DMIF, University of Udine

DMBD – Part I

Entity-relationship model

Andrea Brunello
andrea.brunello@uniud.it

The process of designing a database is typically articulated into different phases:

1. Brainstorming meetings with IT personnel and all interested stakeholders
   - Collection of requirements
   - Design of a conceptual model (e.g., E-R model)
   - The E-R model has a specific notation, the *E-R diagram*, that helps all involved parts to discuss about the future database
2. Translation of the conceptual model into a logical model
   - Typically, a set of translation rules is followed
   - At this stage, a specific DBMS technology must be chosen (e.g., relational DB)
3. Addition to the logical model of details regarding the physical, low-level aspects (e.g., usage of indexes)
   - The physical schema is thus obtained

Conceptual design

- The initial phase of database design is to characterize fully the data needs of the prospective database users
- Such needs are encoded into a conceptual model, that acts as a connecting point between stakeholders and IT personnel
- In this course, we will consider the **Entity-Relationship** model, which makes use of *diagrams* to represent the overall logical structure of a database graphically
- Such diagrams are typically complemented by free text annotations, that describe the requirements that could not be encoded in the diagram, or present the kinds of operations that users are expected to perform on the data

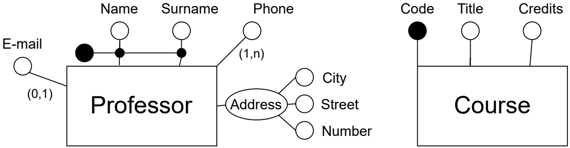
Entity-Relationship model

- An Entity-Relationship model describes a domain by means of:
  - Entity sets
  - Relationship sets
  - Attributes
- There are several ways in which entity sets, relationship sets and attributes can be represented in E-R diagrams
- We will consider the notation proposed in the book:

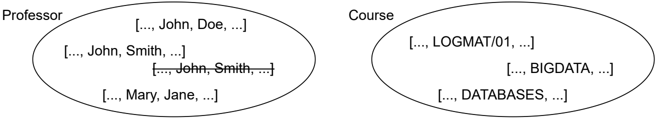
Database Systems - Concepts, Languages and Architectures, P. Atzeni, S. Ceri, S. Paraboschi, and R. Torlone

Entity sets

- An **entity** is an object that exists in the considered domain and is distinguishable from other objects
  - Example in the University domain: a specific professor, student, or course
- An **entity set** is a set of entities of the same type that share the same properties
  - Example: the set of all professors, each characterized by a name, a surname, and a salary
- Such properties, possessed by all members of an entity set, are represented by **attributes**
- A subset of the attributes forms a **primary key** of the entity set, i.e, knowing the values of such attributes, it is possible to uniquely identify a member (entity) of the entity set

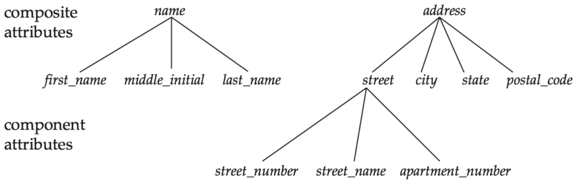


- Entity set *Professor* has the attributes:
  - *E-mail*, optional
  - *Name* and *Surname*, that together make the primary key
  - *Phone*, multi-valued
  - *Address*, composite, and made by *City*, *Street*, *Number*
- Entity set *Student* has the attributes:
  - *Code*, that makes the primary key
  - *Name*
  - *Surname*



- An entity set cannot contain two entities with the same values on the primary key attributes
- Observe that, being a set, each entity set has a *trivial* primary key composed of all the attributes

- Composite attributes can be nested further



- The characteristics of attributes can be combined, e.g., there can be multi-valued composite attributes, as well as optional composite attributes

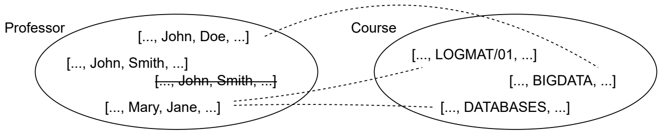
Relationship sets

- A **relationship** is an association among several entities
  - E.g., *Professor:John Doe* may be linked with *Course:BIGDATA* by means of a relationship *Teaches*
  - The relationship is an association between actual entities

- A **relationship set** $R \subseteq E_1 \times E_2 \times \cdots \times E_n$ is a mathematical relation among $n \geq 2$ entities (e_i), each taken from its entity set (E_i)
  - $R = \{(e_1, e_2, \ldots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$
  - where $(e_1, e_2, \ldots, e_n)$ is a *relationship*

- For instance, $(John, Doe, BIGDATA) \in Teaches$ is a relationship belonging to relationship set *Teaches*

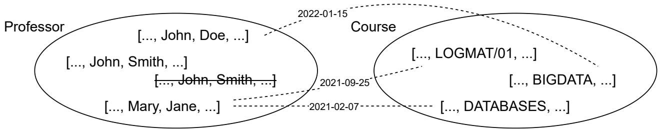
Relationship sets
Intuitive meaning



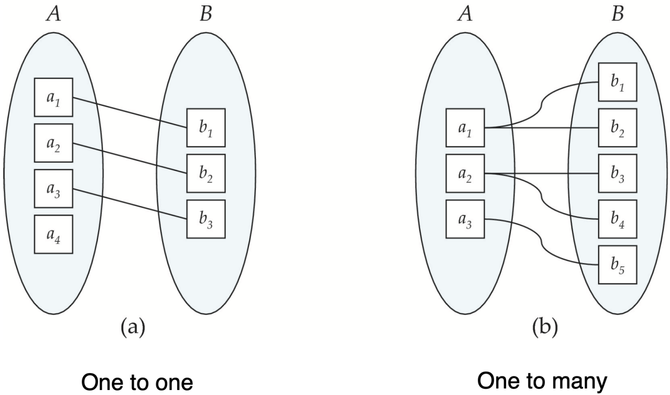
- The picture shows three relationships (dashed lines), which we assume belonging to relationship set *Teaches*:
  - *John Doe* is associated to *BIGDATA*, $(John, Doe, BIGDATA)$
  - *Mary Jane* is associated to *LOGMAT/01*, $(Mary, Jane, LOGMAT/01)$
  - *Mary Jane* is associated to *DATABASES*, $(Mary, Jane, DATABASES)$

Relationships and attributes

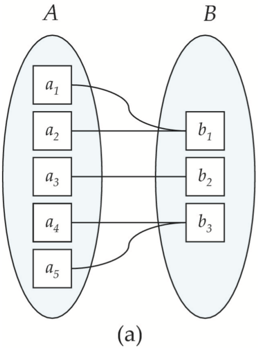
- Attributes may also be associated with relationships, e.g., to track when a professor started teaching a course
- Intuitively, a relationship is thus as follows:
  - $(John, Doe, BIGDATA, 2022-01-15)$



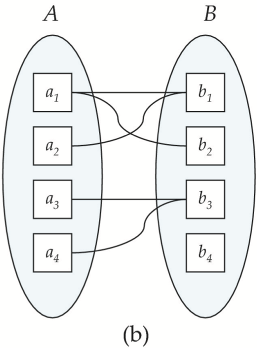
- Given entity sets E_1, E_2, and a relationship set R between them, entities may participate into relationships in different ways, for instance:
  - An entity of E_1 may not participate to any relationship with an entity of E_2 (a prof. may not be teaching any courses)
  - An entity of E_1 may participate to > 1 relationship with an entity of E_2 (a prof. may be teaching more than one course)
- In general, for a binary relationship set we can identify the following **cardinalities**:
  - One to one
  - One to many
  - Many to one
  - Many to Many
- Moreover, entity sets may have a *partial* or *total* **participation** to a relationship set



- Left: A has a partial participation, while B total
- Right: both A and B have a total participation



Many to
one

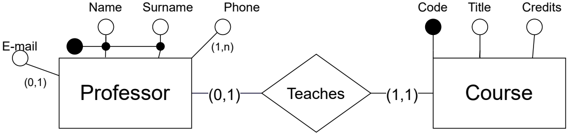


Many to many

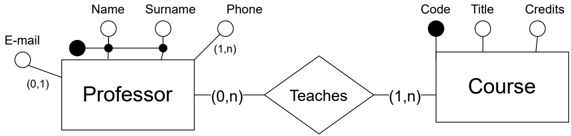
- Left: both A and B have a total participation
- Right: both A and B have a total participation

Relationship sets

Notation for participation and cardinality constraints

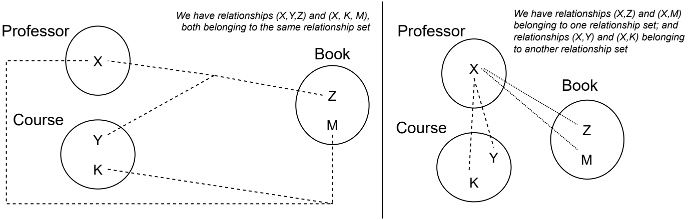


- A professor teaches in at most one course: (0,1)
- A course is taught by one and only one professor: (1,1)
- (1,1) is assumed by default, and can be omitted
- Concerning the relationship set *Teaches*, this means that:
  - A given entity $p_1 \in \mathit{Professor}$ can appear in at most one relationship $(p_1, c) \in \mathit{Teaches}$ where $c \in C$
  - A given entity $c_1 \in \mathit{Course}$ appears in exactly one relationship $(p, c_1) \in \mathit{Teaches}$ where $p \in P$

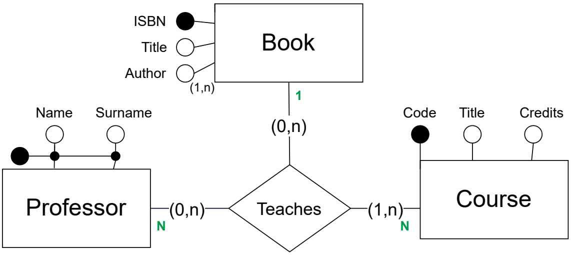


- A professor teaches in zero or more courses: (0,n)
- A course is taught by one or more professors: (1,n)
- Concerning the relationship set *Teaches*, this means that:
  - A given entity $p_1 \in Professor$ can appear in any number of relationships $(p_1, c) \in Teaches$ where $c \in C$
  - A given entity $c_1 \in Course$ appears in at least one relationship $(p, c_1) \in Teaches$ where $p \in P$
- Observe that the only values allowed in the parentheses are 0, 1, n. We cannot specify a number, e.g., 4

- Sometimes, it is necessary to involve more than 2 entities in a relationship (though in a DB most of them are binary, and nearly always at most ternary)
- For instance:
  - *professor X teaches course Y using the book Z*
  - *professor X teaches course K using the book M*



- Observe how the relationships on the right convey less information than those on the left

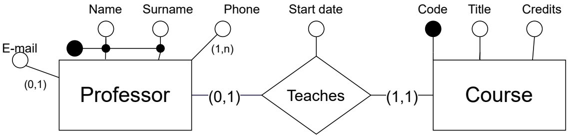


- Participation/cardinality constraints still apply to ternary relationships, although things complicate a little bit
- Here: a professor may teach zero or more courses, each course is taught by at least one professor, and a book is used within zero or more teaching activities, although a professor uses only a book within the same course

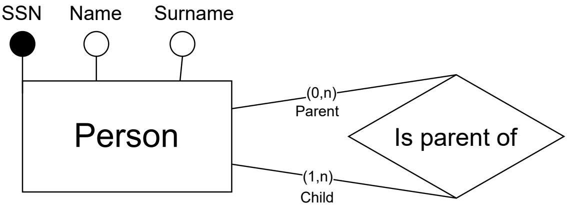
- If I have an entity of *Professor*, how many entities of *Book* and *Course* together can be associated to it?
    - Here, the answer is zero or more, so entity set *Professor* participates with $(0, n)$ to *Teaches*

- As for the green numerosities, they allow us to solve some ambiguity. Without them, we wouldn't know if a professor may appear in more than one relationship because:
    - He may teach more than one course using the same book
    - He may teach only one course but using multiple books
    - He may teach multiple courses using multiple books

- Setting the green 1 close to *Book*, we say that, given a *Professor* and a *Course*, there can be only a *Book* associated to them both. On the contrary, given a *Professor* and a *Book*, we may have more courses (green N close to *Course*)



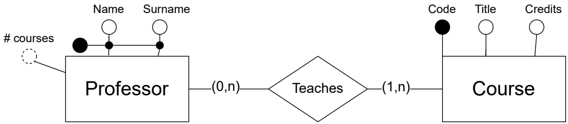
- Relationship sets can also have attributes
- This is quite natural when the attribute tells us something regarding the specific entities involved in the relationship
- For instance, here we track the date in which a given professor started teaching a given course



- Entity sets of a relationship need not be distinct
    - For instance, consider the case in which we want to track the prerequisites of a given course
- Each occurrence of an entity set plays a “role” in the relationship
- In that case, it is useful to write down the **roles** as labels on the arcs

Derived attributes

- A derived attribute is an attribute whose value can be computed from other information
- For instance, the number of courses taught by a professor can be derived from the number of relationships of relationship set *Teaches* he/she participates to
- The derived attribute is depicted with a dashed circle
- The E-R diagram should include notes, written in natural language, on how to calculate the derived attributes contained in it



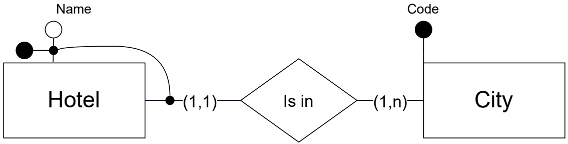
Weak entity sets

- Consider the following scenario:
  - We want to keep track of hotels, each characterized by a name
  - There may be multiple hotels with the same name in different cities, however, given a city, there cannot be two hotels with the same name
  - Each city is univocally identified by a code
- Intuitively, to univocally identify a hotel, its name is not enough; we also need to know the city where it is

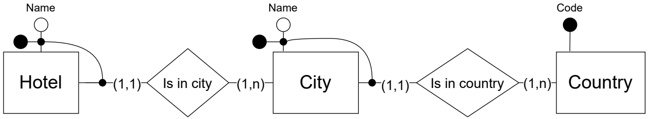
Weak entity sets
(cont.)

- The notion of **weak entity set** formalizes the above intuition. A weak entity set is one whose existence is dependent on another entity, called its **identifying entity**; instead of associating a primary key with a weak entity, to uniquely identify it we use the identifying entity, along with extra attributes called **partial keys**
- Each weak entity must be associated with a *a corresponding* identifying entity; that is, the weak entity set is said to be **existence dependent** on the identifying entity set. The identifying entity set is said to **own** the weak entity set that it identifies. The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**

- Observe the $(1,1)$ cardinality on the *Hotel* side



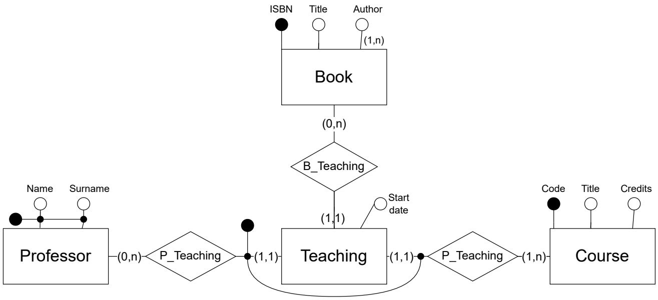
- We may also define chains of weak entity sets



- A weak entity set may depend on more than one identifying entity

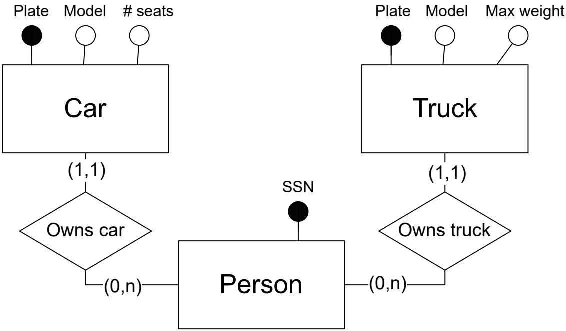
Ternary relationship set reification

- The reification process replaces a ternary relationship set with an entity set and three relationship sets
- The obtained representation is equivalent
- Observe that the introduced entity set is weak with respect to two entity sets

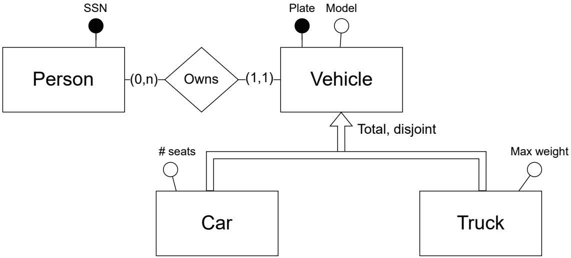


Specialization

- It may happen that different entity sets share some commonalities regarding their attributes or relationships
  - E.g., a *Truck* and a *Car* may both be considered as a *Vehicle*



- The **specialization** construct allows us to handle such a scenario
- It is a similar concept as specialization/generalization in object oriented programming
- A “lower-level” entity set inherits all the attributes and relationship participations of the “higher-level” entity set which it is linked to



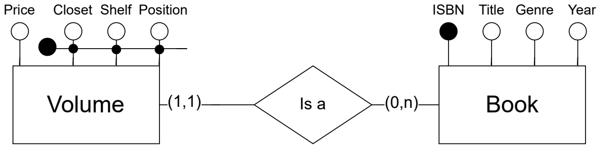
Specialization
(cont.)

- A specialization may be:
  - **Total** or **Partial**
    - **Total**: each higher-level entity corresponds to at least one of the lower-level entities (e.g., a *Vehicle* is a *Car* or a *Truck*)
    - **Partial**: there may be higher-level entities that do not correspond to any lower-level entity (e.g., we may have a generic *Vehicle* which is not a *Car* nor a *Truck*)
  - **Disjoint** or **Overlapping**
    - **Disjoint**: a higher-level entity corresponds to at most one of the lower-level entities (e.g., a *Vehicle* can be a *Car* or a *Truck*, but not both)
    - **Overlapping**: a higher-level entity may correspond to more lower-level entities (e.g., a *Vehicle* can be both *Car* or a *Truck*, for instance think of a pick-up)
- Of course, each lower-level entity corresponds to a higher-level entity
- Always write in the diagram the kind of specialization

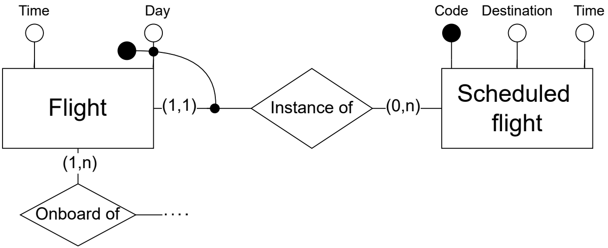
The *instance of* construct

- Consider the following scenario:
  - A library maintains a catalogue of published books. For each book, it records: ISBN, title, genre, and year of publication
  - Not all books are available at the library. Of the available ones, we record the selling price and the location of each copy: closet, shelf, position in the shelf

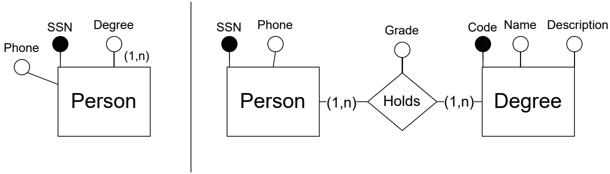
- We may notice the presence of two different entities:
  - *Book*: a “virtual” book, with its ISBN
  - *Copy*: the actual, “physical” volume, which can be considered as an instance of a book



- An airport wants to keep track of flights
- A flight is uniquely identified by a code, and characterized by a destination and a daily scheduled departure time (e.g., the AZ123 flight of 3:50 PM to Rome)
- For each occurrence of a flight, we also want to keep track of the actual departure time, its crew, [. . .]



- When should attributes be used, and when entity and relationship sets instead?
- It really depends on the domain, and on the amount of detail that we want to express



Exercise 1

Let us synthesize the ER conceptual schema for a database of tv series data recording information about episodes, actors and directors, based on the following requirements:

- Let us assume that both actors and directors are uniquely identified by their name and surname, and are characterized by their birth date and their nationality. The same person may be both an actor as well as a director
- Each tv series is characterized by its title, genre, release year, and director. We assume that there is only one director per series

Exercise 1
(cont.)

- It may be the case that distinct tv series have the same title (this is the case, for instance, with remakes). Nevertheless, given a year, there cannot be two series with the same title
- A tv series is composed by a number of episodes, each with an incremental number that uniquely identifies it within the series, and a title
- Different episodes may involve different actors, each with a specific role. For the sake of simplicity, let us assume that, in a given episode, each actor may play one role only

Exercise 2

Let us synthesize the ER conceptual schema of a database for the management of the study program of a university on the basis of the following set of requirements.

- Each course is uniquely identified by its numerical code and characterized by a name, a set of prerequisites (a set of other courses that must be already passed in order to take the exam), and the professor that teaches it (each course is taught by one professor only)

- Each student is uniquely identified by an ID number and characterized by a first name, a last name, an email, an address, and a birth date. Each student is enrolled in one or more courses and has passed a number of them (possibly no one yet). For each course that the student has passed, we record the date of the exam and the obtained grade

- Professors are partitioned in full and associate professors. Each professor is uniquely identified by his/her fiscal code and characterized by a first name, a last name, one or more mobile numbers, possibly an email address, the department to which he/she belongs, and the courses that he/she teaches (one or more). For each full professor, we record the day he/she was named full professor
- Each department is uniquely identified by its name and characterized by an address and a budget