Database: A way to store the data ensuring the persistent content and efficient access.

Database System (DBS): Computer storage systems for the manipulation of related data volume in centralized systems. Meaning that you can:

- Insert new data in existing files
- Delete and add files
- View data content
- Update data.

DBMS: Software that allows to create, keep and access a DB. It manages and guarantees many properties of a DBS (propagation of changes, access control…)

Paradigms of the DBS

Relational: It has a logical structure that always keeps the same type of information. Meaning that if I can not put a number in a column where I have to put the names. If I use 3 columns as a description of a student (DNI, course and grade), I always have to use these 3 columns.
Non-Relational: It has a flexible logical structure that allows to keep the same type of information. Meaning that I can describe a student with 3 columns or 35.

They both guarantee ACID properties: Atomicity, Consistency, Isolation and Durability.

Requirements for a DBS

In order to guarantee a proper operation of all applications using data stored in a DB, the system must:

- Guarantee the data integrity
- Be efficient
- Grant independency between program code and data format

Integrity

Content **consistency** and **coherence** of information stored in the DB.

- Example of incoherence is a DNI with a negative value. Impossible value of the requested information
- Example of inconsistency is that the same query gives different results depending on its implementation. More than one grade for an exam.

Avoid **redundancy** which are incorrect or duplicated values in the requested information.

- Example of redundancy is having multiple key values, when you only need one. Having the DNI, NIA, name in the same table.

It has the following problems:

- Unnecessary disk usage
- Duplication on updates
- Risk of inconsistencies

DBS must guarantee data **integrity when content is modified**. Update on information to all files if one is changed **(change propagation).**

In excel, changing a value in a file is also going to change the same value in other files if it is well “programmed”. If we do not do this, we will have inconsistencies because in one file we will have one value and in another file another value.

- Inserts and updates must be adjusted to the defined **domain** (coherence). For domain, we mean a set of rules that we see for each column or attribute. For example, a DNI can not be negative.

Efficiency

DBS should minimize the maximum time for a query (given for access to all records) based on contents and functional needs of the DB:

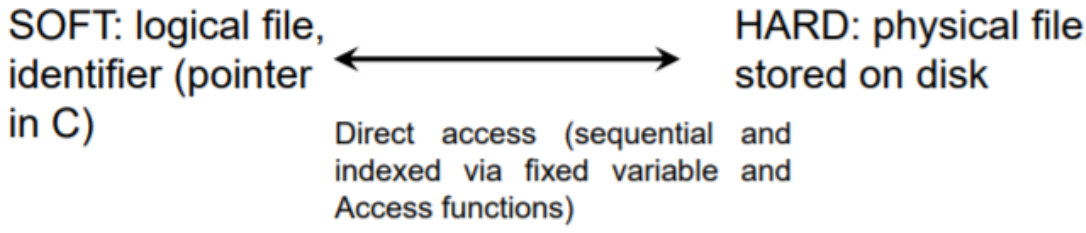
- Storage Structures. Restructuring the disk information distribution (minimize access to disk). How is the data structured inside the disk…
- Compression Techniques: DBS compress the information to be stored into disk (minimize content). How the data is sended or received.
- Characteristics of the devices where the data is stored. Imagine that we have a server with only 6 Gb of RAM and we have to control the connection of more than 1000000 people.
- Concurrence Control of accesses of several users (priorities).

Independence

Imagine that we have a file where the data is stored. Then we need a software to extract the data. This software is the one that controls the DB (because a DB is just a bunch of files).

Which problem will we have if a program that directly accessed these files? In python we define functions with a number of parameters, and if we want to add a new parameter we need to change the function… otherwise we will have an error.
This also happens in DB. Imagine that we add a column in a file (or change something in the design of the DB). I will have to change my way of doing the queries, because now I am not asking for the 6th column but the 7th column (for example).

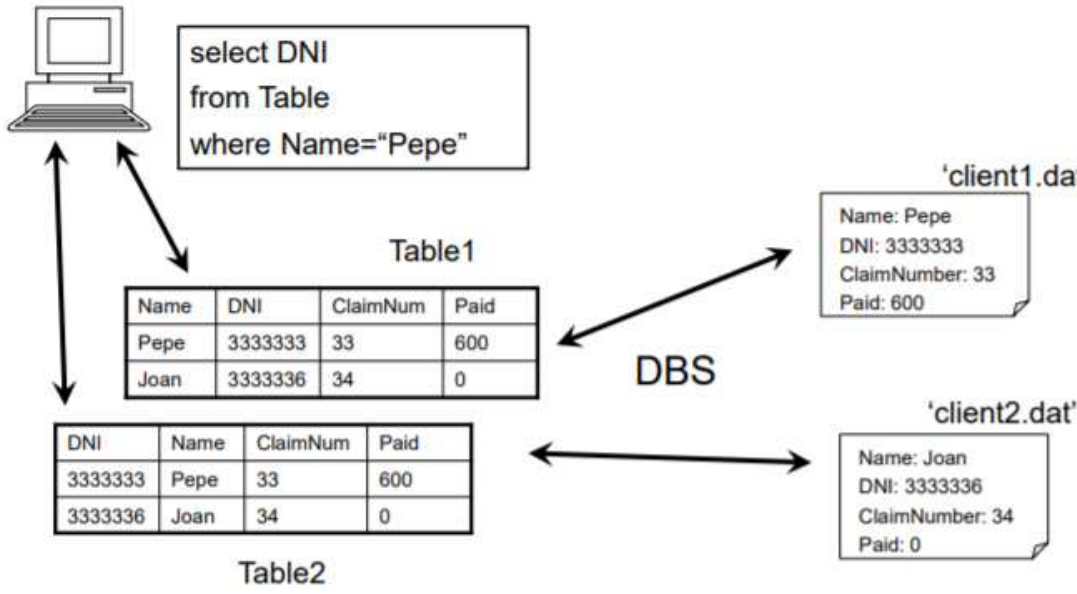
This is not optimal
So, we need something in the middle, between SOFT and HARD that works as a brain.



So, I want to make an independence from the application that is connecting to the database and the real place where the data is stored (files). This is called “independence between the physical and logical part”.

A table is a logical object from which you access the data. These tables are inside and controlled by the DBMS (the element that is in the middle of the physical and logical part working as a brain).

So, to reach the information we first have to go through an object (table).



DBMS must ensure that changes in the storage structure and data access do not affect the applications that use them (because the DBMS is in the middle solving these problems). This is why we say "independence between the logical and physical part".

Components of a DBS

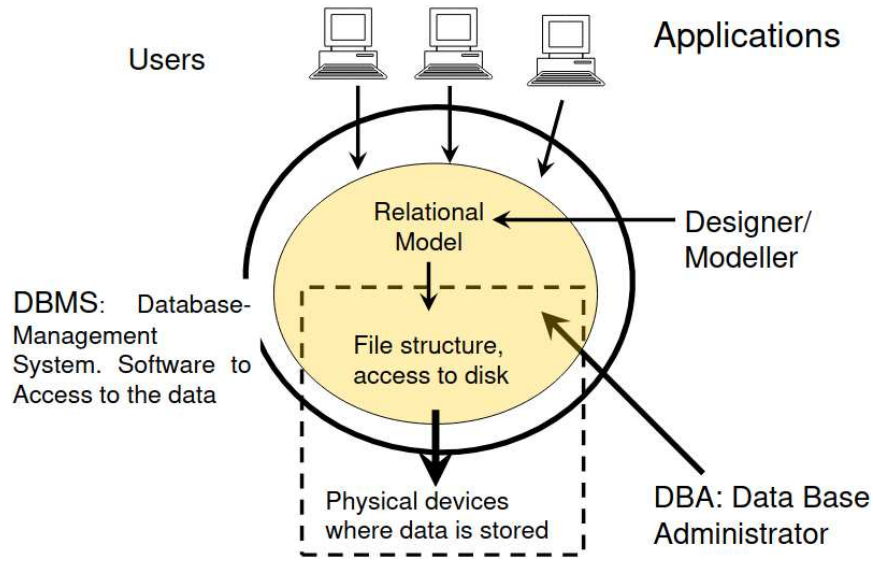
Computers

- DB
- Hardware where the DBS is installed
- Database-Management System (DBMS) is the software. It makes the communication or translation from the humans to the DB and vice versa.
- Applications

Humans

- Modelers
- DB administrators
- Programmers
- Final users

Note: DBS are the tables and DBMS is the brain that controls the tables, the access to the data and the queries from users.



We have the users with applications that connect to a DBS (black circle). Inside the DBS we have the relational model, a composition of tables where we store the information.
This relational model is created by a Designer or Modeler.

Most common architectures:

- ANSI/SPARC
- Client-Server

ANSI/SPARC

Structure the DB according to the description of the data in 3 levels of abstraction:

- External level: Presentation of the data to users (applications that use a relational model)
- Conceptual level: Logical description of data: tables (relational), collections (non-relational)
- Internal level: Organization and storage on physical files (low-level, pointers, indexes…)

Front-end / Back-end

Structuring the DB according to the type of application (software) that are executed on the DB in 2 levels:

- The front end are the applications (client) that connect to the external level. In our case, SQL developer.
- The back end is the software which executes all functions specified in a DBMS.

Since there are infinite views on the external level, we also have infinite front-ends. But only one back-end.

Vocabulary

Candidate key: Minimum set of attributes that uniquely identify each instance.

Primary key: CK selected by the database designer that uniquely identifies an instance. An entity without PK is not well defined.

Foreign key: Field in one table that refers to the PK of another table.

Discriminant: Set of attributes which allow to distinguish between instances of the weak entity that depend on the same instance of a strong entity

Relationship: Association between different related entities instances. Each instance of the relationship is defined by the values of the PKs of the associated entities instances.

Entity: Physical objects that you want to represent in the DB

Weak entity: Entity that has not enough attributes to form a PK and it requires the FK of a strong entity to be identified.

Strong entity: Entity that has enough attributes to form a PK.

Aggregation: Group of relationships and participating entities into a new entity

Specialization: Partition of an entity in specific groups (low-level entities).

Generalization: Process summarizing multiple entities into a high level entity based on common characteristics (the reverse process of specialization).

Simple attribute: Characteristics of an entity. Attributes that have a single value for each instance.

Composite attribute: Attribute that can be splitted into simpler attributes. Date of birth can be splitted into day, month and year. Note that the pieces are always of the same type (otherwise it should be an entity)

Multivalued attribute: The attribute has more than one value for each instance (they are “vectors” of variable length for each instance). Example: Dani alves has 3 phone numbers.

Derivative attributes: Attribute with a value that depends on the value of other attributes

Cardinality: Maximum number of instances of an entity that may be associated with an instance of the other entity involved in a relationship.