Diseño conceptual: Entidad relación

CI-0127 Bases de Datos, Universidad de Costa Rica

Sivana Hamer

1 Database design process

The database design process starts by recollecting requirements and ends with the full database design. The complete process is shown in Fig. 1.

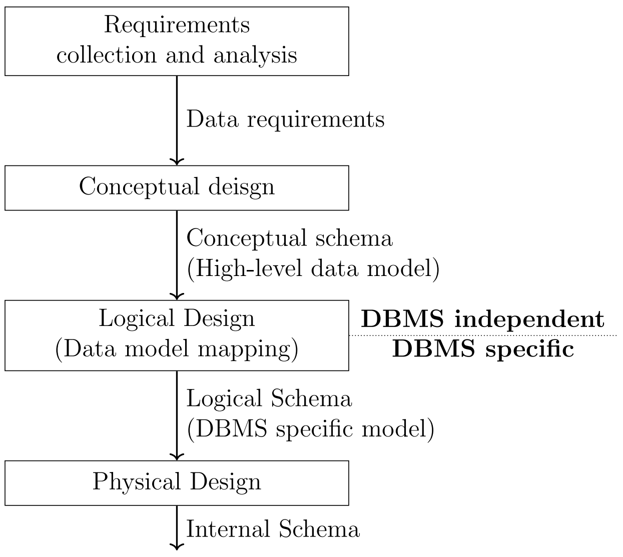
Requirements collection and analysis
Data requirements
Conceptual deisgn
Conceptual schema (High-level data model)
Logical Design (Data model mapping)
DBMS independent
DBMS specific
Logical Schema (DBMS specific model)
Physical Design
Internal Schema

Figure 1: Database design process

1. To start the design, the database designers gather and document the *data requirements* from prospective database users. These requirements can be also gathered with the functional or non-functional requirements of the application, including the operations needed.
2. Based on the data requirements, the data high-level *conceptual design* is defined with a *conceptual schema* based on a data model. These designs do not include implementation details, being easier to understand for users and to create a good design due to not needing to be bogged down by the physical and implementation details. During or after this step, the operations in high level user queries can also be defined.
3. Using the conceptual schema, we can implement it by using a *logical design* or *data model mapping*. This implementation is DBMS specific and may be somewhat automated using tools.
4. Finally, the *physical design* can be defined. This includes the internal storage structure, file organization and indexes.

2 Entity relationship model

The *entity relationship* (E-R or ER) data model is a *high-level or conceptual data model.* In this model, data is described as entities (Section 3), attributes (Section 4) and relationships (Section 5). The model can be graphically represented in an *ER diagram.* For the following examples, we will try to exemplify using a course enrollment system for our university. The ER diagram shown in Fig. 2.

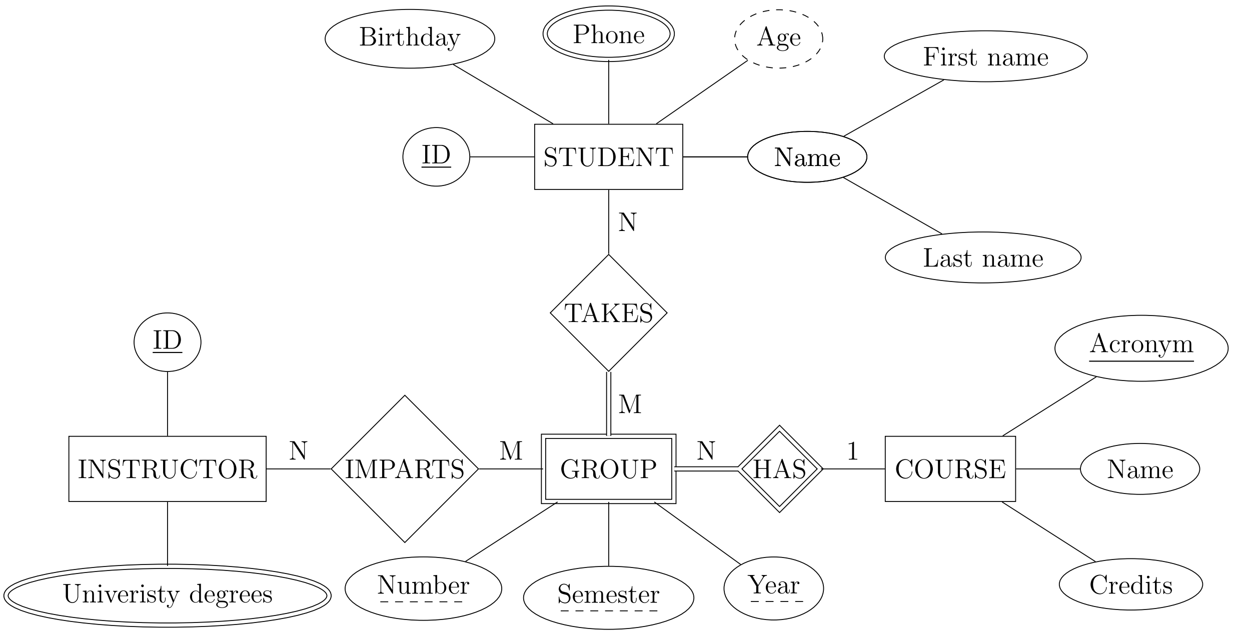

Figure 2: ER diagram for course enrollment

3 Entities

An *entity* is a real-world "thing" or "object" distinguishable from other entities. They may have a real world physical existence (e.g., person, building) or a conceptual existence (e.g., institution). In the course enrollment example entities can include instructors, courses, groups and students. An entity can be represented in an ER diagram as a square, as shown in Fig. 3. Inside the square a name is defined. Entities may be recognized as nouns.

Figure 3: ER diagram representation for an entity

While stored in a database, each entity will have different values for each attribute. For example, there might be instances for a person such as p_1 and p_2. *Entity type* are the set of entities that have the same attributes (e.g., PERSON, BUILDING). The *entity set* or *entity collection* are the collection of entities of a particular entity type in the database at a point of time (for PERSON the $p_1, \cdots, p_n$).

4 Attributes

Attributes are the properties that describe an entity. For example, for a person these could include the name, year of birth, or gender. Each attribute has a *value.* For example, for the previous attributes a value for a person p_1 could be "Sivana Hamer", 1900 and "female", respectively. An attribute is represented by an oval as shown in Fig. 4. Attributes are connected by lines to the associated entity with their names inside the oval. Attributes tend to be the nouns describing another noun (entity).

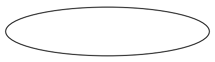

Figure 4: ER diagram representation for an attribute

There are different types of attributes:

Simple vs Composite attributes. *Simple* or *atomic* attributes are not divisible in further parts (e.g, credits). While a *composite* attributes can be divided into smaller parts (e.g., name into first and last name). A composite attribute is represented by ovals in a hierarchical structure as shown in Fig. 5. Furthermore, a composite attribute can be further subdivided into more independ parts (e.g, first name into first and middle name). Composite attributes are used when a user refers to some of the smaller parts and sometimes the whole, but if only the whole is used there is no need to subdivide it into components.

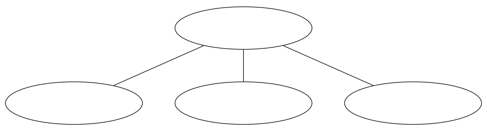

Figure 5: ER diagram representation for a composite attribute

Single-valued vs Multivalued attributes. For some attributes, each particular entity may have only a *single* value. For example, each person only has one birthday. However, there are other attributes that may have different numbers of values for an entity attribute (*multi valued*). For example, a student may be none, one or multiple phone numbers. There might be lower and upper bounds for the number of attributes. The ER diagram representation of a multivalued attribute is an oval with two lines, as shown in Fig. 6.

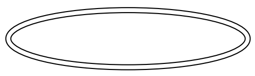

Figure 6: ER diagram representation for a multivalued attribute

Stored vs Derived attributes. A *stored* attribute is saved in the database, while a *derived* attribute is obtained from another attribute. For example, using a birthday attribute for student we can gather the age. The representation of a derived attribute is an oval with dashed lines, as shown in Fig. 7.

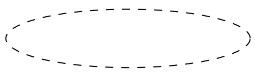

Figure 7: ER diagram representation for a derived attribute

Key vs non-key attributes. Every entity within a set has a unique *key* as a *uniqueness constraint*, called the *key attribute*. For example, every student has an ID that is unique to them. Another example is how every course has a unique acronym in our university. The representation for a key value is shown in Fig. 8. A composite variable can be also used as a key, as long as each of the subparts together provide a unique value. For example, a license plate is unique as the combination of the state and the number. An entity may also have more than one key attribute, but always most have at least one. If an entity has no key attributes, it is a weak entity (Section 6).

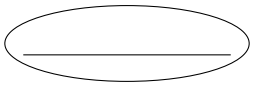

Figure 8: ER diagram representation for a key attribute

- Attributes might also save NULL if for an entity a particular value is not applicable (e.g., middle name if there is none) or if it is unknown (missing or is not known). For example, we might not know someones birthday.
- The values of the attribute can also be constrained between ranges and data types, though in the ER diagram this is not represented. This is called the *domain* of the attribute.
- A *complex* attribute is a multivalued and composite attribute.

5 Relationships

Relationships are associations between several entities. For example, the relationship between the entity STUDENT and GROUP is TAKES. Relationships might be recognized as verbs. It is represented in an ER diagram as a diamond, as seen in Fig. 9. Relationships are connected by lines to the associated entities, with the name inside the diamond.

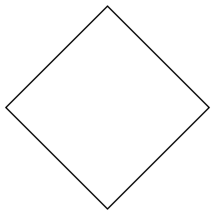

Figure 9: ER diagram representation for a relationship

- The *relationship set* represents all the relationships of the same type.
- The *degree of the relationship* is the number of associated entities to a relationship. For example, a binary relationship has two entities while a ternary has three associated entities. In our current ER diagram, we can only see binary relationships.
- Every entity that participates within a relationship has a particular *role*. These roles are described by role names, though they are not necessary for all the relationships.
- Relationships may be *recursive* or *self-referencing* when the same entity participates within a relationship as different roles. For example, a PERSON in a INSTRUCTing relationship may be the instructor or the apprentice. The representation of this relationship is shown in Fig. 10.

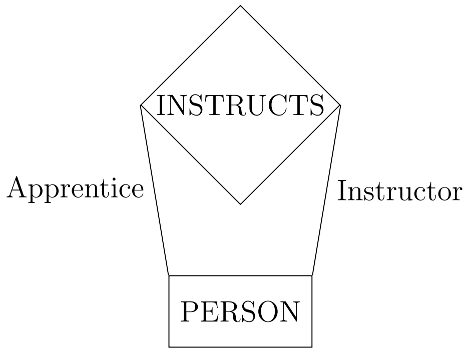

Figure 10: ER diagram for a recursive instructing relationship

- The *cardinality* determines the maximum number of entities and entities can be associated within a relationship set. The cardinality numbers are detailed above the connecting lines of the relationship and the entity. The cardinality can be used for any degree of relationship, though for this example we will use it for two entities in a relationship, A and B. They can be:

  One-to-one (1:1) Where every entity in A is associated at most with one entity in B. For example, every STUDENT has one ID_CARD while every ID_CARD only has one STUDENT.

  One-to-many (1:N). Where every entity in A can be associated with any number of entities (zero or more) in B, but B entities can be associated to at most one entity in A. For example, a COURSE may be given in multiple GROUPs but every GROUP only has one COURSE.

  Many-to-many (N:M). Where every entity in A can be associated with any number of entities in B and every entity in B can be associated with any number of entities in A. For example, STUDENTs can take multiple GROUPs or even none if they just entered university. While, every GROUP is taken by multiple STUDENTs.

- There may be *participation constraints* between entities in a relationship, representing the minimum number of relationships that may participate in a relationship.

  Total participation. Indicates that all entities must participate in the relationship. For example, every GROUP is expected to have a STUDENT (at least 7 in the UCR).

  Partial participation. Indicates that only some entities participate in a relationship. For example, every COURSE is not expected to have a GROUP as it might have not been given yet.

- Relationships may also have attributes. Though for 1:1 or 1:N the attribute may be associated to the entity with the 1 cardinality.

- We can represent the minimum and at most maximum relationships instances in the set using the (min, max) notation. We can see how we pass from min max notation to our ER diagram in Fig. 11.

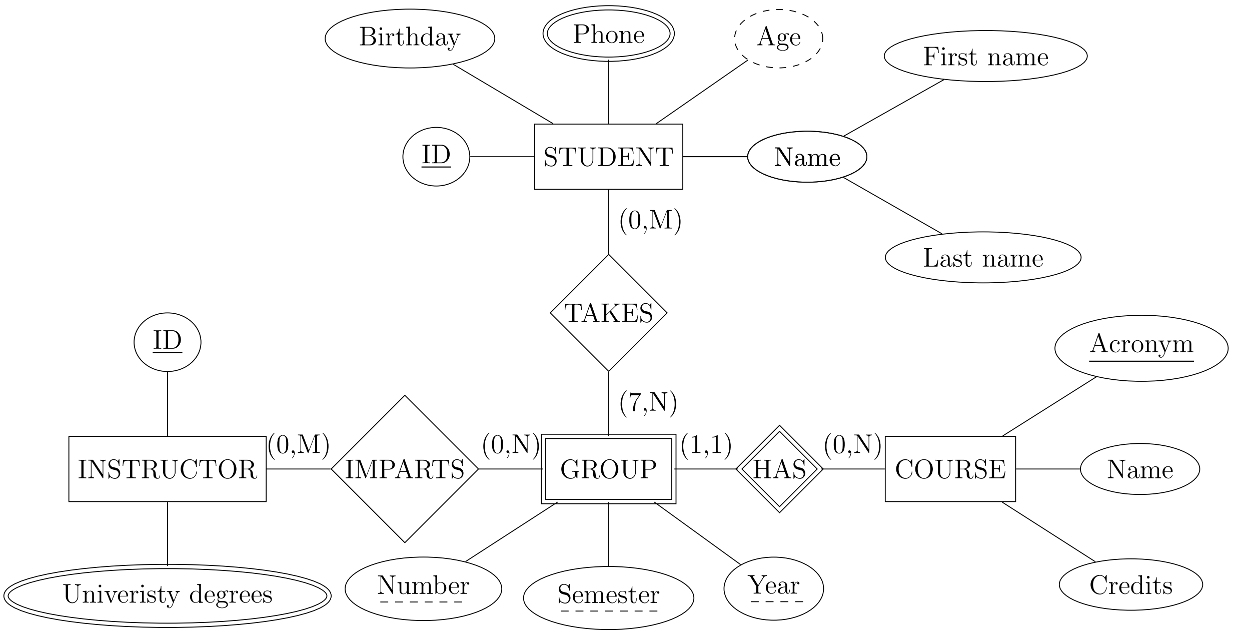

Figure 11: ER diagram for course enrollment using (min, max) notation

6 Weak entities

Entities without keys are *weak entities*, in contrast to entities with keys that are *strong entities.* Weak entities can be identified based on the combination of their attributes and another entity type (i.e., *identifying or owner entity type*). The relationship between the weak and owner entity is called the *identifying relationships.* An example of a weak entity is GROUP with the HAS relationship with COURSE, as every GROUP requires a course. The weak entity type and identifying relationship are represented in the ER diagram as a double lined box (Fig. 12) and diamond (Fig. 13), respectively.

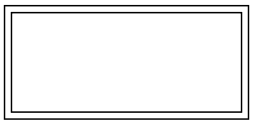

Figure 12: ER diagram representation for a weak entity

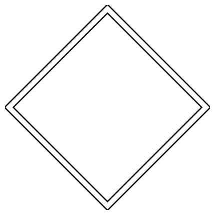

Figure 13: ER diagram representation for an identifying relationship

A weak entity has a participation constraint with their owner entity. Weak entities may require a *partial key* that in combination with the owner entity gives uniqueness. We do not require partial keys in 1 to 1 relationships. Partial keys are denoted with dashed underlining. We can see that for the GROUP, we have the group number, semester and year.

7 Binary versus higher degree relationships

While binary relationships are most common, sometimes there are situations where more than two entities must be involved in a relationship (*higher degree relationship*). As such, higher degree relationships are useful if all the entities are needed based on the semantics of the situations. Usually, three binary relationships are different from a ternary relationship. One can have both binary and higher degree relationships if they provide different meanings and they are all needed for the application. It is more efficient for the DBMS to have binary relationships.

For example, we can represent the relationship of a STUDENT TAKES a COURSE during a SEMESTER with a ternary relationship. The ER diagram is shown in Fig. 14. This represents a STUDENT s taking a COURSE c in a SEMESTER se, as a tuple of (s, c, se). A binary relationship between STUDENT and COURSE, (s, c), would only indicate that they TOOK a course in a semester (we can not be sure when). A binary relationship between COURSE and SEMESTER, (c, se), indicates that a COURSE was GIVEN but not who was it taken by. Finally, a binary relationship between STUDENT and SEMESTER, (s, se), indicates that they took classes in that semester but not which clases. We represented this relationship as binary in our above diagram by modeling the entities and relationships differently. This may be possible and preferable, depending on the semantics of the situation. We could have also added a GROUP entity as a weak entity of COURSE and SEMESTER, however if we did that we would have no need for the ternary relationship as the information within the SEMESTER would be part of the GROUP. We must also must have the SEMESTER as part of the key in GROUP due to the fact that a group may have the same number, but be given in different semesters.

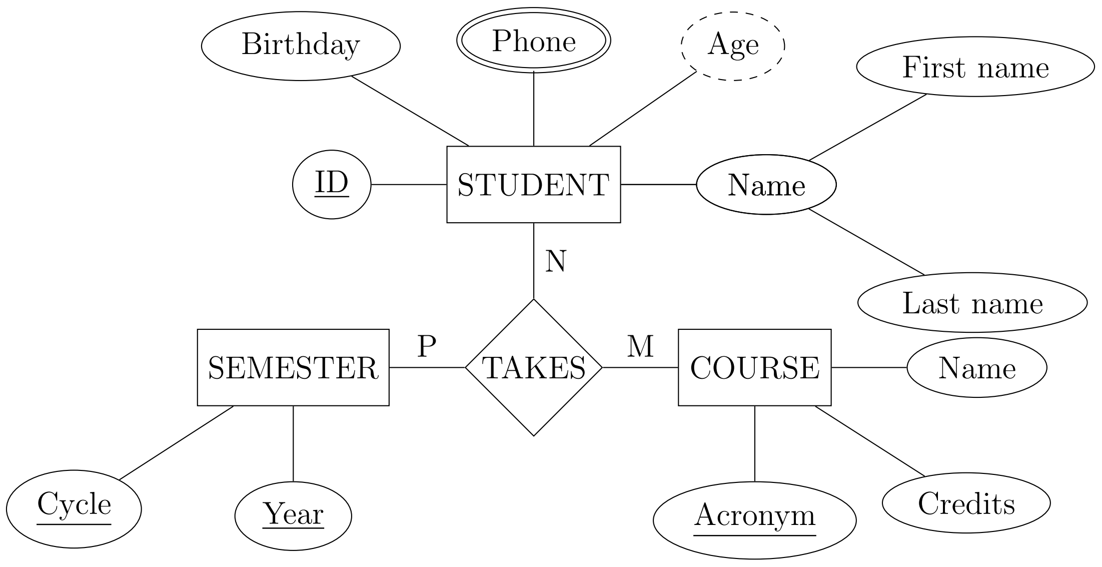

Figure 14: ER diagram for a ternary relationship

8 Naming conventions

- Names should be in singular, as they represent an item of a set.
- Entity and relationship type are written in all uppercase.
- Attributes names have only their initial letter in uppercase.
- Role names are all in lowercase.
- Diagrams are organized to be read from left to right and top to bottom.

References

[1] R. Elmasri and S. Navathe, *Fundamentals of database systems*, 7th ed. Pearson, 2016, chapters 3 and 4.

[2] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, 7th ed. New York, NY: McGraw-Hill, 2020, chapter 6.

[3] C. Faloutsos and A. Pavlo. Lectures #2. [Online]. Available: https://15415.courses.cs.cmu.edu/fall2016/