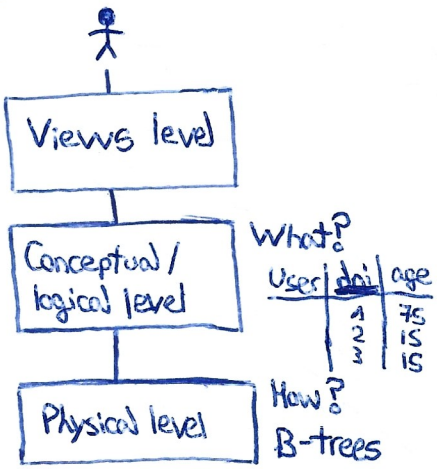
Bases de Datos (BD):

Purpose: (Problems to solve from File system)
- Data Redundancy and Inconsistency
- Dificulty of accesing data
- Data isolation (Multiple Files and Formats)
- Integrity problems (Hard to create, modify constrains)
- Atomicity of updates (Solved with transactions)
- Concurrent access by multiple users
- Security problems.
- Low data independence

Definition:
- Database: Set of interrelated data { data + metadata: Information about the data
- DataBase Management System: Is the software application that interacts with the user, applications and the database to capture and analyze data.
  It allows users to create, update, remove, read data in a database by an interface:
  + Retrieve = Select
  + Modify = - insert - delete - update

Data Models (Types): {They describe: Data, Relationships, Semantics, Constrains}
* Relational Model: Implementation → Defines how the data is stored (in tables (relations), with tuples (rows), attributes (columns), with keys, constrains, ...)
* Entity-Relationship data model: Design → Defines what the database should look like {Entities (Real-world objects), attributes (properties of the entities) and relations (conections between entities)}
* Object-based data models
* Semi-structured data model (XML)
* Network model
* Hierarchical model

Abstraction levels:



Schemas and Instances:
- Schemas are the structure = (Fixed)
  - Logical = What data is stored and how is related
  - Physical = How data is stored (Files, tables, ...)
- Instances are the data at a given point (Changes)

+ Independence of physical data = Means that we can change the Physical schema (level) without affecting the logical one. And vice versa.

Data Definition Language (DDL):
Defines the DB structure (schema) {CREATE, DROP, ALTER

Data Manipulation Language (DML):
Works with the data inside the tables (instances) {Insert, select, Update, Delete
Types:
- Pure → for theoretical and research purposes (More procedural/imperative)
- Commercial → Used in real-world db systems (More declarative)

SQL query language:
- Non procedural (Declarative)
- Takes several tables as input returns always one as output
- Is NOT turing machine equivalent

DB Users:
- Native (Final Users) = No knowledge about db.
- Application Programmers: Use python, C#, Java with db connectivity
- Sophisticated: Query the db directly without writing application programs
- Specialized: Work with non-traditional db like CAD.

Transaction management:
- A transaction is a collection of operations that are performed as a block (cannot be divided). If during the execution a Failure happens, all changes are undone (rollback) to maintain db consistency.
- Ensures that the db remains consistent despite system Failures
- Properties ACID: Atomicity, Consistency, Isolation and Durability

DataBase Administrator (DBA):
- Person who has control over the database.

Entities: (Instances) (A specific person, company, event)
↳ Have attributes: Value, domain

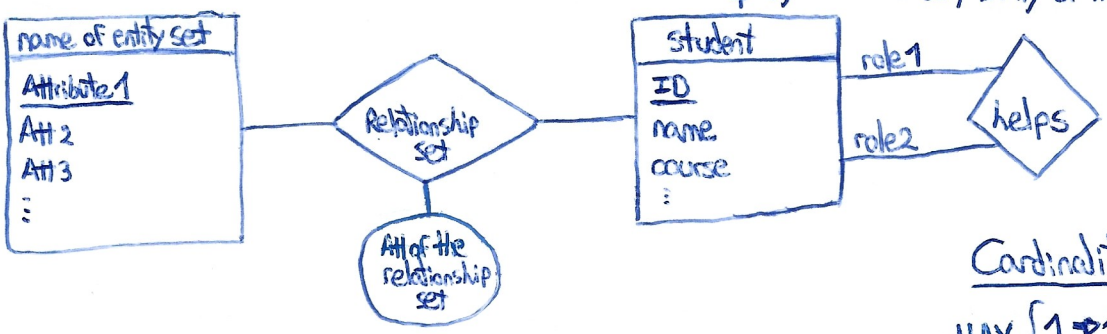
Relationship: Conection of 2 or more entities

Entity Sets: (Classes) (The group of entities of the same type)

Relationship Sets: Collection of similar relations between entities.

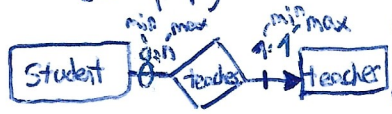
E-R Diagram:

If an attribute is underlined is a primary key: Identifies uniquely each member/ entity of the set/table



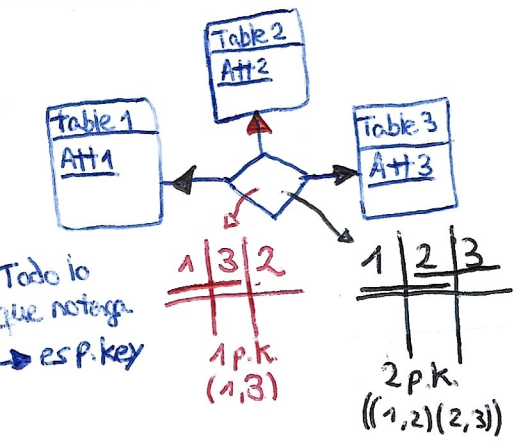
Cardinality:

- MAX { 1 ↔ 1 : (→) or (1); +1 ↔ n : (—) or (n) }
- MIN { ∅ : (0); 1 : (+) }



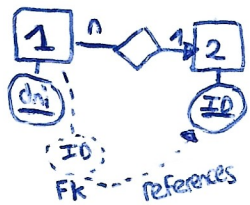
Each student has only 1 teacher
Each teacher can have 1 or more students

Non-binary Rela. Sets:

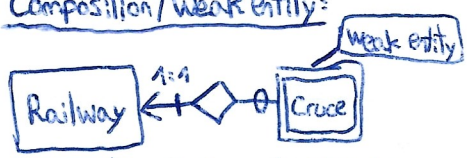


Todo lo que no tenga → es p.key

Foreign keys:

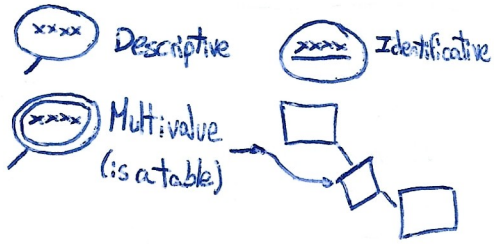


Composition / Weak entity:

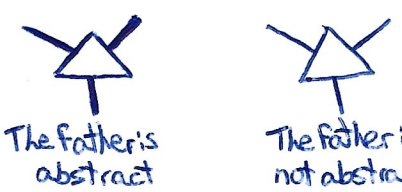


IF railway is removed, cruce is also removed as ◻/ Weak
Pk of railway is also the Pk in cruce

Types of Attributes:



Generalization / Specialization:



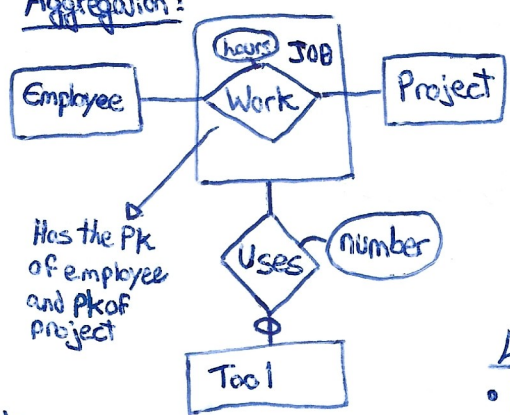
The father is abstract

The father is not abstract

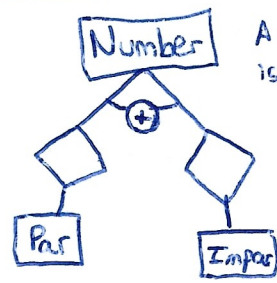
IS A → Cannot be both types

IS A$_S$ → Can be both types of childs

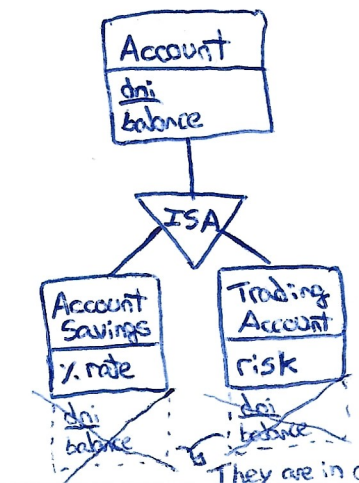
Aggregation:



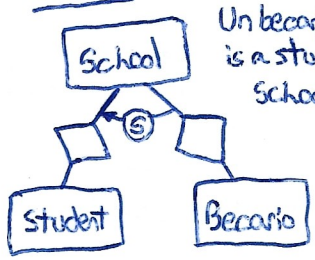
Exclusion: ⊕ → XOR



A number if it is par → is not impar



Subset:



Un becario en x school is a student in the same school

Life cycle of a Data Base:

- Requirement Analysis (of the universe of discourse)
- Conceptual Design (E-R model / view integration)
- Logical Design (E-R → Relational model)
- Refinement Through use (optimization / denormalization)
- Physical Design (Creation of indexes)
- Implementation (constrains, security and applications)
- Testing, monitoring and maintenance

Bases de datos (BD):

Relational Algebra:

Selection $\sigma_p(r)$ → where

Projection $\Pi_{a_1,a_2}(r)$ → Select

Cross Join $r \times s$

Natural Join $r \bowtie s$ (Derived from cross Join)

Rename $\rho_x(r)$

Union $r \cup s$

Diference $r - s$

Intersection $r \cap s = r-(r-s)$

ARMSTRONG RULES:

Reflexivity	$A \to B$	$B \subseteq A$
Augmentation	$A \to B$	$wA \to wB$
Transitivity	$A \to B, B \to C$	$A \to C$
Decomposition	$A \to BC$	$A \to B, A \to C$
Union	$A \to B, A \to C$	$A \to BC$
Pseudotransitivity	$A \to B, Bw \to C$	$Aw \to C$

Definition of $V \to W$ $r(R)$:

$\forall t_1, t_2 \in r \ (t_1[v] = t_2[v] \to t_1[w] = t_2[w])$

Definition Superkey: Candidate key + any other attribute (or not)

Any X where $X^+ = R$ (all posible attributes) (The last superkey is the set of all attributes)

Definition Candidate key:

A minimal superkey where no attributes can be removed without losing $X^+ = R$ (candidate key)

(There can be several candidate keys; one may be chosen as primary key)

Functional Dependencies (constrains):

$F = \{ A \to ABC, \ldots, DE \to FG \}$ = Original set of functional dependencies

F^+ is an equivalent closure

$F_c = \{ PC \to city, \ city \to province, \ DNI \to name, PC \}$ = Minimal cover clousure (without redundancy)

X^+ are the reachable attributes from X, and from the reachable of those, ...

Steps for F_c: (Removing estraneous attributes)

1- Remove trivial dependencies $A \to A$

2- Remove right attributes by transitivity $A \to BC, B \to C \rightarrow A \to B, B \to C$

3- Remove left attributes by pseudotransitivity $A \to B, AB \to C \rightarrow A \to B, A \to C$

Clousure F^+ of F given X:

$R = (A, B, C, \ldots)$ → All posible attributes

$X = (A, B)$ → Set of R (attributes that I have)

$F = \{ AB \to C, C \to DE, \ldots \}$

$(A,B)^+ = \{A, B, C, D, E, \ldots\}$

If $X^+ = R \to X \in F^+$

And X is a superkey

Steps for 3NF:

0. F should be F_c
1. Each dependency $X \to A$ is a table: $R_i = (X, A)$ if $X \to A, X \to B$ then $R_i = (X, A, B)$
2. If no candidate key is in any table $k^+ = R \to R_i = (k)$ (Create a table with 1 candidate key)
3. If $R_A(U)$ with F_A and $R_B(V)$ with F_B and $U \in V$ then check if F_A is in F_B and drop $R_A(U)$

Boyce-Codd Normal Form:

For each R_i either:

- $X \to y$ is trivial
- X is a superkey

3NF:

For each R_i either:

- R_i is in BCNF
- All Y is prime (part of a candidate key)

Is more important to Preserve Dependencies that Remove Redundancy

Decomposition:

$$r(R) \rightarrow r_1(R_1), r_2(R_2), \cdots, r_n(R_n)$$

Types:

- Lossy (very bad): Phantom rows appear $r < r_1 \bowtie r_2 \bowtie r_3 \bowtie \cdots r_n$
- Lossless (don't lose info): $r = r_1 \bowtie r_2 \bowtie \cdots \bowtie r_n$ $\quad R = R_1 \cup R_2 \cup R_3 \cdots \cup R_n$

Dependency Preservation: (DP)

$$F \leftrightarrow F_1 \cup F_2 \cup F_3 \cup \cdots \cup F_n$$

Example of relational algebra:

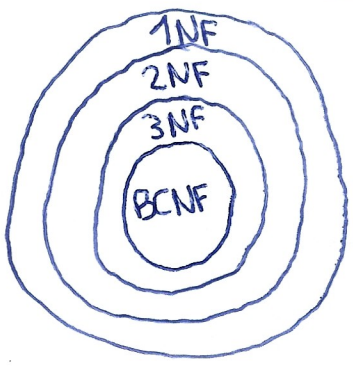
Select DNI
From Person
Where DNI>10
OR name≠'Paco'

$$\rightarrow \Pi_{(DNI)}\left(\sigma_{(DNI>10)}(Person) \cup \sigma_{(name \neq 'Paco')}(Person)\right) = \Pi_{(DNI)}\left(\sigma_{(DNI>10 \;//\; name \neq 'Paco')}(Person)\right)$$

```
Select e.Name AS EmpName, d.DeptName AS Dept
FROM Employee As e INNER JOIN Department AS d ON e.DeptID = d.DeptID
WHERE e.Salary > 50000;
```

$$\rho_{F(EmpName, Dept)}\left(\Pi_{(e.Name, d.deptName)}\left(\sigma_{(e.Salary>50000)}\left(\rho_e(Employee) \bowtie_{e.deptID=d.DeptID} \rho_d(Department)\right)\right)\right)$$

Types of Normal Forms:



1NF: Atomic (Indivisible) Values for every attribute (No lists/sets)
Primary keys in each table
No repeated columns with Phone, Phone1, ...
Same domain and type for each row

2NF: 1NF + No nonprime attribute is partially dependent on any candidate key.

3NF: 2NF + Every non-trivial FD $(X \rightarrow A)$ either
- X is a superkey of R
- Every Y attribute is prime (part of a candidate key)

BCNF: $\forall X \rightarrow X$ (trivial) or $X \rightarrow Y$ (non-trivial: $X \rightarrow X$) where X is a superkey of R

Normalization Steps:

0. Start with a canonical cover F_c
1. Compute candidate keys for R.
2. For each Functional dependency in F: $(X \rightarrow A)$
   - Generate an R_i
   - Compute the F_i constraint of F for R_i
   - Compute candidate keys for R_i
3. If a dependency is repeated in several R_i leave 1.
4. If no candidate key of R appears in a R_i create an additional R_i with $F_i = \{\}$
5. Mark the normal form (3NF or BCNF) of every R_i
6. Check lossless join (with $\bowtie$)
7. Check dependency preserved ($F \leftrightarrow F_1 \cup F_2 \cup \cdots \cup F_n$)