

Unit 1: Information storage and retrieval systems.

2022/23

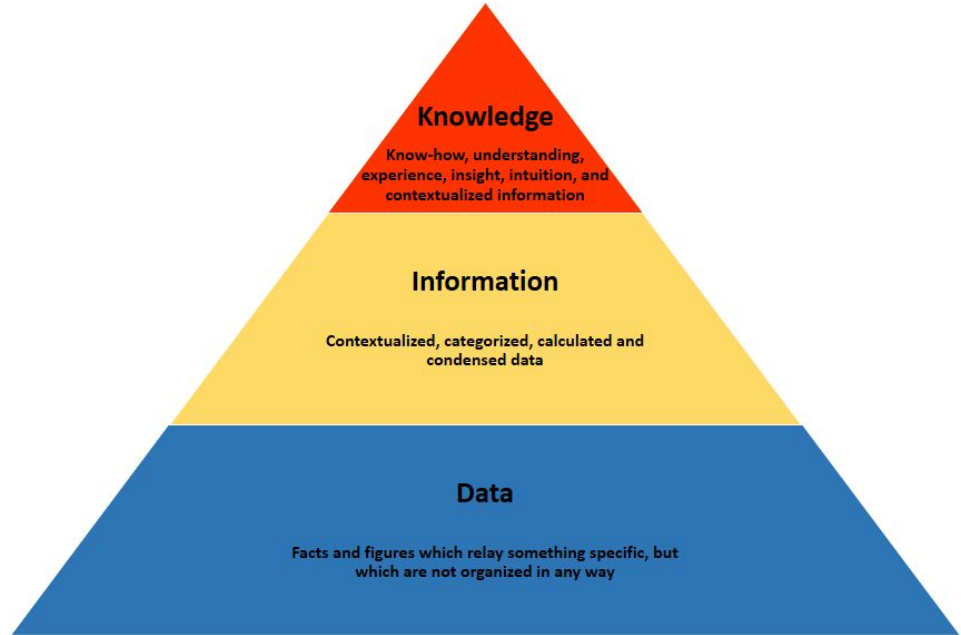
Goals

- Students completing this unit can:
 - Analyze **logical storage systems** and their functions.
 - Identify **types of databases** according to their data model.
 - Identify types of databases according to location of information.
 - Assess the usefulness of a **database management system (DBMS)**.
 - Describe the **functions and advantages of a DBMS**.
 - Identify the **components of a database management system**.
 - Describe the **characteristics of the different logical data models**.
 - **Classify different DBMS**.
 - Distinguish the **physical, conceptual and internal schemas** of a given database.

Contents

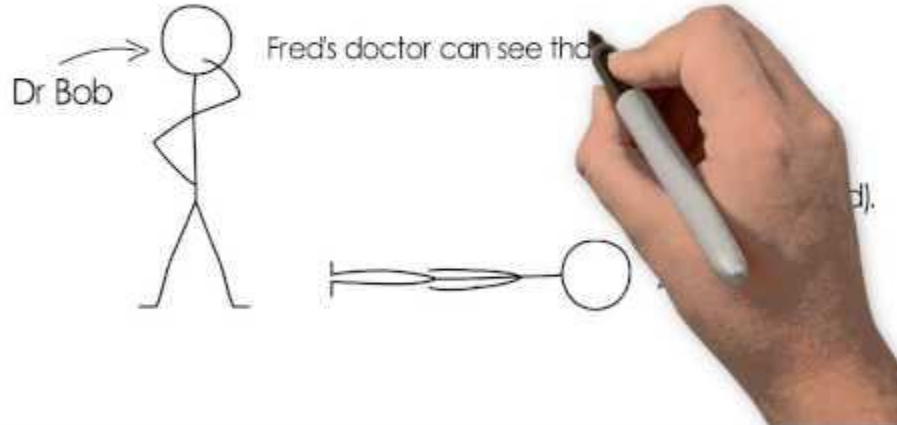
- 1.1. Data, information and knowledge.
- 1.2. Information Retrieval and Storage System.
- 1.3. Files.
- 1.4. Databases.
- 1.5. Database management systems (DBMS).
- 1.6. Centralized and distributed databases.

1.1. Data, information and knowledge.



1.1. Data, information and knowledge.

Data + Context = Information.



Data, Information and Knowledge

Topic 4.1.01

Charlie Broomfield
@grunitedias

What is data?

130/90
140/90
160/100
170/100

This is data.
Data is raw facts or figures.
Data has no context.

What do these figures mean?

We learn that these figures are Fred's blood pressure readings.
The data now has context and turns from data into information.



Data + Context = information.

Dr. Bob



Fred's doctor can see that his blood pressure has increased to dangerous levels and prescribes him some medication immediately.

Fred, 'Poor Fred'.



Summary

Data is raw facts or figures
information is data that has been processed by a computer and has context
Knowledge is derived from information by applying rules to it

Information + Rules = Knowledge

Information becomes knowledge when we apply rules to it. In this case the rules about healthy and unhealthy blood pressure are applied to Fred's readings.

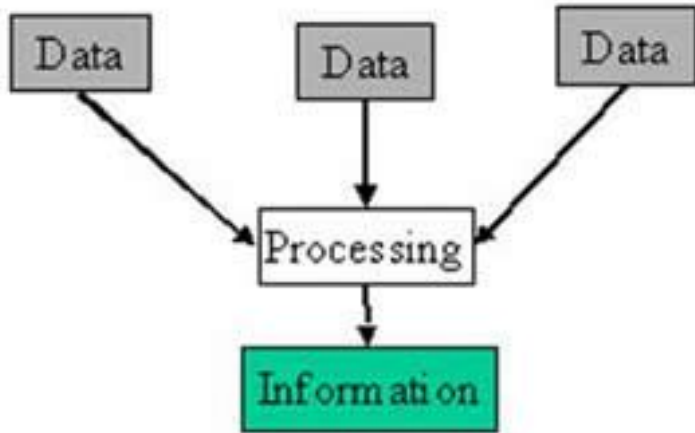


Fred gets better, and lives happily ever after.

Author:
[Charlie Broomfield](#)

1.1. Data, information and knowledge.

Information is created from data



Data is nothing but facts and statistics stored or free flowing over a network, generally **it's raw and unprocessed**. For instance: Your name, your address, your age, your favourite football team, etc.

Data becomes **information** when it's processed, turning it into something meaningful. For instance: When an enterprise processes the ages of the employees to fire many of them.

data + context = information

1.2. Information Retrieval and Storage System.



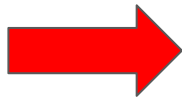
1.2. Information Retrieval and Storage System.



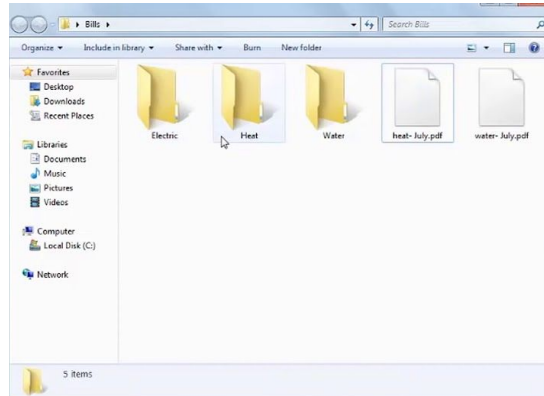
1.2. Information Retrieval and Storage System.

- An **Information Retrieval (and Storage) System** is a system that is capable of storage, retrieval, and maintenance of information. Information in this context can be composed of **text** (including numeric and date data), images, audio, video and other multimedia objects. **Evolution:**

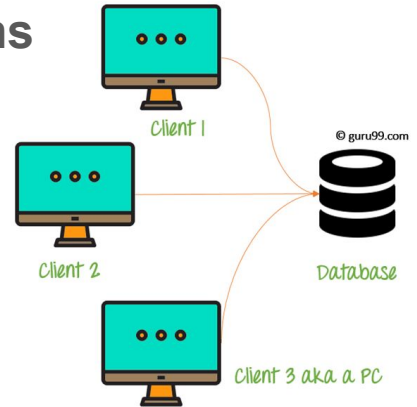
Manual files and folders



Computer file systems



Database management systems



1.2. Information Retrieval Systems.

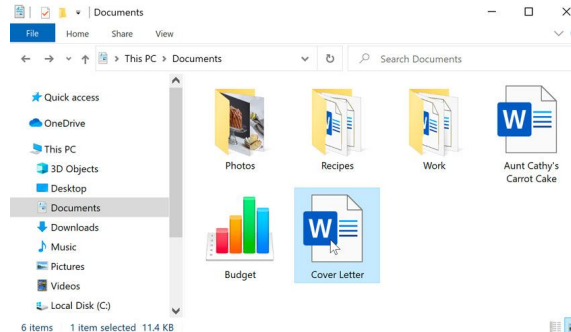


Manual files and folders: ➡ **Computer file systems:** ➡

Set of labeled folders whose content was related and stored in a file cabinet.



Data stored in computer files and application programs accessed them to obtain requested reports for decision-making.



Database management systems:

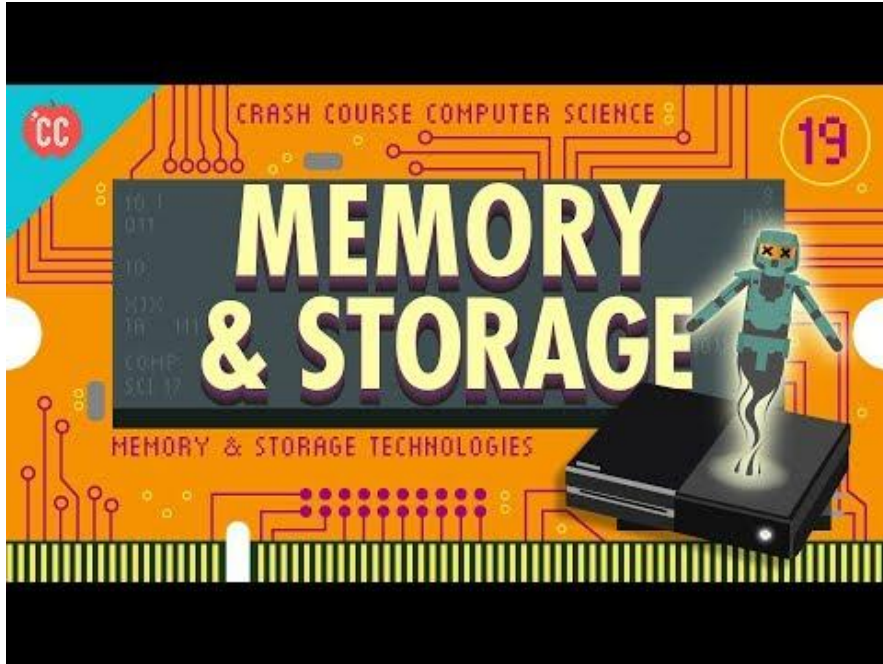
A database is a set of related data organized and structured with information about something.

Any change in the structure of data will not affect the application programs that use it. Information is managed using DBMS.

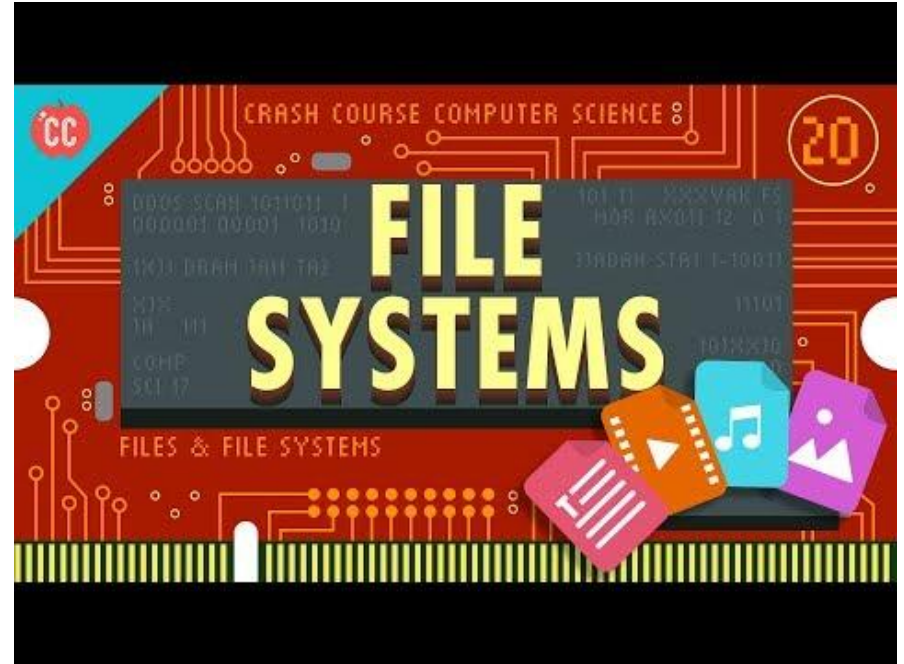
1.3. Files.



1.3. Files: Introduction.



Memory & Storage: Crash Course Computer Science #19

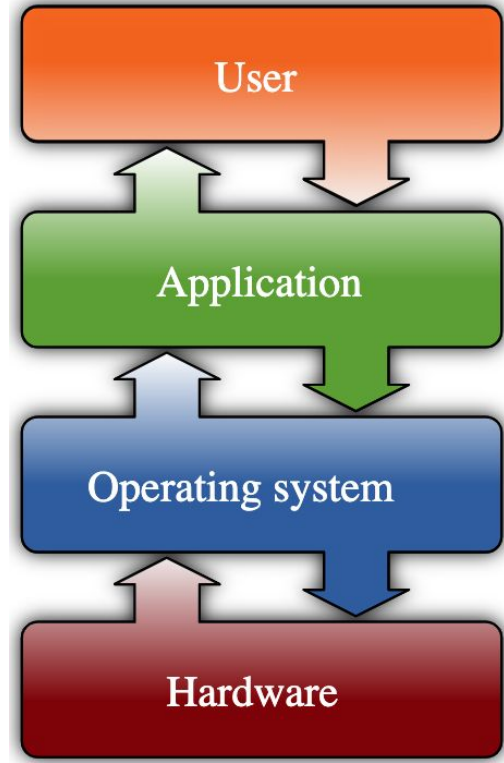


File Systems: Crash Course Computer Science #19

1.3. Files: File systems.

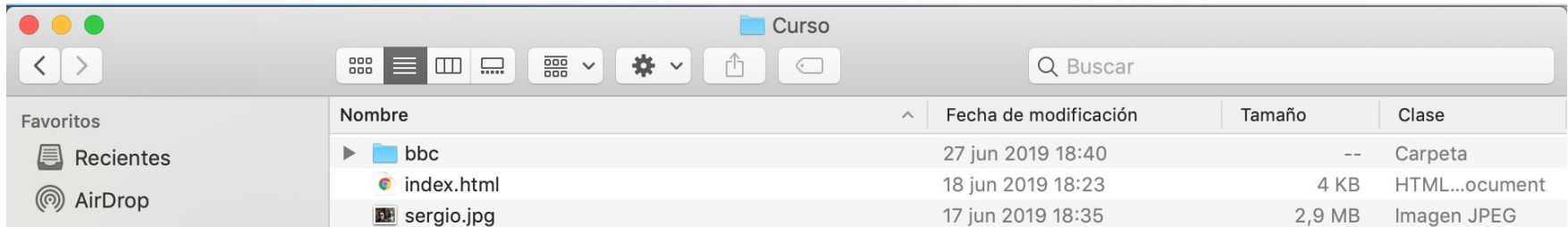
- “A **file system** (or filesystem) controls how data is stored and retrieved in a computer:
 - Without a file system, information placed in a storage medium would be one large body of data with no way to tell where one piece of information stops and the next begins (storage mediums are designed to store 0s and 1s).
 - By separating the data into pieces and giving each piece a name, the information is easily isolated and identified. Taking its name from the way paper-based information systems are named, each group of data is called a "file".
 - The structure and logic rules used to manage the groups of information and their names is called a 'file system'.”

(Wikipedia)

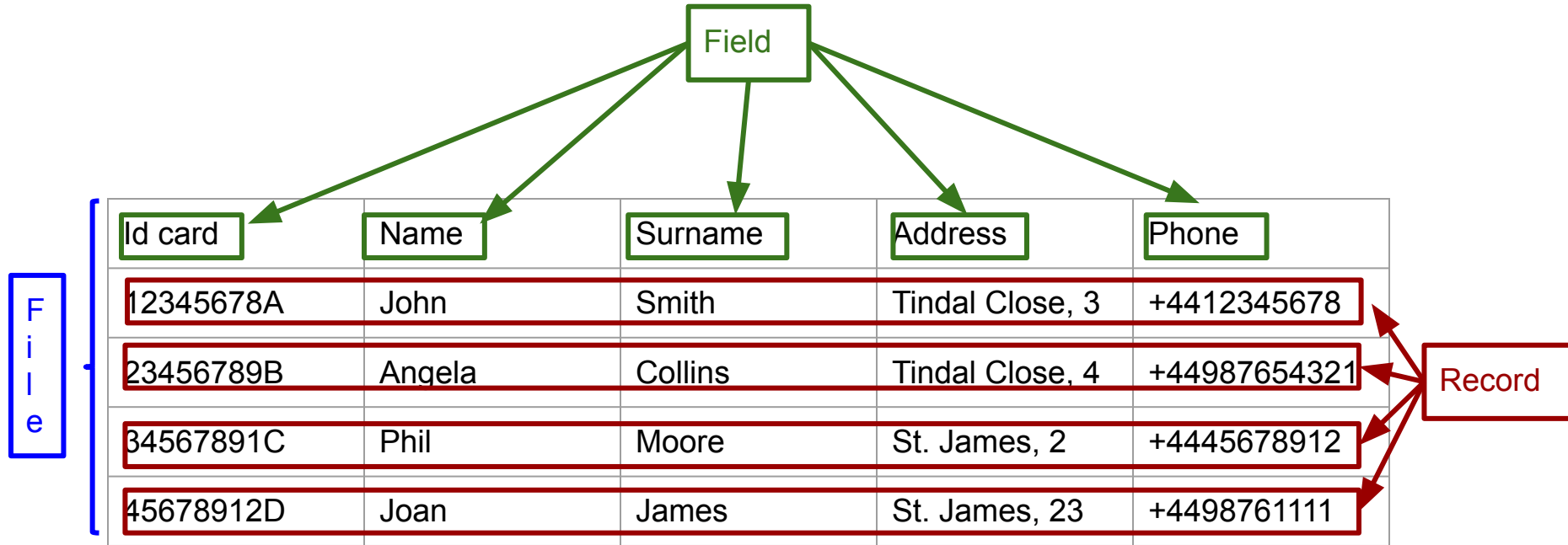


1.3. Files: Definition.

- **File:** A folder or box for holding loose papers together and in order for easy reference.
- **Computer file:** “A file is a container in a computer system for storing information. Files used in computers are similar in features to that of paper documents used in library and office files. There are **different types of files** such as text files, data files, directory files, binary and graphic files, and these different types of files store different types of information. In a computer operating system, files can be stored on optical drives, hard drives or other types of storage devices”. (techopedia.com)



1.3. Files: Concepts.



1.3. Files: Concepts.

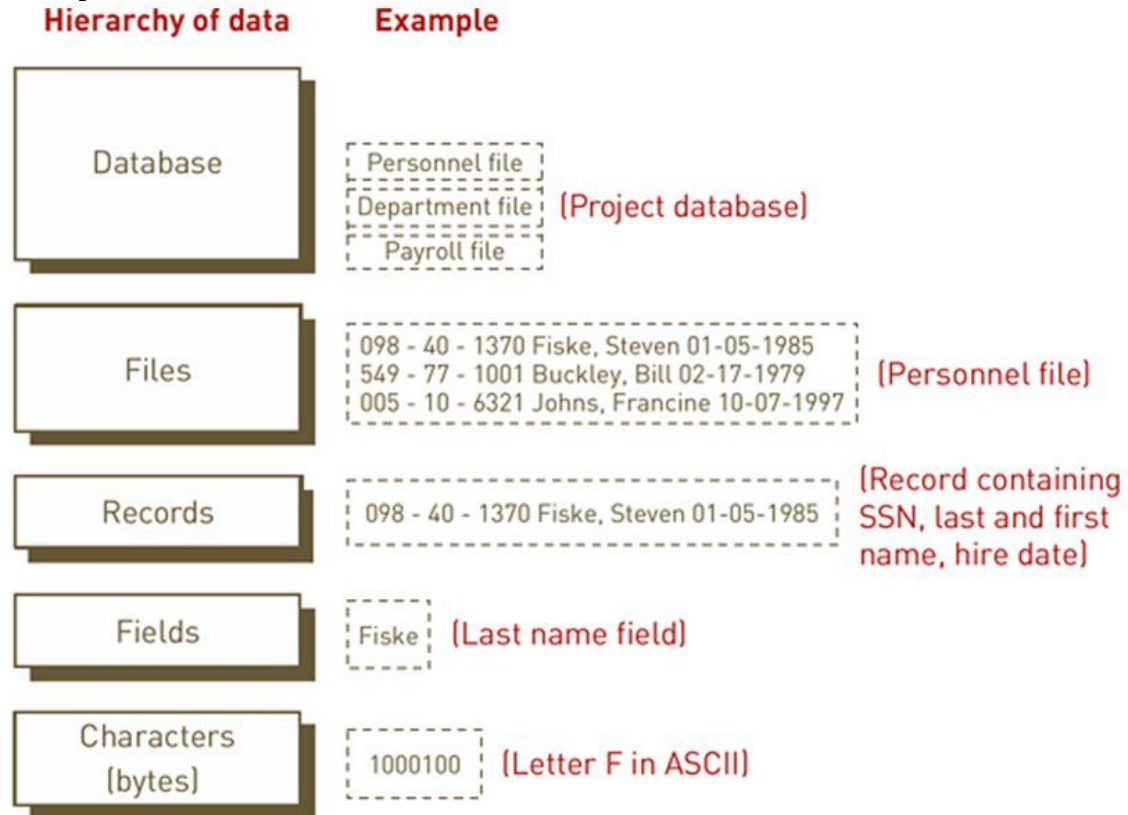
- **Concepts about files:**
 - **Data:** In this case, the information we need to store.
 - **Field:** It is a character or set of characters that has specific meaning. It is used to define and save data. It is the smallest unit of information created with sense by itself.
 - **Record (Logical Record):** It is a set of logically related fields that describe a person, place or thing. It is also the unit of treatment of data files.

1.3. Files: Concepts.

- **Concepts about files:**

- **File:** It is a set of related records (in this example file with customer data).
- **Block (Physical Record):** Amount of information that is transferred between the media in which the file is stored and the main memory from the computer in a single R/W operation; since it is impossible to have the entire file in memory because of its size to process it.
- **Blocking factor:** Number of records that fit into a physical block and are transferred in an R/W operation.

1.3. Files: Concepts. The Hierarchy of Data



1.3. Files: Operations.

- **File operations:**

- **Open:** Prepare the file to process it.
- **Close:** Close the file (to prevent processing it).
- **Read:** Get information from the file.
- **Write:** Save information to the file.
- **Seek:** Place the reading pointer in a specific position (it cannot be done in all types of files).
- **End of file (eof):** To point out if we have reached the end of the file.

```
#!/usr/bin/env python3
```

```
print("I'm a simple program to read a file!\n")
```

```
path = input("Enter a path to a file: ")
```

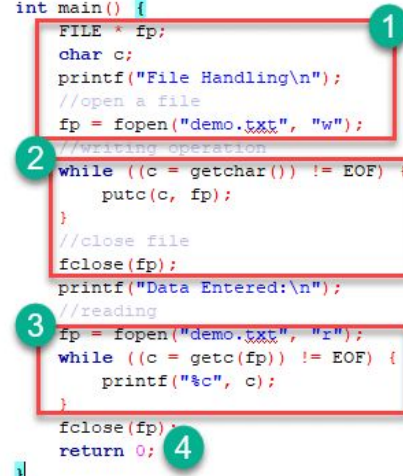
```
f = open(path, "r")
```

```
for x in f:
```

```
    print(x)
```

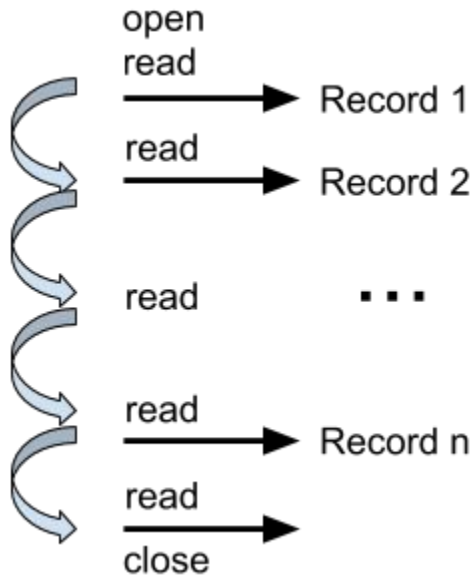
```
f.close()
```

```
#include <stdio.h>
int main() {
    FILE * fp;
    char c;
    printf("File Handling\n");
    //open a file
    fp = fopen("demo.txt", "w");
    //writing operation
    while ((c = getchar()) != EOF) {
        putc(c, fp);
    }
    //close file
    fclose(fp);
    printf("Data Entered:\n");
    //reading
    fp = fopen("demo.txt", "r");
    while ((c = getc(fp)) != EOF) {
        printf("%c", c);
    }
    fclose(fp);
    return 0;
}
```



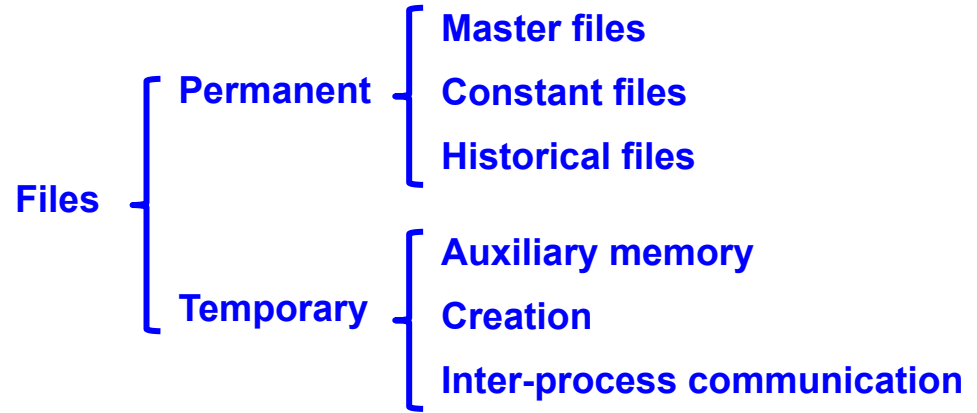
1.3. Files: Operations.

Operations sample in a sequential file:



Field 1	Field 2	...	Field m
End Of File (EOF)			

1.3. Files: Types.



- **Basically, two types of files:**

- **Permanent:** Usually contain information relevant to an application. They contain the current state of the data that can be modified from the application (**master files**), fixed data for the application (**constant files**), or data that were considered current in a previous period or situation (**historical files**).
- **“Temporary files**, or foo files (.TMP), are files created to temporarily contain information while a new file is being made. It may be created by computer programs for a variety of purposes; principally **when a program cannot allocate enough memory for its tasks, when the program is working on data bigger than the architecture's address space, or as a primitive form of inter-process communication.**” (Wikipedia)

1.3. Files: Devices.

- **Types of devices (access mode):**
 - **Sequential access media:** Store records sequentially. It is sometimes the only way of accessing the data, to access a record you must process all the records before it. For instance, **magnetic tape data storage**.
 - **Direct-access storage device (DASD):** It is a secondary storage device in which each physical record has a discrete location and a unique address. Store either sequential or direct access files. For instance, **hard disk drives**.

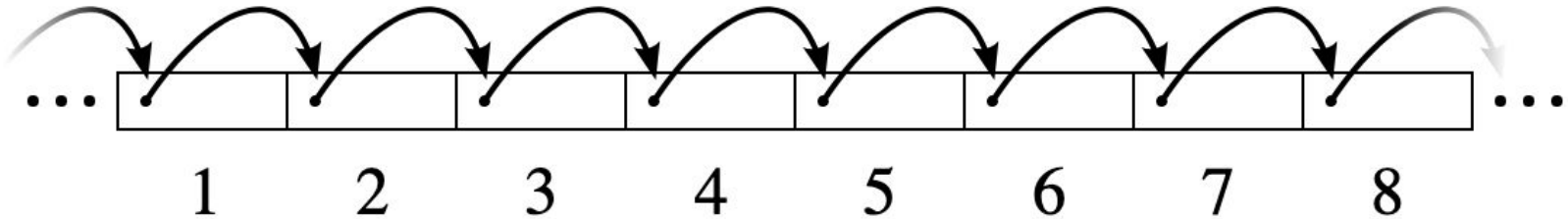


1.3. Files: Access Methods.

- **Access to a file record:** Procedure used to select it. The procedure is linked to the type of support where the file is stored.
- **Access methods:**
 - Sequential
 - Direct or random access

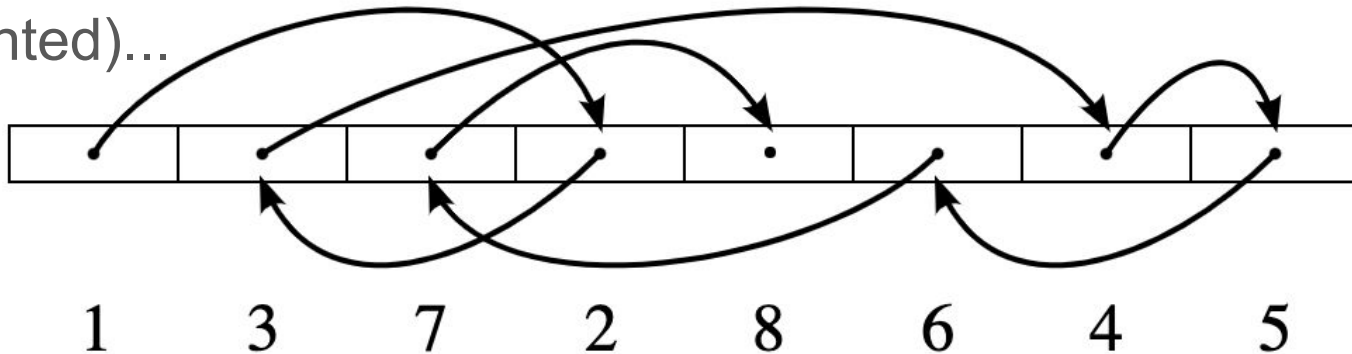
1.3. Files: Sequential Access.

- The records are read **sequentially from the beginning of the file until** the searched record is found or until the end of the file (**eof**) is reached.
- **Device types:** Sequential, direct-access.
- A modern example can be a **cassette tape**...



1.3. Files: Random Access.

- **Random access** (or direct access) is the ability to access an arbitrary record of a sequence in equal time as any other. It is typically contrasted to sequential access.
- **Device types:** Direct-access.
- A modern example can be a CD (you can skip to the track wanted)...



1.3. Files: File Allocation Methods.

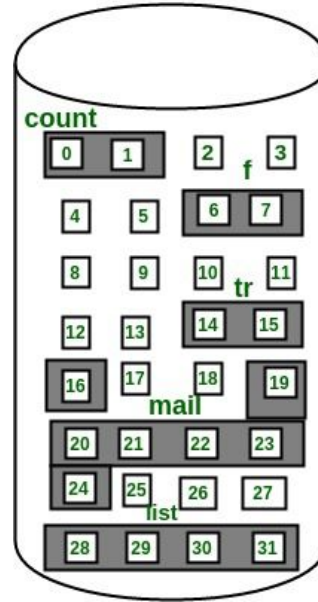
- The **allocation methods** define how the files are stored in the disk blocks. There are three main file allocation methods:
 - **Contiguous Allocation**
 - **Linked Allocation**
 - **Indexed Allocation**
- The main idea behind these methods is to provide:
 - **Efficient disk space utilization.**
 - **Fast access to the file blocks.**

1.3. Files: Contiguous Allocation.

- In this case, **each file occupies a contiguous set of blocks on the disk.**
- If a file requires **n blocks** and is given a **block b as the starting location**, then the blocks assigned to the file will be: **b, b+1, b+2, ... b+n-1.**
- **blocks occupied by the file = ending block - starting block + 1**

1.3. Files: Contiguous Allocation.

- The **directory entry** for a file with **contiguous allocation** contains
 - Address of starting block
 - Length of the allocated portion.
- The file 'mail' in the following figure starts from the block 19 with length = 6 blocks. Therefore, it occupies 19, 20, 21, 22, 23, 24 blocks.



Directory

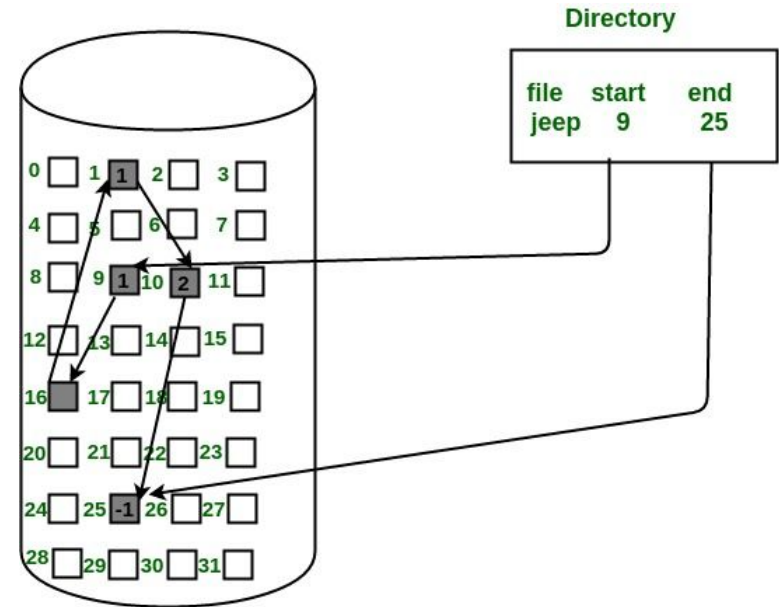
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

1.3. Files: Contiguous Allocation.

- Advantages:
 - Both the **Sequential and Direct Accesses** are supported by this. For direct access, the address of the kth block of the file which starts at block b can easily be obtained as $(b+k)$.
 - This is **extremely fast** since the number of seeks are minimal because of contiguous allocation of file blocks.
- Disadvantages:
 - This method suffers from both **internal and external fragmentation**. This makes it **inefficient** in terms of **memory utilization**.
 - **Increasing file size** is difficult because it depends on the **availability of contiguous memory** at a particular instance.

1.3. Files: Linked List Allocation.

- Each file is a **linked list of disk blocks** which need **not** be **contiguous**. Blocks can be scattered anywhere on the disk.
- The directory entry contains a pointer to the starting and the ending file block. **Each block contains a pointer to the next block occupied by the file.**
- The file 'jeep' in following image shows how the blocks are randomly distributed. The last block (25) contains -1 indicating a null pointer and does not point to any other block.



1.3. Files: Linked List Allocation.

- Advantages:

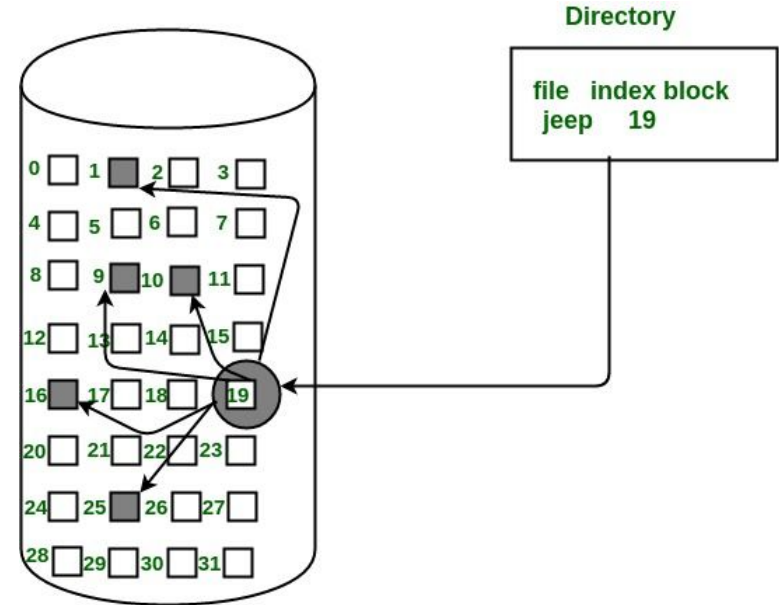
- This is very **flexible in terms of file size**. File size can be increased easily since the system does **not** have to look for a **contiguous chunk of memory**.
- This method does **not** suffer from **external fragmentation**. This makes it relatively better in terms of memory utilization.

- Disadvantages:

- Because the file blocks are distributed randomly on the disk, a **large number of seeks are needed to access every block individually**. This makes linked allocation slower.
- **It does not support random or direct access**. We can not directly access the blocks of a file. A block k of a file can be accessed by traversing k blocks sequentially (sequential access) from the starting block of the file via block pointers.
- **Pointers required** in the linked allocation incur some extra overhead.

1.3. Files: Indexed Allocation.

- In this scheme, a special block known as the **Index block** contains the pointers to all the blocks occupied by a file. Each file has its own index block. The i th entry in the index block contains the disk address of the i th file block. The directory entry contains the address of the index block as shown in the image:



1.3. Files: Indexed Allocation.

- Advantages:

- This **supports direct access** to the blocks occupied by the file and therefore provides fast access to the file blocks.
- It **overcomes** the problem of **external fragmentation**.

- Disadvantages:

- The pointer overhead for indexed allocation is greater than linked allocation.
- For very small files, say files that expand only 2-3 blocks, the indexed allocation would keep one entire block (index block) for the pointers which is inefficient in terms of memory utilization. However, in linked allocation we lose the space of only 1 pointer per block.

1.3. Files: More complex allocation schemes.

- For **files** that are **very large**, **single index block** may not be **able to hold all the pointers**. Following mechanisms can be used to resolve this:
 - **Linked scheme**
 - **Multilevel index**
 - **Combined Scheme**

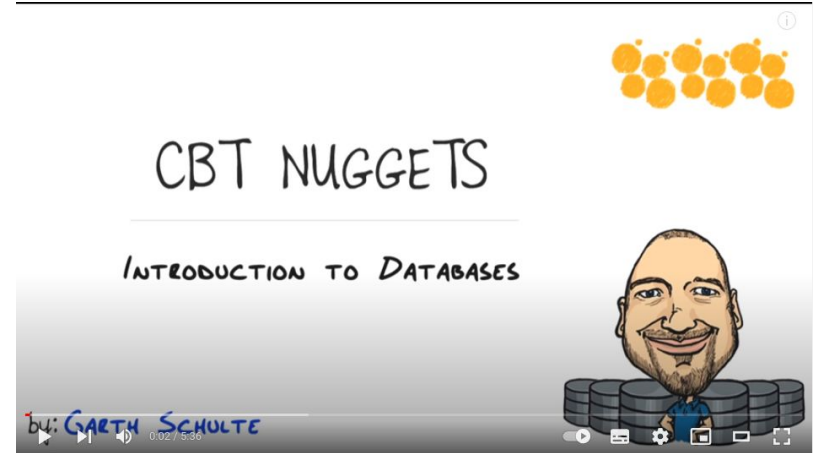
1.3. Files: All this is more complex.

- **Nevertheless, all the topics about files are more complex and you'll see in subjects about operating systems...**

1.4. Databases.



1.4. Databases: DBMS and DB?



1.4. Databases: DBMS and DB?

Database Management System (DBMS):

- A collection of data related to each other, structured and organized, and
- A set of programs to access and manage this data.

The collection of these data is called **database** (DB). In other words, a database is a set of organized and integrated data that belongs to the same context systematically stored for later use. In this sense, a library could be considered as a database composed mostly of papers and textbooks printed on paper and **indexed to query them**.

1.4. Databases: Working with files.

- **Before DBMS** (70s), **information was managed using files** in the filesystem supported by the operating system running (data was stored in file systems organized into records with fields).
- Thus, the **definition of the data** -and also the **access and manipulation-**
was coded within the application programs instead of being stored independently.
- This is a great **inconvenient** when you work **with very large volumes of information**. It's a **better idea** to **separate data** contained in files **from programs** that use it.
- **The aim** is to structure and organize data in such a way that it can be accessed independently of the programs that manage them.

1.4. Databases: Working with files.

Disadvantages of storing data in files:

- Data redundancy and inconsistency.
- Physical-logical data dependence.
- Difficulty accessing data.
- Data separation and isolation.
- Difficulty for concurrent access.
- Dependence on the structure of the file with the programming language.
- Data security problems.
- Data integrity problems.

1.4. Databases: Working with files.

- **Data redundancy and inconsistency:**
 - Redundancy -> Data duplicated
 - Inconsistency -> Duplicated data differs

Employees' Skills

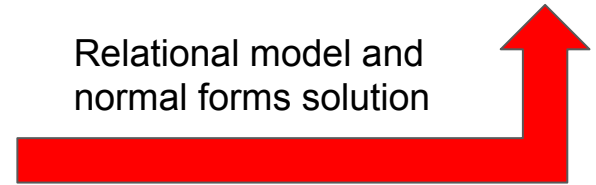
Employee ID	Employee Address	Skill
426	87 Sycamore Grove	Typing
426	87 Sycamore Grove	Shorthand
519	94 Chestnut Street	Public Speaking
519	96 Walnut Avenue	Carpentry

Employee ID	Name	Surname	Address
426	John	Smith	87 Sycamore Grove
519	Michael	Jordan	94 Chestnut Street

Employee ID	Skill ID
426	TYP
426	SHO
519	CAP
519	PSP

Skill ID	Name
TYP	Typing
SHO	Shorthand
PSP	Public Speaking
CAP	Carpentry

Relational model and
normal forms solution



1.4. Databases: Working with files.

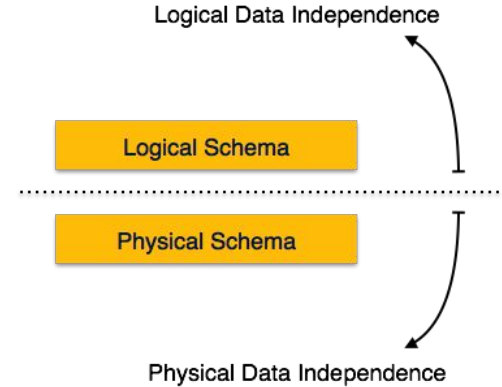
- **Physical-logical data dependence:**

- Physical structure of the data -definition of files and records- is coded in the application programs. Any change in that structure involves the programmer identifying, modifying and testing all the programs that manipulate those files.

- **Difficulty in accessing data:**

- For instance, when a new query is needed -that was not defined at the beginning- it implies the need to code the necessary application program. And that isn't efficient...

This is difficult to understand if you haven't work with files with a programming language...



Read:

https://www.tutorialspoint.com/dbms/dbms_data_independence.htm

```
#include <stdio.h>
int main() {
    FILE * fp;
    char c;
    printf("File Handling\n");
    //open a file
    fp = fopen("demo.txt", "w");
    //writing operation
    while ((c = getchar()) != EOF) {
        putc(c, fp);
    }
    //close file
    fclose(fp);
    printf("Data Entered:\n");
    //reading
    fp = fopen("demo.txt", "r");
    while ((c = getc(fp)) != EOF) {
        printf("%c", c);
    }
    fclose(fp);
    return 0;
}
```

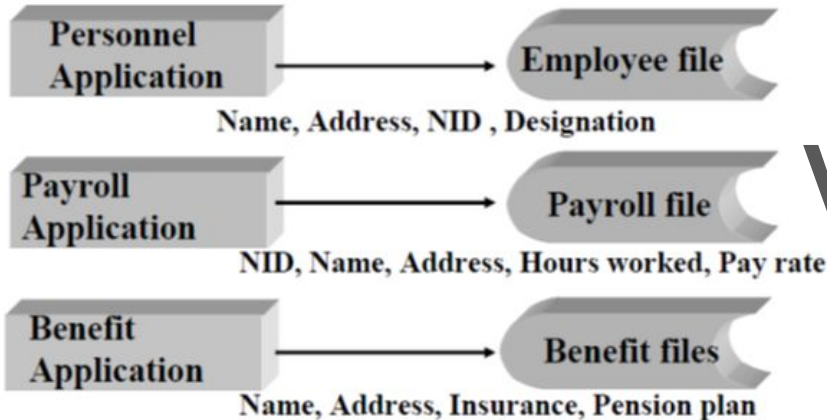
The code is annotated with four red boxes and numbers 1 through 4. Box 1 highlights the file opening operation: `fp = fopen("demo.txt", "w");`. Box 2 highlights the writing operation: `while ((c = getchar()) != EOF) { putc(c, fp); }`. Box 3 highlights the reading operation: `fp = fopen("demo.txt", "r"); while ((c = getc(fp)) != EOF) { printf("%c", c); }`. Box 4 highlights the file closing operation: `fclose(fp);`.

1.4. Databases: Working with files.

You'll understand everything when you work with files in Java...

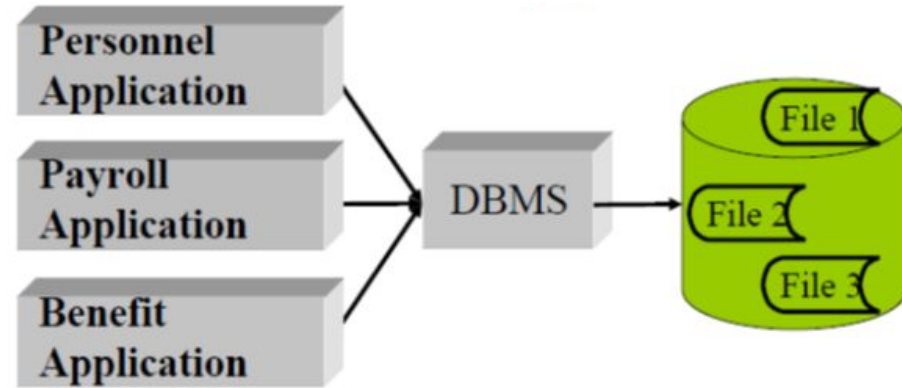
- **Separation and isolation of data:**
 - Data distributed in several files, with different formats, etc. It's difficult to write new programs that ensure the correct manipulation of data. Synchronization between files could be necessary...

File Approach



VS

Database Approach



Read: <http://conceptformate.blogspot.com/p/file-base-vs-database.html>

You'll understand everything when you work with files in Java...

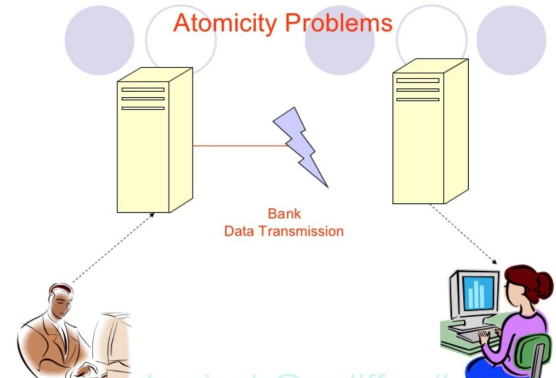
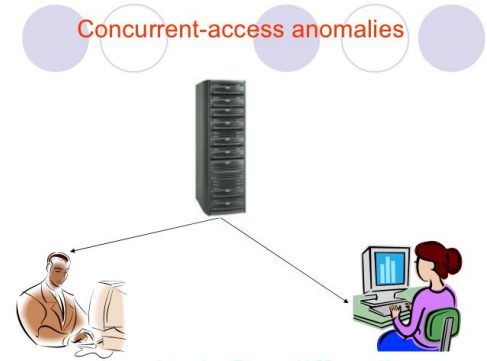
1.4. Databases: Working with files.

- **Difficulty concurrent access:**

- Concurrency: More than one user working at the same time with the the same file/s (update the data simultaneously, update data while reading, etc.). Concurrent updates can result in inconsistent data.

- **Atomicity problems:**

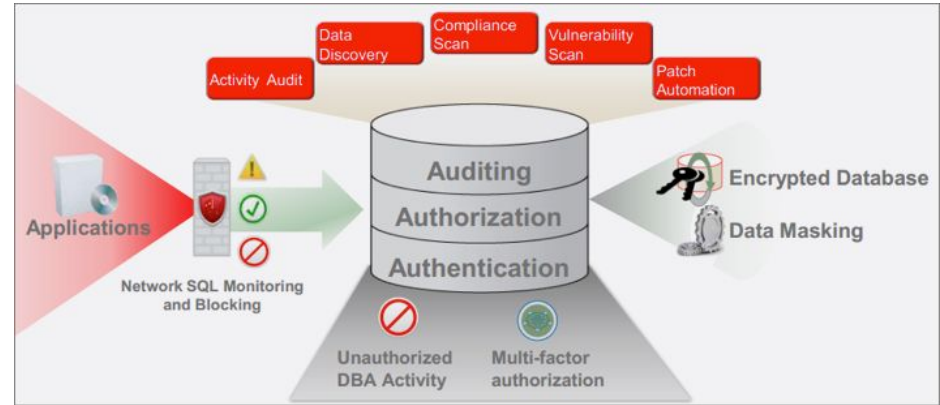
- Many data operations must be atomic. For instance, if you transfer money between two bank accounts you subtract money to the first account and you add money to the destination account (and the operation must be atomic!).



You'll understand everything when you work with files in Java...

1.4. Databases: Working with files.

- **Problems with security of data:**
 - It's difficult to implement security restrictions coding.
- **Data integrity problems:**
 - Values stored in files must be consistent. For example, you won't be able to insert a mark to a student that he/she isn't enrolled. This involves adding a large number of lines of code in your programs. This issue is more complicated when there are restrictions that involve several data in different files.



Primary Table

CompanyId	CompanyName
1	Apple
2	Samsung

Related Table

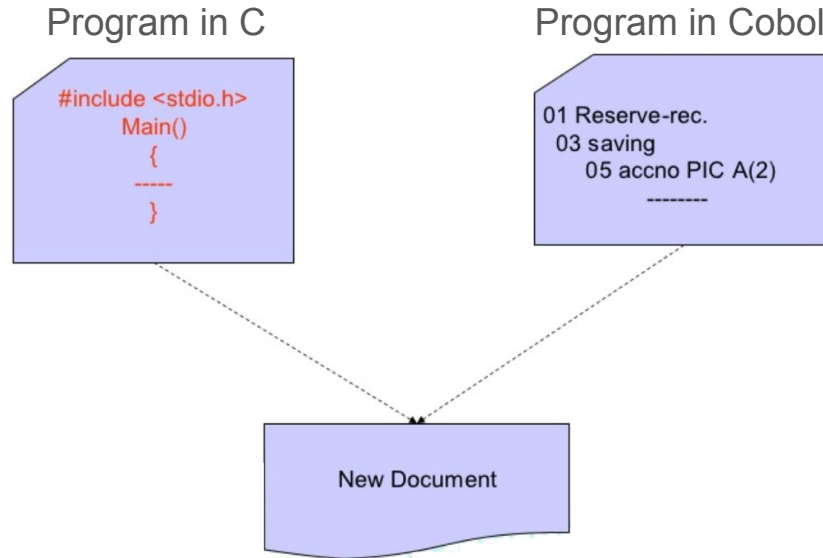
CompanyId	ProductId	ProductName
1	1	iPhone
15	2	Mustang

Associated Record ✓
Orphaned Record ✗

You'll understand everything when you work with files in Java...

1.4. Databases: Working with files.

- **Dependence on file structure with the programming language:**
 - File structure is defined within the programs coded with a programming language. This implies that file formats are incompatible. The incompatibility between files generated by different programming languages makes data difficult to process.



1.5. Database management systems (DBMS)



PostgreSQL

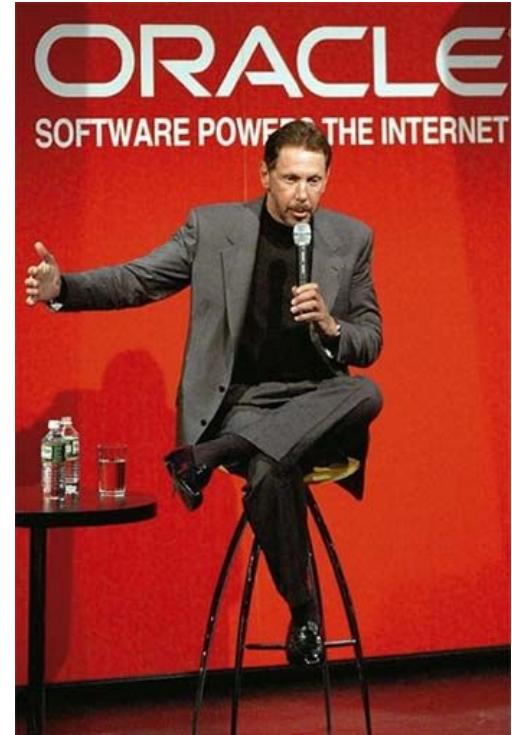


ORACLE[®]
DATABASE

1.5. DBMS: Working with files.

So, during **early computer days**, data was **collected and stored in files...**

Larry Ellison (co-founder of Oracle) was amongst the firsts who realised the need for a software based Database Management System.



1.5. DBMS: Characteristics of a RDBMS.

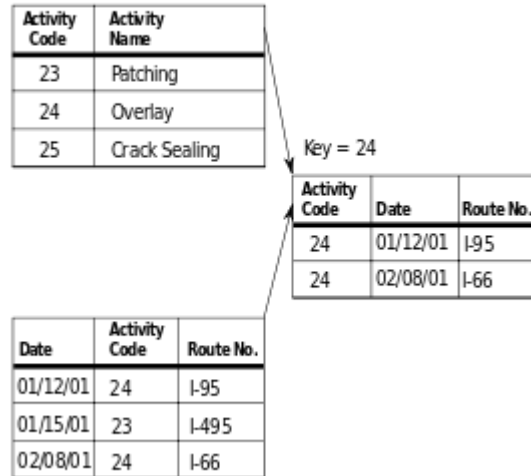
A DBMS is an application that allows users to define, create and maintain a databases providing controlled access to them. A DBMS must provide the following services:

1. Data stored into tables.
2. Reduce Redundancy.
3. Data Consistency.
4. Support Multiple user and Concurrent Access.
5. Query Language.
6. Security.
7. Transactions.
8. Backup and recovery.

1.5. DBMS: Characteristics of a RDBMS.

1. Data stored into tables: In **relational DBMS**, **data** is stored into **tables** (created inside a **database**). DBMS also allows to have **relationships** between tables which makes the data more meaningful and connected. DDL: **Data Definition Language**.

Relational Model



1.5. DBMS: Characteristics of a DBMS.

2. Reduce Redundancy: DBMS follows **Normalization** which divides the data in such a way that repetition is minimum.

Table with repeated information:

Employee ID	Employee Address	Skill
426	87 Sycamore Grove	Typing
426	87 Sycamore Grove	Shorthand
519	94 Chestnut Street	Public Speaking
519	96 Walnut Avenue	Carpentry

Employee's table



Employee ID	Name	Surname	Address
426	John	Smith	87 Sycamore Grove
519	Michael	Jordan	94 Chestnut Street

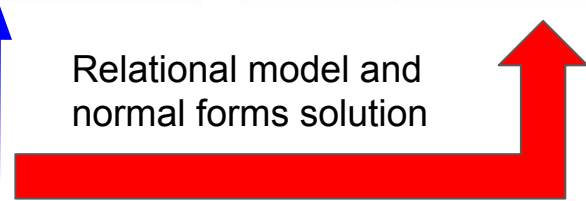
Employee ID	Skill ID
426	TYP
426	SHO
519	CAP
519	PSP

Skill ID	Name
TYP	Typing
SHO	Shorthand
PSP	Public Speaking
CAP	Carpentry

Relational model and
normal forms solution

Employee-Skill's table

Skill's table



1.5. DBMS: Characteristics of a DBMS.

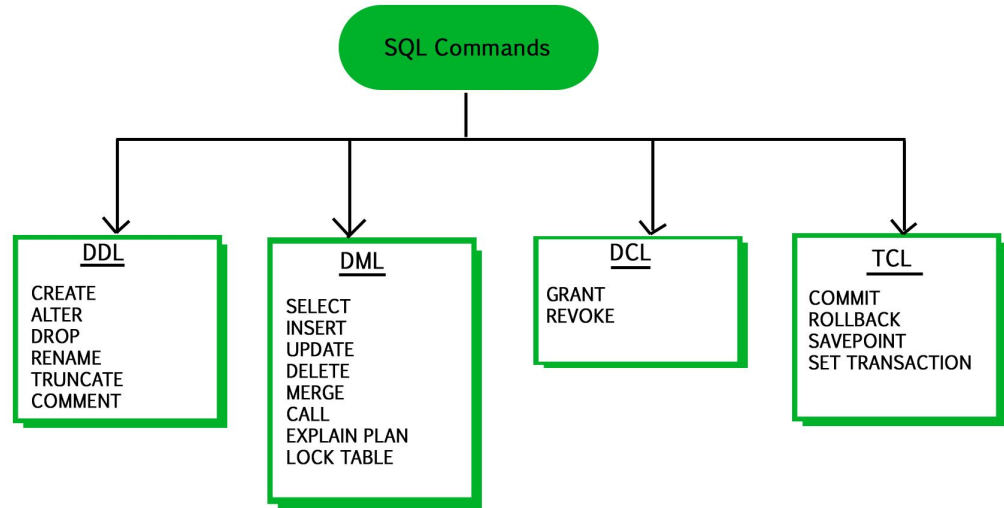
3. Data Consistency: If we reduce redundancy, we can ensure more consistency. In the last slide, what is Michael Jordan's address? That's a good example of redundant and NOT consistent data!

4. Support Multiple user and Concurrent Access: DBMS allows multiple users to work on it (update, insert, delete data) **at the same time** and maintaining data consistency.



1.5. DBMS: Characteristics of a DBMS.

5. Query Language: DBMS provides users with a simple Query language, using which data can be easily fetched, inserted, deleted and updated in a database. Another name: DML (*Data Manipulation Language*). Basically: SELECT, INSERT, UPDATE and DELETE.



```

SELECT `Team`, SUM(`Points`) As `Points`, SUM(`Won`) As `Won`, SUM(`Drawn`) As `Drawn`, SUM(`Lost`) As `Lost`
, SUM(`GF`) As `GF`, SUM(`GA`) As `GA`, (SUM(`GF`) - SUM(`GA`)) AS `GD` FROM
(
  (SELECT `home` AS `Team`
  , SUM(CASE
    WHEN `scoreHome` > `scoreAway` THEN 3
    WHEN `scoreHome` = `scoreAway` THEN 1
    ELSE 0
  END) AS `Points`
  , SUM(CASE
    WHEN `scoreHome` > `scoreAway` THEN 1
    ELSE 0
  END) AS `Won`
  , SUM(CASE
    WHEN `scoreHome` = `scoreAway` THEN 1
    ELSE 0
  END) AS `Drawn`
  , SUM(CASE
    WHEN `scoreHome` < `scoreAway` THEN 1
    ELSE 0
  END) AS `Lost`
  , SUM(`scoreHome`) AS `GF`
  , SUM(`scoreAway`) AS `GA`
  FROM sergiopmiPES.`tblmatch`
  WHERE `id` = 1
  GROUP BY `home`)
  UNION ALL
  (SELECT `away` AS `Team`
  , SUM(CASE
    WHEN `scoreAway` > `scoreHome` THEN 3
    WHEN `scoreAway` = `scoreHome` THEN 1
    ELSE 0
  END) AS `Points`
  , SUM(CASE
    WHEN `scoreAway` > `scoreHome` THEN 1
    ELSE 0
  END) AS `Won`
  , SUM(CASE
    WHEN `scoreAway` = `scoreHome` THEN 1
    ELSE 0
  END) AS `Drawn`
  , SUM(CASE
    WHEN `scoreAway` < `scoreHome` THEN 1
    ELSE 0
  END) AS `Lost`
  , SUM(`scoreAway`) AS `GF`
  , SUM(`scoreHome`) AS `GA`
  FROM sergiopmiPES.`tblmatch`
  WHERE `id` = 1
  GROUP BY `away`)
) `tmpTable`
GROUP BY `Team` ORDER BY `Points` DESC;

```

Team	Points	Won	Drawn	Lost	GF	GA	GD
Dog	16	5	1	0	21	7	14
Flip	8	2	2	2	11	7	4
Ernie	4	0	4	2	10	14	-4
Monkey	3	0	3	3	8	22	-14

tblmatch

Nombre	Tipo
<u>id</u>	int(11)
<u>home</u>	varchar(6)
<u>away</u>	varchar(6)
<u>scoreHome</u>	tinyint(4)
<u>scoreAway</u>	tinyint(4)

tbltournament

Nombre	Tipo
<u>id</u>	int(11)
<u>date</u>	date
<u>closed</u>	tinyint(1)

tbltournament

id	date	closed
1	2017-10-28	1

tblmatch

id	home	away	scoreHome	scoreAway
1	Dog	Ernie	2	2
1	Dog	Flip	2	1
1	Dog	Monkey	7	1
1	Ernie	Dog	1	4
1	Ernie	Flip	1	1
1	Ernie	Monkey	3	3
1	Flip	Dog	2	3
1	Flip	Ernie	1	0
1	Flip	Monkey	1	1
1	Monkey	Dog	0	3
1	Monkey	Ernie	3	3
1	Monkey	Flip	0	5

1.5. DBMS: Characteristics of a DBMS.

6. Security: Protect data from unauthorised access or system crashes.

7. DBMS supports **transactions**, which allows us to better handle and manage data integrity in real world applications where multi-threading is extensively used. Basically: COMMIT, ROLLBACK, SAVEPOINT, ROLLBACK and SET TRANSACTION (TCL: Transactional Control Language).

8. Backup and recovery mechanisms to restore information in case of failures in the system (it could be included in point 6).

1.5. DBMS: Advantages and disadvantages of DBMS.

Advantages:

- Minimal data duplicity or data redundancy.
- Easy retrieval of data using the Query Language.
- Reduced development time and maintenance need.
- With Cloud Datacenters, we now have Database Management Systems capable of storing almost infinite data.
- Seamless integration into the application programming languages which makes it very easier to add a database to almost any application or website.

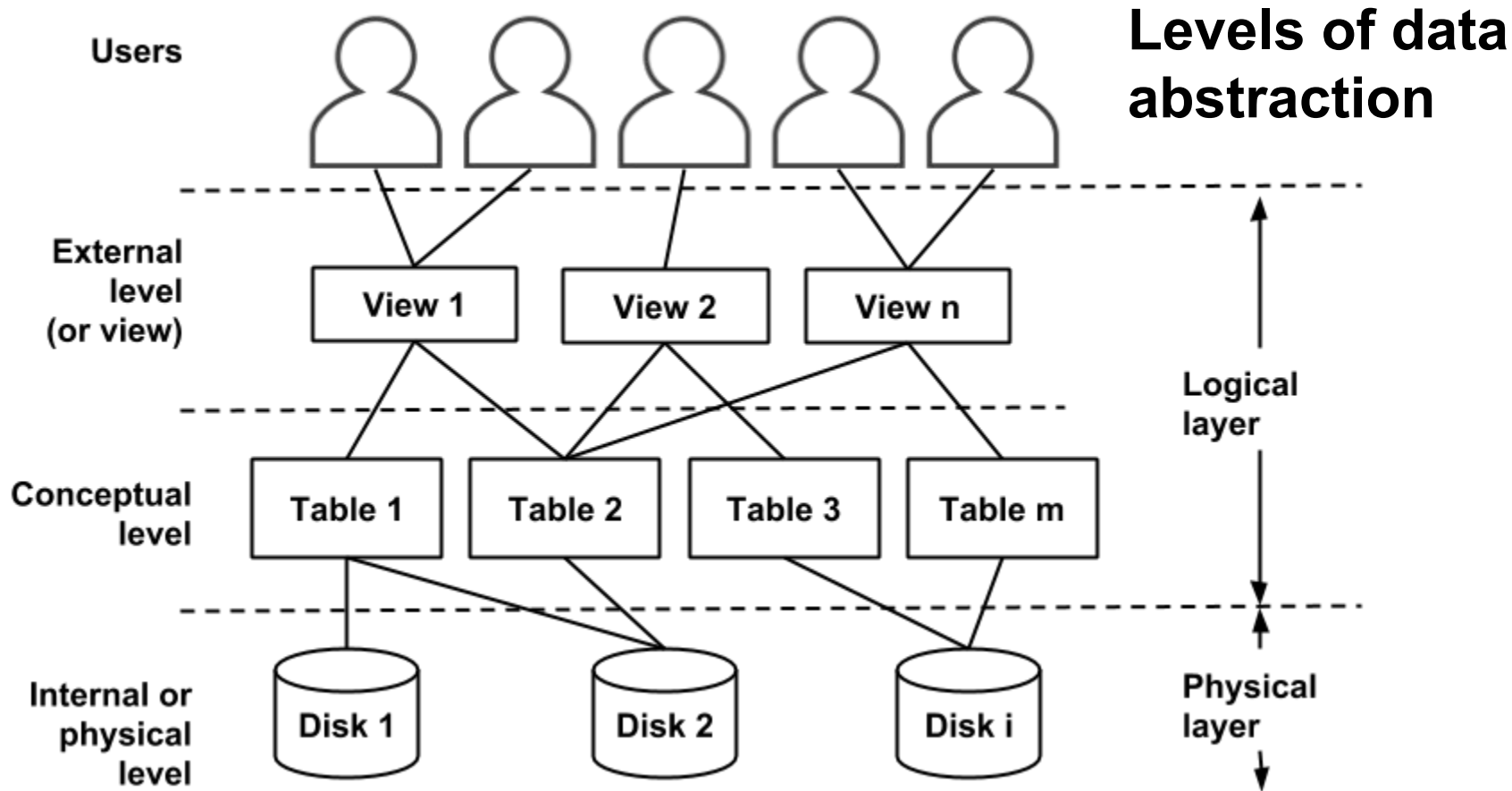
Disadvantages:

- Its complexity.
- Licensed DBMSs are generally costly.
- They are large in size.

1.5. DBMS: Architecture of database systems.

In 1975, the ANSI-SPARC committee (American National Standard Institute - Standards Planning and Requirements Committee) proposed a **three-tier architecture for DBMS** with the goal of **separating application programs from physical DB**. In this architecture the schema of a DB is defined in three different layers of abstraction:

- Internal or physical level.
- External level (or view).
- Conceptual level.



1.5. DBMS: Architecture of database systems.

- **Internal or physical level:** The closest to physical storage (how data is stored in the computer). It describes the physical structure of the DB into an internal scheme. This scheme is defined by a physical model and describes the details of how data is physically stored (files that contain the information, its organization, types of records, methods to access the records, maximum length of data stored, etc.)
- **Conceptual level:** It describes the structure of the entire DB for a group of users through a conceptual scheme. This scheme describes entities, attributes, relationships, user operations and restrictions, hiding the details of the physical storage structures to users of this level. Represents the information contained in the DB.
- **External level (or view):** The closest to the users. Several external schemes or views of users are described. Each view describes the part of the database that interests a group of users at this level (representing the individual view of a user or a group of users).

1.5. DBMS: Architecture of database systems.

With the proposed three-tier architecture the concept of data independence is introduced. Two types of independence are defined:

- **Logical independence:** The ability to modify the conceptual schema without having to alter external schemas or application programs. You can modify the conceptual schema to extend the DB or to reduce it (for instance, if an entity is deleted, external schemas –that do not refer to it– won't be affected).
- **Physical independence:** The ability to modify the internal schema without having to alter either the conceptual or the external schema. For instance, new data files can be added because the ones in use are full. Physical independence is easier to achieve than logic, since it refers to the separation between applications and physical storage structures.

The **Data dictionary** plays an important role in independence and to achieve it you need more hardware resources.

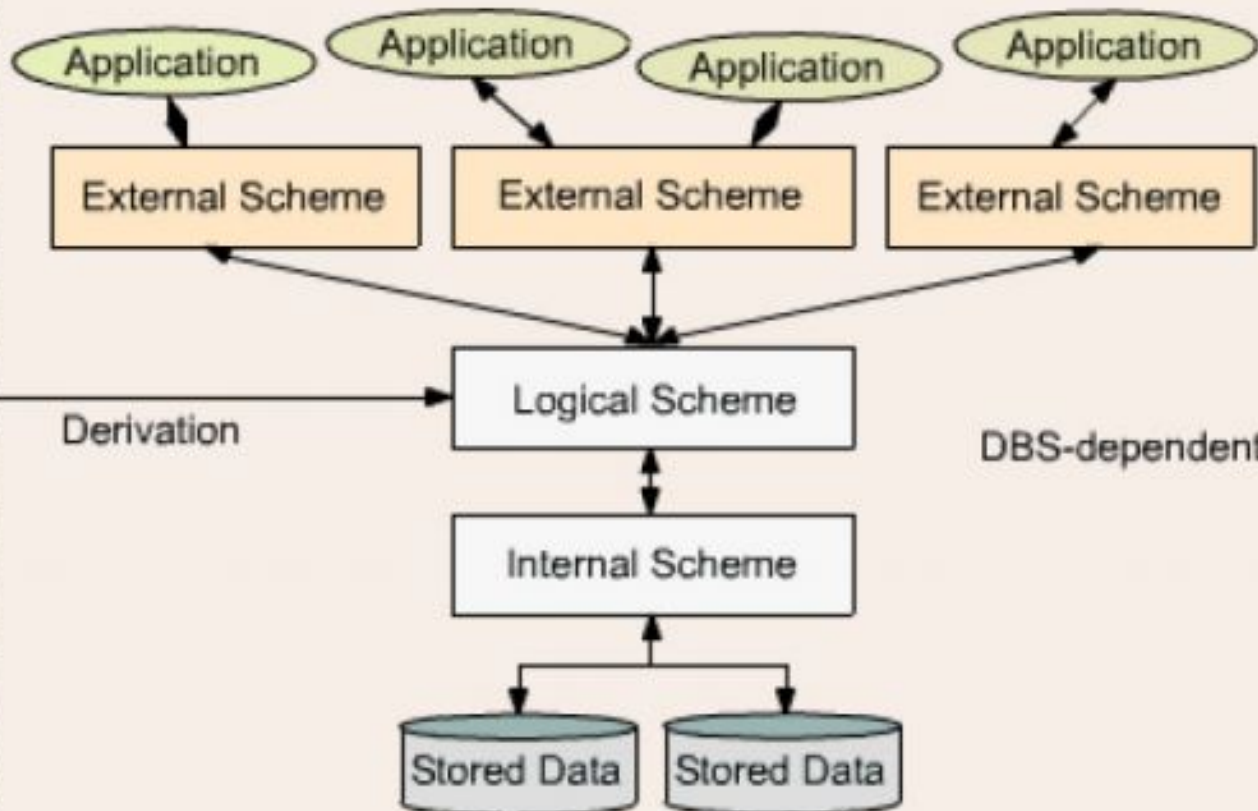


Data Modelling

Conceptual Scheme

DBS-independent

3-Scheme-Architecture



1.5. DBMS: Architecture of database systems.

- **Data dictionary:**

- The data dictionary is a structure of tables and views (read-only) that contains information such as:

- The definitions of all the schemas in the DB (tables, indexes, views, clusters, synonyms, sequences, procedures, functions, packages, triggers, etc.)

PostgreSQL:

System

Catalogs

- The allocated space used by a schema.

- The users of the database.

- The privileges and roles for each user.

- Information about those who have accessed and updated schemas.

- General information of the database.

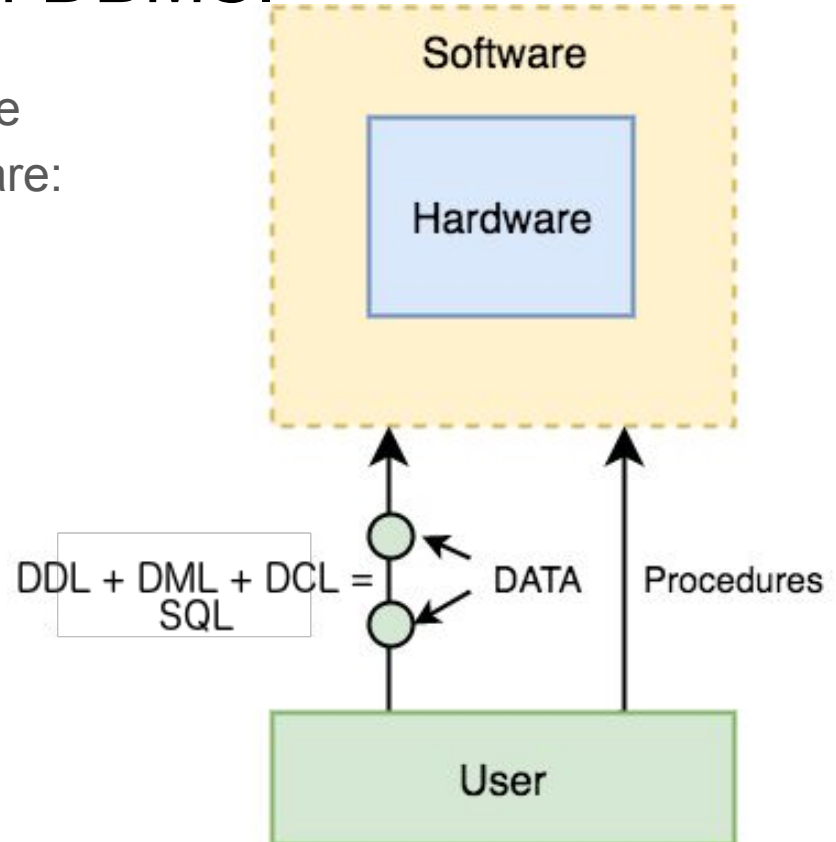
- The data dictionary is created when the database is created. To accurately reflect the status of the BD status at all times, the data dictionary is automatically updated by the DBMS in response to specific actions (such as when the structure of the DB is changed).

- In a nutshell: **The data dictionary stores metadata:** data about data.

1.5. DBMS: Components of DBMS.

The database management system can be divided into five major components, they are:

1. Hardware
2. Software
3. Data
4. Procedures
5. DDL + DML + DCL + TCL = SQL
6. Users.



1.5. DBMS Components: Hardware.

Hardware: CPU, RAM, hard disks, I/O channels for data, and any other physical component involved before any data is successfully stored.

When we run Oracle or MySQL on our personal computer, then our computer's Hard Disk, our Keyboard, our computer's RAM, ROM, etc. All become a part of the DBMS hardware.



1.5. DBMS Components: Software.

Software: The main component, as this is the program which controls everything. The DBMS software is more like a [wrapper](#) around the physical database, which provides us, with an easy-to-use interface to store, access and update data.

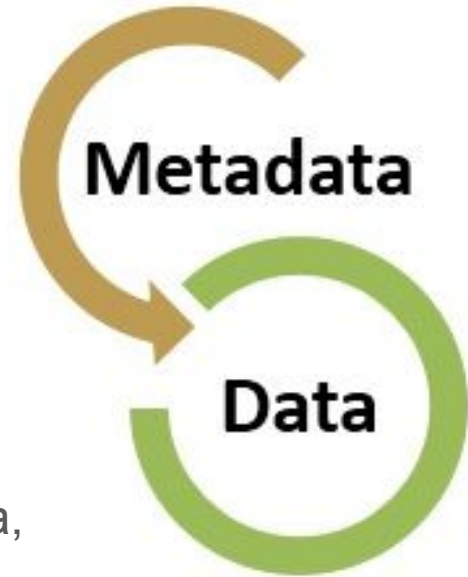
The DBMS software is able to understand SQL commands (and execute them into their databases).



1.5. DBMS Components: Data.

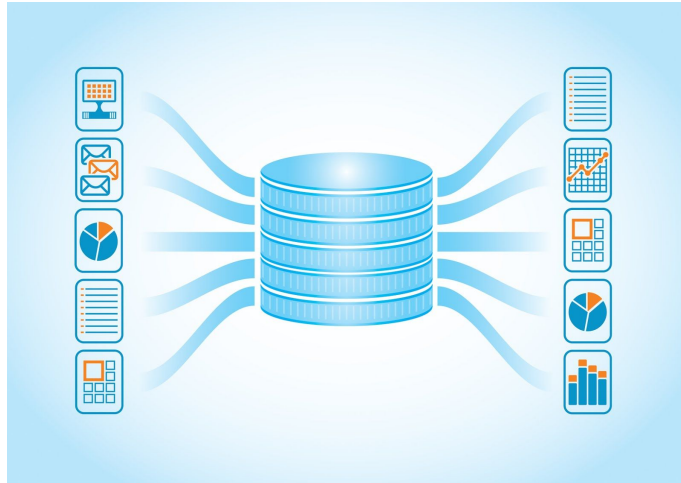
Data: The resource for which DBMS was designed. The motivation behind the creation of DBMS was to store and use data.

Metadata is data about the data. This is information stored by the DBMS to better understand the data stored in it. For example: When I store my Name in a database, the DBMS will store when the name was stored in the database, what is the size of the name, is it stored as related data to some other data, or is it independent, all this information is metadata. (Think about metadata in JPG files). This is the Data Dictionary...



1.5. DBMS Components: Procedures.

Procedures: General instructions to use a database management system. This includes procedures to setup and install a DBMS, to login and logout of DBMS software, to manage databases, to make backups, to generate reports, etc.



1.5. DBMS Components: DDL.

DDL (*Data Definition Language*):

- It's the language to define the database schema.
- All DDL have special instructions to define each of the three schemes of the DB: physical, conceptual or sub-scheme.
- The DBMS parses DDL and stores all the specifications of the DB schema in the **Data Dictionary**. The Data Dictionary contains metadata, that is, "data about data" of the DB.

1.5. DBMS Components: DDL.

DDL (*Data Definition Language*):

```
CREATE TABLE IF NOT EXISTS `tbltournament` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `date` date NOT NULL,  
  `closed` tinyint(1) DEFAULT '0',  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=2 ;
```

```
CREATE TABLE IF NOT EXISTS `tblmatch` (  
  `id` int(11) NOT NULL,  
  `home` varchar(6) NOT NULL,  
  `away` varchar(6) NOT NULL,  
  `scoreHome` tinyint(4) NOT NULL,  
  `scoreAway` tinyint(4) NOT NULL,  
  PRIMARY KEY (`id`,`home`,`away`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

1.5. DBMS Components: DML.

DML (*Data Manipulation Language*):

- It's a language that allows users to access or manipulate the data that you are granted to access according to your subschem.
- There are basically two types of DML:
 - Procedural
 - The user specifies what data he/she needs and how to get it.
 - Non-procedural
 - The user specifies what data he/she needs, but without indicating how to obtain it.

1.5. DBMS Components: DML.

DML (*Data Manipulation Language*):

```
SELECT *
```

```
FROM `tbltournament`;
```


1.5. DBMS Components: DCL.

DCL (*Data Control Language*)

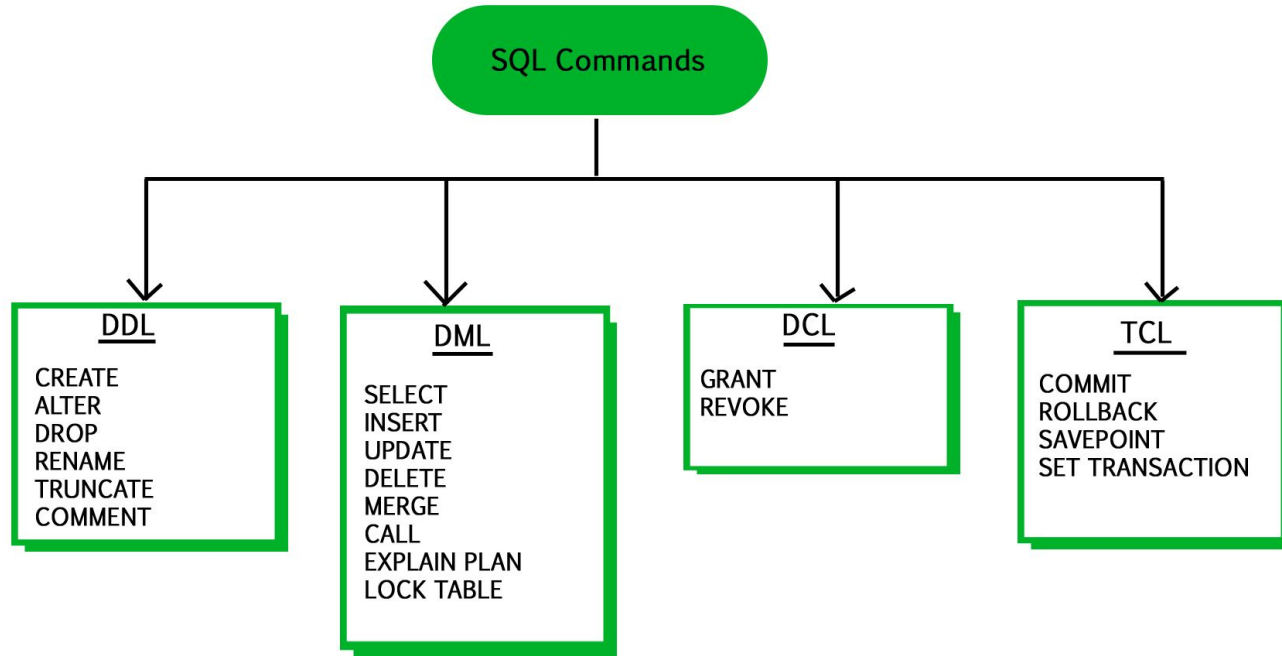
- It's a language provided by the *Database Management System* that includes SQL commands to control access to the data contained in the database (the administrator will use them).
- Some examples of commands included in the DCL are the following:
 - GRANT: Allows one or more users or roles to perform tasks.
 - REVOKE: Allows you to delete permissions that have previously been granted with GRANT.

1.5. DBMS Components: TCL.

TCL (*Transactional Control Language*)

- TCL commands are used to **manage transactions** in the database. These are used to manage the changes made by DML-statements. It also allows statements to be grouped together into logical transactions..
- Some examples of commands included in the DCL are the following:
 - **COMMIT**: Commit command is used to permanently save any transaction into the database.
 - **ROLLBACK**: This command restores the database to last committed state. It is also used with savepoint command to jump to a savepoint in a transaction.
 - **SAVEPOINT**: Savepoint command is used to temporarily save a transaction so that you can rollback to that point whenever necessary.

1.5. DBMS Components: SQL.



1.5. DBMS Components: Users.

Application Programmer or Software Developer: He/she codes the application programs that use the database. These programs are written in a programming language (Java, PHP, Visual Basic, C++, etc.) and the database is accessed using a specific language (SQL) that the DBMS can understand.

End User: You can access the database in two ways:

- The application programs mentioned in the previous point.
- An interactive interface included as part of the DBMS. E.g., a query language processor.

Database Administrators (DBA): He/she is the one who manages the complete database management system. DBA takes care of the security of the DBMS, its availability, managing the license keys, managing user accounts and access, etc.

You in June!



You now!

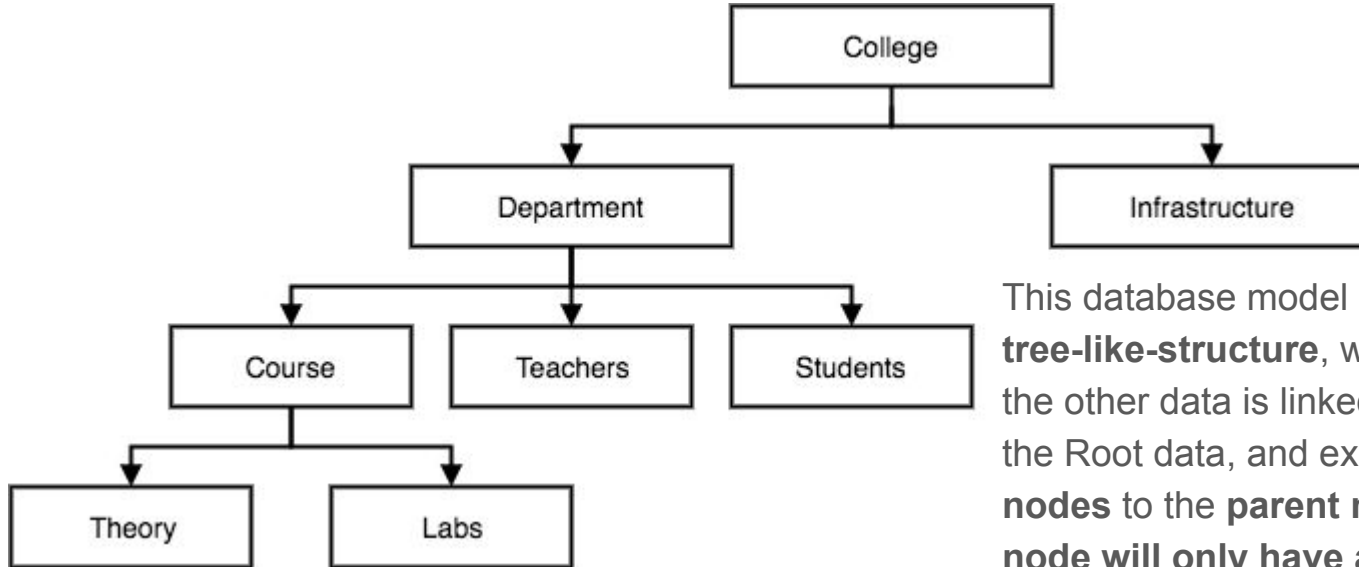


1.5. DBMS: Database Models.

A **database model** defines the **logical design and structure** of a database and defines **how data will be stored**, accessed and updated in a database management system. While the **Relational Model is the most widely used** database model, there are other models too:

- Hierarchical Model
- Network Model
- Entity-relationship Model
- Relational Model

1.5. DBMS: Hierarchical Model.



This database model organises data into a **tree-like-structure**, with a **single root**, to which all the other data is linked. The hierarchy starts from the Root data, and expands like a tree, adding **child nodes** to the **parent nodes**. In this model, **a child node will only have a single parent node**.

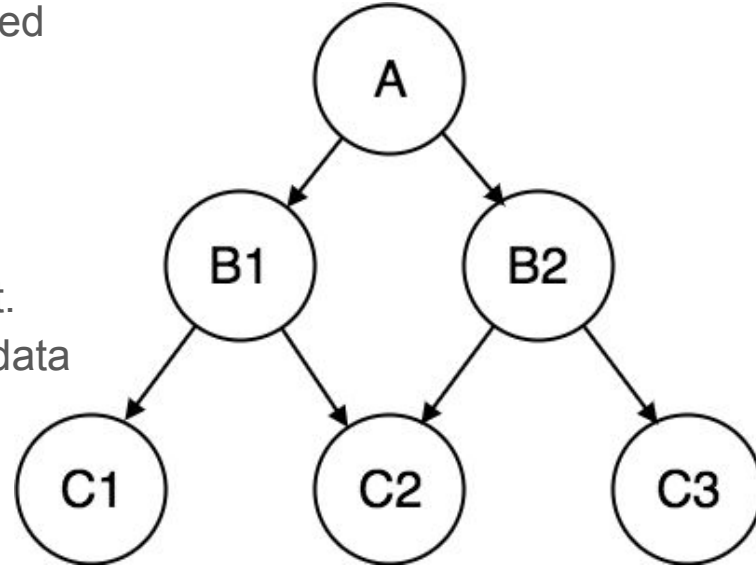
This model efficiently describes many real-world relationships like index of a book, recipes, etc. Moreover, data is organised into tree-like structure with a one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students.

1.5. DBMS: Network Model.

This is an **extension of the Hierarchical model**. In this case data is organised more like a **graph**, and are allowed to have **more than one parent node**.

Moreover, in this model data is **more related** as more relationships are established. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map **many-to-many** data relationships.

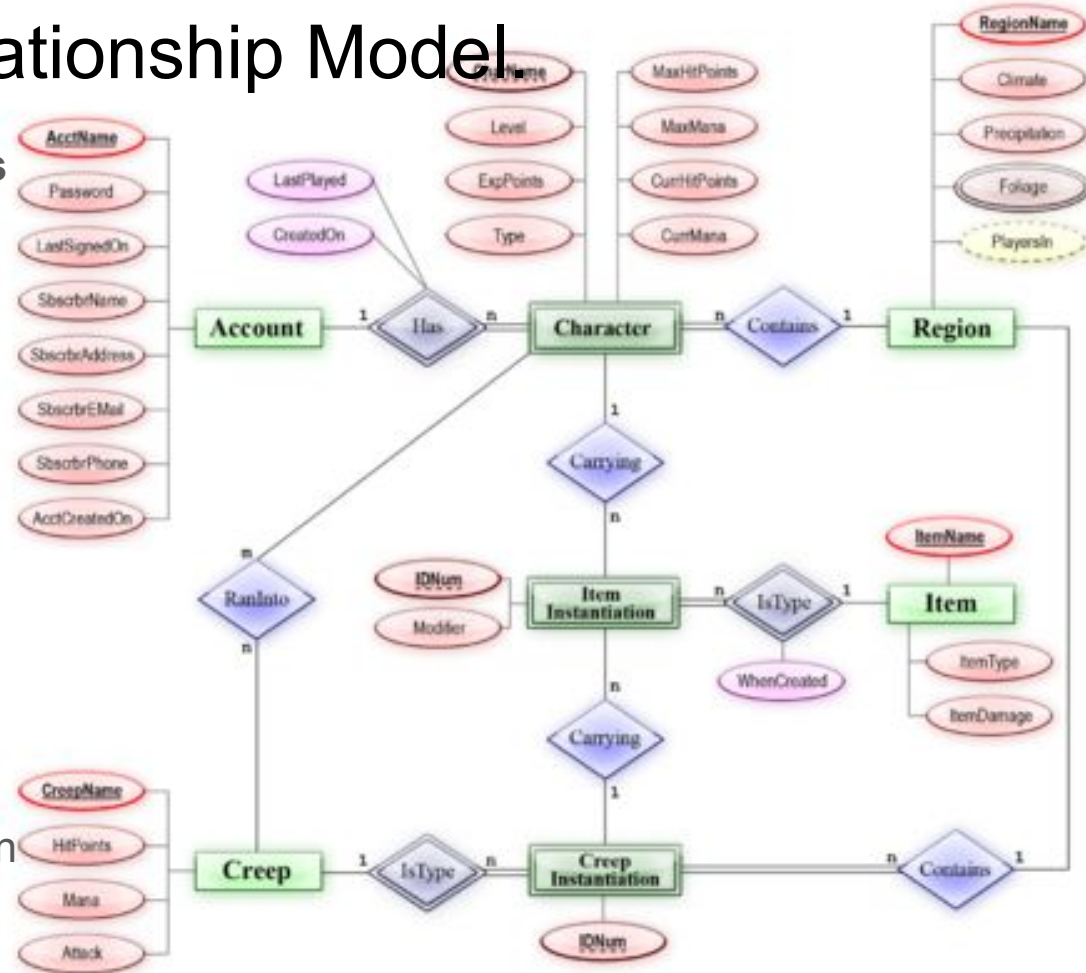
This was the most widely used database model, before Relational Model was introduced.



1.5. DBMS: Entity-relationship Model

In this database model, **relationships** are created by dividing objects of interest into **entities** and its characteristics into **attributes**. Different entities are related using relationships.

E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand. This model is good to design a database, which can then be turned into tables in relational model (explained below).



1.5. DBMS: Relational Model.

In this model, data is organised in **two-dimensional tables** and the **relationship is maintained by storing a common field**. All the information related to a particular **item** is stored in **rows** of that table, and the **attributes** are defined by the **columns**.

Introduced by **E.F Codd in 1970**, and since then it has been the most widely used database model.

student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

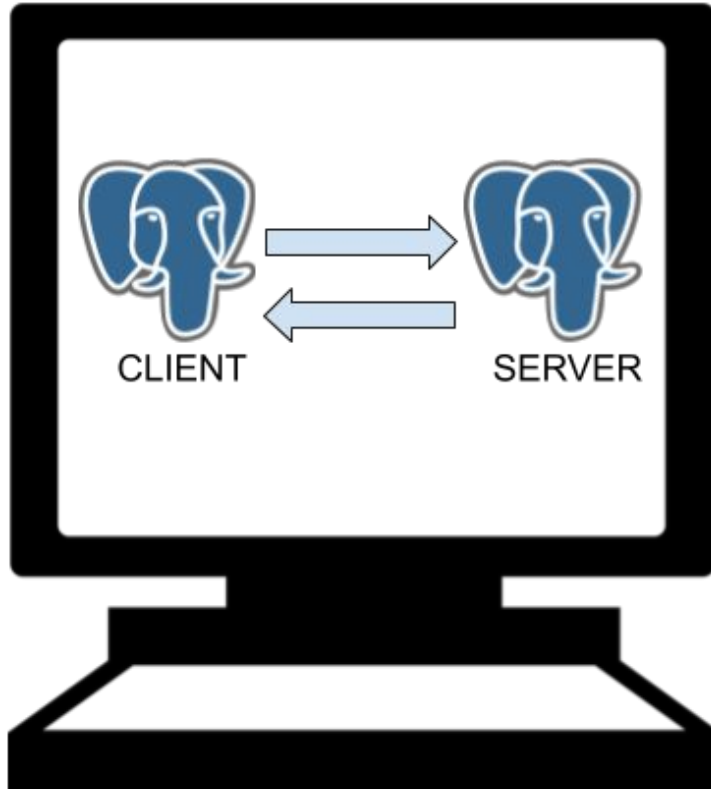


student_id	subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88

1.6. Centralized and distributed databases.

- Client-server model configurations for a DBMS:
 - Client and server in the same host.
 - DB in the server and users access to the DB through their clients using a network.
 - Database distributed among several servers. Users access from their clients and they don't need to know where data is.

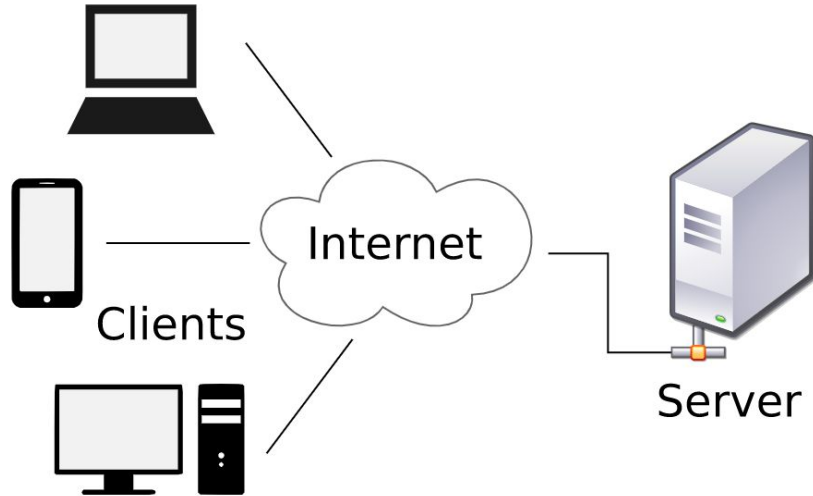
1.6. Centralized and distributed databases.



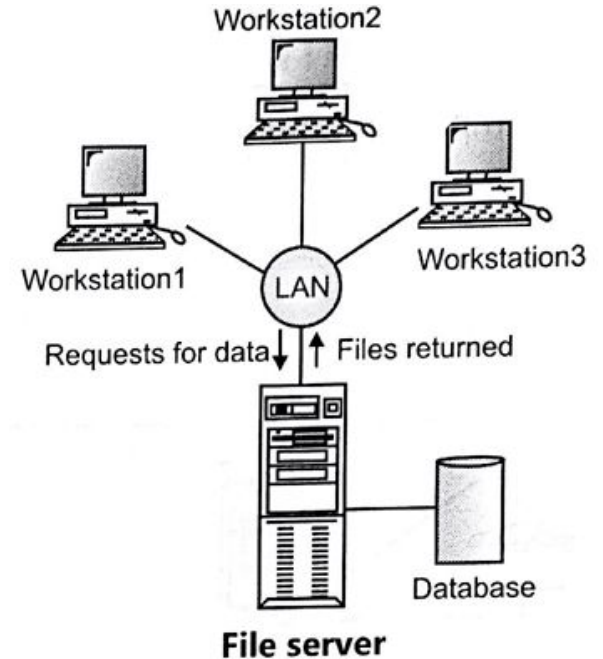
*We'll work with
client and server
in the same host.*

1.6. Centralized and distributed databases.

Client-server architecture:

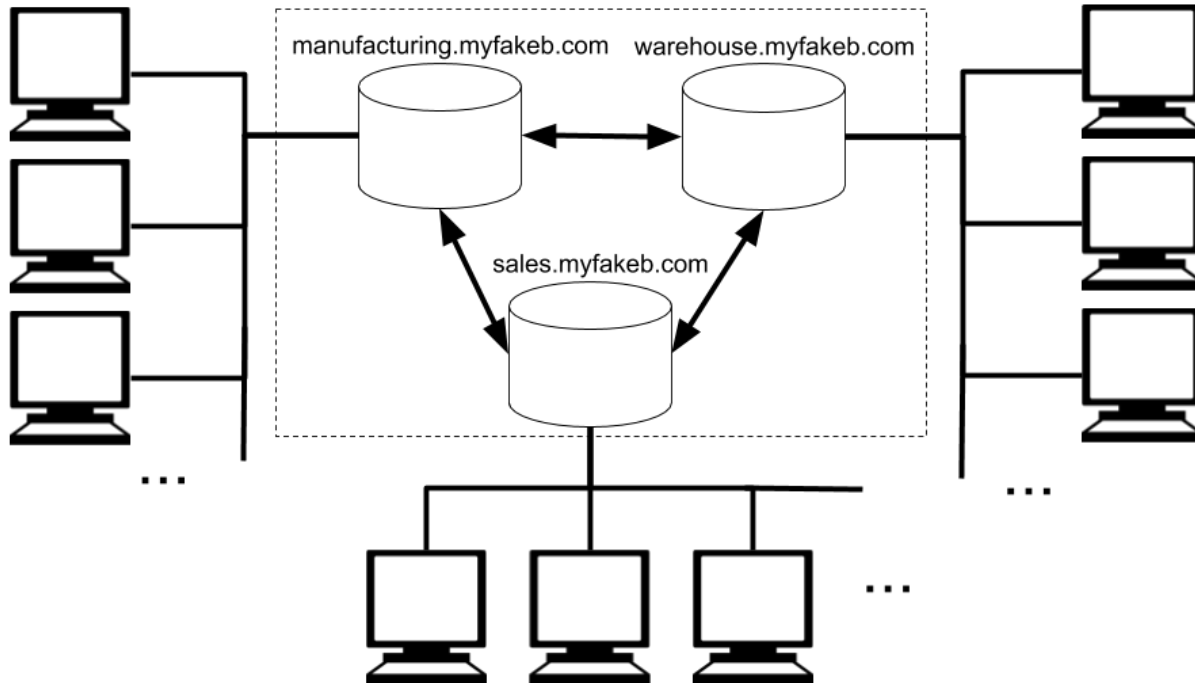


DBMS with a client-server architecture:



1.6. Centralized and distributed databases.

Distributed database:



Source: Oracle.

Sources.

- **M. J. Ramos, A. Ramos and F. Montero.** Sistemas gestores de bases de datos (Chapter 1, pag. 7-15). McGrawHill: 1th Edition, 2006.
- **Abraham Silberschatz, Henry F. Korth and S. Sudarshan.** *Database System Concepts (Chapter 1)*. McGrawHill: 6th Edition, 2010.
- Apunts de la UIB del professor Miquel Manresa (1996).
- <https://www.studytonight.com/dbms/>
- <https://web.cs.ucdavis.edu/~green/courses/ecs165a-w11/6-storage.pdf>
- https://en.wikipedia.org/wiki/Sequential_access
- https://en.wikipedia.org/wiki/Random_access
- <https://www.geeksforgeeks.org/file-allocation-methods/>