Unit 3: Relational Databases Design

1 - Motivation.

2 - E/R Diagrams (Entity/Relationship)

- Entities, attributes, keys
- Relationships: degree, cardinality and correspondence
- Dependency relationships

3 - The extended E/R model (multi-valued and composed attributes, hierarchies and generalizations, associations)

4 - Converting an E/R diagram to the relational model

5 - Functional dependencies

6 - Normal forms (normalization)

Motivation

Once the relational model is known, how are we tackling the database creation?

- From the data
- From a model

While other old models (hierarchical, network) give rise to designs that show little resemblance to reality, the relational model is very suitable for working with data, however it does NOT have a direct correspondence with the real world.

Solution: embrace a methodology that, via an intermediate model, actually reflects the world reality in a closer, more intuitive and understandable way

Options: object-oriented databases, entity-relationship model

Entity-Relationship (E/R) Model

- Defined by Chen in the 70s
- It does NOT take into account the physical implementation of the data, only the conceptual level
- It is based on objects (called entities), which are abstractions of reality with their corresponding attributes, and associations or interactions between them (relationships).
- It is expressed in the form of very intuitive diagrams

In practice:

1) Design of the E/R diagram

2) “Translation” to the relational model

3) Implementation in a real DBMS using SQL

Features

- High-level abstraction of reality → direct and natural, without modifying the essence or fundamental properties
- Physical independence of the storage device
- Requirements:
  - Primary key
  - Obligation of establish relationships between entities → no isolated elements
  - It is not possible to link relationships between them

Elements of the E/R model

Entities:

Any person, organisation, thing, etc. we want to store data from:

The features of the entities are fixed (invariable) over time, and only those that are relevant or useful to the context of the problem will be kept in the model.

In the case that the list of properties is modified later, the cause will be that they were previously incomplete.

Elements of the E/R model

Strong and weak entities

Entities can be tagged as either strong or weak.

We will call a strong entity to any that can exist on its own independently, whereas a weak entity is the opposite – that is, any whose that will require a strong one to make sense. In other words, there is a dependency on the strong one.

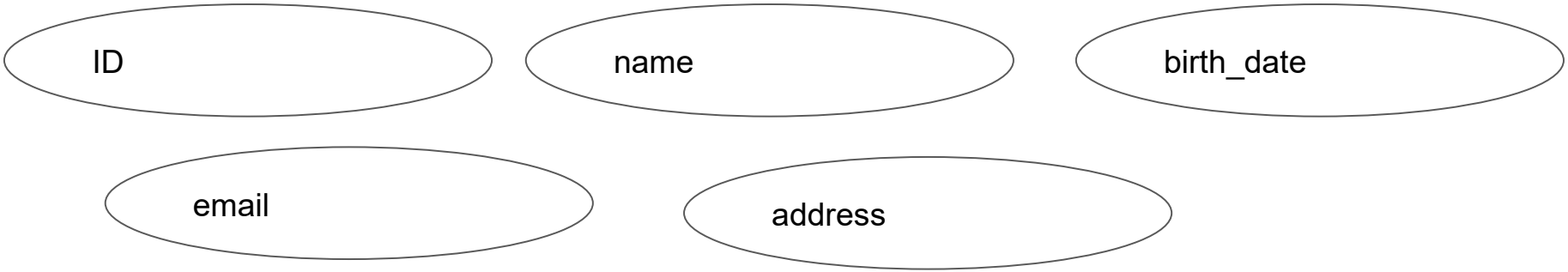
As we can see, we will use a double box to represent a weak entity.

As a consequence → the attributes of a weak entity are insufficient to form a candidate/primary key.

Elements of the E/R model

Attributes: each of the properties selected for an entity that we need to store information about.

Representation: horizontal ellipse + name



Attributes can belong to both entities (mandatory) and relationships

Elements of the E/R model

Attributes can be (or be part of) keys:

- **Superkey**: any set of attributes that identify each instance of an entity in a unique way. There must be one at the least (all attributes)
- **Candidate key** → the minimum superkey (those not selected as primary are called alternate keys)
- **Primary key** → the candidate key selected by the designer of the database to identify the elements of the entity → it must be known, invariable over time, and with no null values. It is shown underlined.
- **Foreign key** → as we already know, it refers to another relation (entity) and it is represented with double underline.

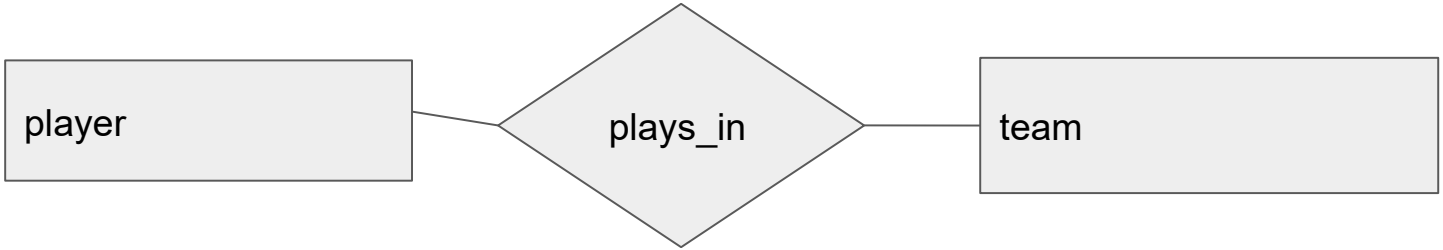
Elements of the E/R model

Relationships: the associations between several entities (2 or more). Therefore their presence is linked to them → they cannot exist independently.

How are they named?

- With describing verbs
- By joining terms from the names of the related entities

Representation: with diamonds



Degree and cardinality of a relationship

Cardinality: it represents the participation of each entity in the relationship

Types:

- 1:1 (one to one)
- 1:N (one to many)
- N:N (many to many)

Degree: the expression of how many entities are involved in the relationship:

- Unary (or reflexive)
- Binary
- Ternary

Extended E/R model

The Entity-Relationship model has received extensions in later revisions to make it more powerful, flexible and representative.

For instance, we can find new elements such as:

- Hierarchy or **specialisation** (is-a) → refinement or sub-type of a given entity
- **Aggregation**: it represents the action of taking a part of the diagram (entities with interrelationships) as an entity for abstraction.
- **Reflexive** relationships: they allow an entity to establish a relationship with itself.
- **Exclusivity**: a special type of relationship in which only one of the possible options can be taken (e.g. car and fuel). They are represented with a bow.

Transformation of the E/R diagram to the relational model

This process is straightforward. I must be done via the following rules:

- Entities: each one is transformed into a relational model table directly, with the same name, attributes and primary key
- Weak entities: the same as a normal entity, but also adding the primary key of the strong entity on which it depends. Its primary key will always include the attributes of the imported primary key.
- Relationships of degree N:N: in this case, a table is created for the relationship. In addition to its attributes, it will import the primary keys of those involved entities (therefore, they will be foreign keys). The set of both will be the primary key of the table.

Transformation of the E/R diagram to the relational model

- 1:N relations: in this case it is NOT necessary to create a new table. Instead, its attributes and the primary key of entity with cardinality 1 will be imported into the table with cardinality N
- 1:1 relationships: this kind of relationships, not being present often, are treated as just one element all in all. Unless we wish to keep the data separated for better understanding or visibility, the solution is to directly create a single table with all the attributes of the involved entities and those from the relationship, if any.

Problems from the relational design

Despite the advantages of the process to get the database representation, the resulting tables may still have issues:

- Redundancies → leading to an overrepresentation of the data, thus deriving in potential insert, update or delete anomalies
- Ambiguity → in two similar tables similar field names can be used to refer to different data
- Inconsistencies → often derived from data redundancy. The same data saved in two separate attributes as if they were different is an example
- Loss of information → the deletion of a record might trigger data disappearing

All these problems can be solved with a correct analysis methodology, although others are derived from the existing dependencies.

Functional dependencies

Dependence → a set of restrictions or constraints imposed on particular attributes on the tables.

Notation: C1 → C2

Determinant → Dependent

C1 functionally determines C2

C1 and C2 are subsets of attributes of a relation R

Each value of C1 corresponds to a unique value of C2.

Example: NSS → name

(NSS: Social Security Number)

Dependency Types

- Functional
  - Trivial
  - Elementary (A, B → B)
  - Transitive (A → B, B → C, then A → C)
  - Complete (if A → B, but no subset of A can imply B)
- Multi-valued (when there are rows that imply the presence of other rows in the same table, e.g. car models and colours)
- Natural product (if a given table T can be recreated by natural joining other tables, each one with a subset of attributes of T)

Normalization theory

Steps:

1. Obtaining an ER diagram
2. Transform the diagram into equivalent tables
3. Optimise the previous tables, eliminating undesirable effects

The third phase is carried out with the help of the normalisation theory, which is based on a series of rules that our design must comply with.

These rules are called normal forms, and cause the the tables (belonging to the design) to progressively split into smaller ones with the same information.

Each normal form complies with the previous one.

Normal forms

- The normal forms of the process are:
  - 1NF
  - 2NF
  - 3NN
  - BCNF (Boyce-Codd normal form)
  - 4NF (multi-valued attributes)
  - 5NF (combination or projection)
  - 6NF (primary key + 1 attribute)

1st normal form (1NF)

A database is in 1NF if all its tables also are.

A table will fulfil the first normal form if all of its attributes are atomic → each one has a single value.

This restriction will always be fulfiled by the tables obtained from the transformation of the original E/R diagram.

Solution (when present): split records if they have multiple values

2nd normal form (2NF)

A database is in 2FN if all its tables are in 2FN too.

A given table is in 2NF if it already is in 1NF, and all its non-primary attributes are functionally dependent on the corresponding key.

In other words: there is a functional dependency between the primary key and all the other attributes, so we can guarantee that it has been chosen correctly.

Solution (when present): rectification of the choice of the primary key

3rd normal form (3NF)

A database is in 3FN if all its tables are in 3FN too.

A table is in 3NF if it is already in 2NF, and all of its non-primary attributes do not show any transitive dependencies between them.

If this circumstance occurs (there are dependencies indeed), this is pointing out that part of the table could be fragmented.

Solution (when present): split the table in two, separating the attributes involved in the dependencies (each on a separate table)

Boyce and Codd normal form (BCNF)

A database is in BCNF if all its tables are in BCNF too.

A table is in BCNF if it already is in 3NF, and each determinant of a dependency is a candidate key.

This normal form can sometimes lead to loss of functional dependencies, so it will not always be convenient to apply it.

Example in…

https://en.wikipedia.org/wiki/Boyce-Codd_Normal_Form

Recommendations for obtaining diagrams

- Set stages to identify: entities, attributes and keys
- Then, establish stages to identify: relationships, attributes and degrees
- The design of a database is a creative task → there are no mechanical or automatic rules or procedures, although we can follow some recommendations or principles:
  - Nouns / proper names → entities, attributes and instances
  - Verbs → relationships between entities
  - Prepositions → interrelationships