Politecnico di Torino

DBMG

Database design

Database design

- Entity-Relationship Model
- Conceptual design
- Logical design
- Normalization

Entity-Relationship Model

Database design

Entity-Relationship Model

- Life cycle of an information system
- Database design
- Entities and Relationships
- Attributes
- Identifiers
- Generalization
- Documenting E-R Schematics
- UML and E-R

Life cycle of an information system

Database design

Database design

- The design of a database is one of the activities of the process of developing an information system
  - must be seen in the broader context of the life cycle of an information system

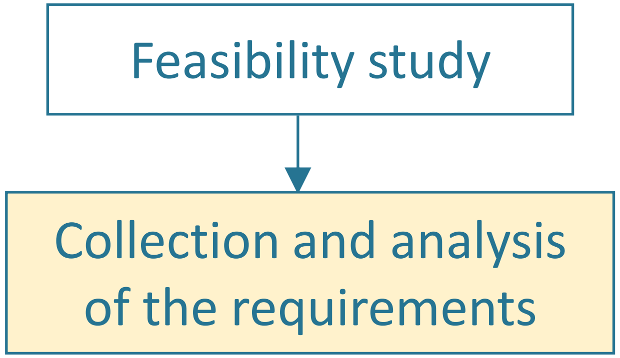
Life cycle of an information system

Determination of the costs of the different alternatives and the priorities for the implementation of each system component

Feasibility study

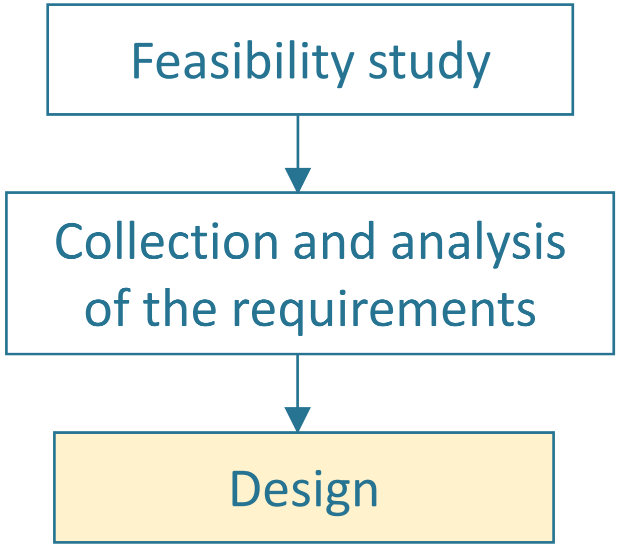
Life cycle of an information system

- Definition of the properties and functionalities of the information system
- Requires user interaction
- Produces a comprehensive, but informal, description of the system to be implemented



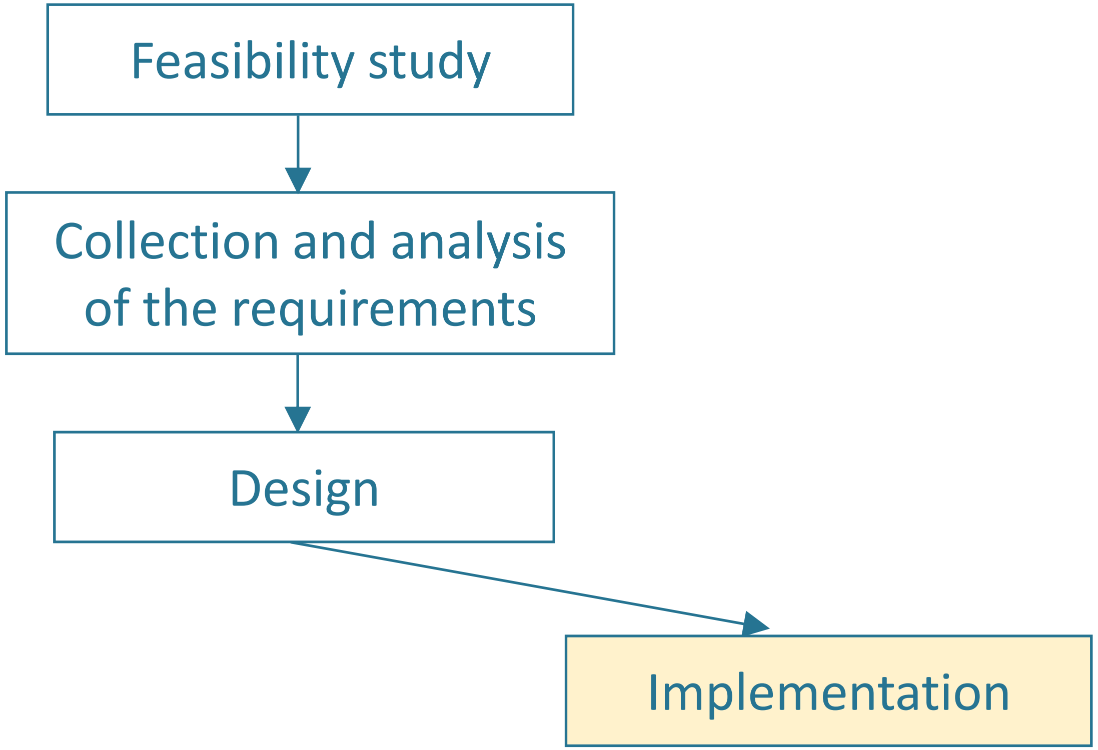
Life cycle of an information system

- Divided into data and application design
- Produces formal descriptions



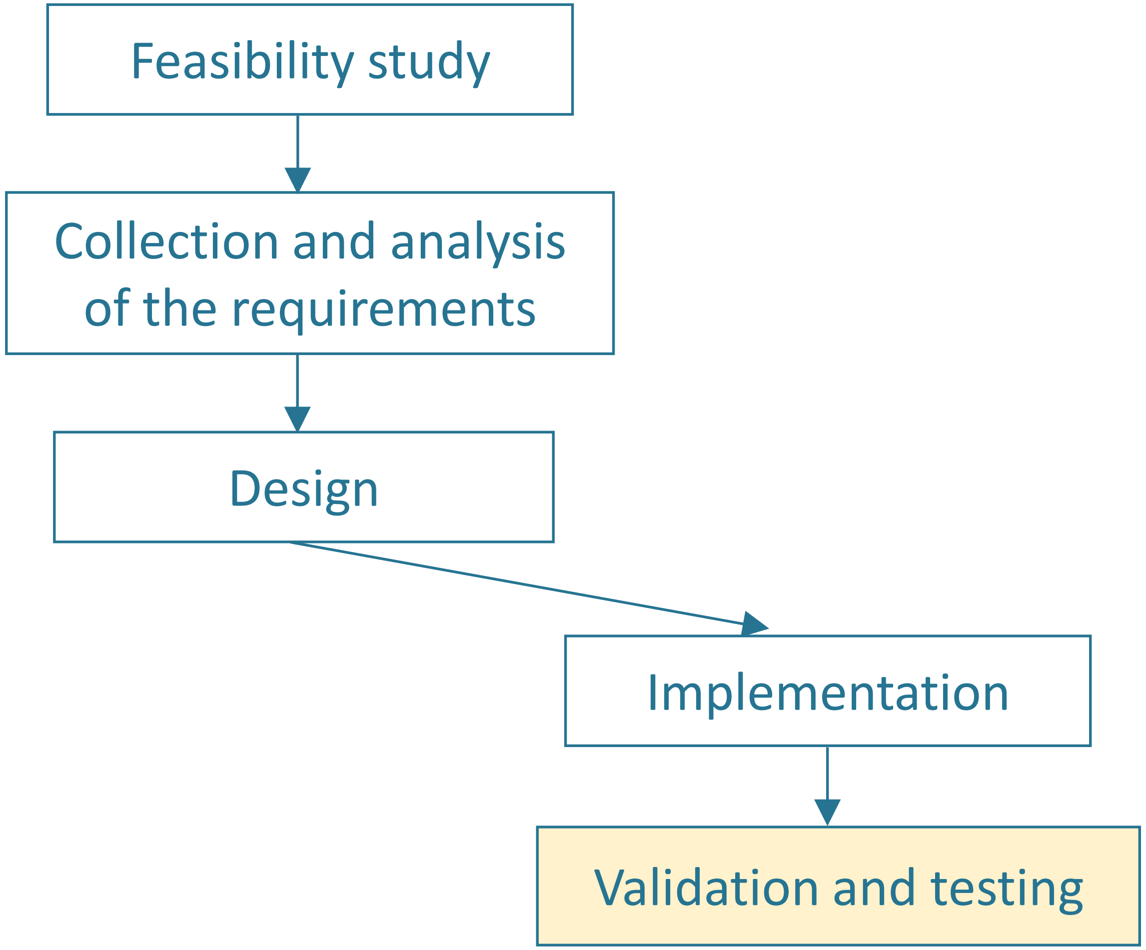
Life cycle of an information system

- Implementation of the information system according to the characteristics defined in the design phase



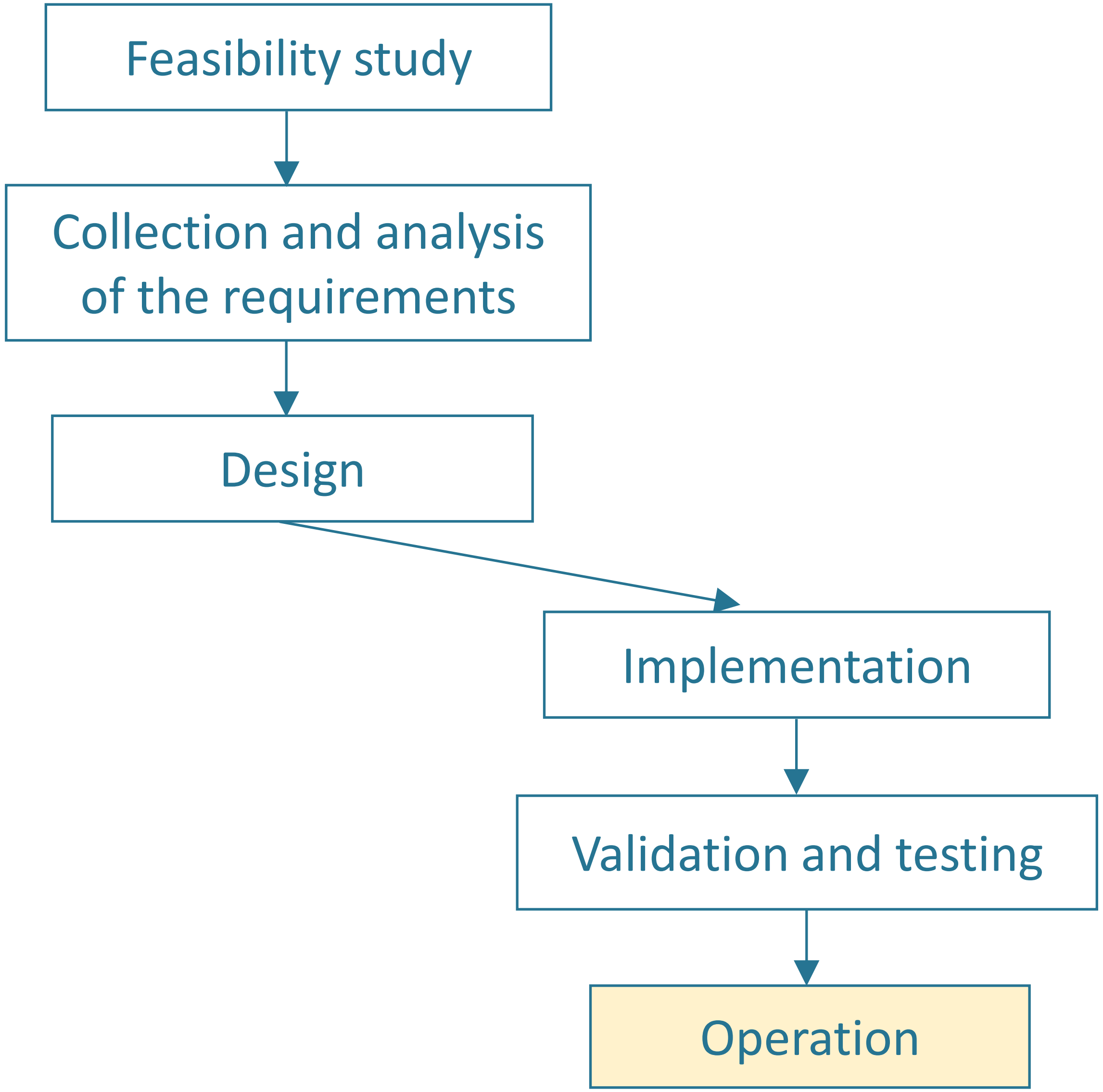
Life cycle of an information system

- Verification of the correct functioning and quality of the information system
- It can lead to changes in requirements or design revision



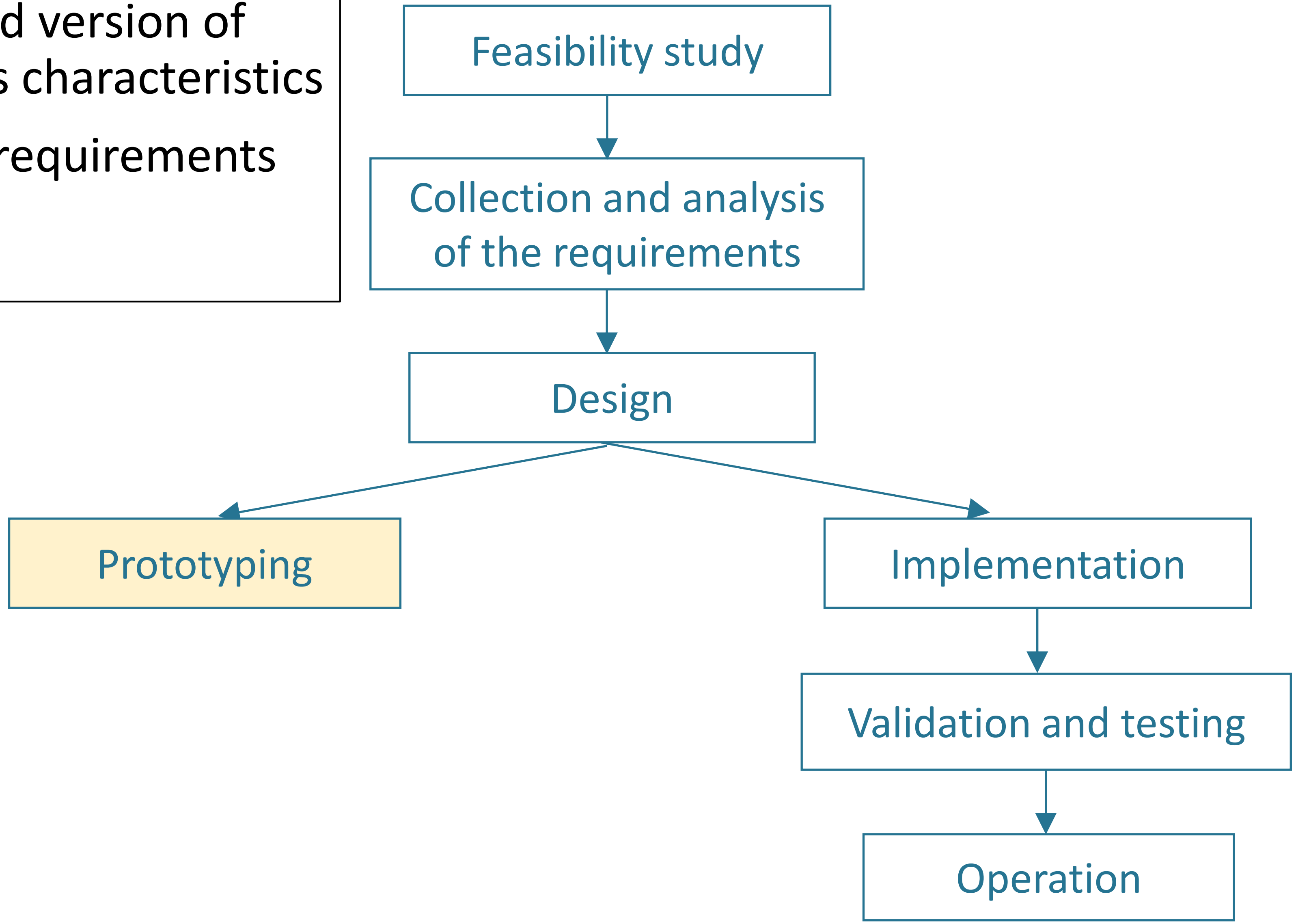
Life cycle of an information system

- System operation
- Requires maintenance and managing operations

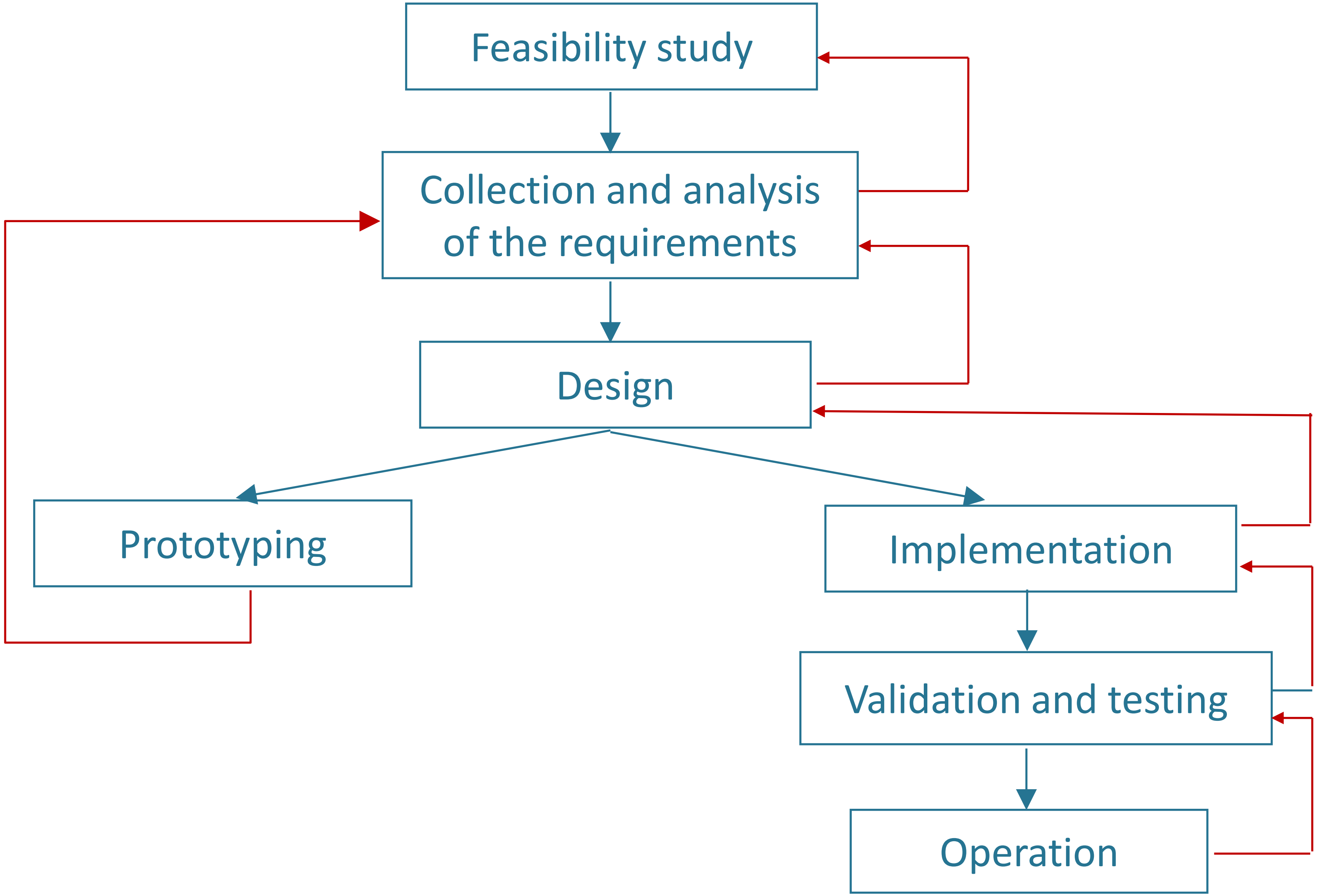


Life cycle of an information system

- Quickly create a simplified version of the system to evaluate its characteristics
- It can lead to changes in requirements or design revision



Life cycle of an information system



Database design

Database design

Database design

- The database is an important component of the entire system
- Data-driven design methodology
  - the design of the database precedes that of the applications that use it
  - greater attention to the design phase than to the other phases

Design Methodology

- A design methodology consists of
  - decomposition of the project activity into successive and independent phases
  - strategies to be followed in the various phases and criteria for choosing the best strategy
  - reference models to describe the input and output data of the various phases

Properties of the methodology

Generality

- can be used regardless of the problem and the tools available

Quality of the result

- in terms of correctness, completeness and efficiency with respect to the resources used

Ease of use

- of both strategies and reference models

Data-driven design

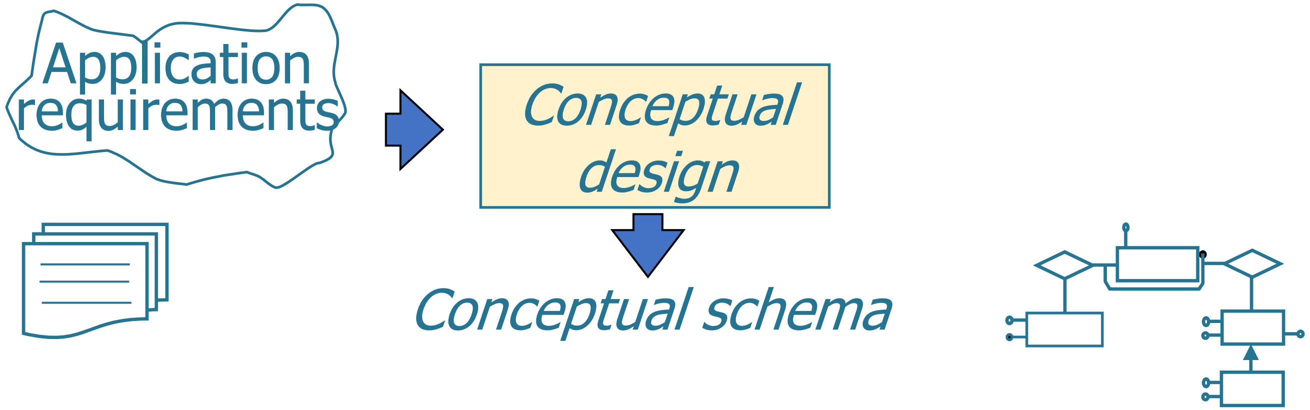
- For databases, methodology based on separating two key decisions
  - *what* to represent in the database
    - conceptual design
  - *how* to represent it
    - logical and physical design

Stages of database design

Informal specification of the reality of interest

- Application properties
- Application functionalities

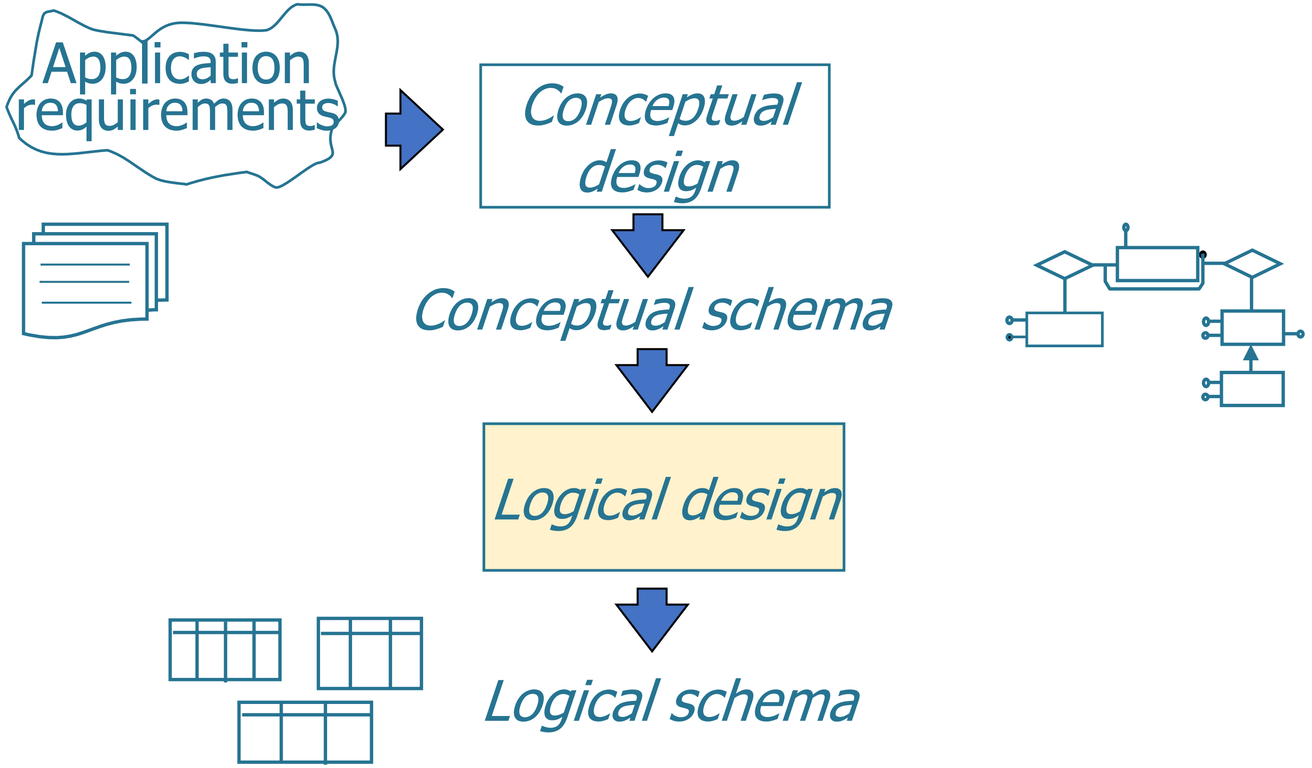
Stages of database design



Representation of informal specifications in the form of a conceptual diagram

- formal and complete description, referring to a conceptual model
- Independent from implementation aspects (data model)
- the aim is to represent the information content of the database

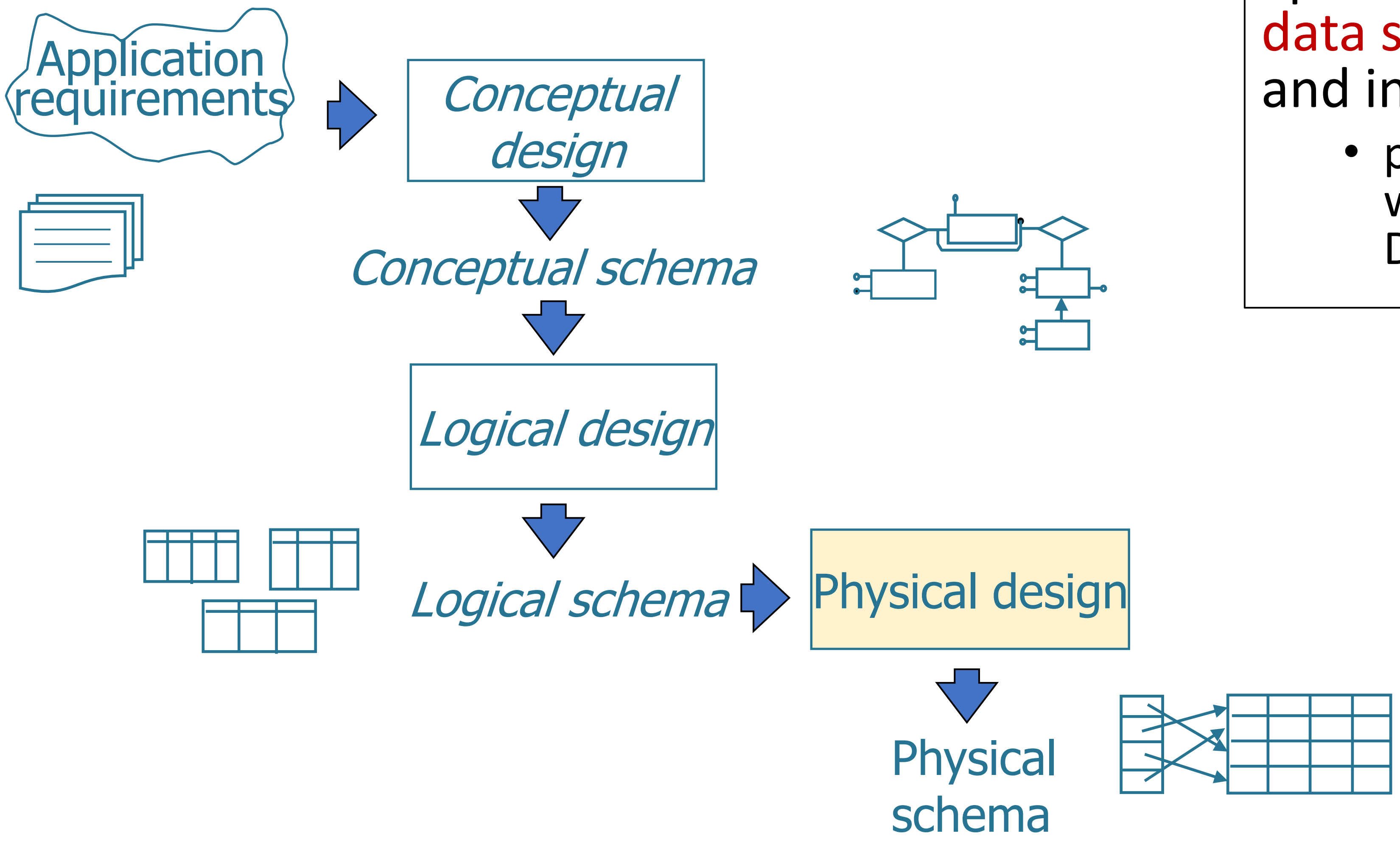
Stages of database design



Translating the conceptual schema into the logical schema

- depends on the chosen data logic model
- takes into account the optimization of data processing operation
- schema quality verified by formal techniques (normalization)

Stages of database design



Specification of the physical data storage parameters (File and index organization)

- produces a physical model, which depends on the chosen DBMS

Entity-Relationship Model

Database design

The E-R (Entity-Relationship) model

- It is the most widely used conceptual model
- Provides constructs to describe specifications about data structure
  - in a simple and understandable way
  - with a graphic formalism
  - independent of the data model, which can be chosen later
- Several variants are available

Main elements of the E-R model

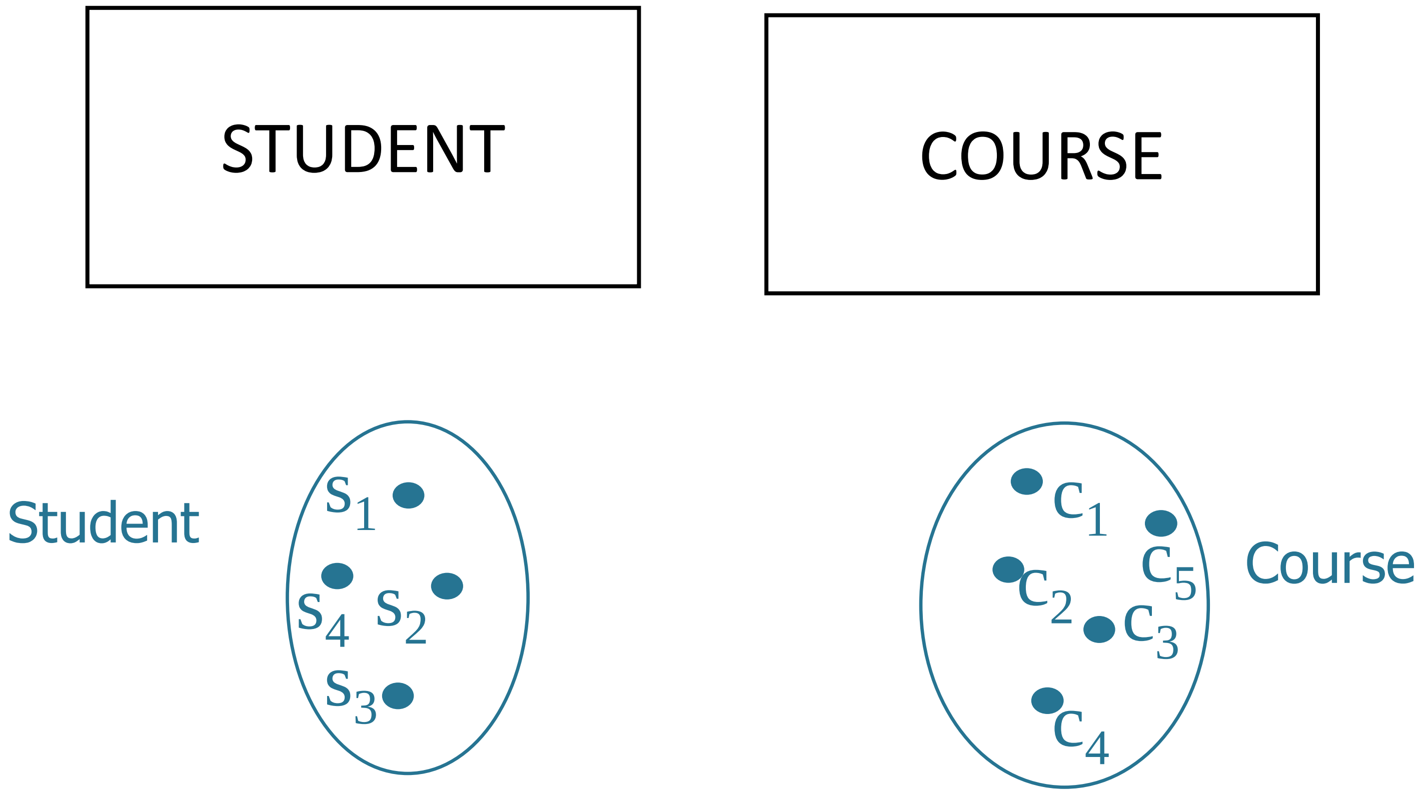
- Entity
- Relations
- Attributes
- Identifiers
- Generalizations and subsets

Entity

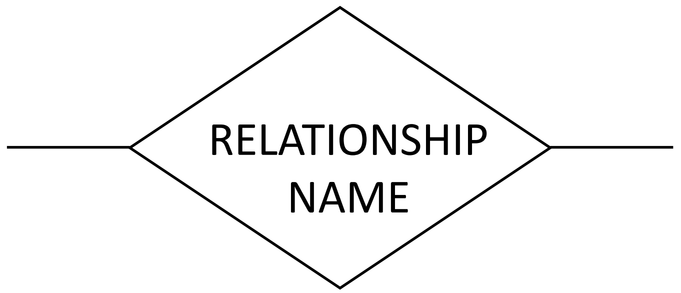
Entity name

- It represents classes of real-world objects (people, things, events, ...), which have
  - common Properties
  - autonomous existence
- Examples: Employee, Student, Article
- An occurrence of an entity is an object of the class that the entity represents

Example of entities

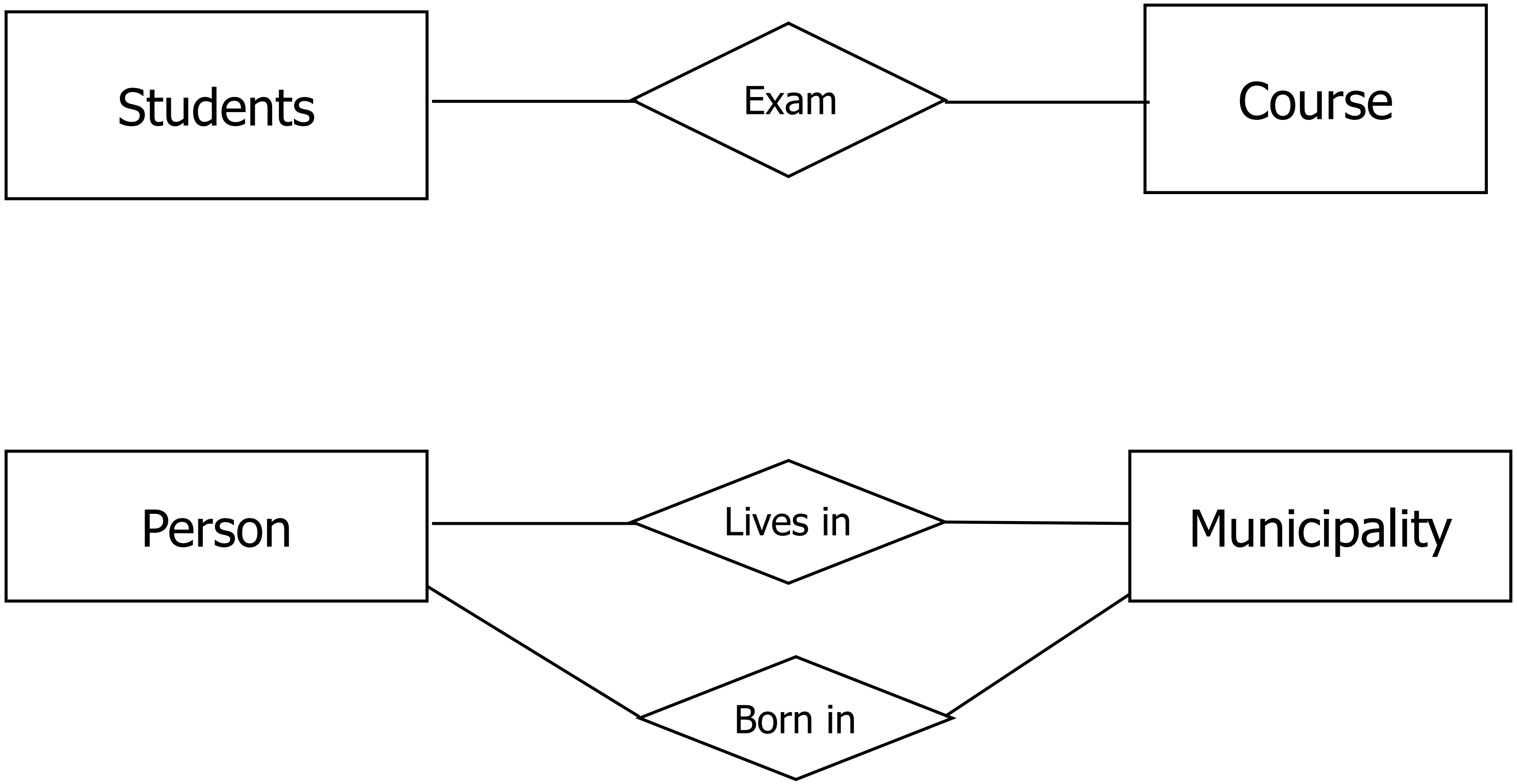


Relationship



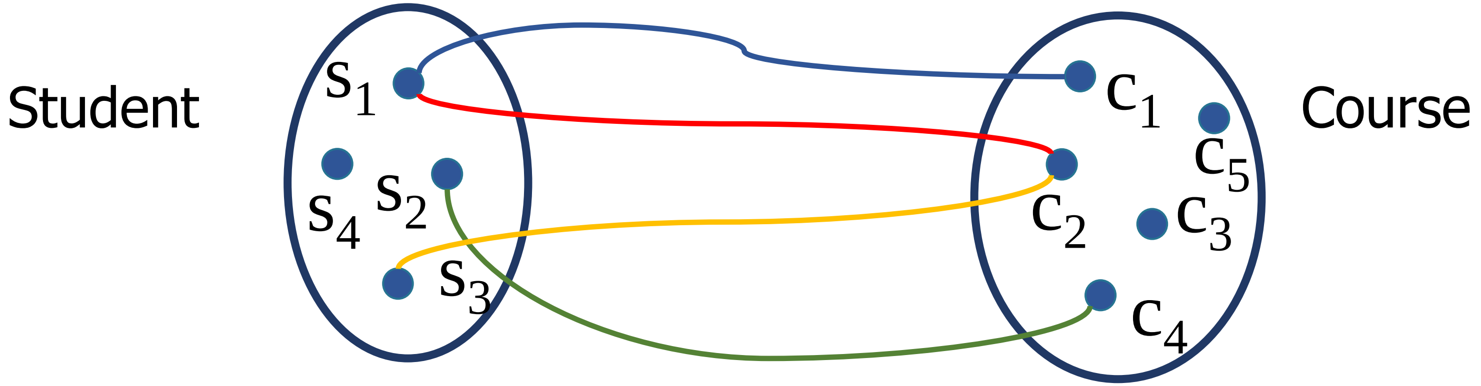
- Represents a logical link between two or more entities
- Examples: exam between student and course, residence between person and municipality
- Not to be confused with the relation of the relational model
    - sometimes referred to as association

Relationship examples



Occurrences of a relationship

- An occurrence of a relationship is an n-tuple (pair in the case of a binary relationship) consisting of occurrences of entities, one for each of the entities involved
- There can be no identical n-tuples

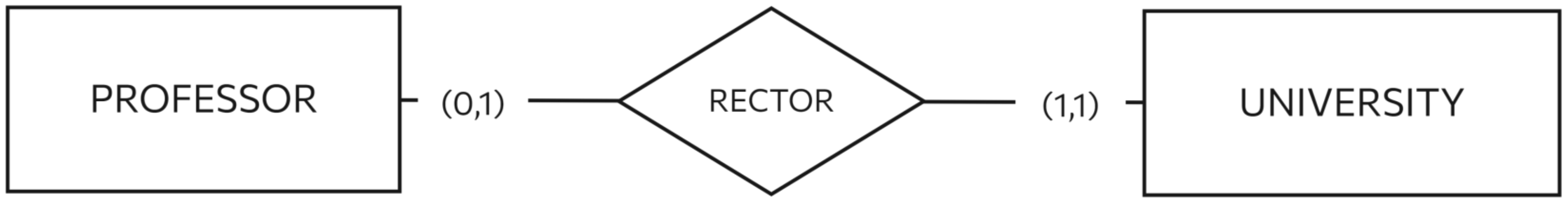


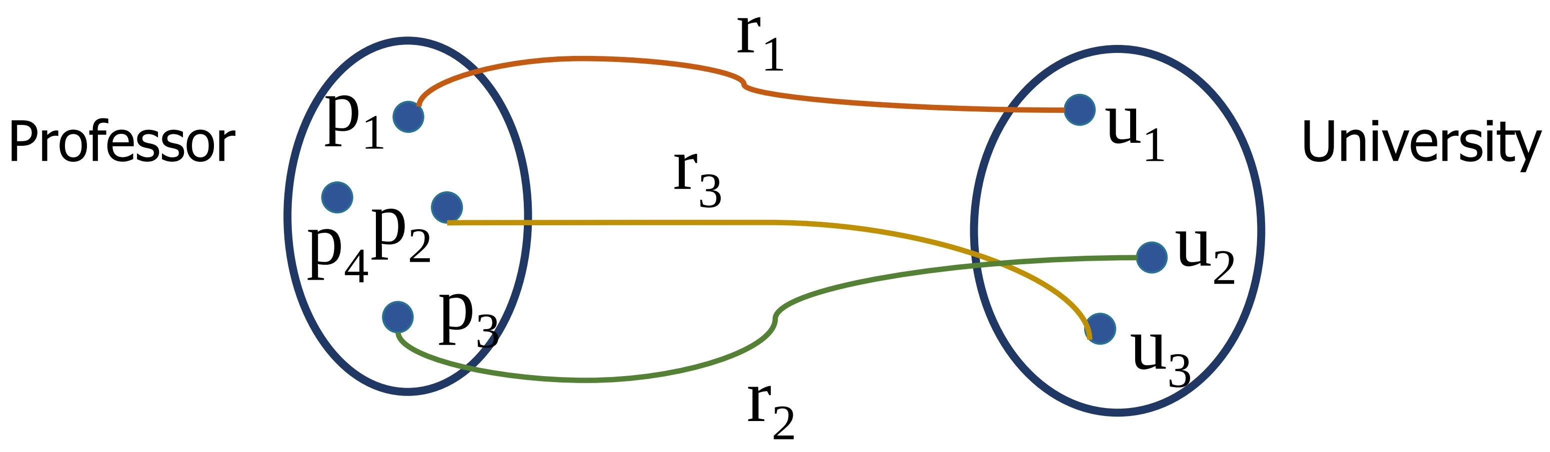
Cardinality of binary relationships

- They are specified for each entity that participates in a relationship
- Describe the minimum and maximum number of occurrences of a relationship in which an occurrence of an entity can participate
  - minimum can be either
    - 0 (optional participation)
    - 1 (participation required)
  - maximum varies between
    - 1 (at most one occurrence)
    - N (arbitrary number of occurrences)

Cardinality of binary relationships

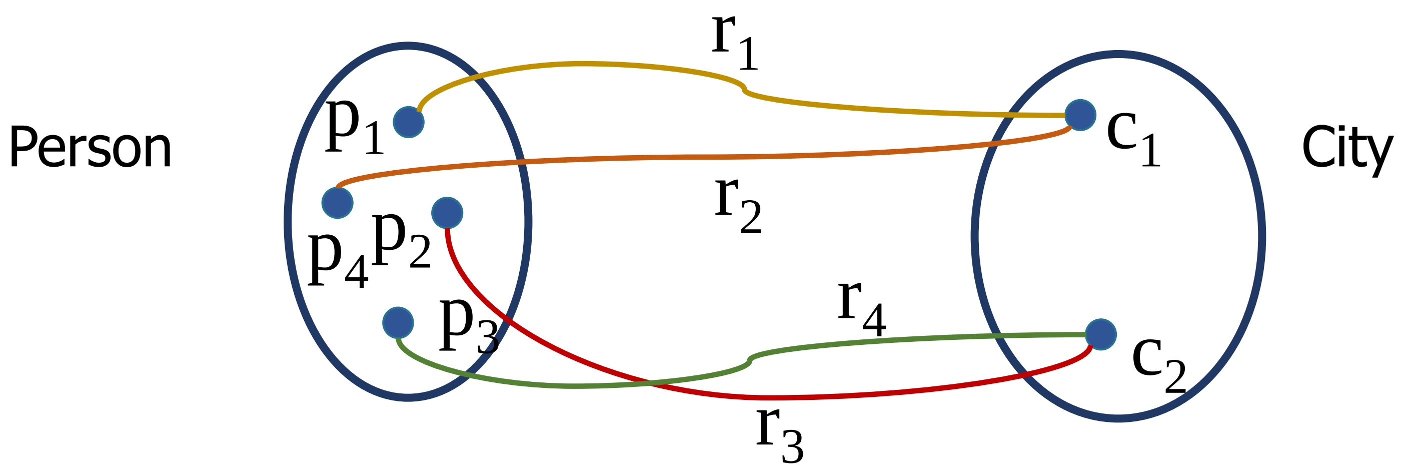
- 1-to-1 relationship





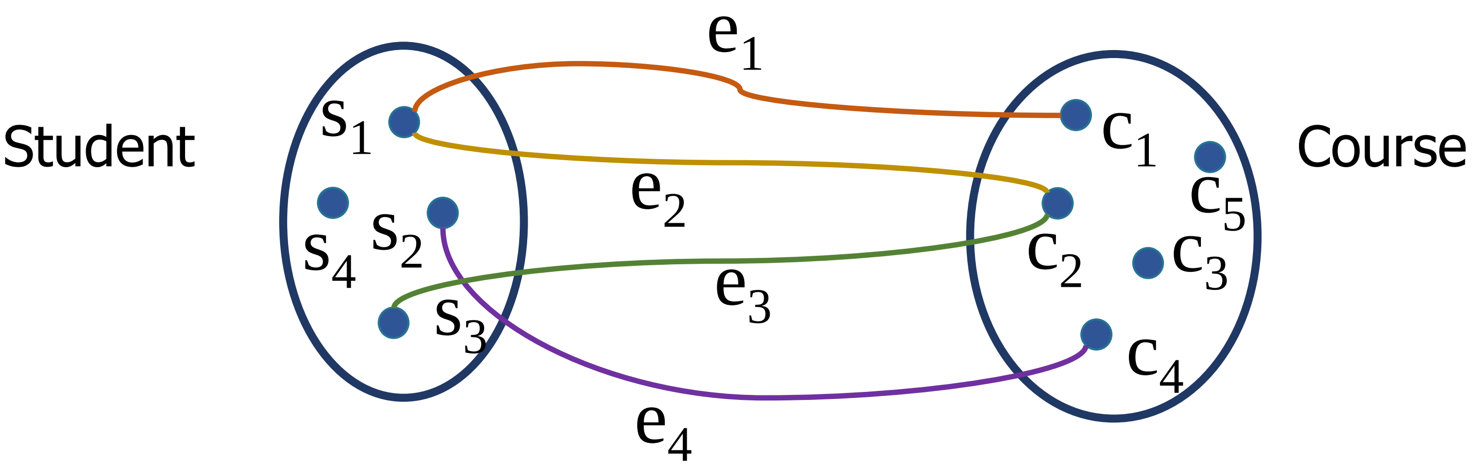
Cardinality of binary relationships

- 1-to-N (many) relationship

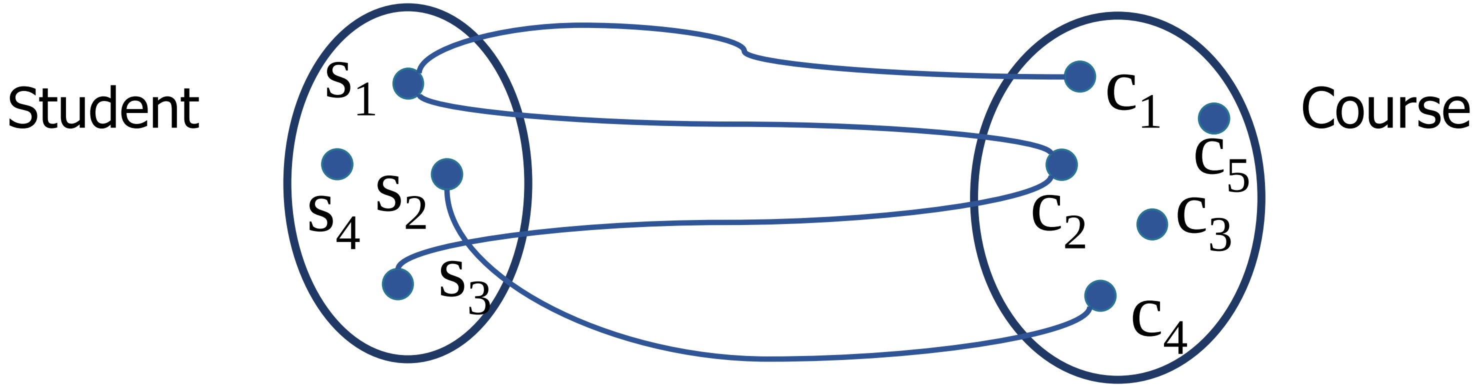


Cardinality of binary relationships

- N-to-N (Many to Many) relationship



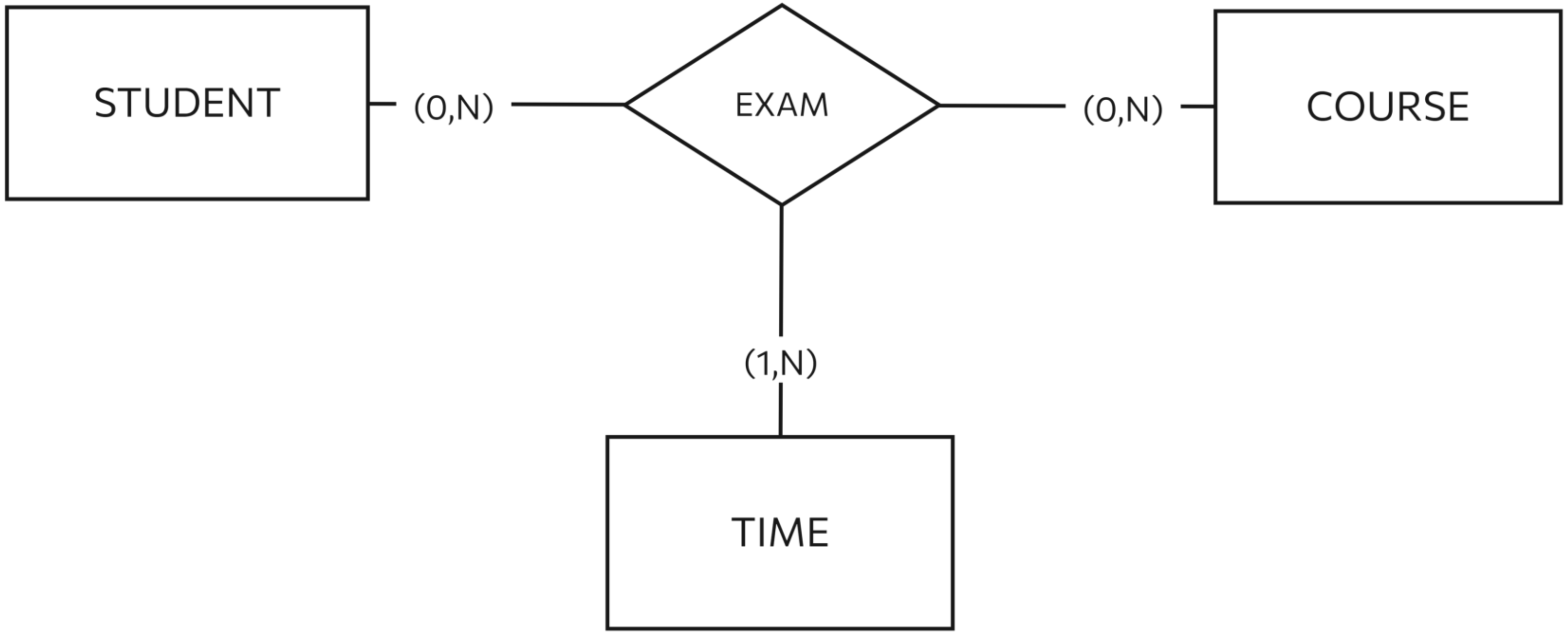
Limitations of binary relationships



- It is not possible for a student to take the same exam more than once

Ternary relationship

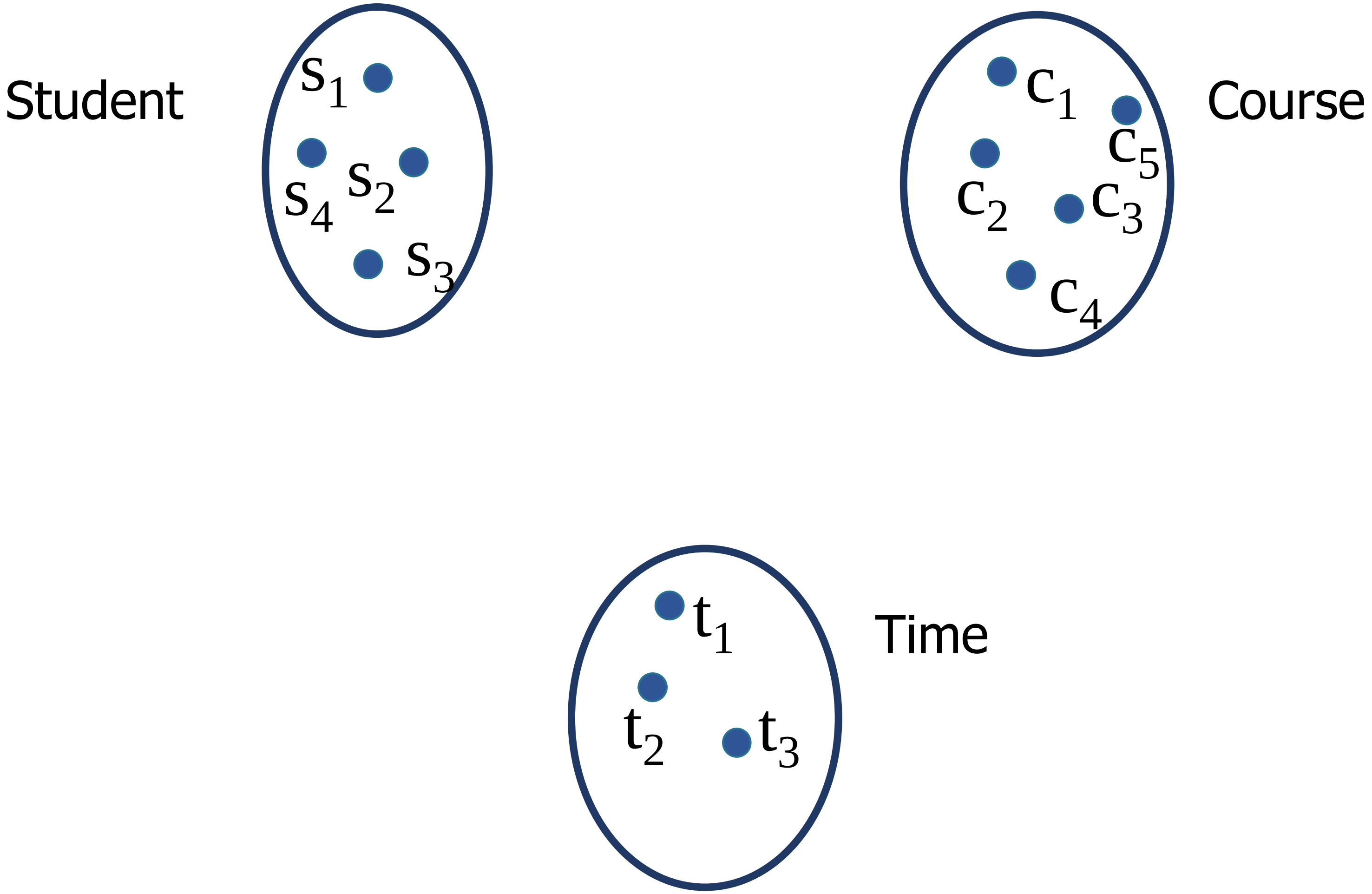
- A student may take the same exam more than once at different times.



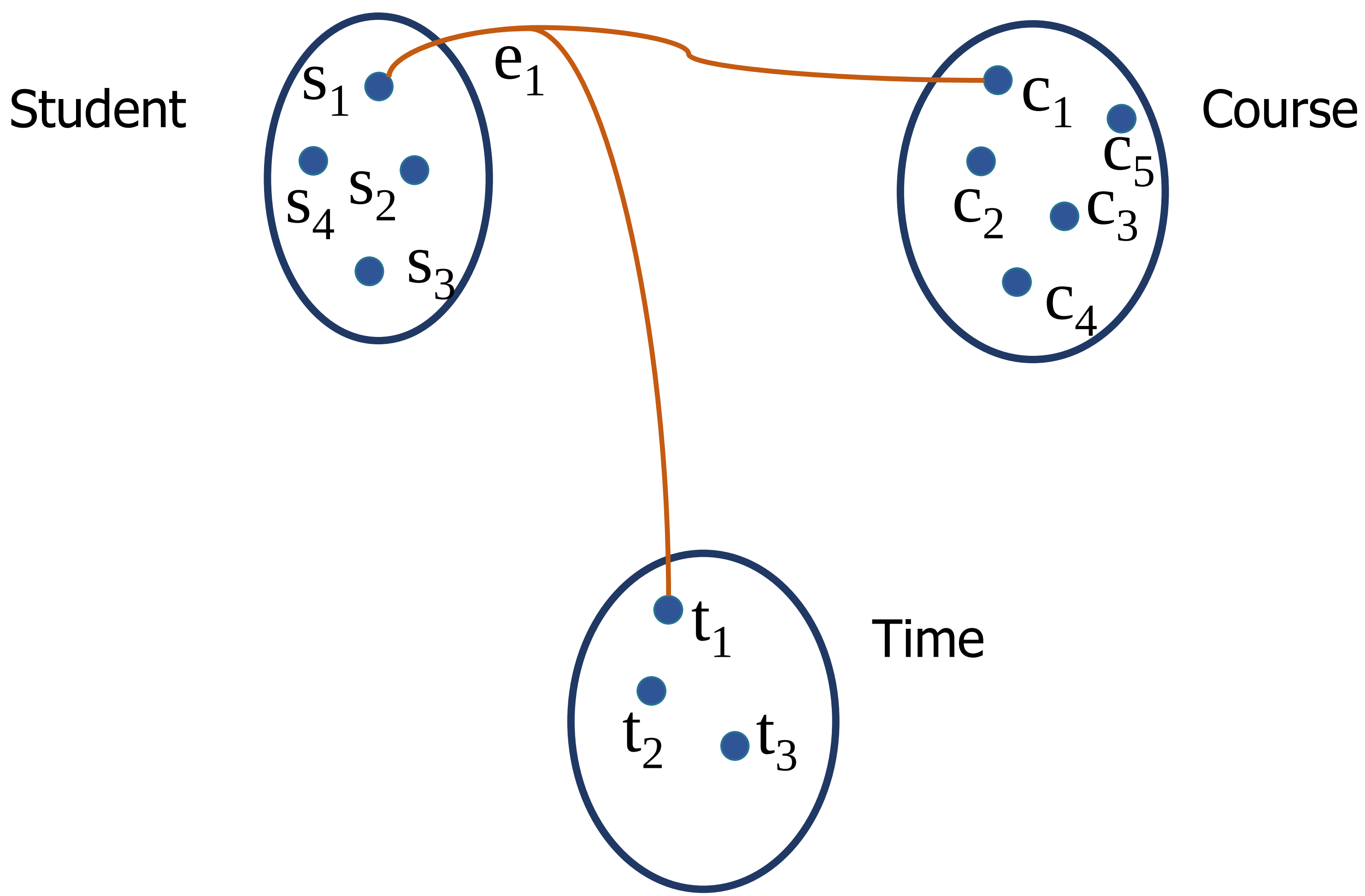
- Example of an instance of the EXAM report

s_1 c_1 t_1
s_1 c_1 t_2
...

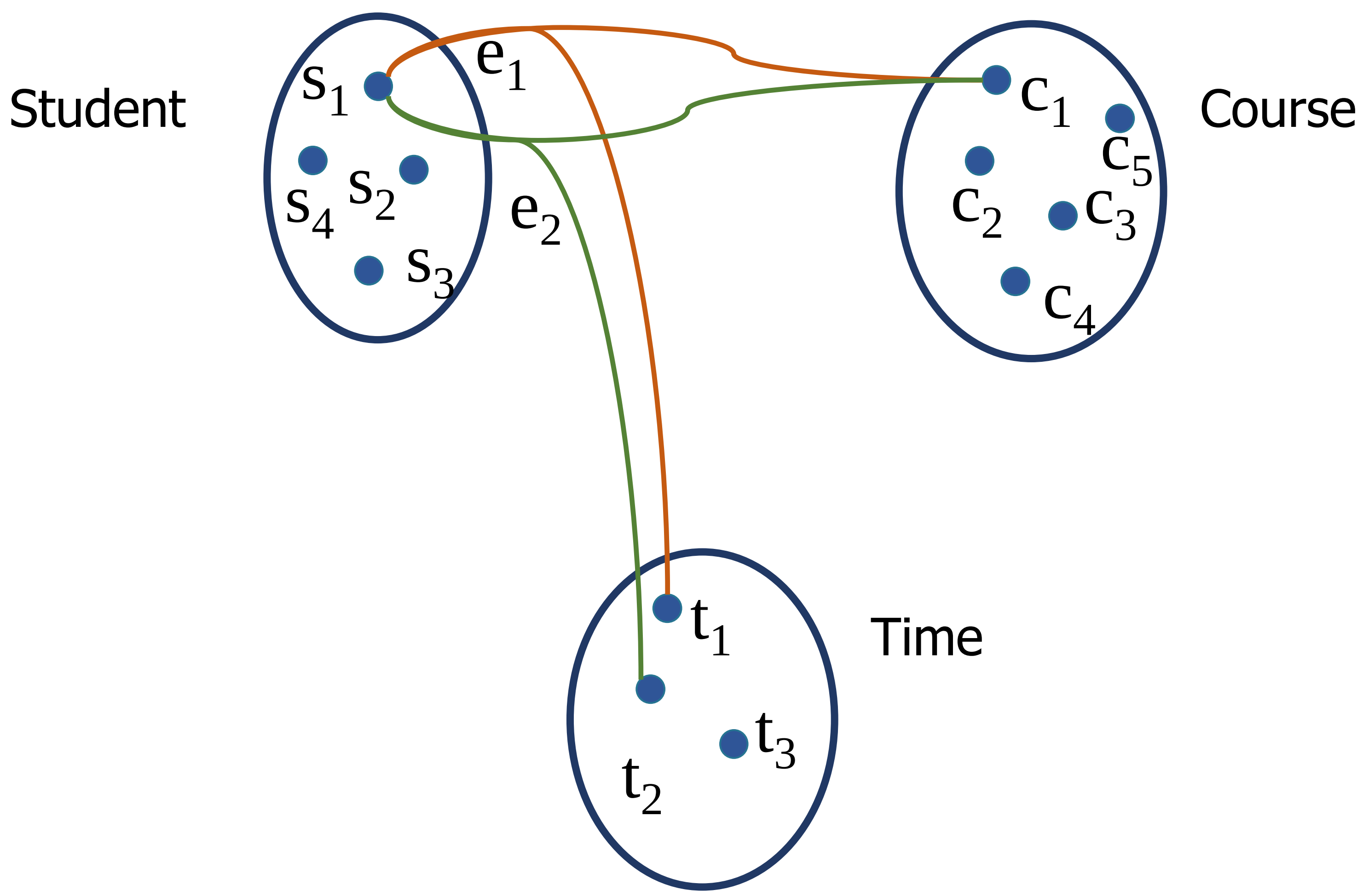
Occurrences of a ternary relationship



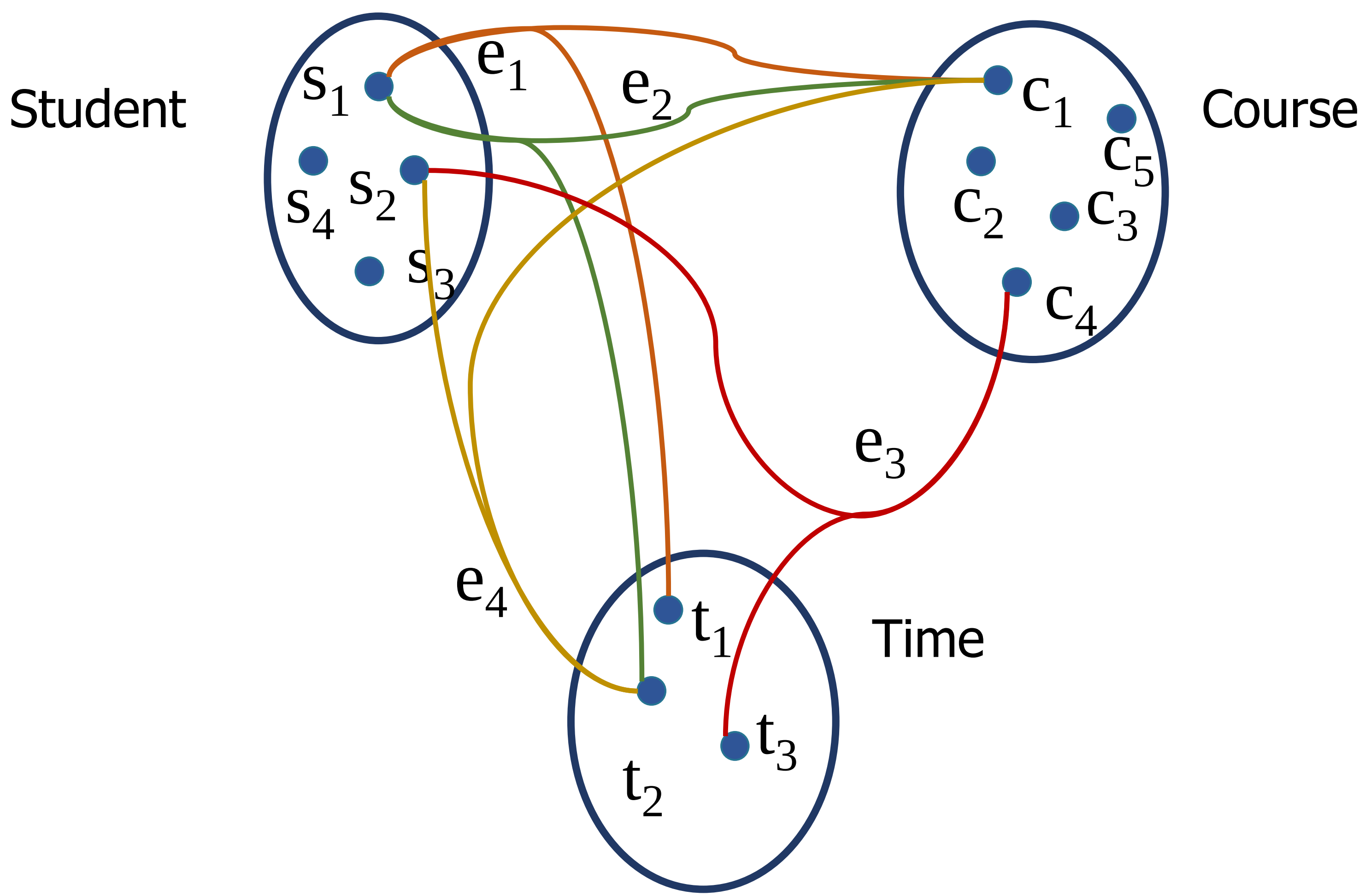
Occurrences of a ternary relationship



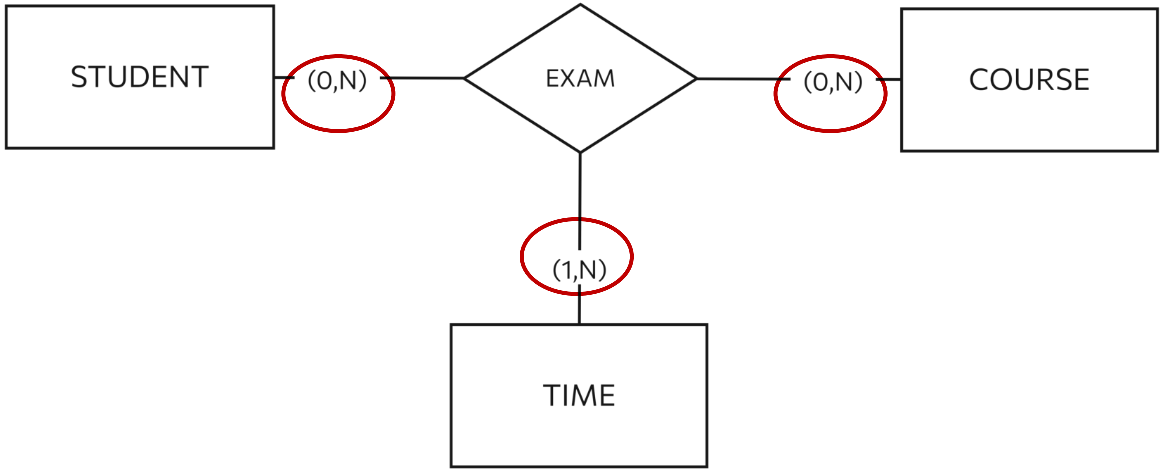
Occurrences of a ternary relationship



Occurrences of a ternary relationship

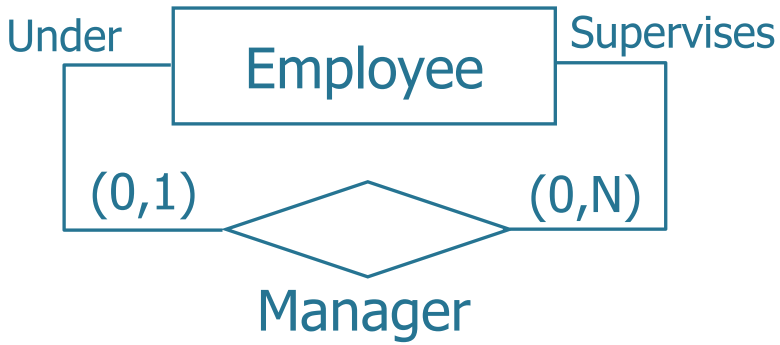


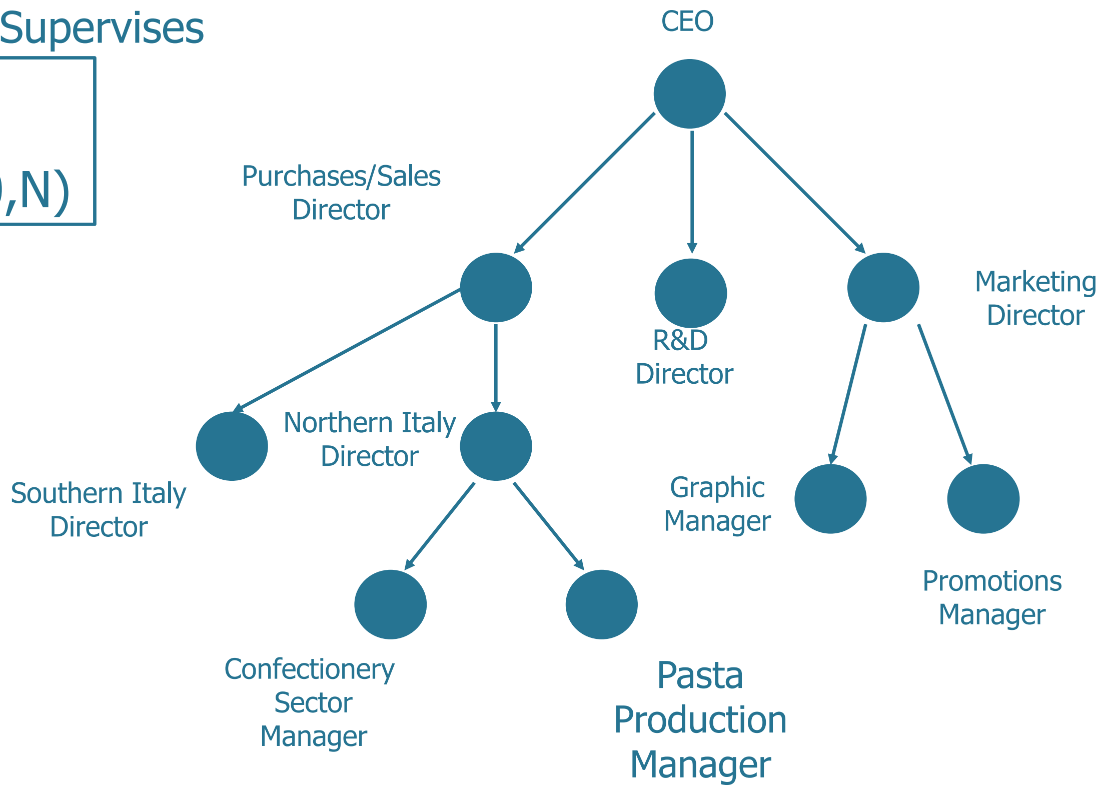
Cardinality of ternary relationships



- Minimum cardinalities are rarely 1 for all entities involved in a relationship
- The maximum cardinalities of an n-ary relationship are (practically) always N
  - if the participation of an entity E has a maximum cardinality of 1, it is possible to eliminate the n-ary relationship and associate the entity E with the others by binary relations

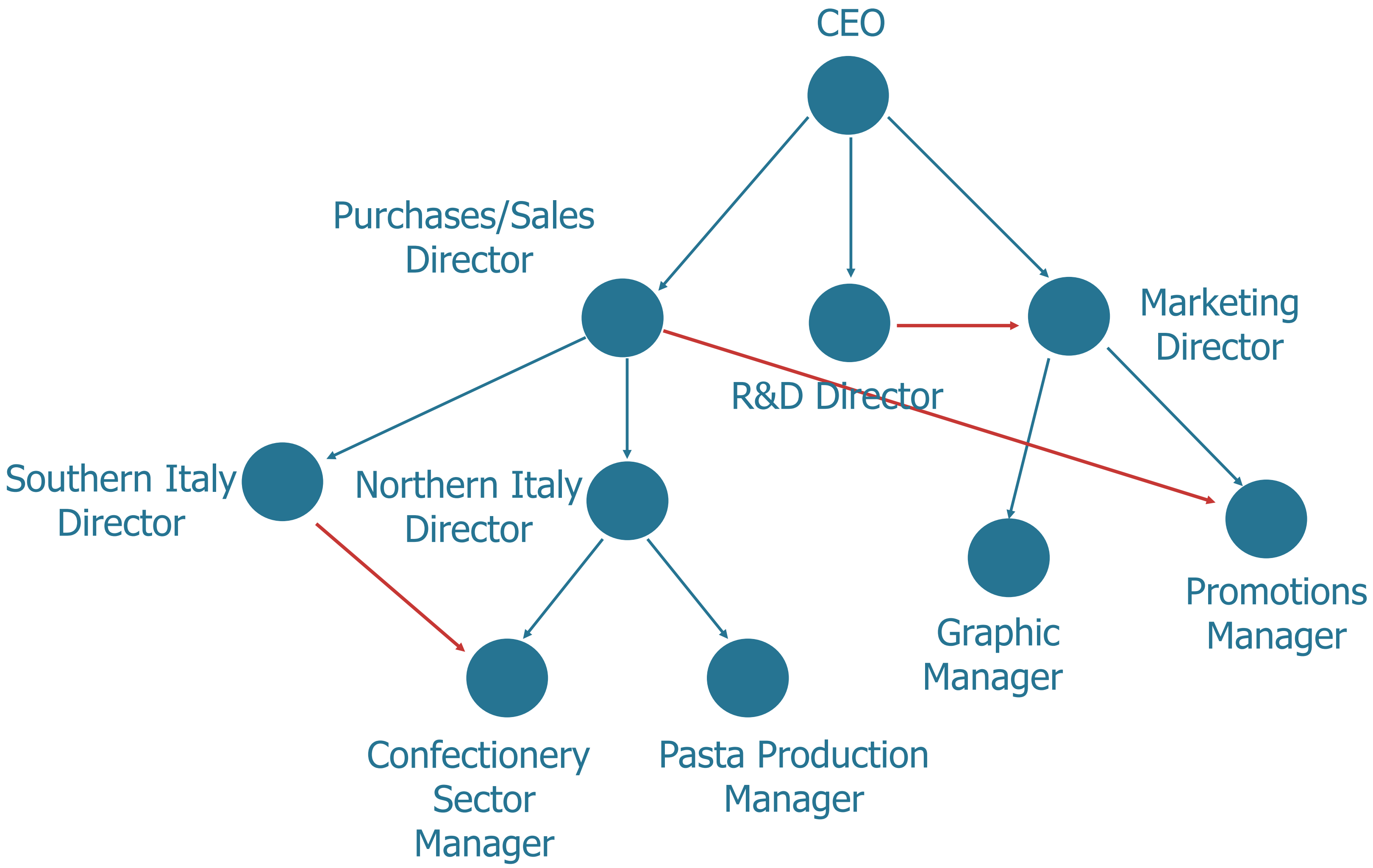
Recursive relationship



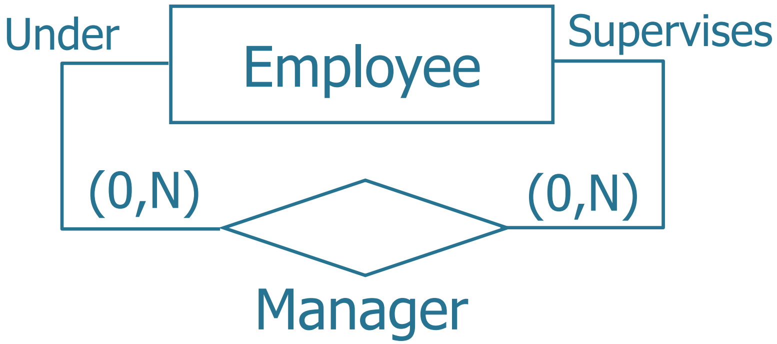


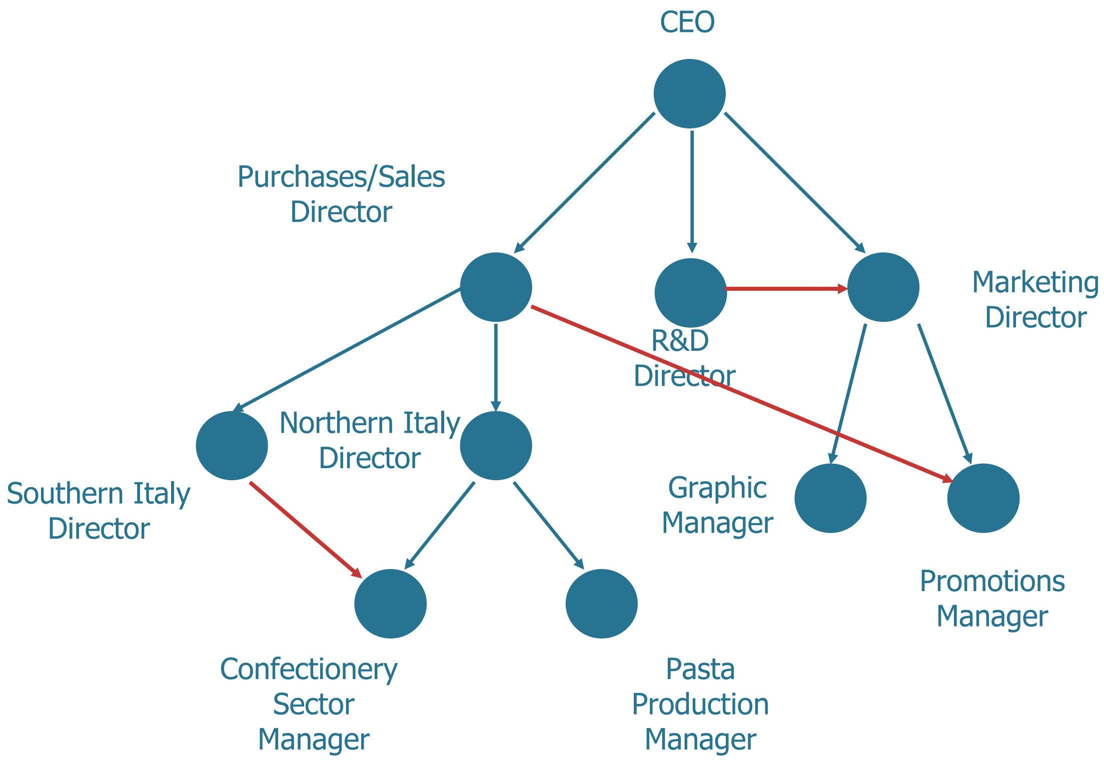
- Relationship between an entity and itself
- If the relationship is not symmetrical, the two roles of the entity must be defined

Recursive relationship



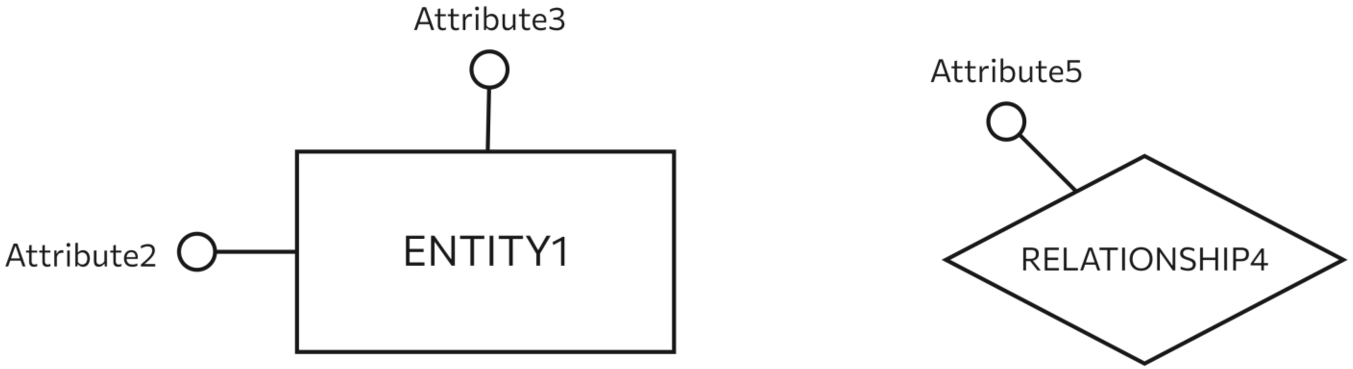
Recursive relationship





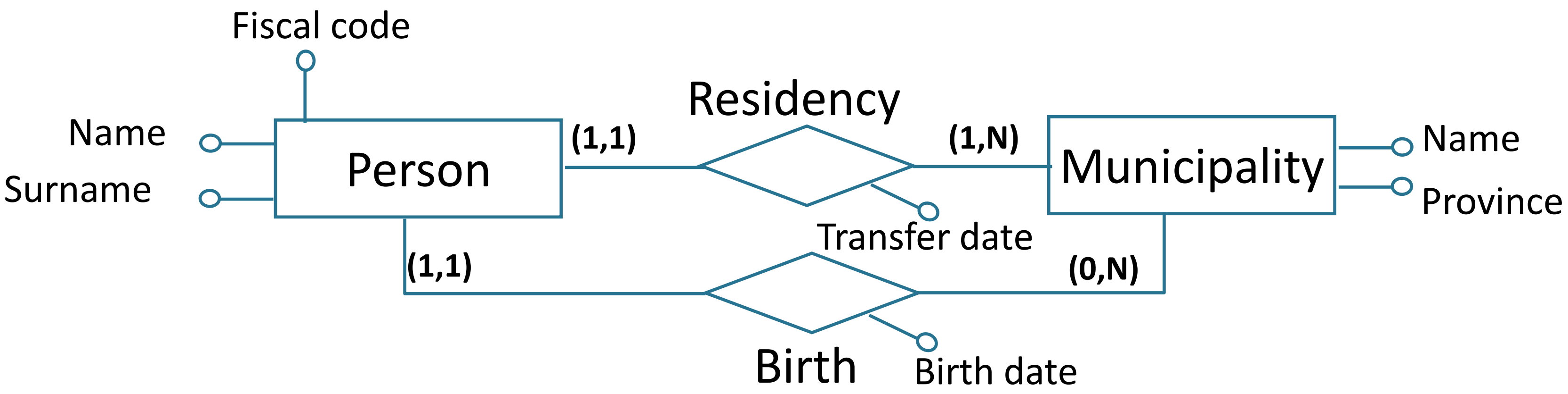
- An employee might have several managers

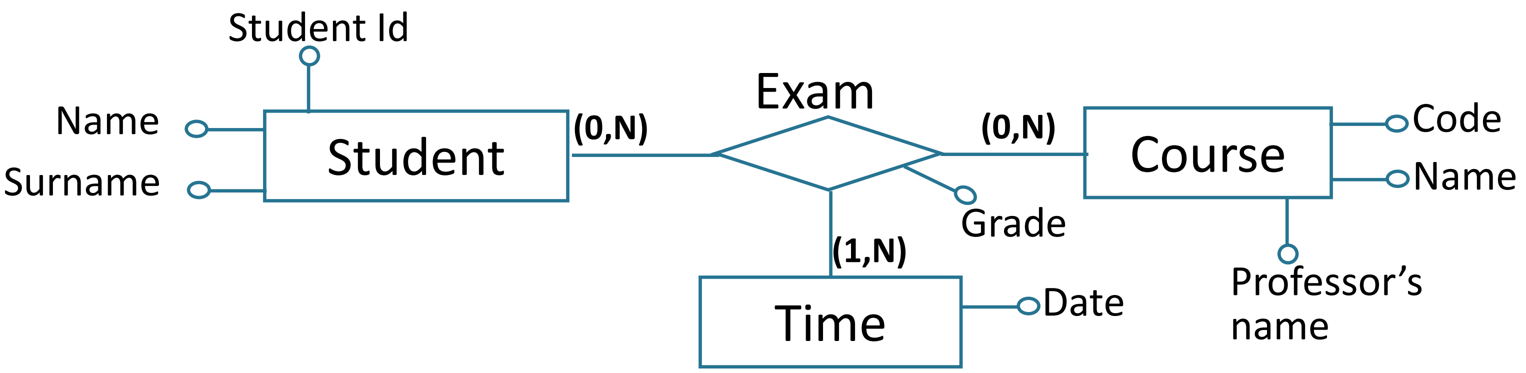
Attribute



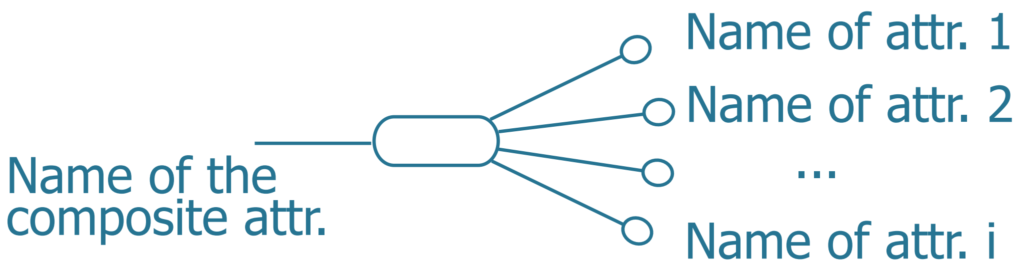
- Describes an elementary property of an entity or relationship
- Examples
  - surname, first name, student ID are attributes that describe the student entity
  - grade is an attribute that describes the exam relationship
- Each attribute is characterized by the domain, the set of admissible values for the attribute

Examples of attributes

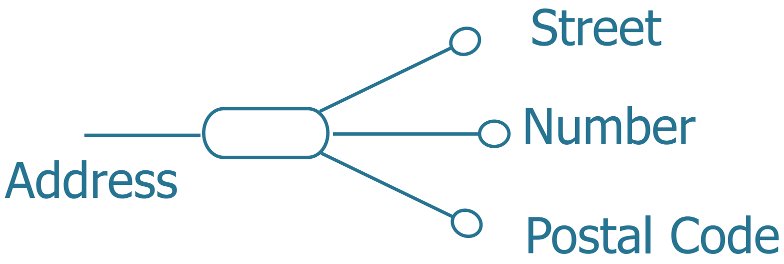




Composite attribute

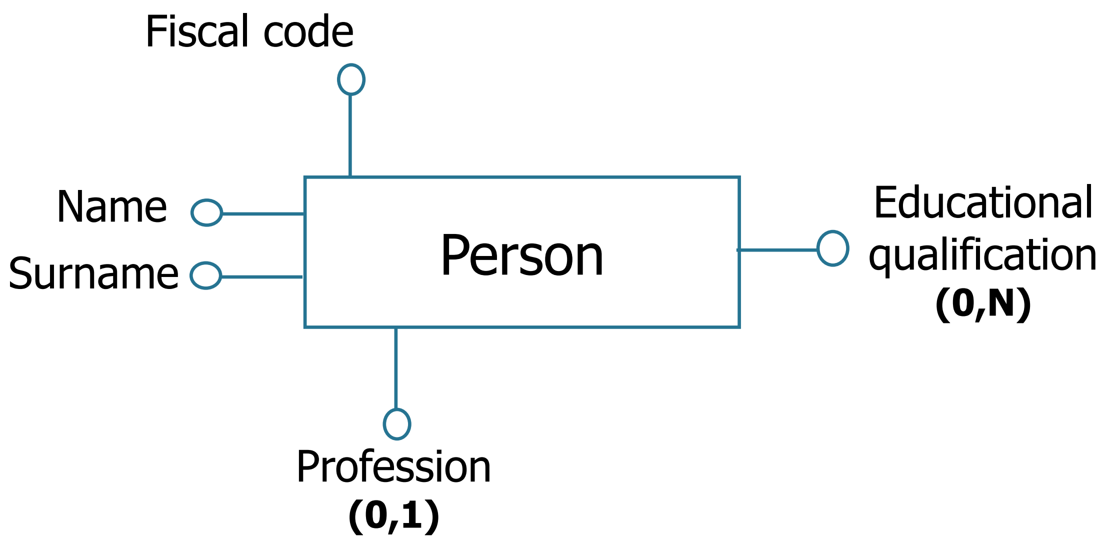


- Group of attributes that have closely connected meanings or uses.
- Example:



Cardinality of an attribute

- Can be specified for entity or relationship attributes
- Describes the minimum and maximum number of attribute values associated with an occurrence of an entity or relationship
  - if it is omitted it corresponds to (1,1)
  - minimum 0 corresponds to an attribute that admits a null value
  - maximum N corresponds to an attribute that can have more than one value for the same occurrence (multivalued attribute)



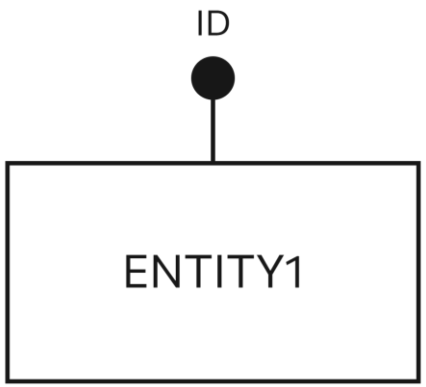
Identifier

- It is specified for each entity
- Describes the concepts (attributes and/or entities) of the schema that allow you to uniquely identify the occurrences of the entities
  - each entity must have at least one identifier
  - there can be more than one appropriate identifier for an entity
- The identifier can be
  - internal or external
  - simple or composite

Internal identifier

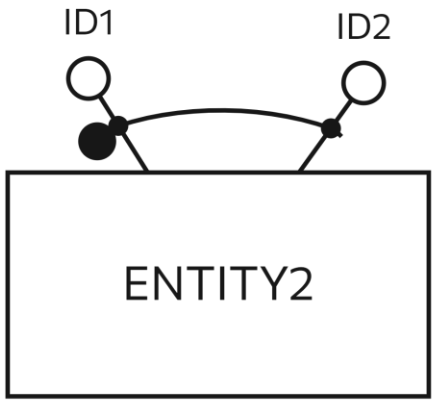
Simple

- consisting of a single attribute



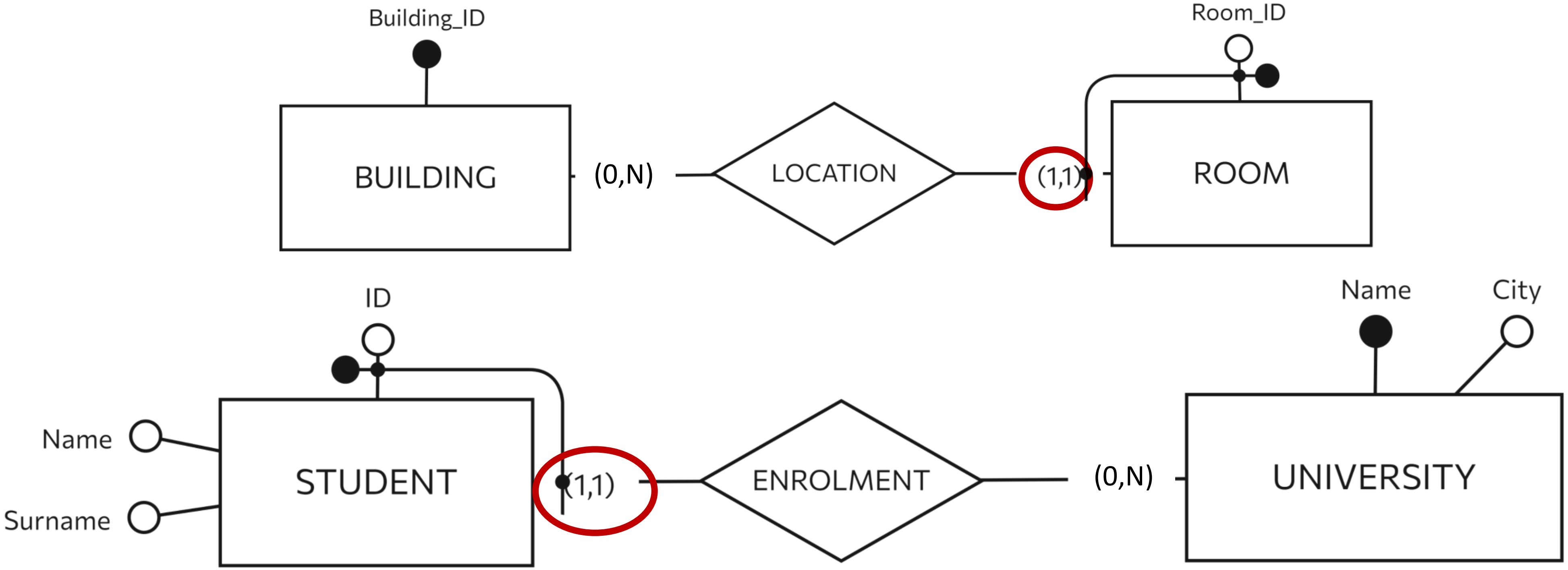
Composite

- consisting of multiple attributes



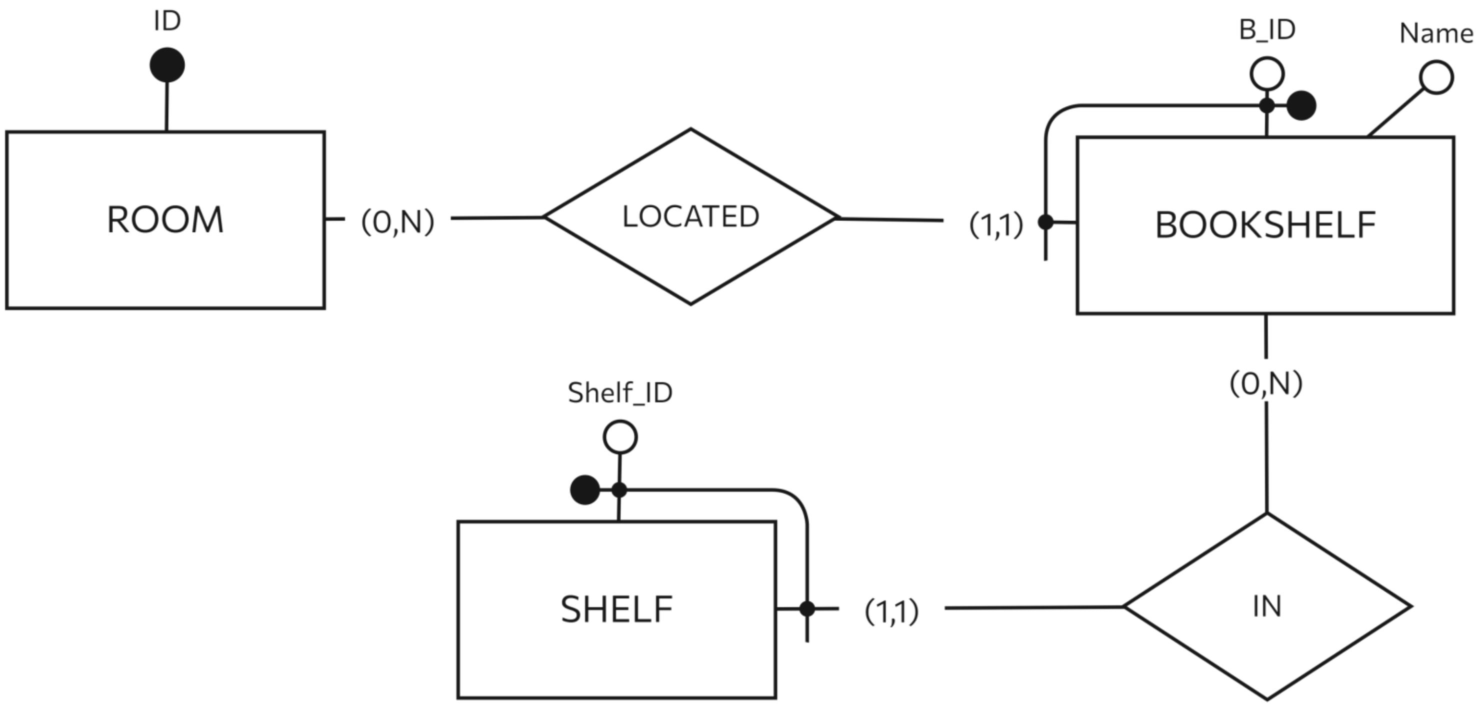
External identifier

- An entity that does not have internal attributes sufficient to define an identifier is called a weak entity
- The weak entity must participate with cardinality (1,1) in each of the relationships that provide part of the identifier



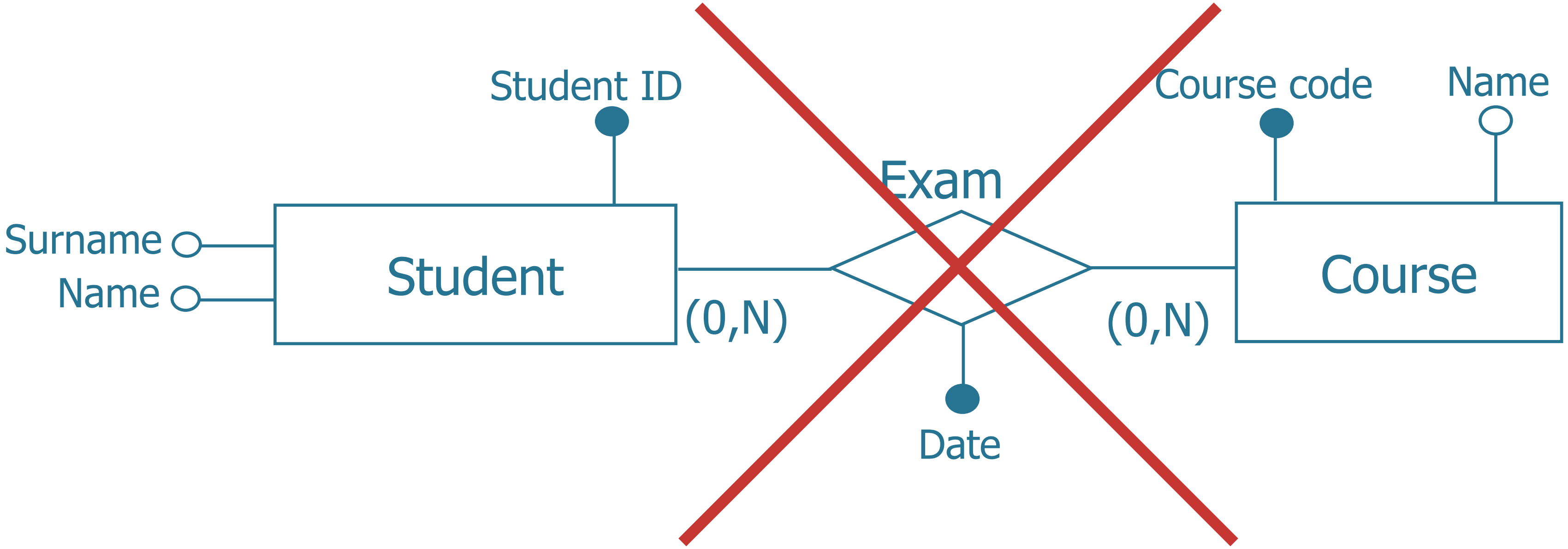
Remarks

- An external identifier can involve an entity that is itself externally identified
  - No identification cycles should be generated



Remarks

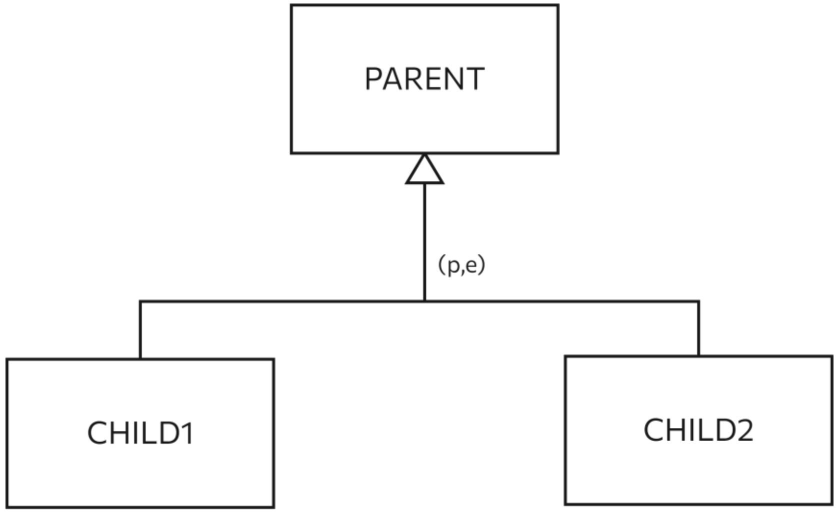
- Relationships do *not* have identifiers



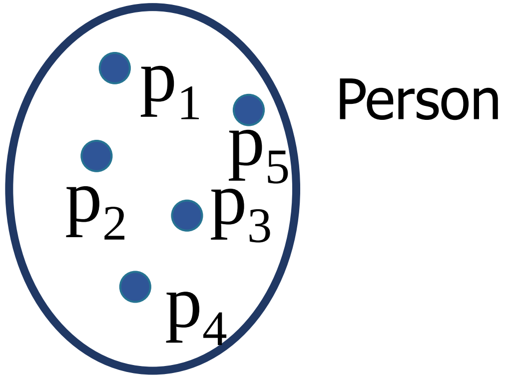
Generalization

It describes a logical link between an entity E and one or more entities E_1, E_2,..., E_n, that are particular cases of E.

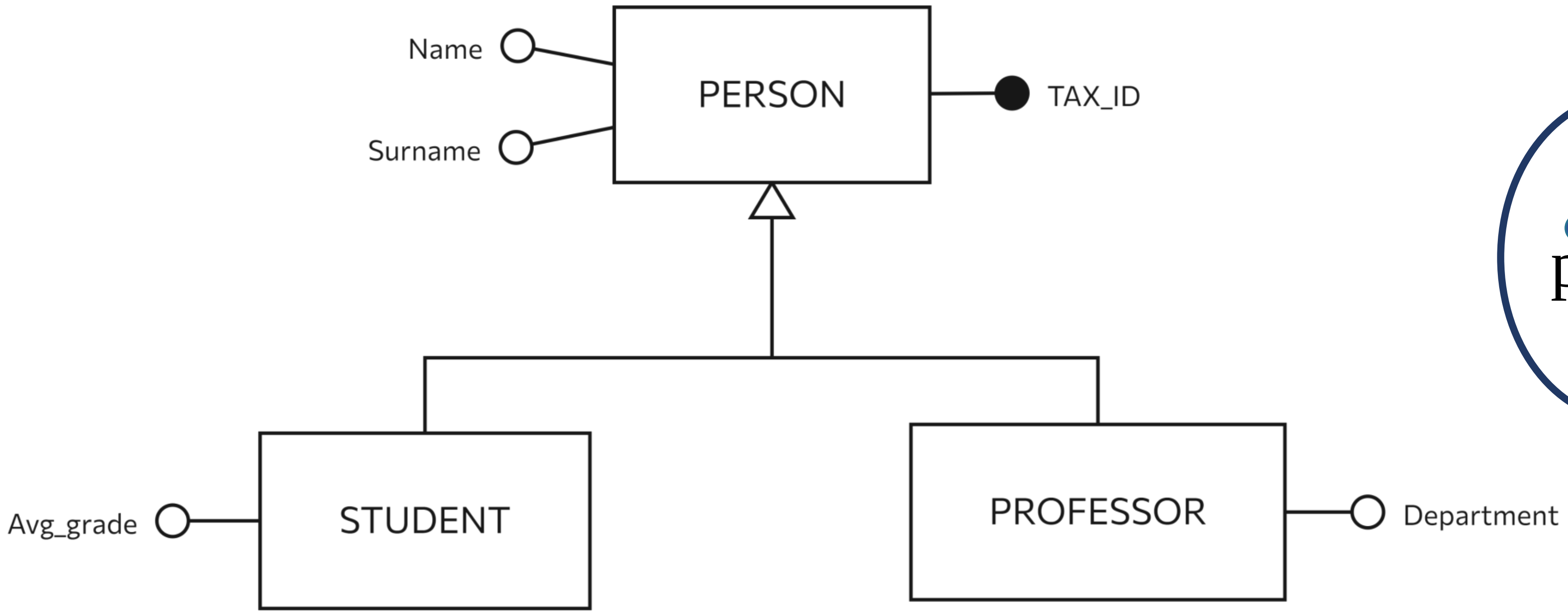
- E is called parent entity, and is a generalization of E_1, E_2,..., E_n
- E_1, E_2,..., E_n are called child entities, and are specializations of E

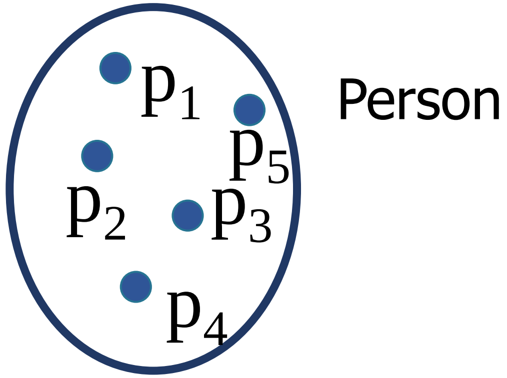


Generalization: example

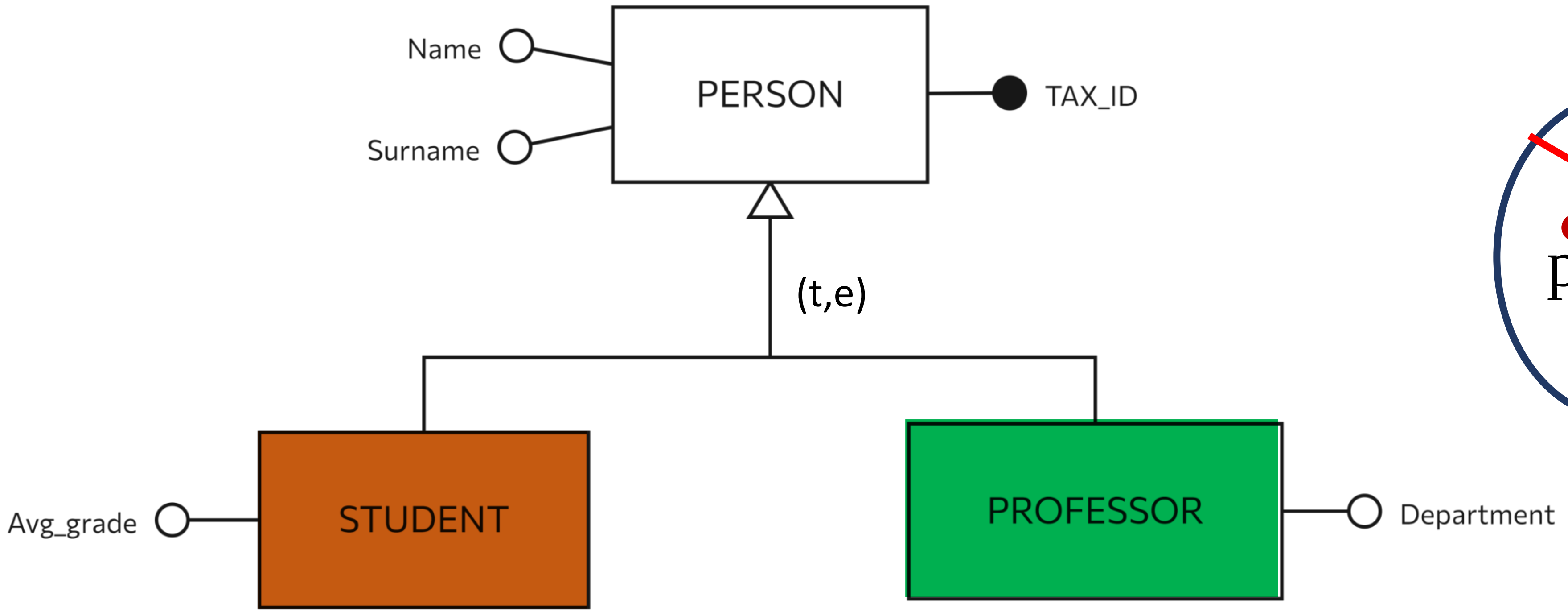


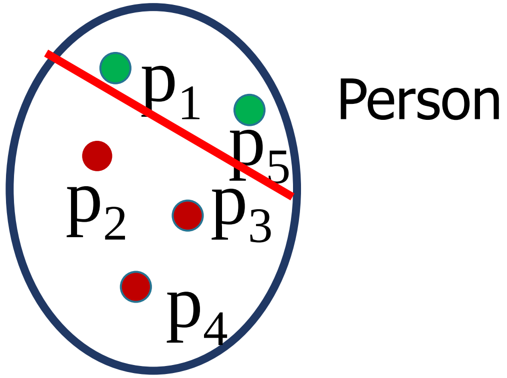
Generalization: example



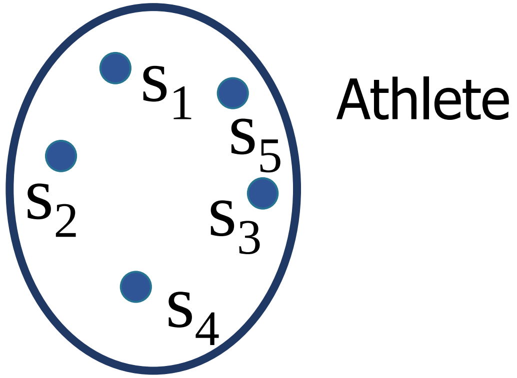


Generalization: example

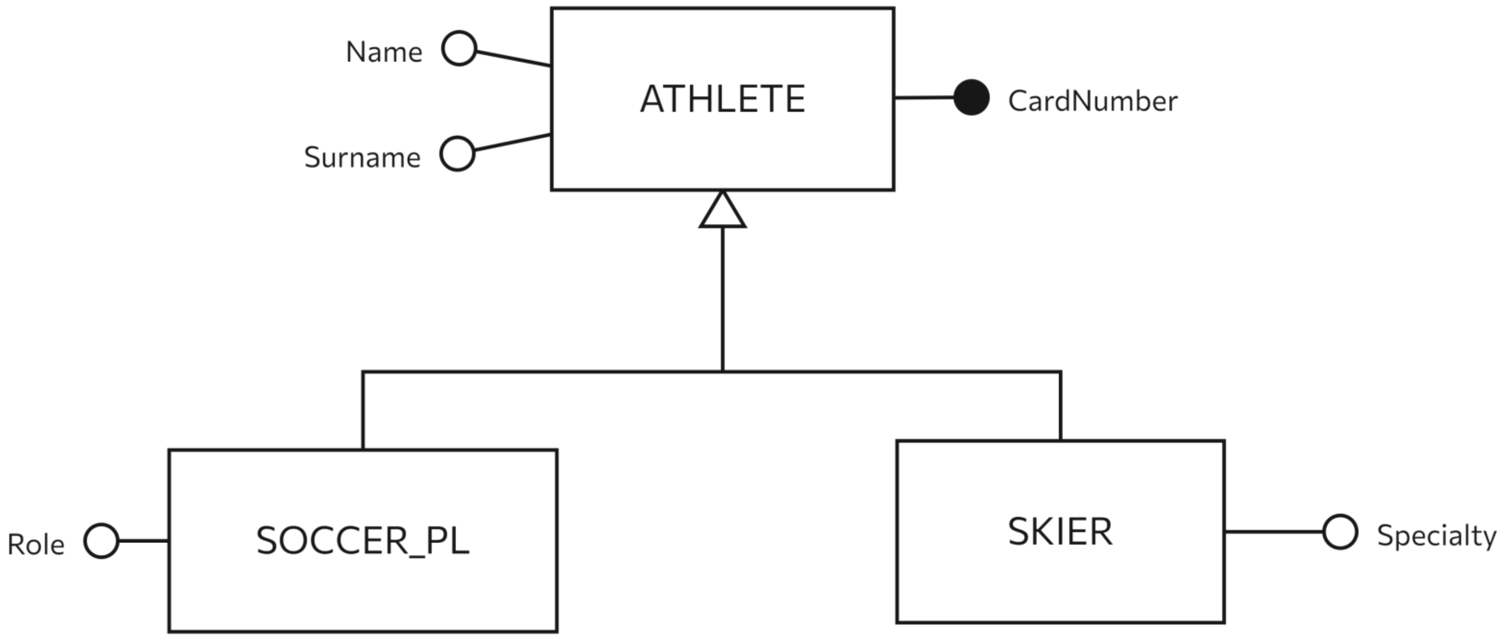


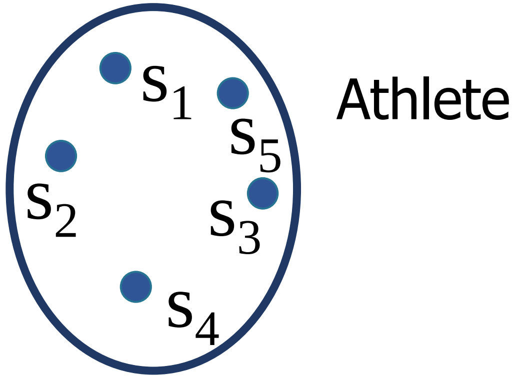


Generalization: example

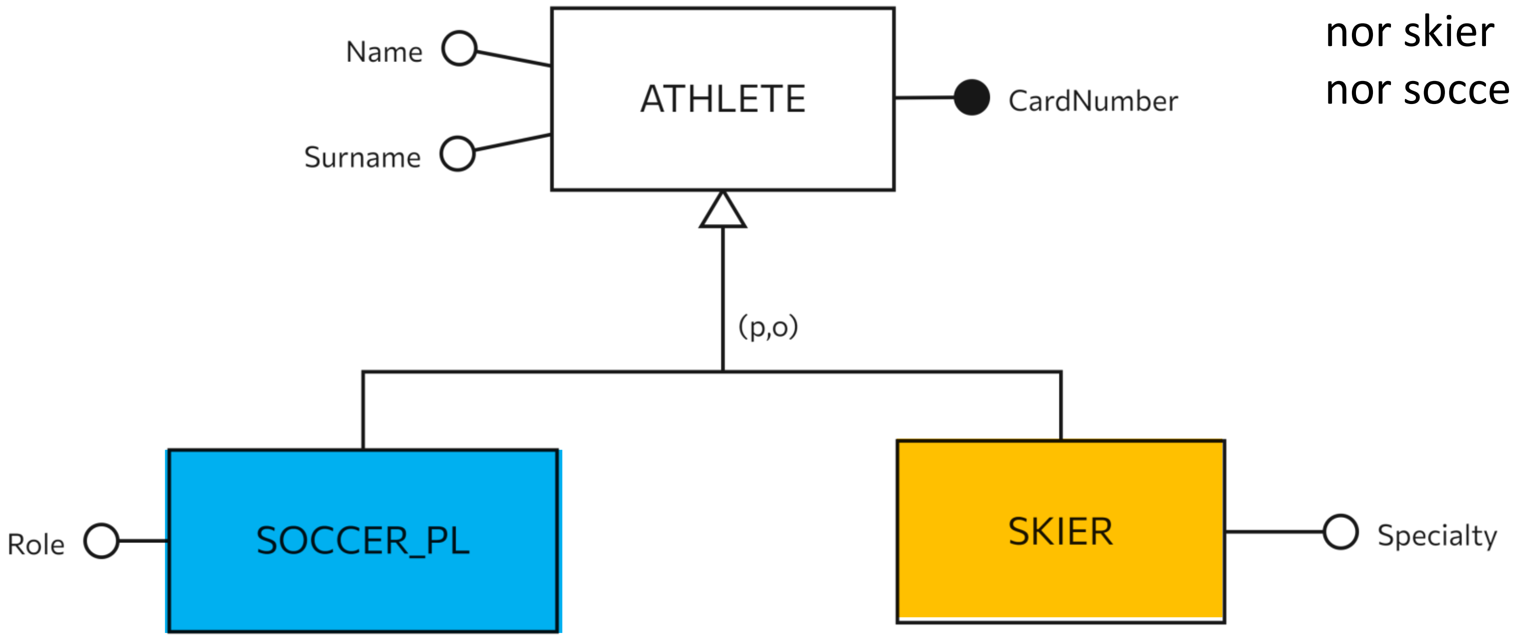


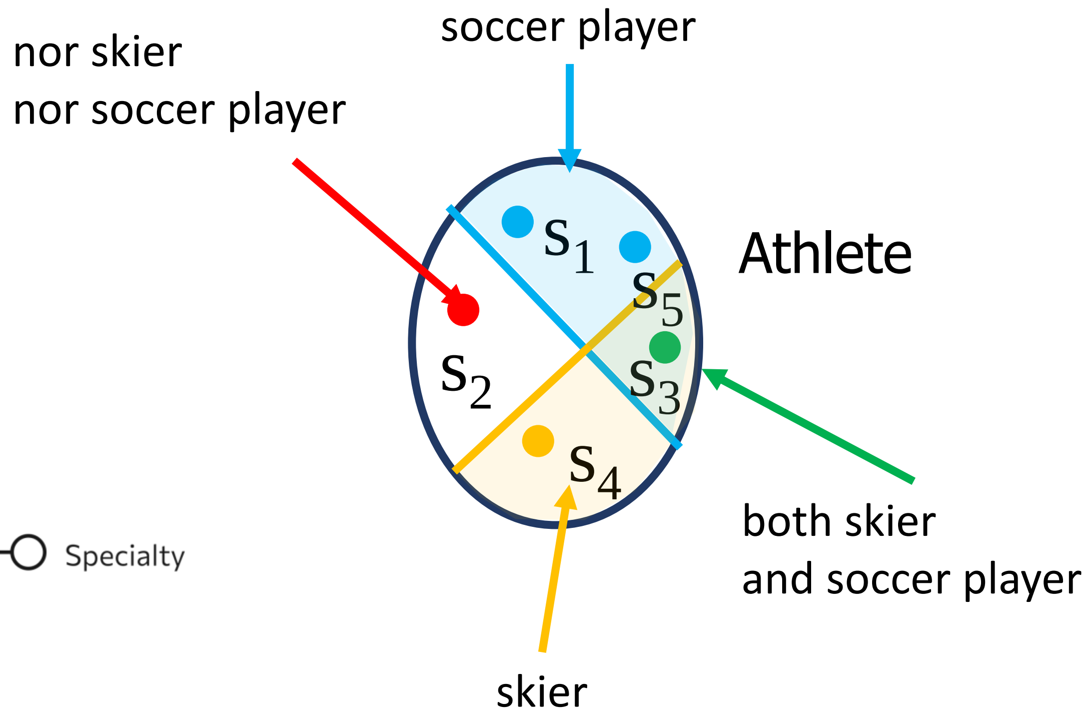
Generalization: example





Generalization: example





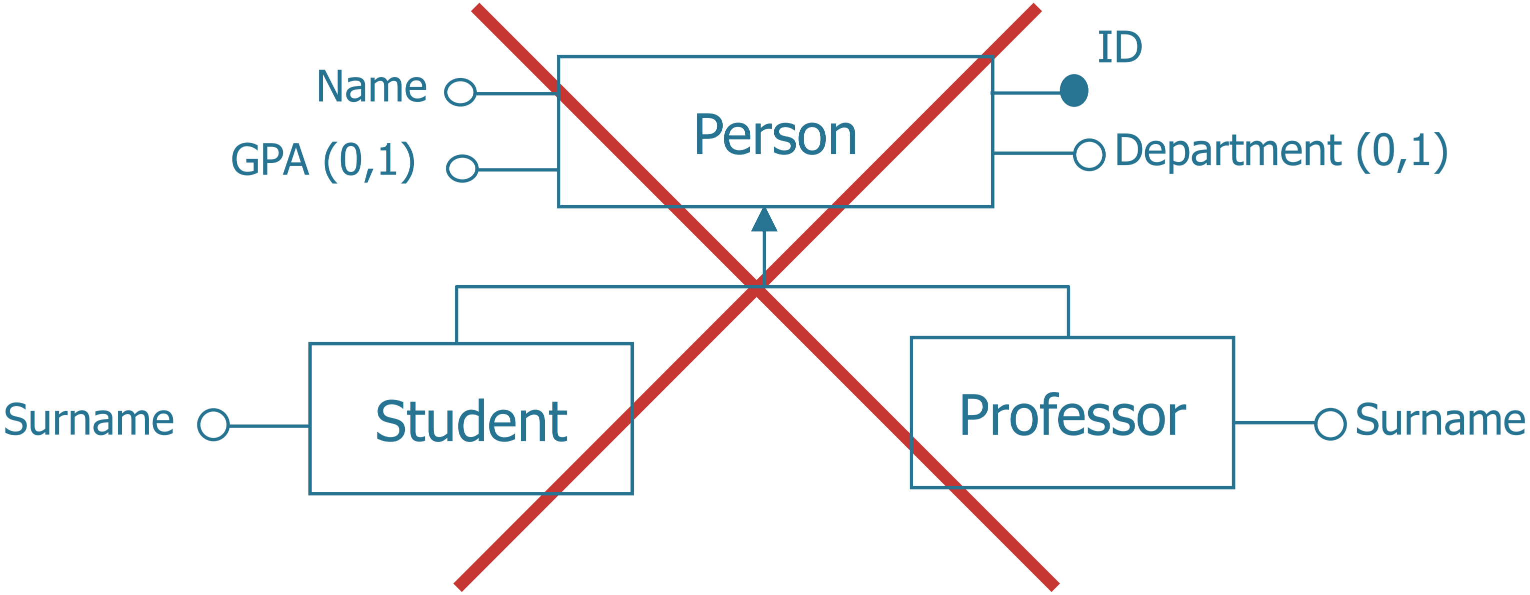
Generalization: properties

- Each occurrence of a child entity is also an occurrence of the parent entity
- Each property of the parent entity (attributes, identifiers, relationships, other generalizations) is also a property of each child entity
  - property known as inheritance
- An entity can be involved in multiple different generalizations

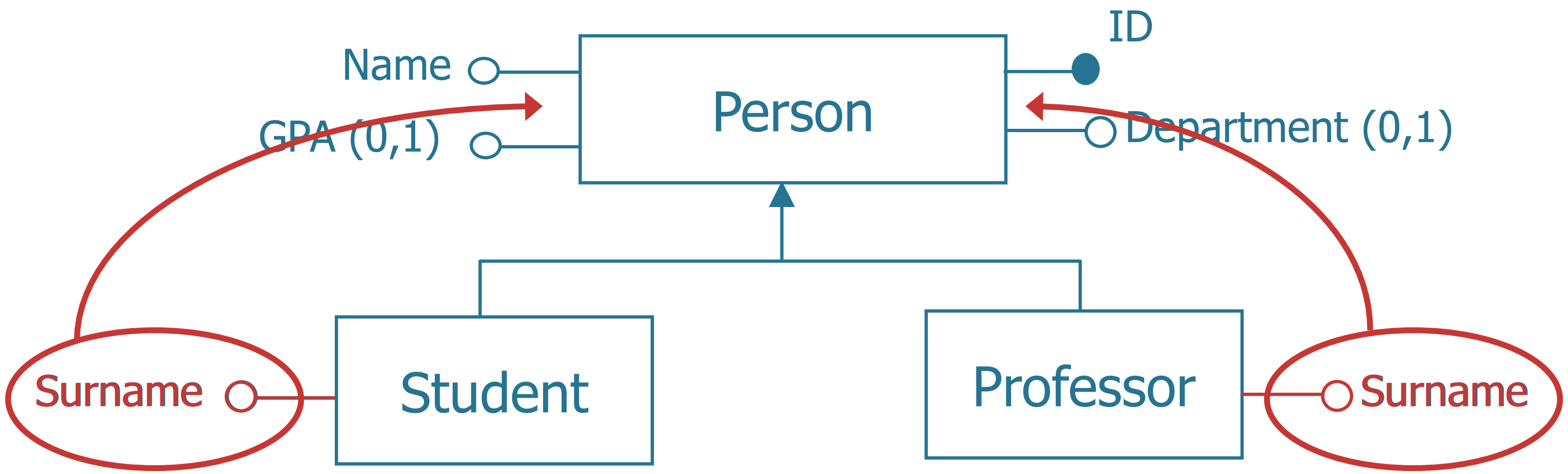
Generalization: properties

- Orthogonal characteristics
  - *total* generalization if each instance of the parent entity is an instance of at least one of the child entities, *partial* otherwise.
  - *exclusive* if each instance of the parent entity is at most one instance of one of the child entities, *overlapping* otherwise.

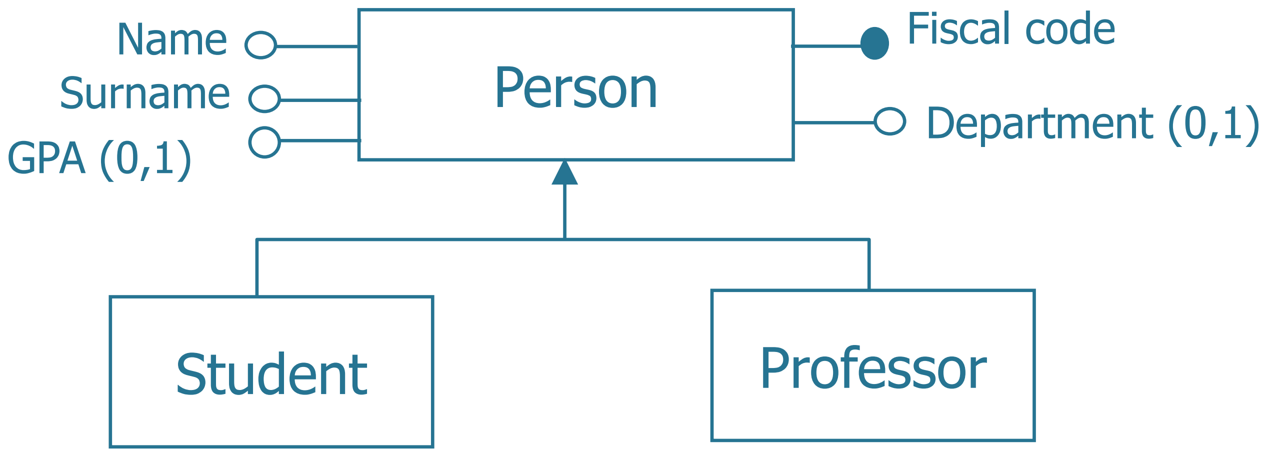
Generalization: incorrect example



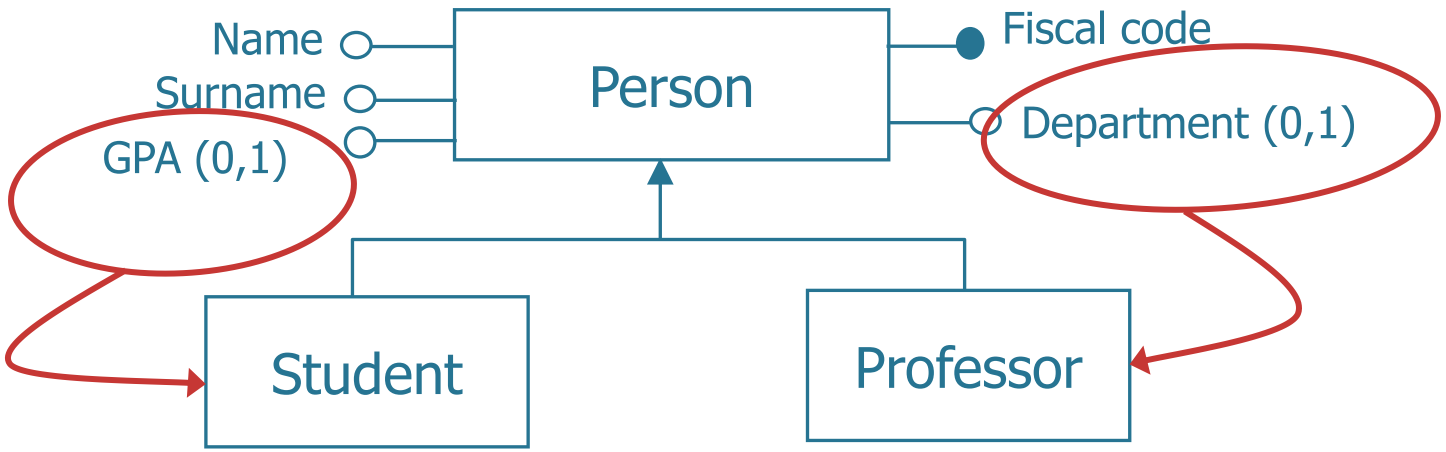
Generalization: incorrect example



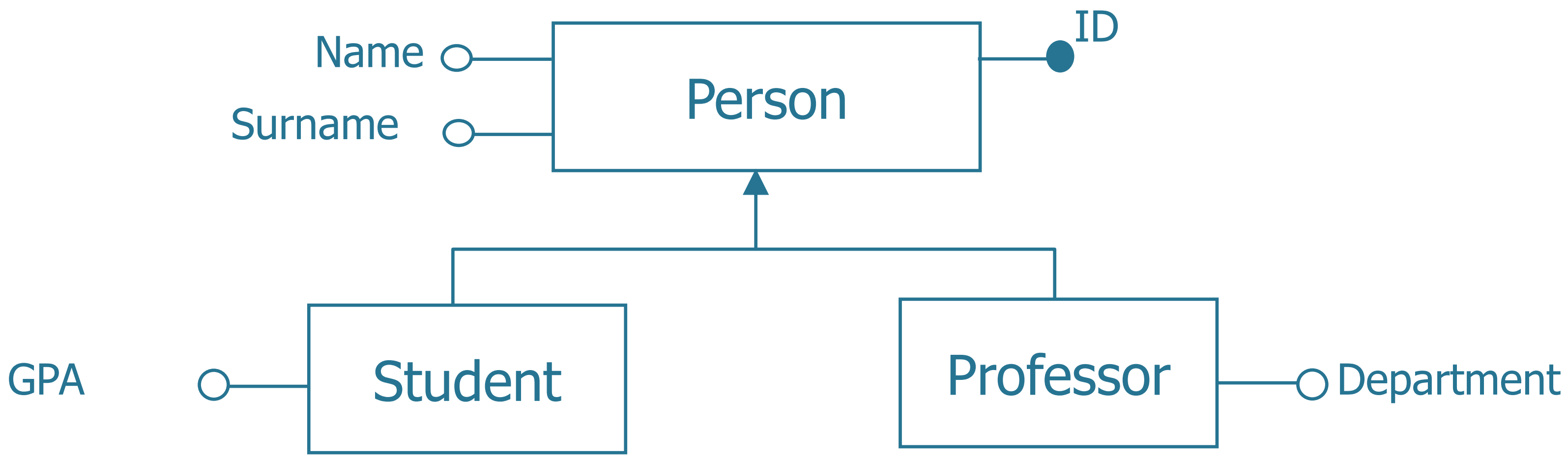
Generalization: incorrect example



Generalization: incorrect example

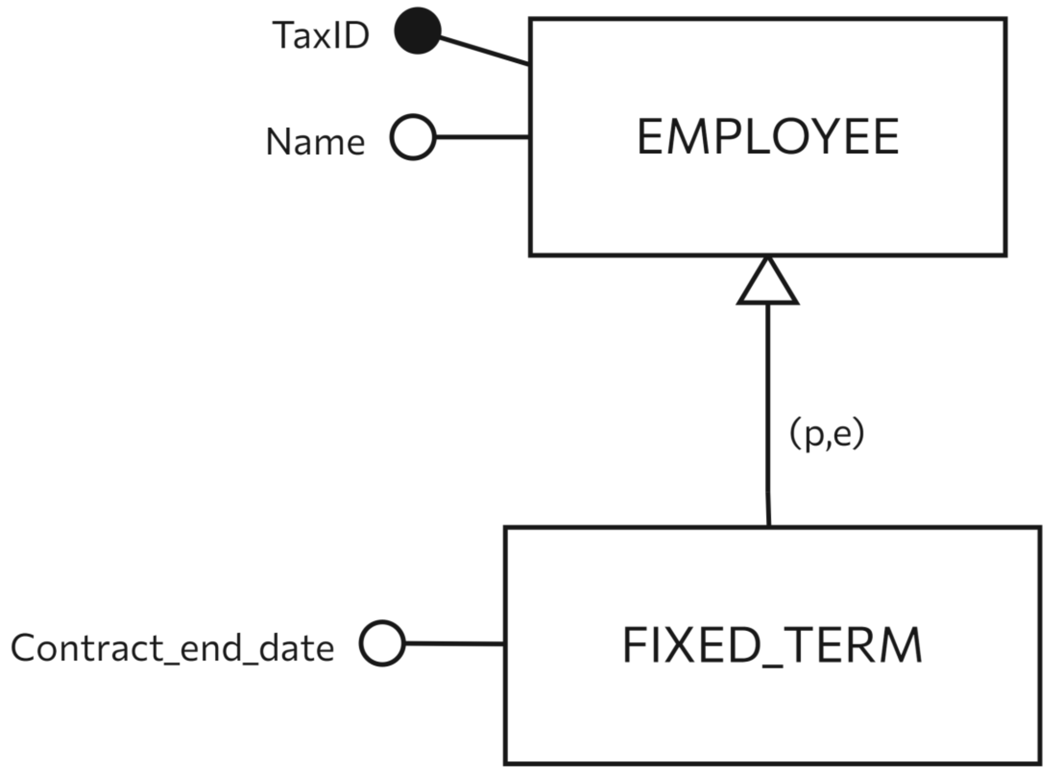


Generalization: correct example



Subset

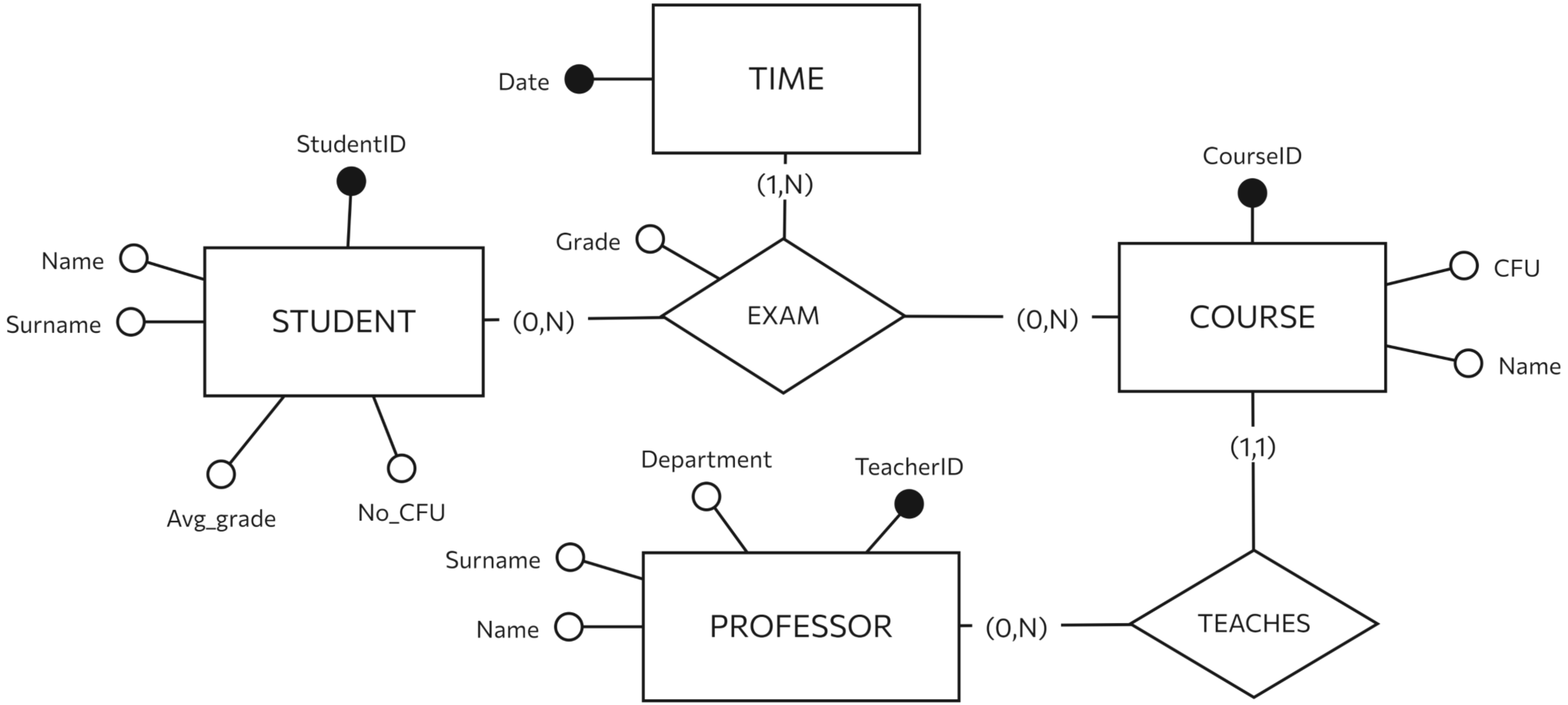
- Particular case of generalization with only one child entity
  - the generalization is always partial and exclusive



ER Model Documentation

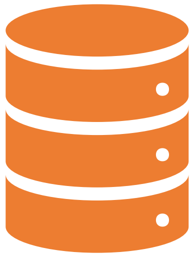
Database design

Documenting E-R models



DBMG

Documenting E-R models



Data Dictionary

Enrich the E-R schema with natural language descriptions of entities, relationships, and attributes

Data dictionary: example

Entity	Description	Attributes	Identifier
Student	University student	Student ID, Surname, Name, CFU acquired, Grades average	Student ID
Professor	University professor	Professor ID, Department, Surname, Name	Professor ID
Course	Courses offered by the university	Course code, Name, CFU	Course code
Time	Dates on which exams were taken	Date	Date

Data dictionary: example

Relationship	Description	Entities involved	Attributes
Exam	It associates a student to the exams taken and memorizes the mark obtained	Student (0,N), Course (0,N), Time (1,N)	Grade
Holder	It associates each course to the professor who teachers the course.	Course (1,1), Professor (0,N)	

Documenting E-R models

Data Dictionary

Enrich the E-R schema with natural language descriptions of entities, relationships, and attributes

Data Integrity Constraints

They cannot always be explicitly stated in an E-R scheme

They can be described in natural language

Data integrity constraints: examples

Integrity constraints	
RV1	The grade of an exam can only take values between 0 and 30
RV2	Each student cannot pass the same exam twice
RV3	A student may not take more than three exams for the same course during the same academic year

Documenting E-R models

Data Dictionary

Enrich the E-R schema with natural language descriptions of entities, relationships, and attributes

Data Integrity Constraints

They cannot always be explicitly stated in an E-R scheme

They can be described in natural language

Data Derivation Rules

Explicitly define that a concept of the schema can be obtained (by inference or arithmetic calculation) from other concepts of the schema

Derivation rules: examples

Derivation rules	
RD1	The number of credits acquired by a student is obtained by adding the number of credits of the courses for which the student has passed the exam
RD2	The average mark is obtained by calculating the average of the marks of the exams passed by a student

UML vs ER

Database design

UML and ER

UML (Unified Modeling Language)

- Modeling a software application
  - structural and behavioural aspects (data, operations, processes and architectures)
- Rich formalism
  - class diagram, actor diagram, sequence diagram, communication diagram, state diagram,...

ER

- Modeling a database
  - structural aspects of an application
- elements tailored to the modelling of a database

UML vs ER

- Different formalisms
- The class diagram of an application is different from the E-R schema of the database
- The class diagram, even if designed for different uses, can be adapted for the description of the conceptual design of a database

- Main Differences of UML vs ER
  - no standard notation to define identifiers
  - ability to add notes to comment on diagrams
  - possibility to indicate the direction of navigation of an association (not relevant in the design of a database)