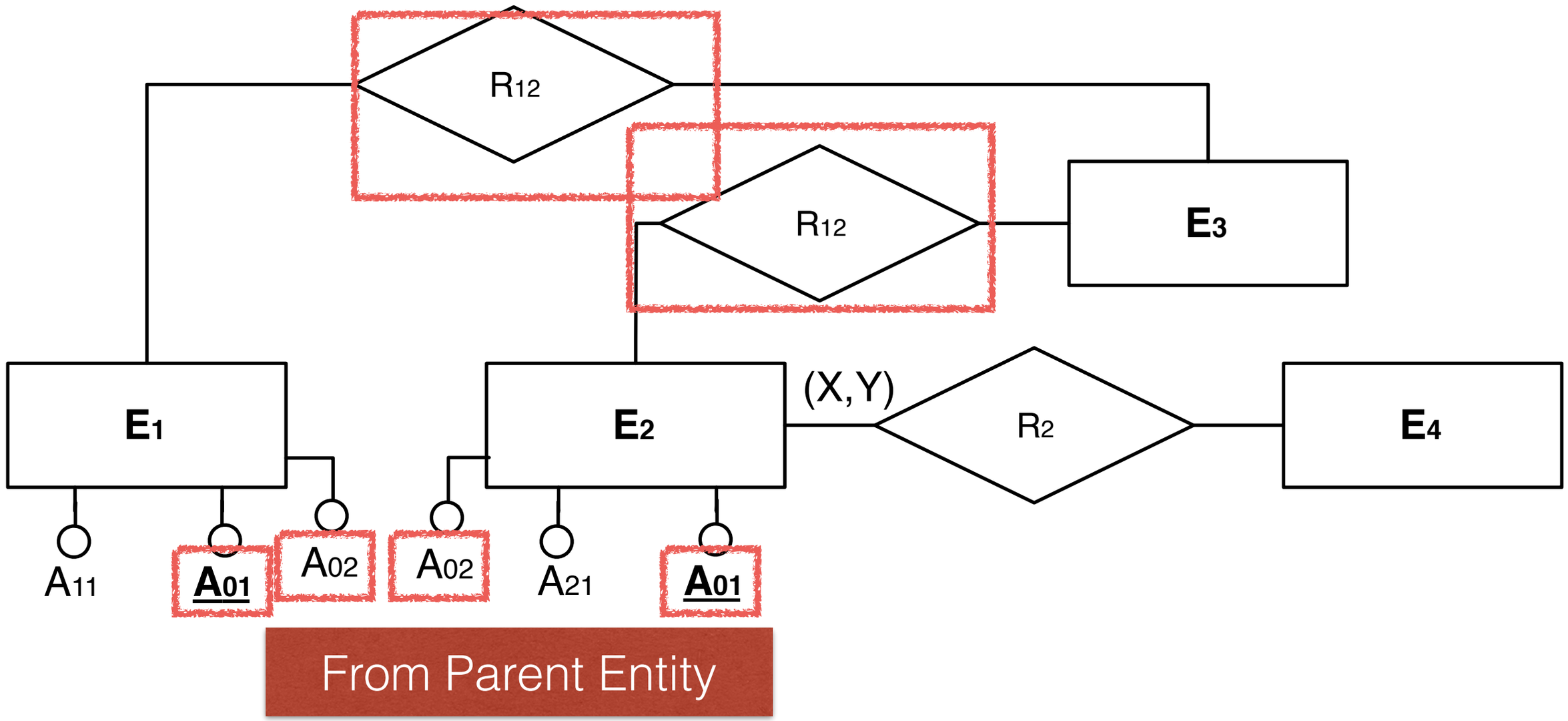


Parent Collapsing



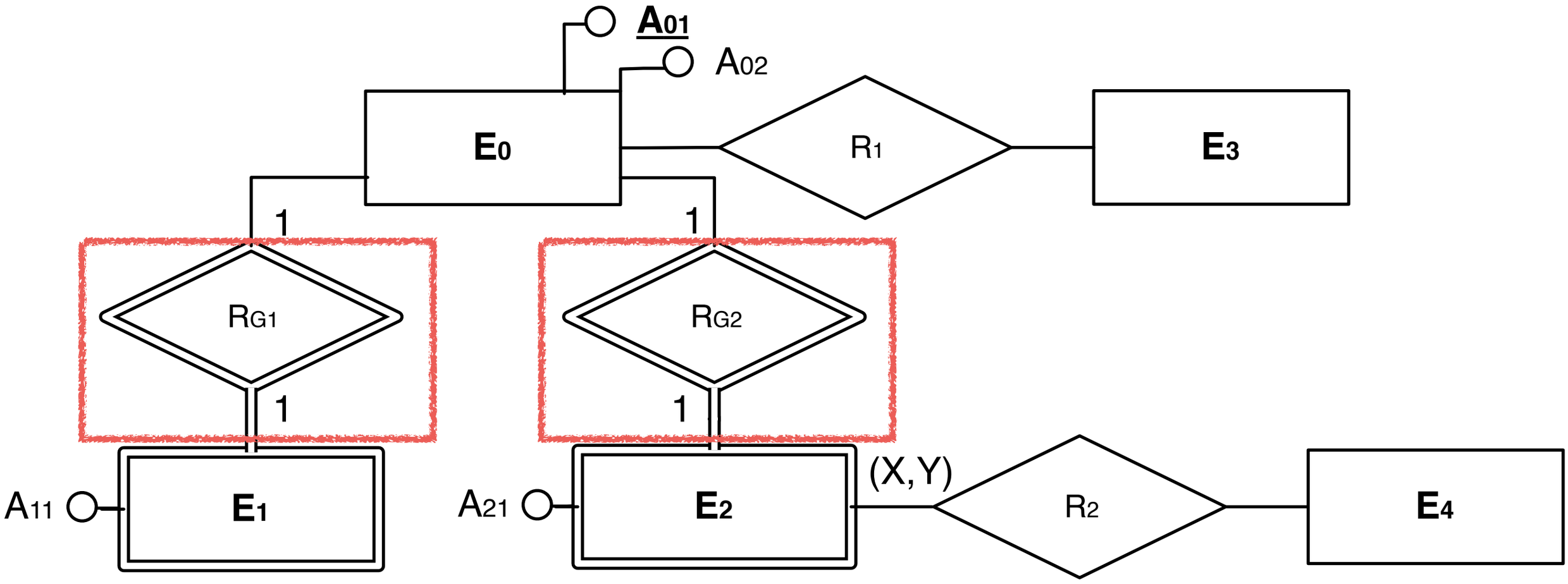
- More appropriate when
  - Child entities introduce little differences (so result in few NULL values)
  - Access operations do not need to distinguish between parent and child entities

Child Collapsing



- Possible only if the **specialisation** is **total**
- More appropriate when access operations need to **distinguish** between different child entities (more efficient)

Substitute with Relationships

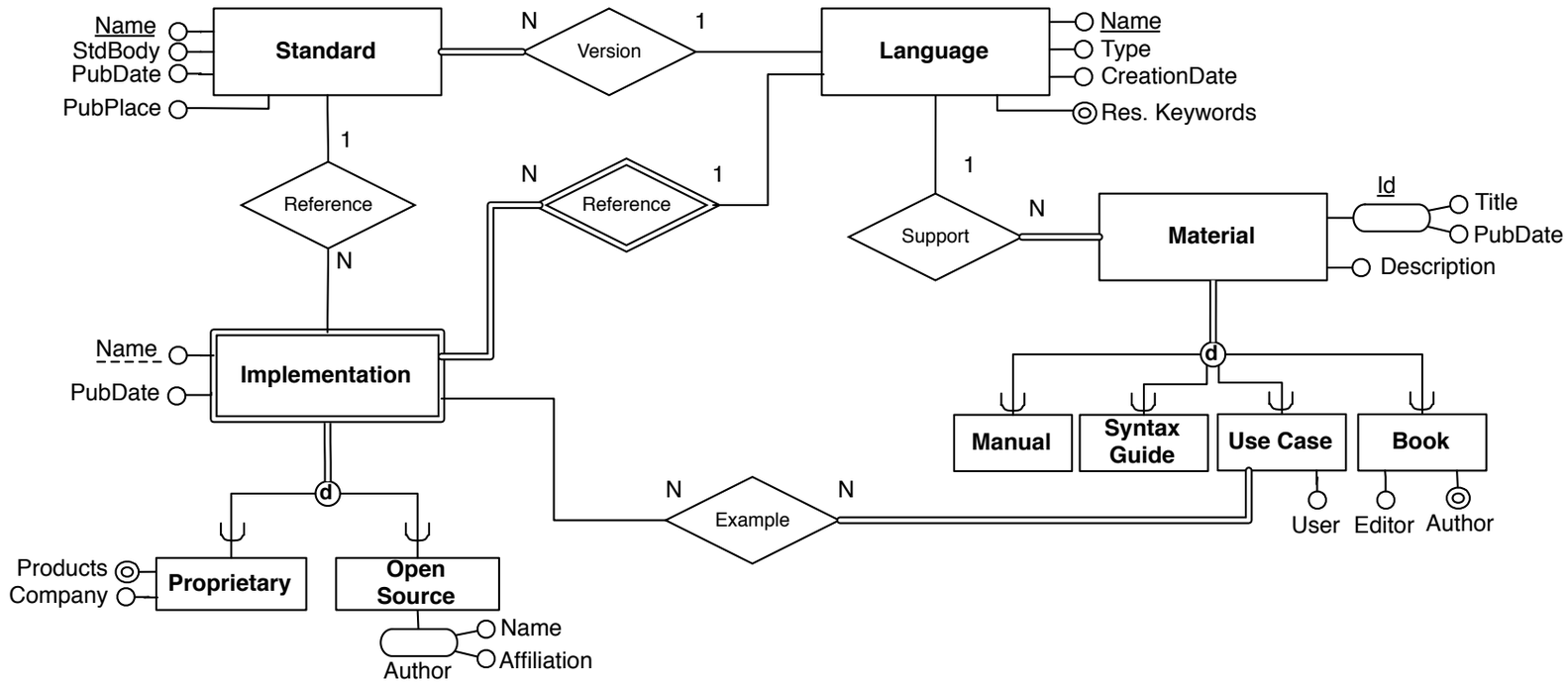


- **Most General Solution**, and always applicable
- But reconstructing the original information can be **expensive**

ASQ Exercise (3 min)

Consider the restructuring of the EER diagram in figure, performed with a *parent-collapsing* specialisation removal method

Which of the following statements is/are correct? There can be more than one



1

The Material entity participates in the Example relationship with a partial participation constraint.

2

The Implementation entity includes 5 attributes, but only the Author attribute becomes optional because it is multi-valued

3

The Implementation entity features 6 attributes, but only 3 become optional

4

The Example relationship changes its cardinality from N:N to 1:N because not all the instances from the Material entities will be related to an instance of the Implementation entity

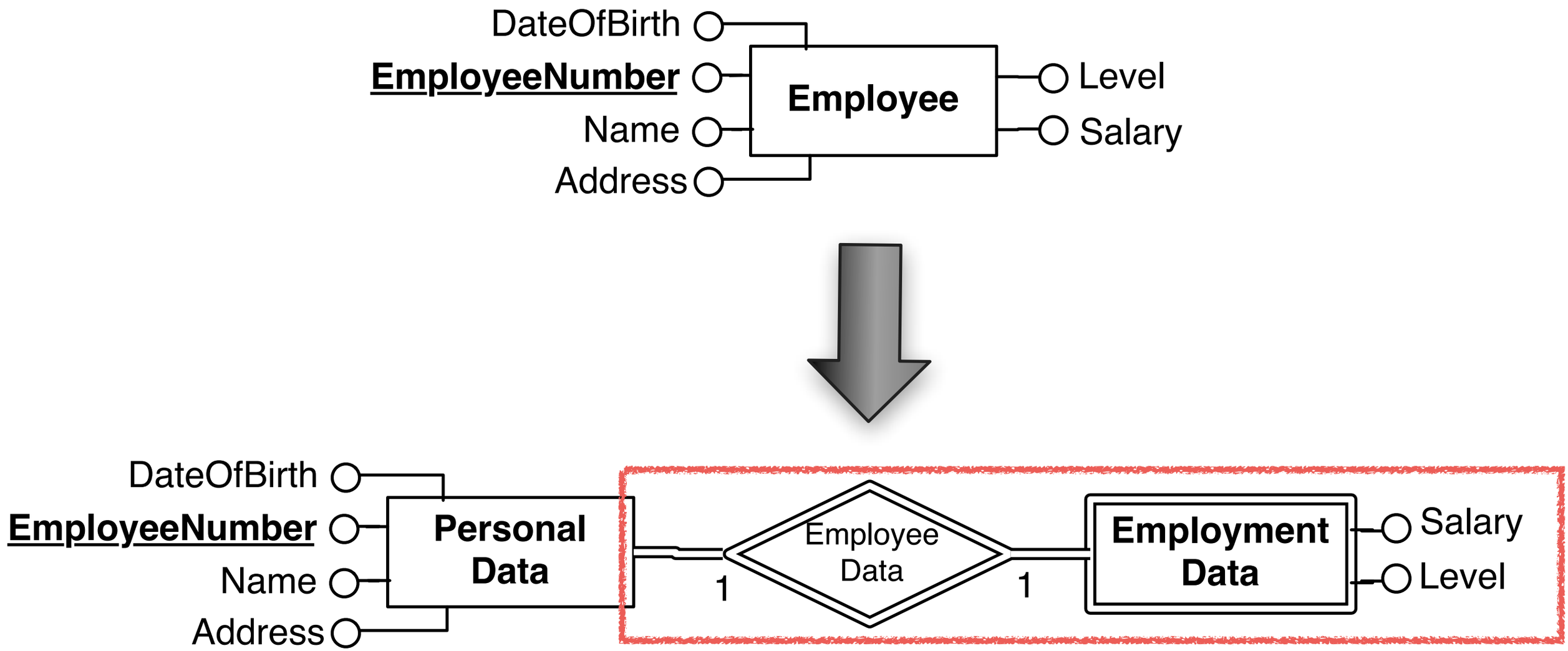
Partitioning and Merging of Concepts

- Entities and relationships of an E-R schema can be **partitioned** or **merged**

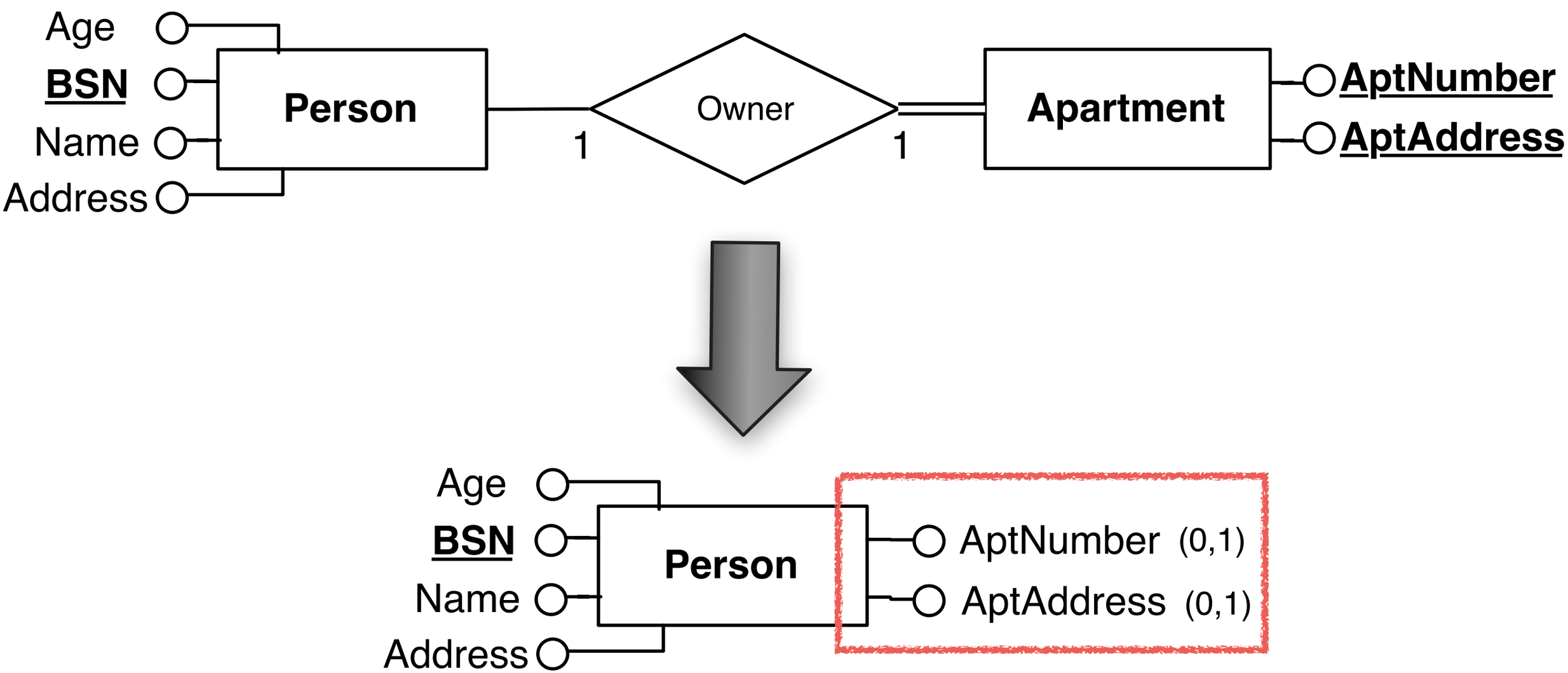
- Better representation of different concepts

- Improve the efficiency of operations. Accesses are reduced by:
  - Separating attributes of the same concept that are accessed by different operations
  - Merging attributes of different concepts that are accessed by the same operations
- The same criteria as for redundancies are valid in making a decision about this type of restructuring

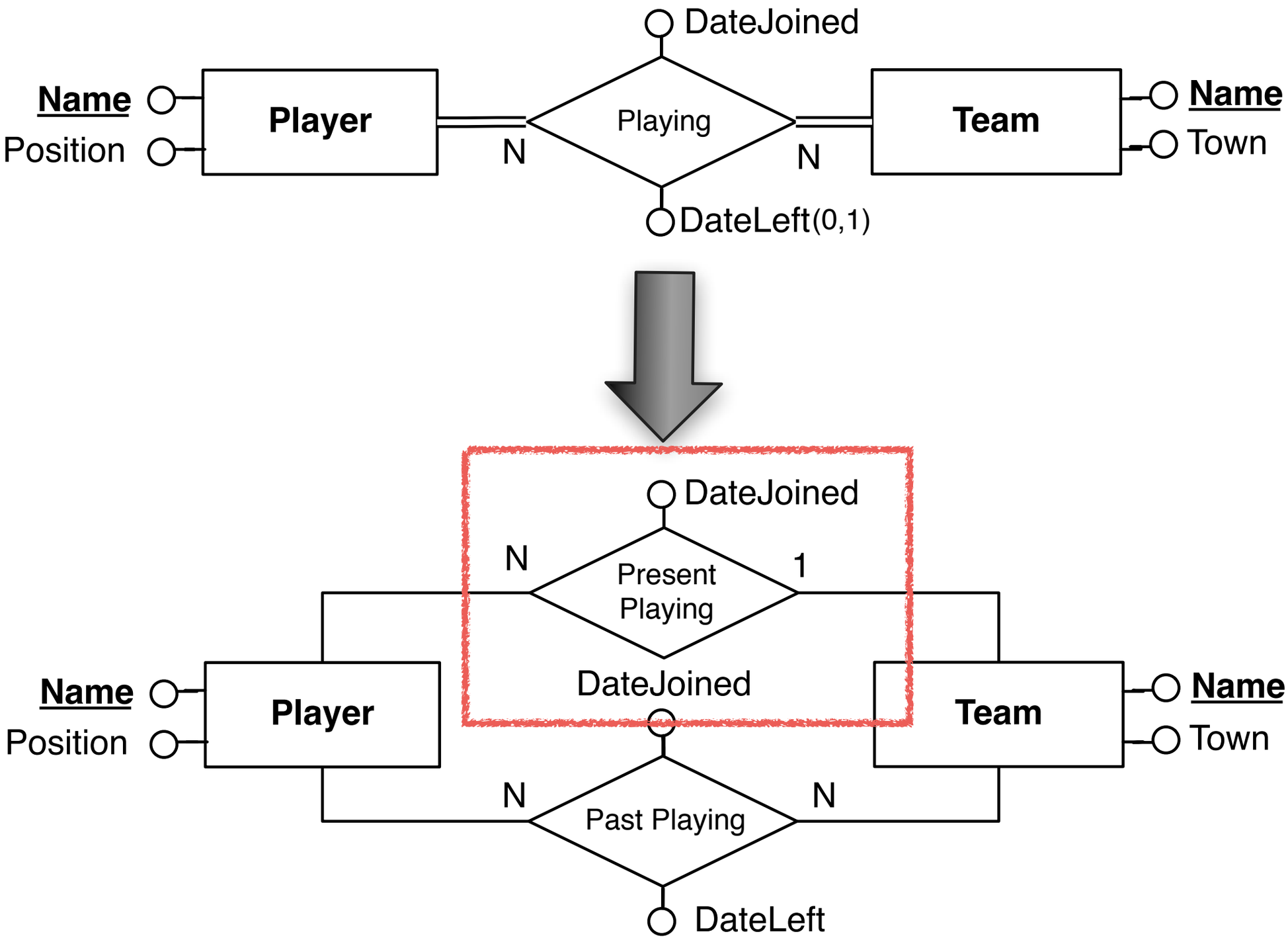
Example Schema: Entity Partitioning



Example Schema: Entity Merging



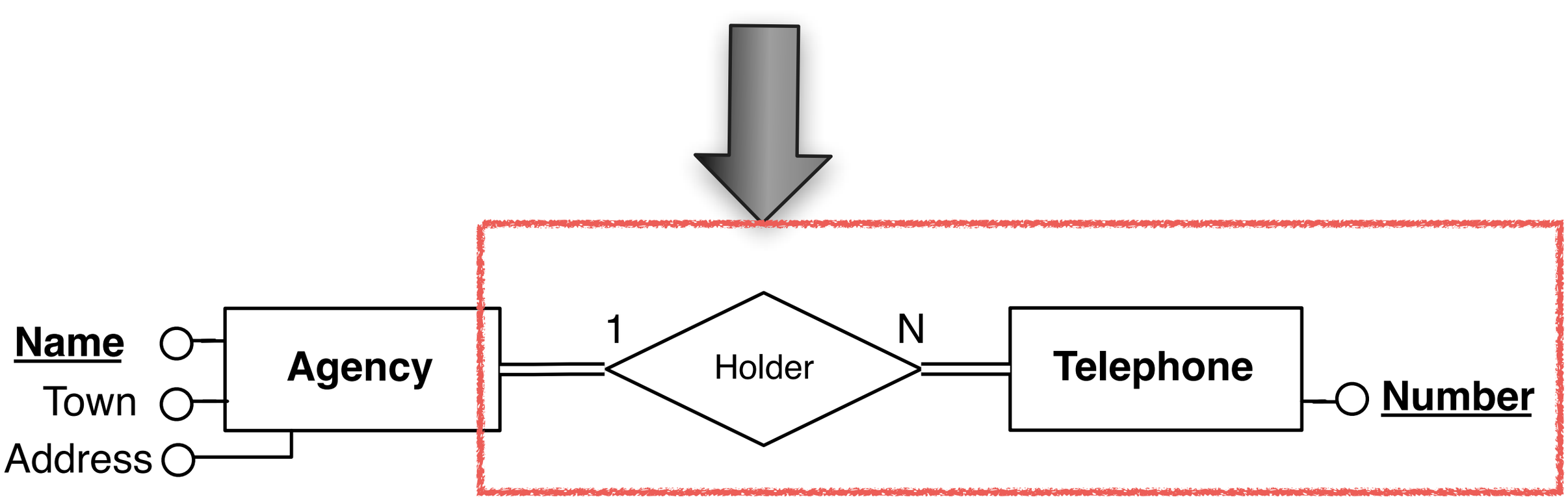
Example Schema: Relationship



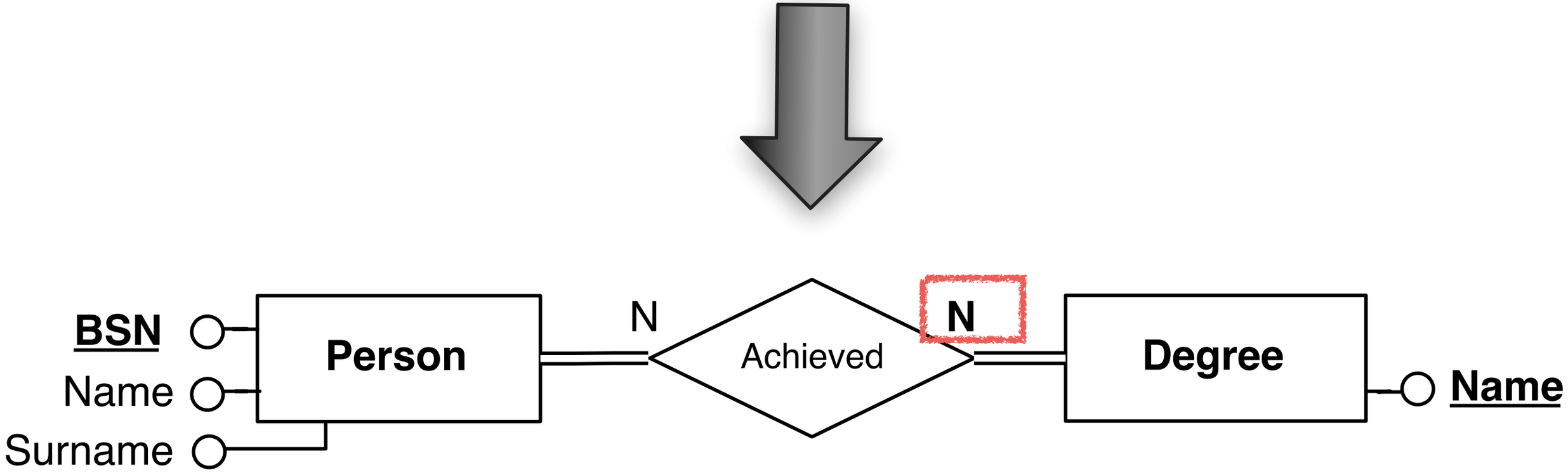
Removal of Multi-Value Attributes

- No representation in the relational model
- Multivalued attribute represented by a **new entity** associated to the original entity
  - Cardinalities of new relationships depend on cardinality of original attribute
  - But also on the nature of the attribute!

Example Schema: Multi-Value / I

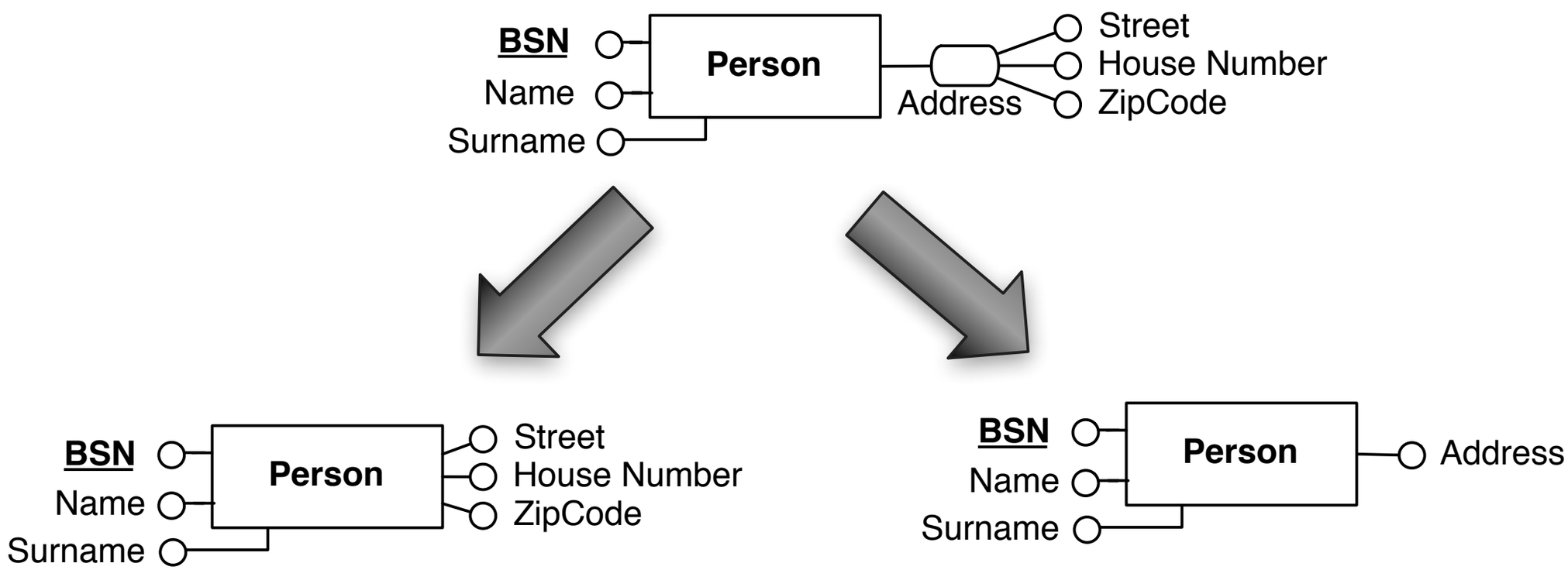


Example Schema: Multi-Value /2



Removal of Complex Attributes

- No representation in the relational model
- Two alternatives:
  - Collapse sub-attributes
    - Better if need to access the single attributes
    - Or composite attribute is candidate identifier
  - Introduce a new attribute which concatenates the others
    - Better if access to the whole information



Selection of Primary Identifiers

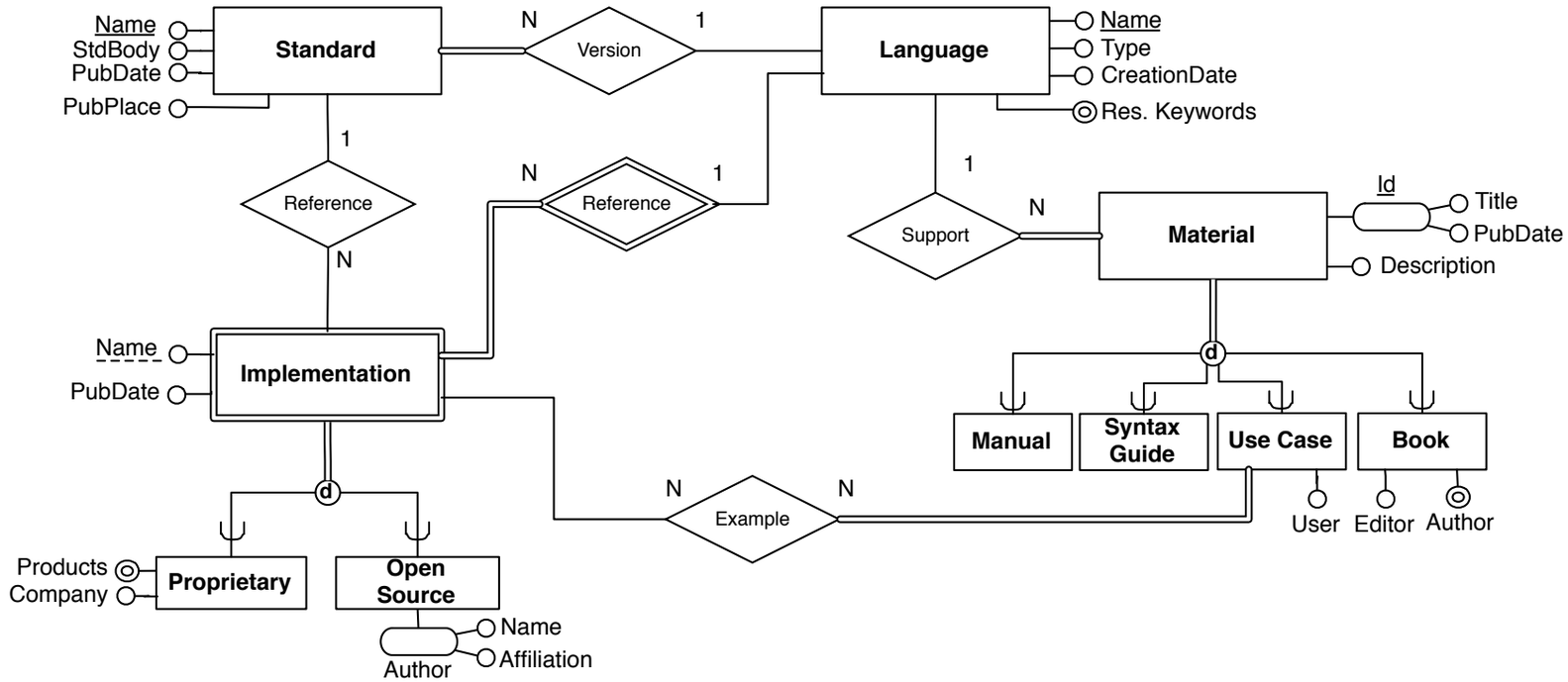
- Required to define **primary keys** for tables
- A good identifier
  - no NULL values
  - **One** or **few** attributes
  - **internal attribute**, better than external ones
  - Is often used by many access operations

- If none of the candidate identifiers satisfies the above requirements
  - introduce further attribute to the entity
  - special values (often called codes) generated solely for the purpose of identifying occurrences of the entity

ASQ Exercise (1 min)

Consider the restructuring of the EER diagram in figure.

Which of the following statements about the *Title* and *PubDate* attributes of the *Material* entity is correct?



1. They are merged into a single *id* attribute
2. They are preserved as independent attributes, and an *Id* attribute is a new primary identifier
3. They are preserved as independent attributes, and together they become primary identifiers
4. They are discarded from the model

Translation into the Relational Model

Creation Of The Relational Model

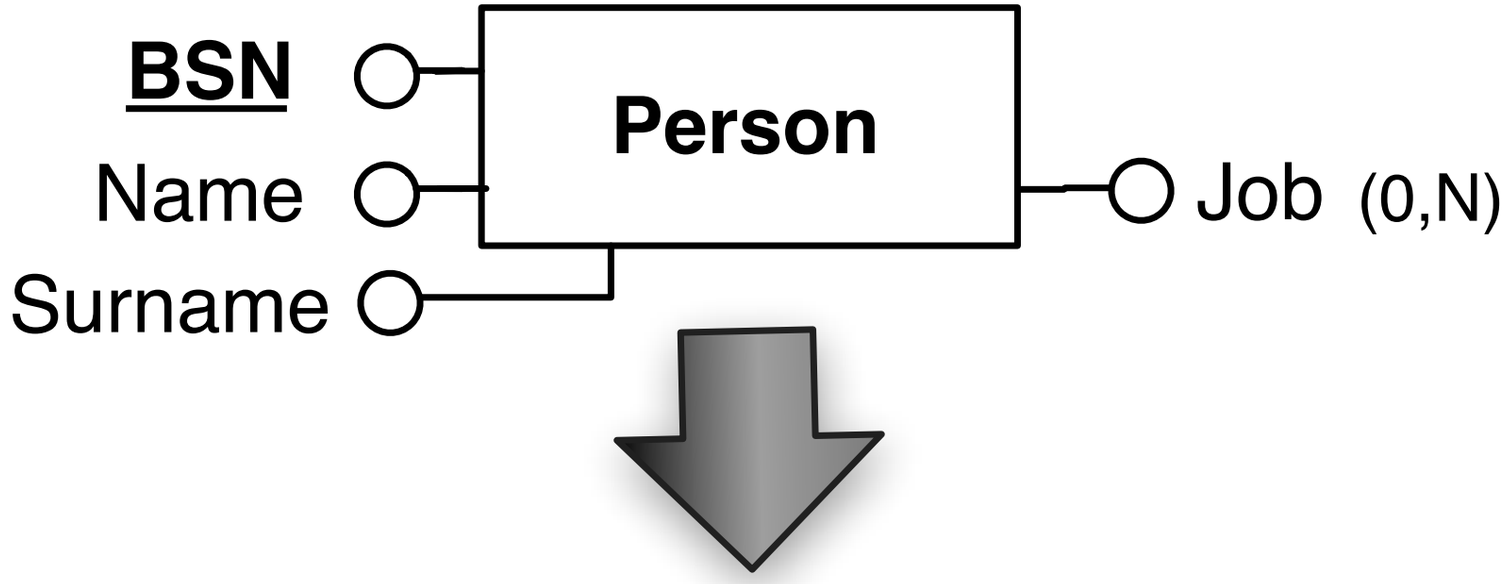
- Transformation between different data models

- Starting from a (restructured) E-R schema, **an equivalent** relational schema is constructed
  - **Equivalent**: a schema capable of representing the same information

- Transformations
  - Each entity maps to a table having the same attributes
  - Mapping of relationships considers maximum cardinality

Example of Entity Transformation

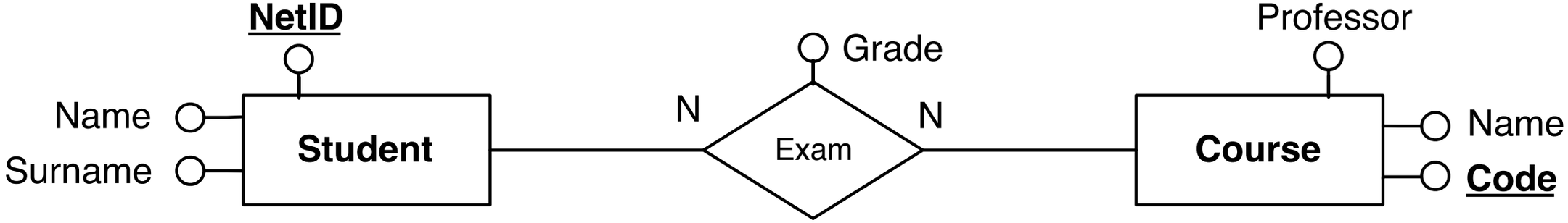
- Primary Key is underlined
- Optional attributes are denoted by an *****

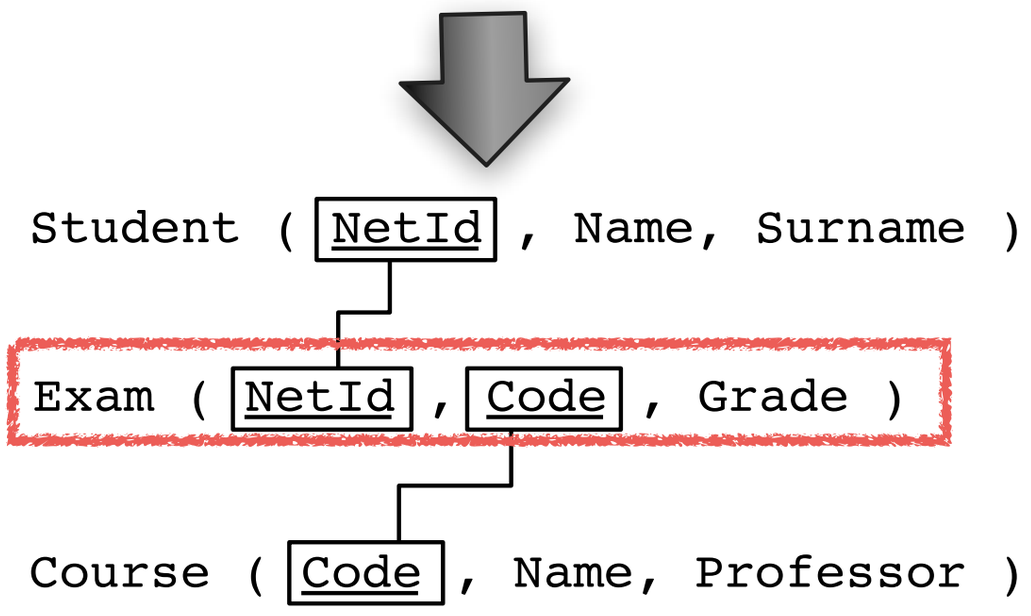


Person (BSN, Name, Surname, Job*)

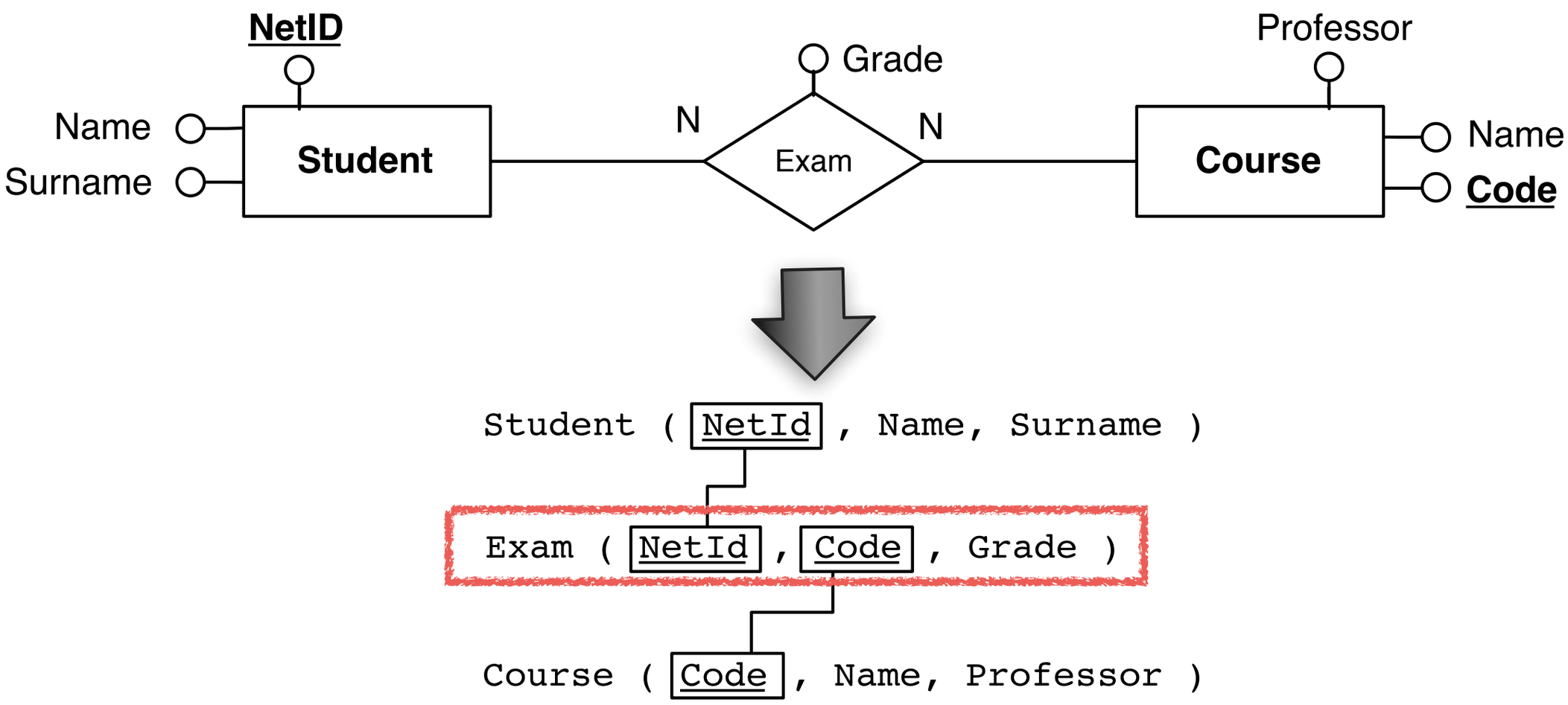
Translation of `Many to Many` binary relationship

- Each many to many relationship maps to a **new table**
  - Primary key is the **combination** of the primary keys of the related entities
  - Name of attributes in the relationship table can be changed (in case of recursive relationships)



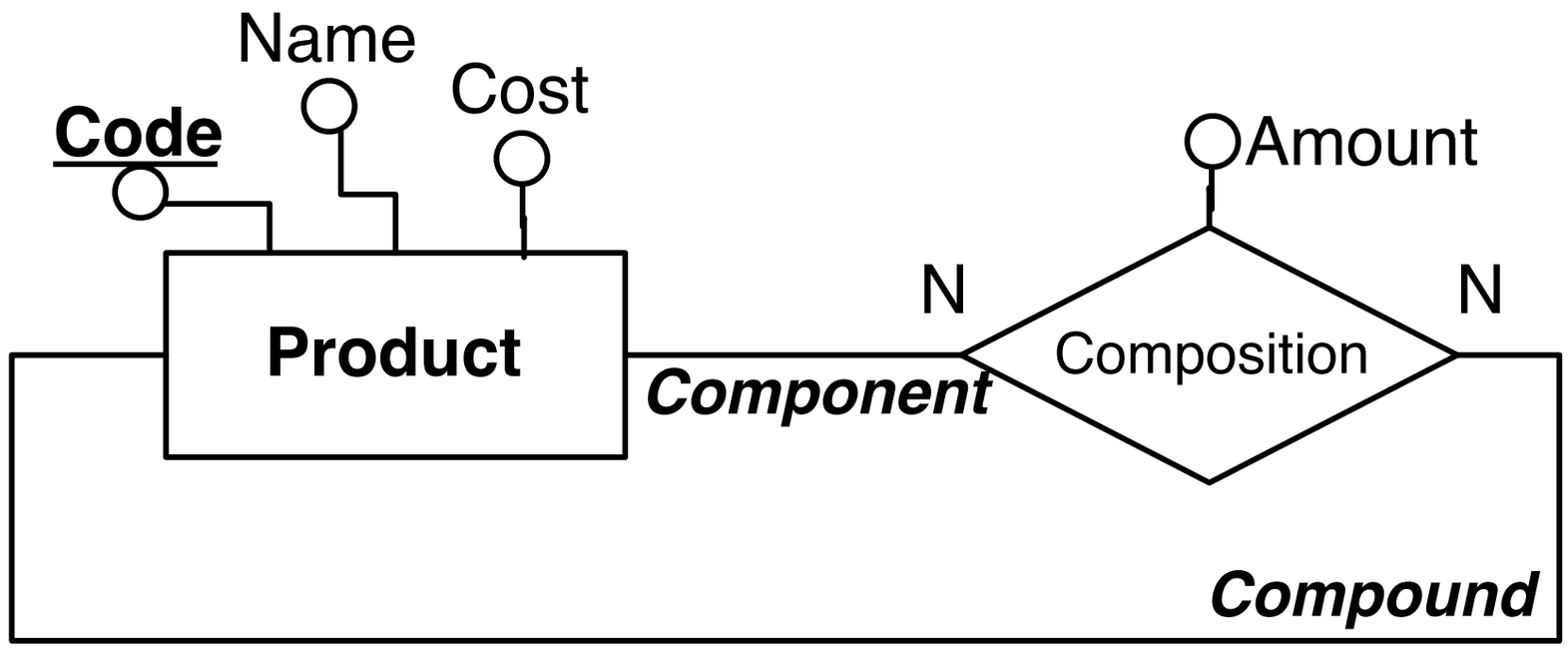


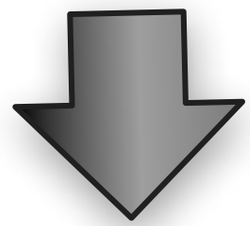
Example of Referential Integrity Constraint




```
CREATE table Exam
 (
   NetId      CHARACTER(6) REFERENCES Student(NetId)
   Code      CHARACTER(6) REFERENCES Course(Code),
   Grade DECIMAL (4) DEFAULT 0,
   PRIMARY KEY (NetId, Code)
```

Recursive Many To Many



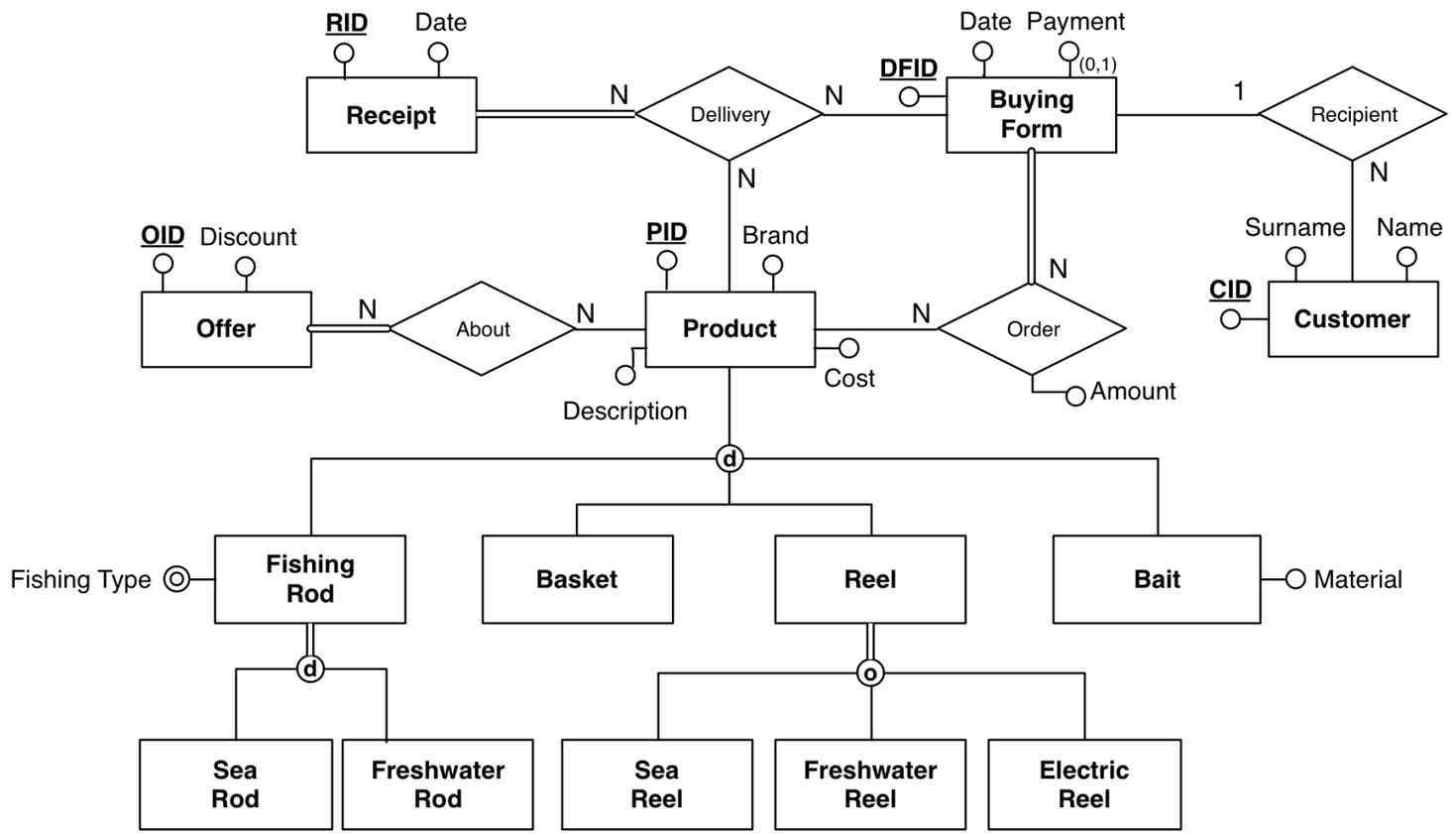


```
Product ( Code , Name, Cost )

Composition ( CodeComponent , CodeCompount , Quantity )
```

Consider the transformation of the EER diagram in figure.

Which of the following statements about the *Order* relationship is/are correct? (There can be more than one)

(I min)



1. The resulting table will include 3 attributes

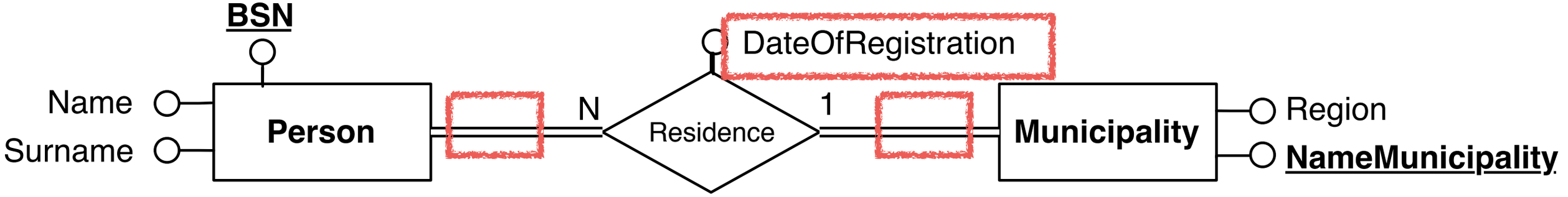
2. Given that the *Product* table has optional participation, the *PID* attribute not be part of the primary key of the resulting table

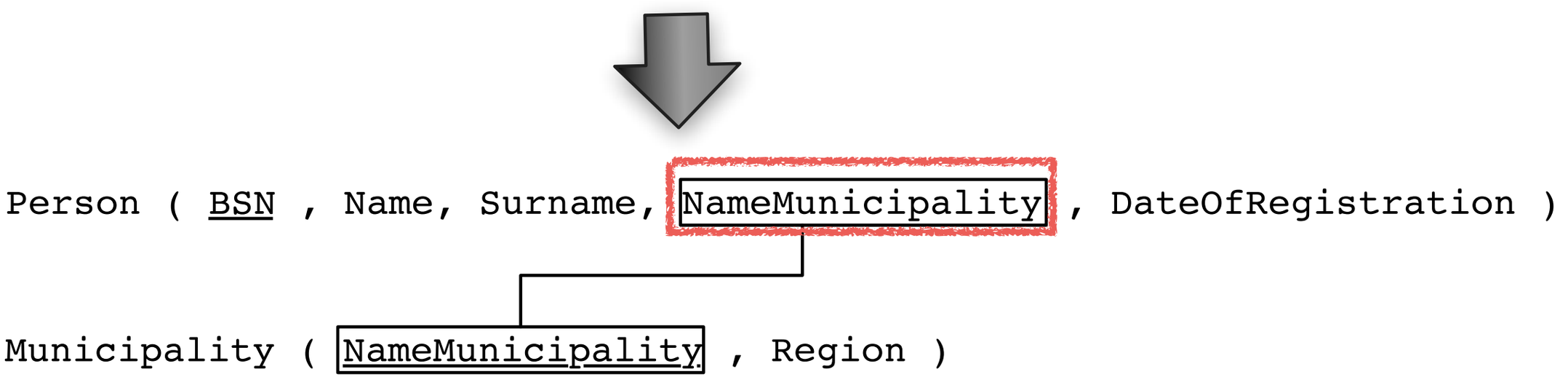
3. The *Amount* attribute will also be part of the primary key of the resulting table

4. The resulting table will also include the *Date* and *Payment* attributes

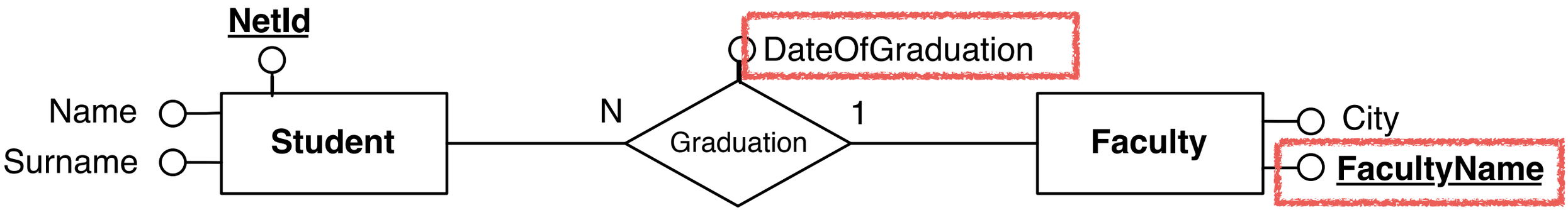
Translation of `One to Many` binary relationship

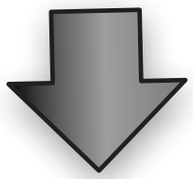
- Two possible translations
  - Attributes (**foreign key**)
  - New table (**relationship relation**)





Example One To Many





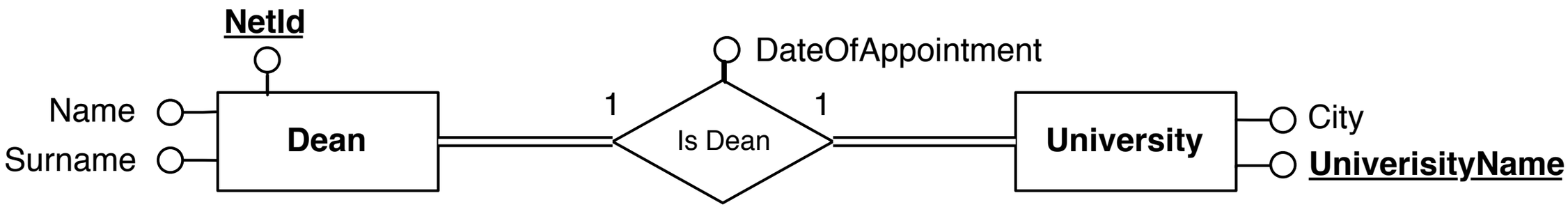
```
Student ( NetId , Name, Surname, FacultyName* , DateOfGraduation*)

Faculty ( FacultyName , City )
```

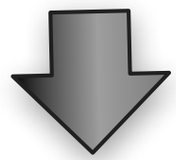
Translation of `One to One` binary relationship

- There can be several translations, according to the **minimum cardinality value**
  - Total participation on both roles
  - Total participation on one role
  - Optional participation on both roles

Example One To One /1

- Total participation on both roles: **Foreign key** approach
  - On one of the two sides





Dean (NetId , Name, Surname, UniversityName , DateOfAppointment)

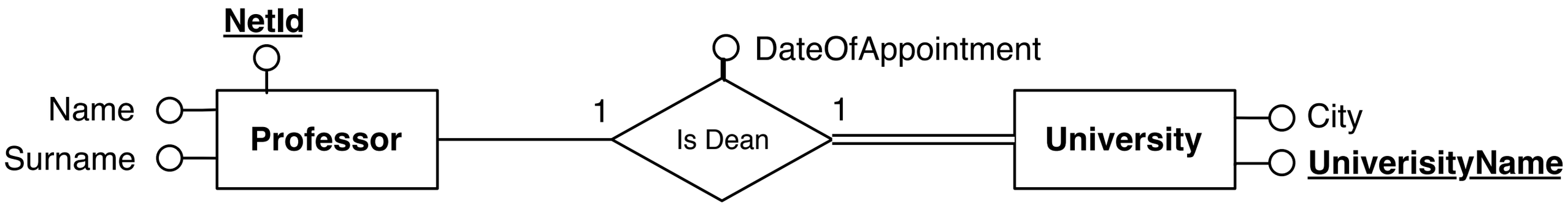
University (UniversityName , City)

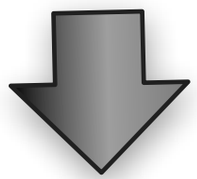
Dean (NetId , Name, Surname)

University (UniversityName , City, NetId , DateOfAppointment)

Example `One To One /2`

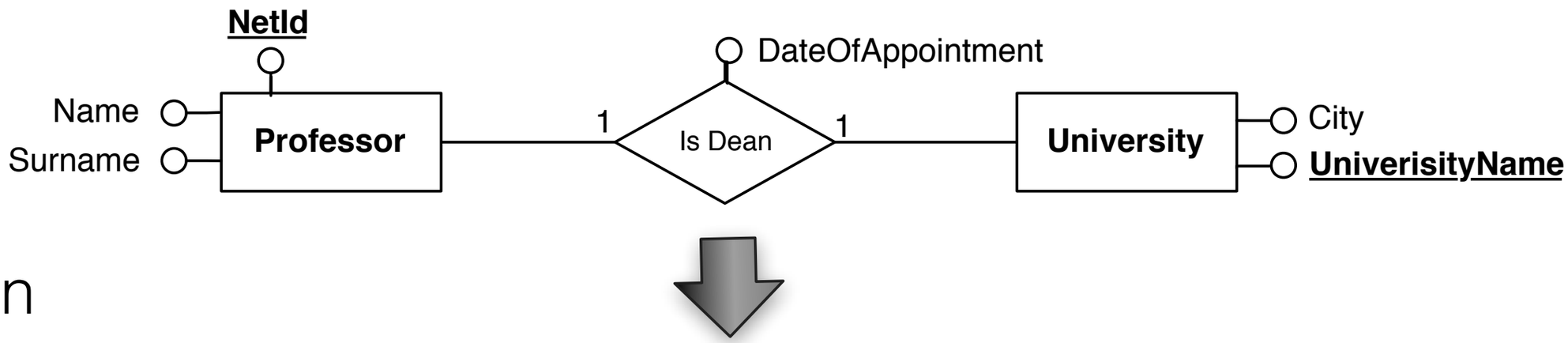
- Total participation on one role: **Foreign key** approach





```
Professor ( NetId , Name, Surname )

University ( UniversityName , City,  NetId , DateOfAppointment)
```

Example One To One /3



Optional participation on both roles

- **Foreign key** approach
- **Relationship relation** approach

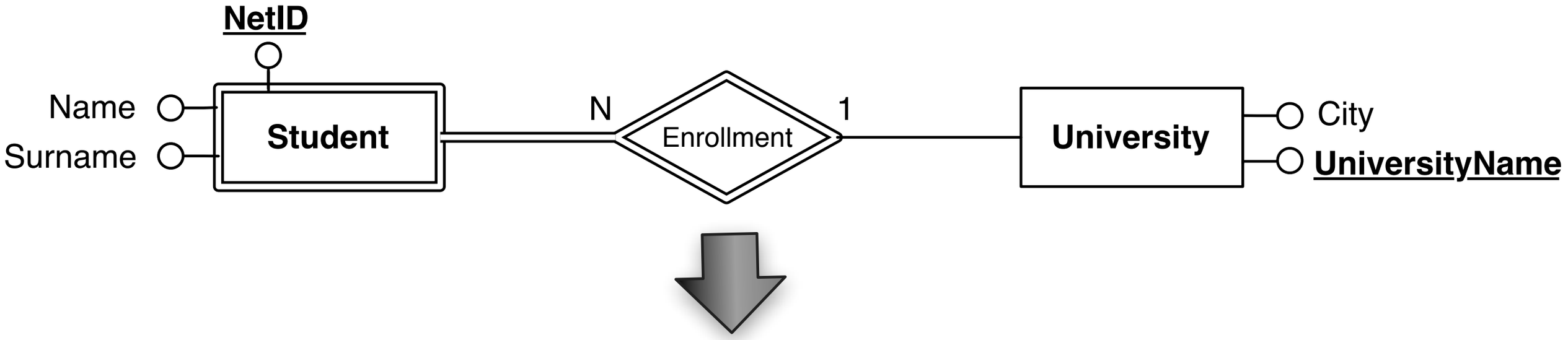
```
Professor ( NetId , Name, Surname )

Dean ( UniversityName ,  NetId , DateOfAppointment)

University ( UniversityName , City )
```

```
Professor ( NetId , Name, Surname )

Dean ( UniversityName , NetId , DateOfAppointment)

University ( UniversityName , City )
```

```
Professor ( NetId , Name, Surname )

University ( UniversityName , City, NetId* , DateOfAppointment* )
```

Entity With External Identifier

- Relationship is represented with the identifier



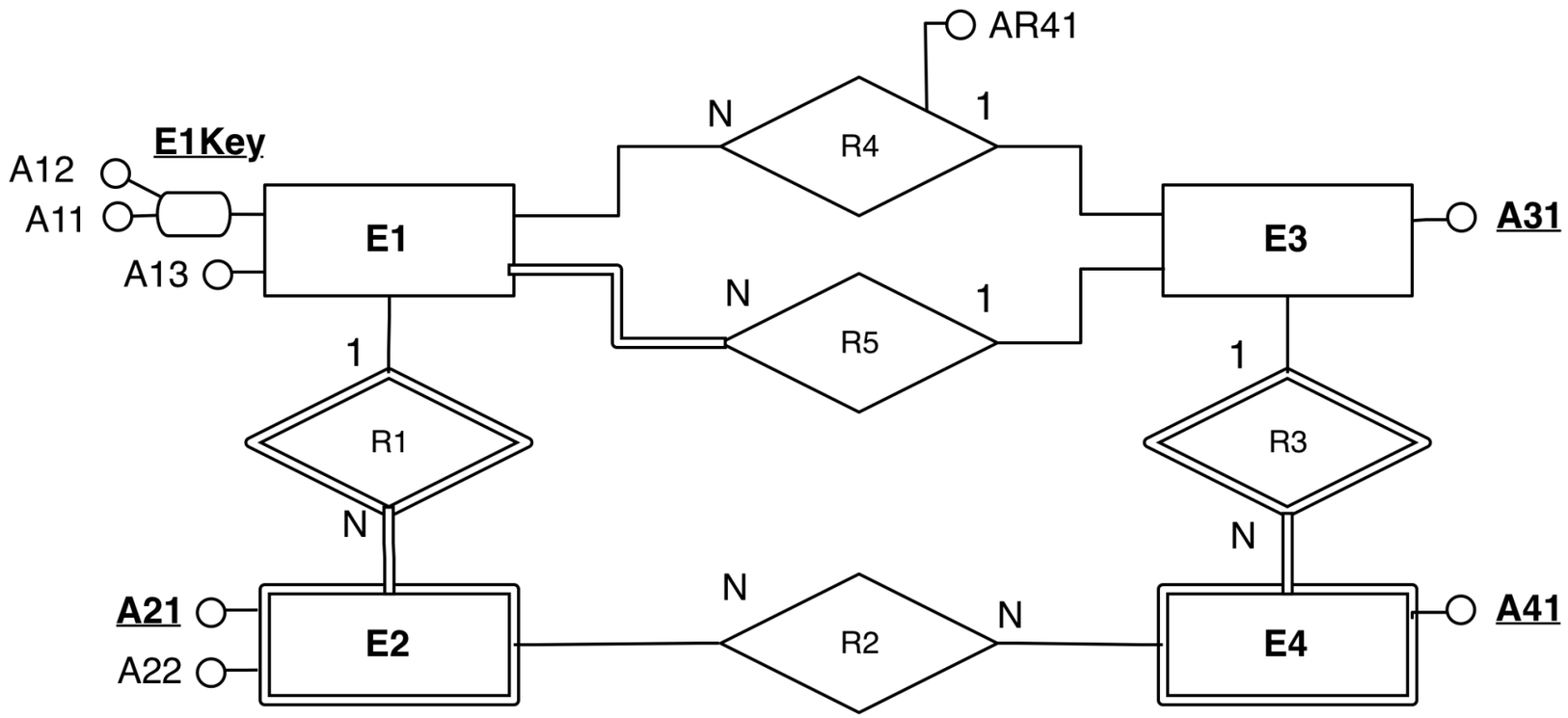
University (UniversityName , City)

Student (NetId , UniversityName , Name, Surname)

ASQ Exercise (1 min)

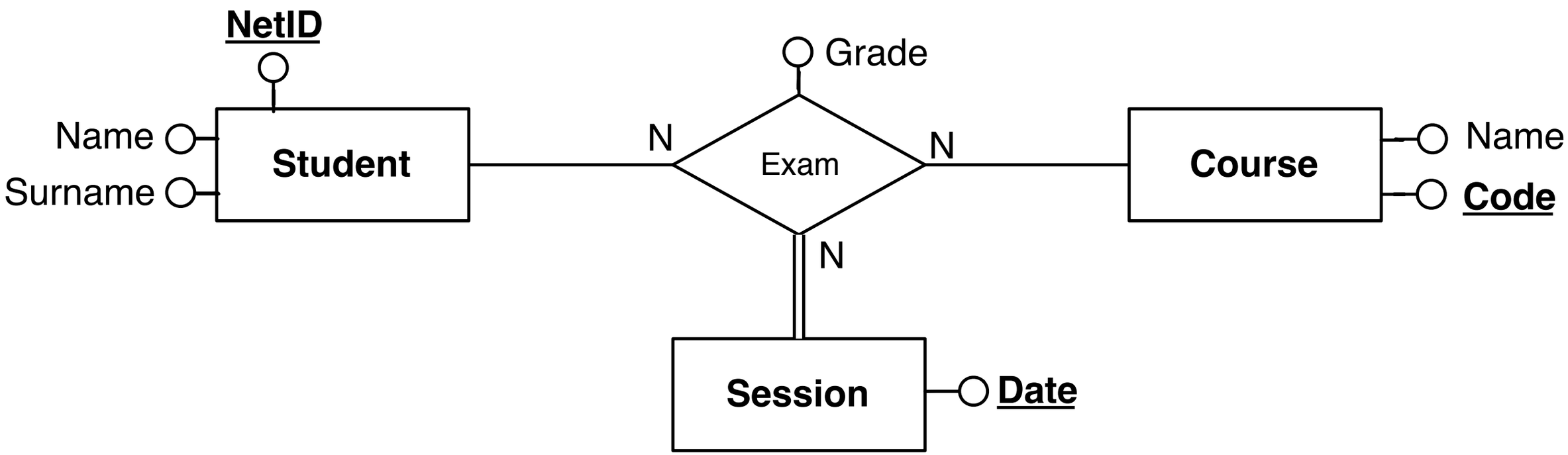
Consider the EER diagram in figure.

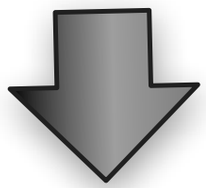
Which is the minimum number of tables resulting from the transformation of the EER schema in Figure 1 into a logical schema?



1 5 2 6 3 7 4 8

Ternary Relationships





```
Student ( NetId , Name, Surname)

Exam ( NetId , CourseName , Date , Grade )

Course ( CourseName , Name )

Session ( Date )
```

Readings

- **Database Book**
  - *Chapter **9** of the Database book*

End of Lecture