

# INTRODUCTION TO DATABASES

## Database Foundations

.

Author: Eng. Carlos Andrés Sierra, M.Sc.  
[cavirguezs@udistrital.edu.co](mailto:cavirguezs@udistrital.edu.co)

Lecturer  
Computer Engineer  
School of Engineering  
Universidad Distrital Francisco José de Caldas

2024-III



# Outline

- 1 Software Components and Applications
- 2 Glosary
- 3 DataBase Classification
- 4 Relational Database Design
- 5 Entity-Relation Model (MER)
- 6 DataBase Management Systems — DBMS
- 7 DataBases Infrastructure
- 8 DevOps
- 9 Data Engineering



# Outline

1 Software Components and Applications

2 Glosary

3 DataBase Classification

4 Relational Database Design

5 Entity-Relation Model (MER)

6 DataBase Management Systems — DBMS

7 DataBases Infrastructure

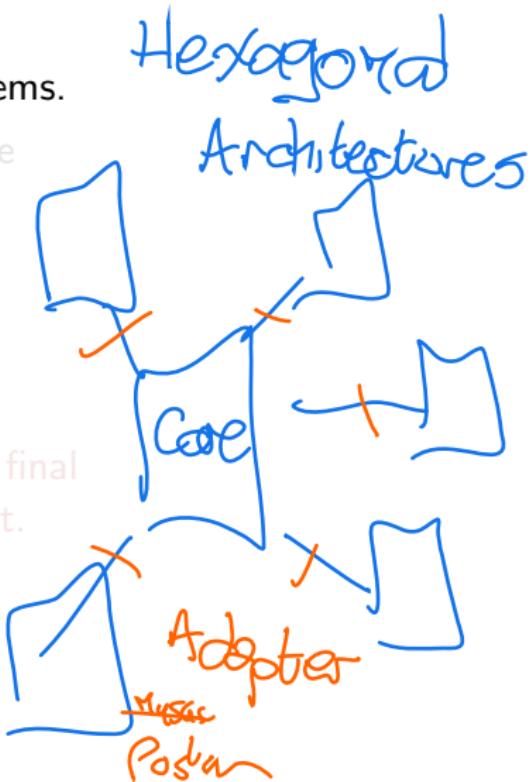
8 DevOps

9 Data Engineering



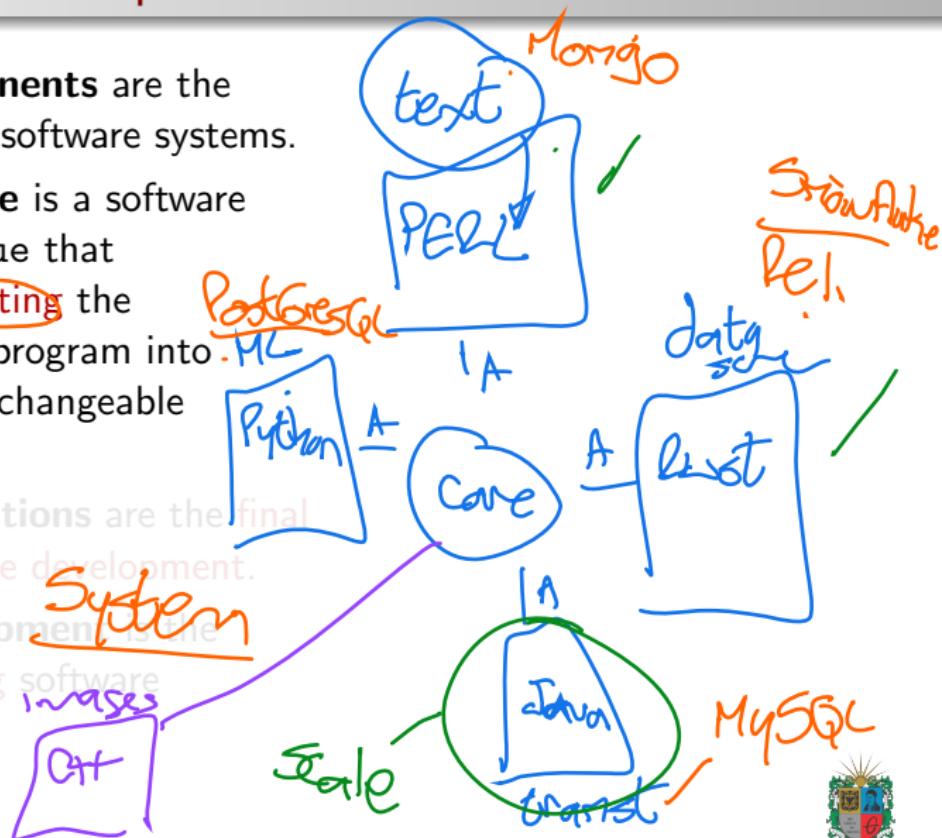
# Modular Software Components

- **Software Components** are the building **blocks** of software systems.
- Modular Software is a software design technique that emphasizes **separating** the **functionality** of a program into independent, interchangeable **modules**.
- **Software Applications** are the final product of software development.
- **Software Development** is the process of creating software applications.



# Modular Software Components

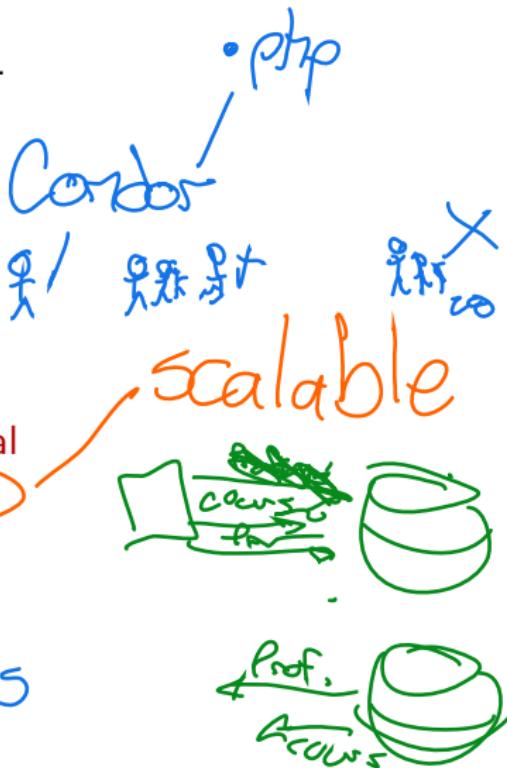
- **Software Components** are the building **blocks** of software systems.
- **Modular Software** is a software design technique that emphasizes **separating** the **functionality** of a program into independent, interchangeable **modules**.
- Software Applications are the final product of software development.
- Software Development is the process of creating software applications.



# Modular Software Components

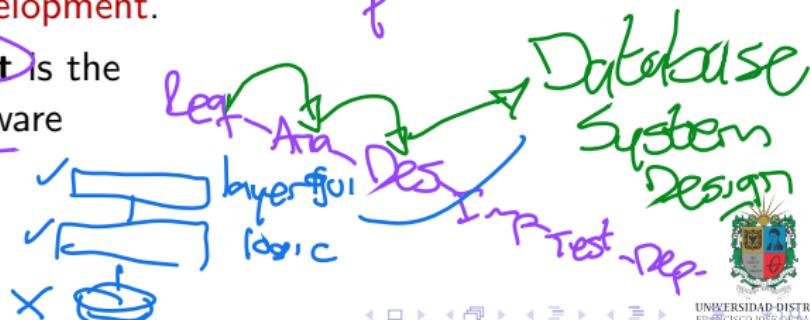
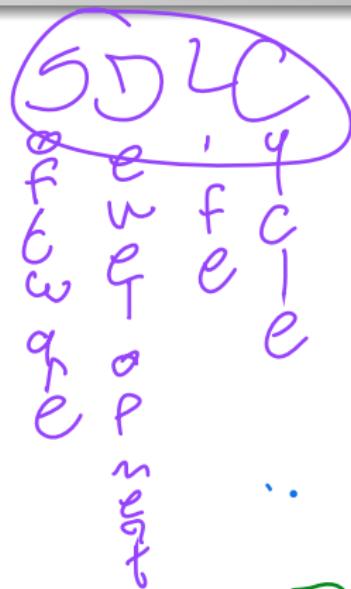
- **Software Components** are the building **blocks** of software systems.
- **Modular Software** is a software design technique that emphasizes **separating** the **functionality** of a program into independent, interchangeable **modules**.
- **Software Applications** are the **final product** of **software development**.

**Flexibility**  
black Friday  
high demand



# Modular Software Components

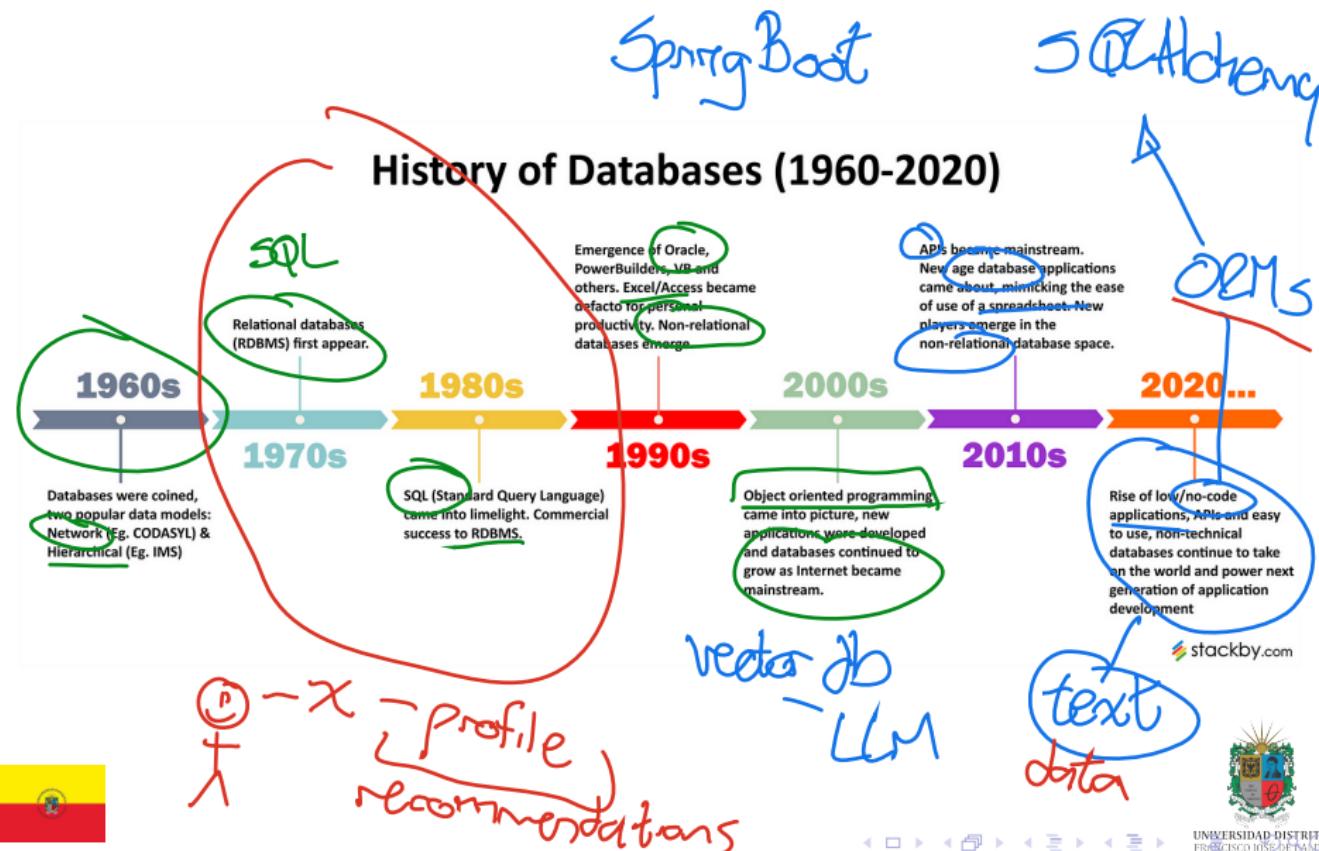
- **Software Components** are the building **blocks** of software systems.
- **Modular Software** is a software design technique that emphasizes **separating** the **functionality** of a program into independent, interchangeable **modules**.
- **Software Applications** are the **final product** of **software development**.
- **Software Development** is the process of **creating** software **applications**.



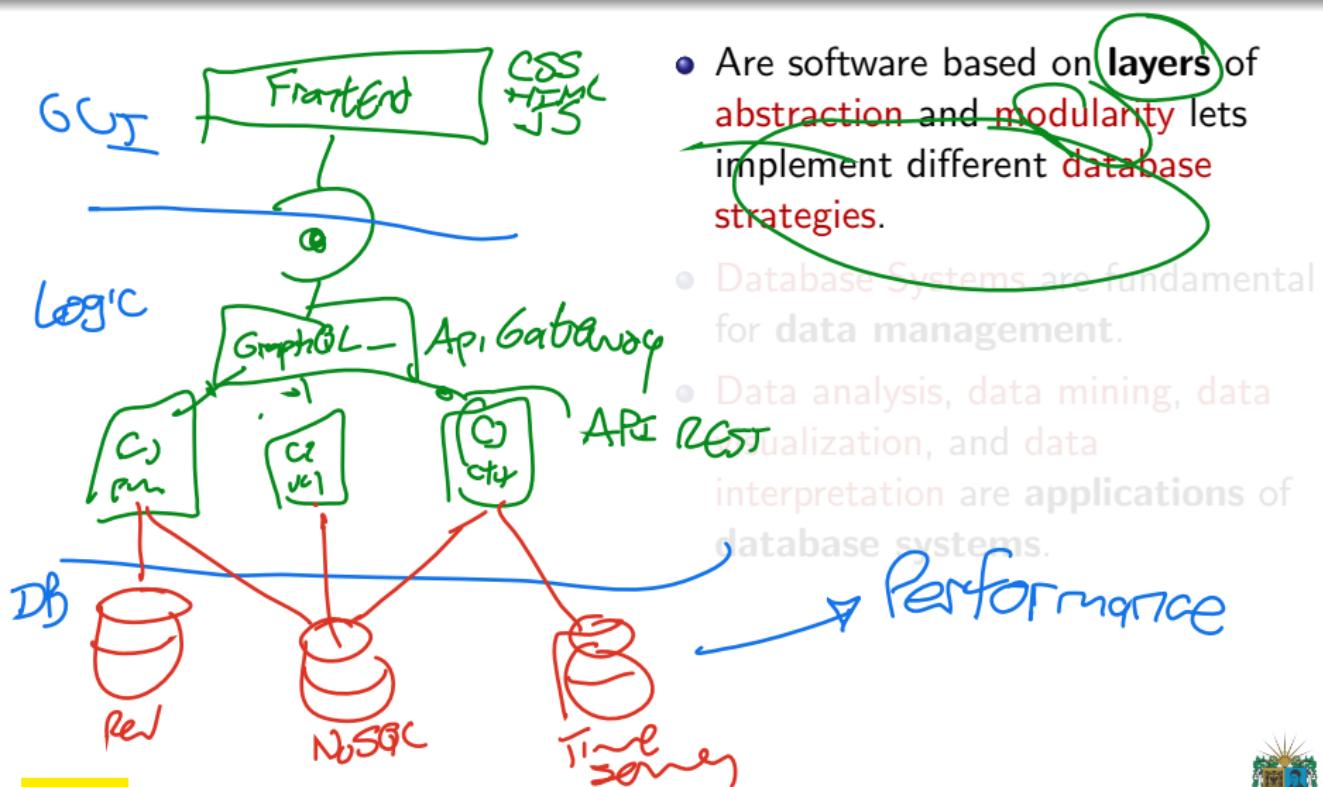
# History of Data



# History of DataBases



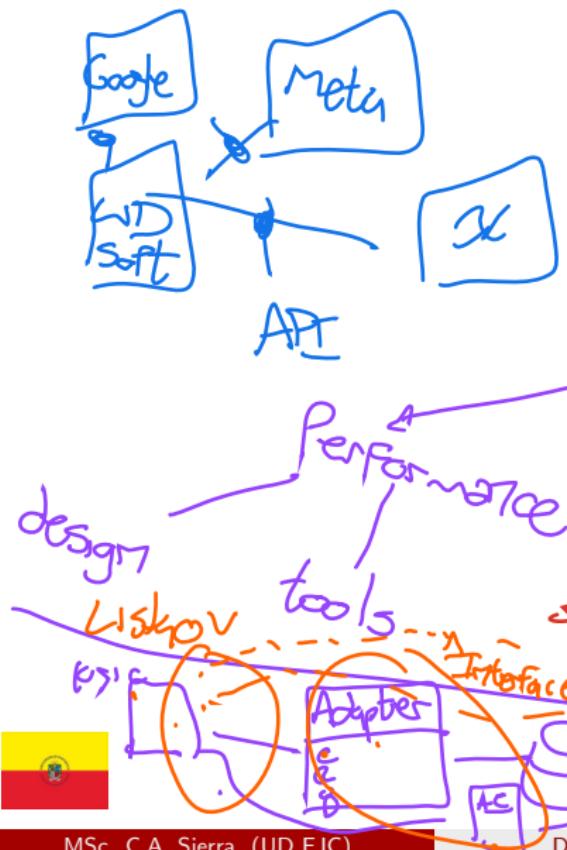
# Applications



- Are software based on **layers** of abstraction and modularity lets implement different database strategies.
- Database Systems are fundamental for data management.
- Data analysis, data mining, data visualization, and data interpretation are applications of database systems.



## Applications



- Are software based on **layers** of abstraction and **modularity** lets implement different **database strategies**.
  - **Database Systems** are fundamental for **data management**.

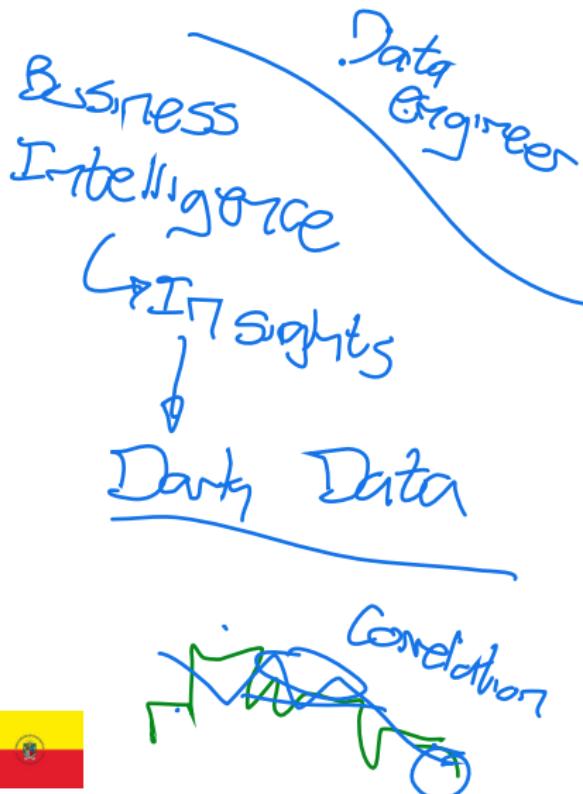
Data analysis, data mining, data visualization, and data  
Access Grants

## External API

vermerab. / tres



# Applications

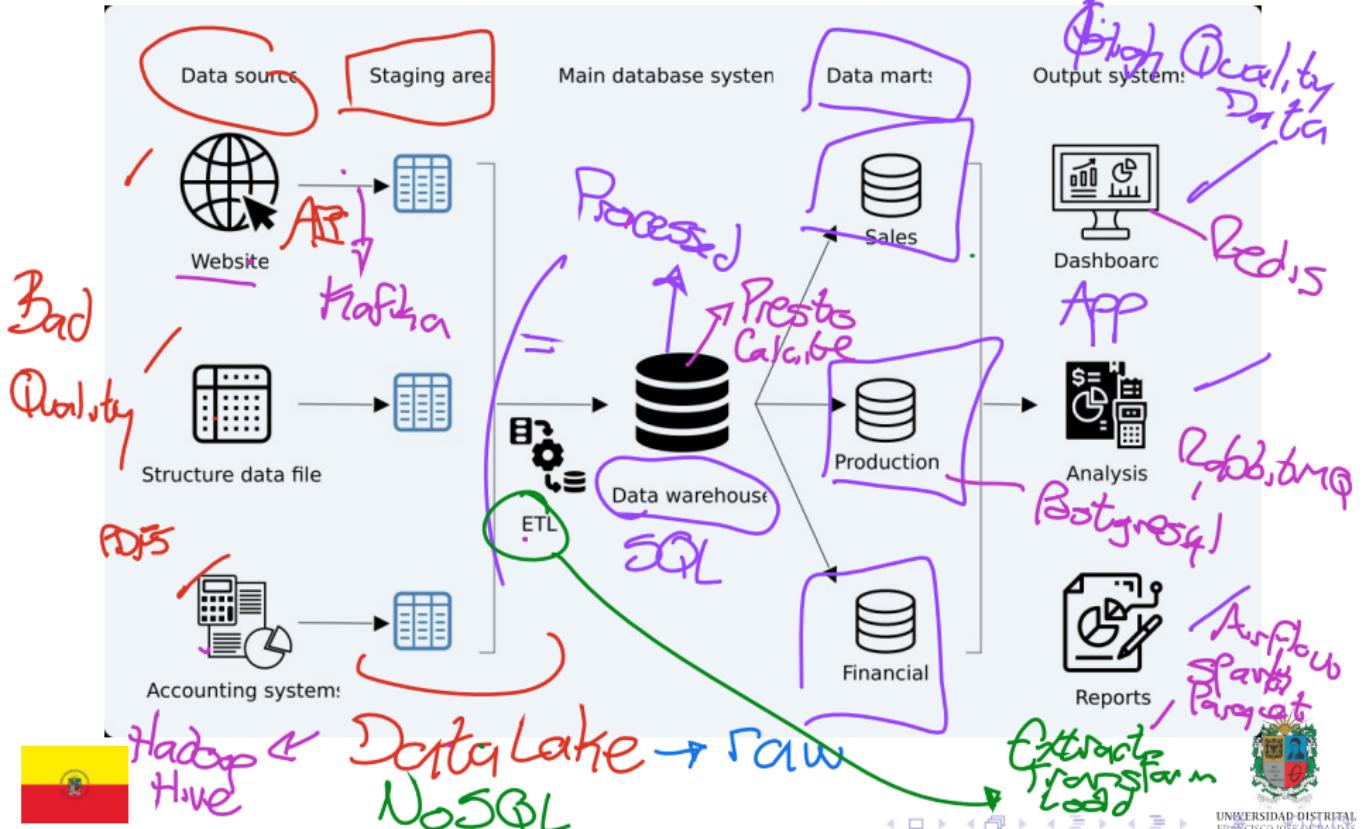


- Are software based on **layers of abstraction** and **modularity** lets implement different **database strategies**.
- Database Systems are fundamental for **data management**
- **Data analysis, data mining, data visualization, and data interpretation** are **applications** of **database systems**.

Dashboard



# Case of Study: DataBase System



# Outline

- 1 Software Components and Applications
- 2 Glossary



- 3 DataBase Classification

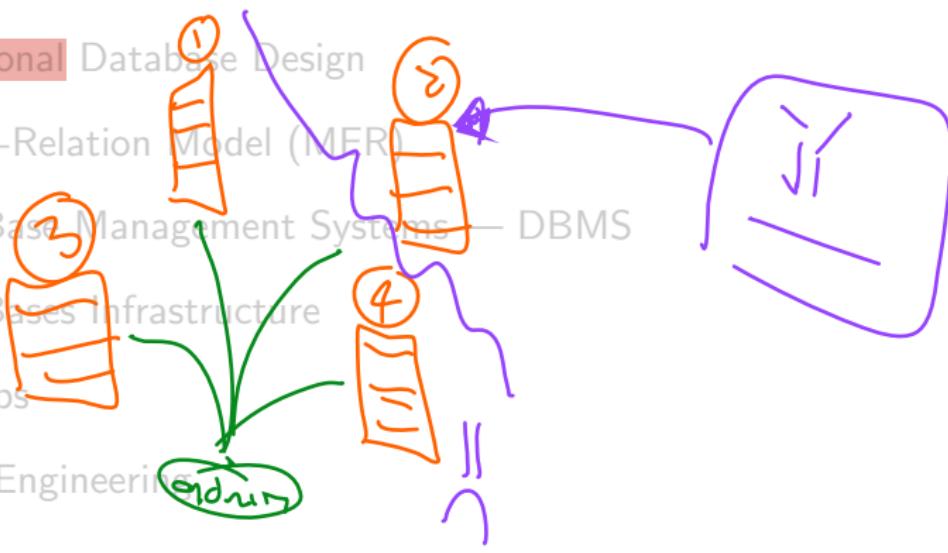
- 4 Relational DataBase Design

- 5 Entity-Relation Model (MER)

- 6 DataBase Management Systems

DBMS

- 7 DataBases Infrastructure



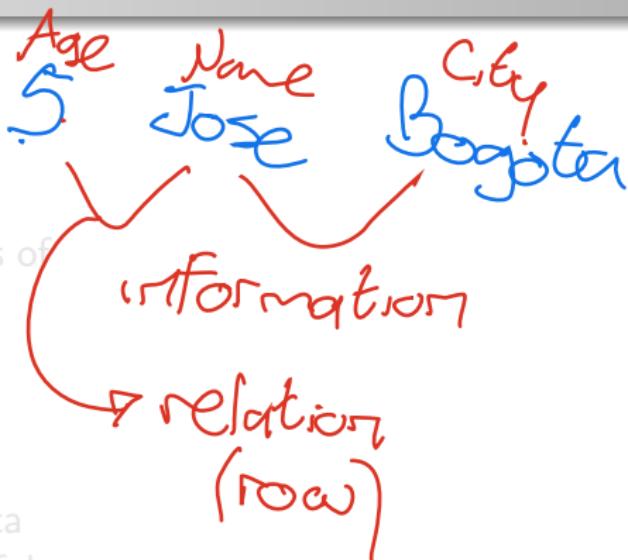
- 8 DevOps

- 9 Data Engineering



# From Data to Information

- **Data:** is a set of values of **qualitative** or **quantitative** variables
  - **Data Management:** is the process of collecting, storing, processing, and analyzing data.
  - **numbers**
  - **numbers**
  - **categories**
  - **category**
- 10**
- 10**
- 10**



# From Data to Information

- **Data:** is a set of **values** of **qualitative** or **quantitative** variables.
- **Data Management:** is the process of **collecting**, **storing**, **processing**, and **analyzing** data.

**Sources**

→ **formats**

→ **quality**

→ **ELT**

→ **ELT(L)**

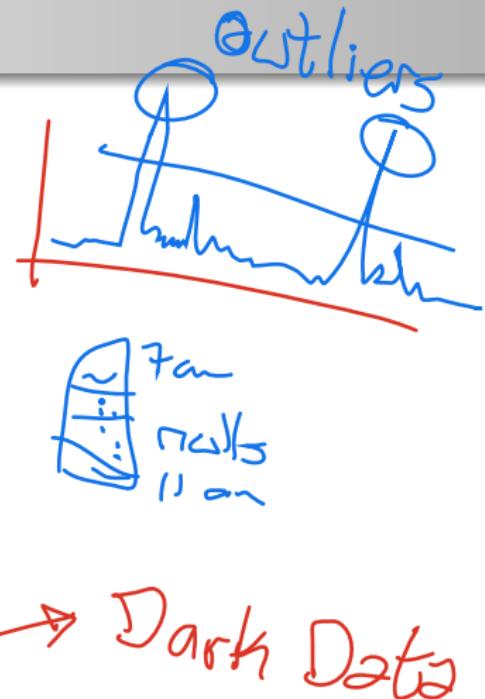
→ **offline**

→ **online**



# From Data to Information

- **Data:** is a set of **values** of **qualitative** or **quantitative** variables.
- **Data Management:** is the process of **collecting**, **storing**, **processing**, and **analyzing** data.
- **Data Analysis:** is a process of **inspecting**, **cleaning**, **transforming**, and **modeling** data with the goal of **discovering** useful **information**, informing **conclusions**, and supporting **decision-making**.



# Structured and Unstructured Data

Power BI  table

WAN - Web - colores

## Structured Data

vs

## Unstructured Data

Name 

Can be displayed  
in rows, columns and  
relational databases



Numbers, dates  
and strings

Estimated 20% of  
enterprise data (Gartner)

20%

Requires less storage



Easier to manage  
and protect with  
legacy solutions



natives

relationship

redundancy



Cannot be displayed  
in rows, columns and  
relational databases



Images, audio, video,  
word processing files,  
e-mails, spreadsheets

80%

Estimated 80% of  
enterprise data (Gartner)



Requires more storage

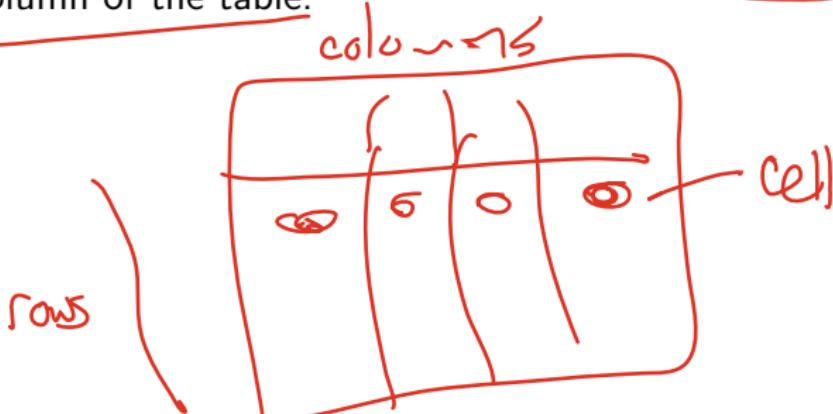


More difficult to  
manage and protect  
with legacy solutions



# Tables, Columns and Rows

- **Table** is a collection of related data held in a structured format within a database.
- **Column** is a set of data values of a particular simple type, one for each row of the table.
- **Row** is a set of data values of a particular relationship, one for each column of the table.



# Primary and Foreign Keys

- **Primary Key** is a unique identifier for a record in a data set.
- **Foreign Key** is a column or group of columns in a table that links to a primary key in another table.

*Country*

code	name
S7	Colombia
S9	Argentina

PK



code	City	Country
1	Bogotá	S7
2	Buenos	S9
3	Cordoba	S9
4	Mérida	S7
5	Salta	S9
6	Patagonia	S9



# Key-Value Data Structures

No redundancy → 90s

- **Key-Value Data Structures** are a type of data structure that can map keys to values.
- **Key** is a unique identifier for a record in a data fragment. **Value** is the data that is associated with the key.



# CRUD Operations

- **CRUD** is an acronym for **Create, Read, Update, and Delete**.
- **Create** is the process of **adding new records** to a **data set**.
- **Read** is the process of **retrieving records** from a **data set**.
- **Update** is the process of **modifying records** in a **data set**.
- **Delete** is the process of **removing records** from a **data set**.

4 operations  
for all apps

Law of Demand



# Outline

- 1 Software Components and Applications
  - 2 Glossary
  - 3 DataBase Classification
  - 4 Relational Database Design
  - 5 Entity-Relation Model (MER)
  - 6 DataBase Management Systems — DBMS
  - 7 DataBases Infrastructure
  - 8 DevOps
  - 9 Data Engineering
- Domain-Driven Design**
- 



# DataBase Classification

- DataBase is a collection of data that is organized so that it can be easily accessed, managed, and updated.
- Relational DataBase is a type of database that stores and provides access to data points that are related to one another.
- NoSQL DataBase is a type of database that provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.



# DataBase Classification

- DataBase is a collection of **data** that is **organized** so that it can be **easily accessed, managed, and updated.**
- Relational DataBase is a type of **database** that stores and provides access to **data points** that are **related** to one another.
- NoSQL DataBase is a type of **database** that provides a mechanism for storage and retrieval of **data** that is modeled in **means** other than the **tabular relations** used in relational databases.

DOC



# DataBase Classification

*E~S~S*

- DataBase is a collection of **data** that is **organized** so that it can be **easily accessed, managed, and updated.**
- Relational DataBase is a type of **database** that stores and provides access to **data points** that are **related** to one another.
- **NoSQL DataBase** is a type of **database** that provides a mechanism for **storage and retrieval of data** that is **modeled in means** other than the **tabular relations** used in **relational databases.**

*Student*

{ id: 1,

name: "Asd",

career: 13

*Career*

{ id: 13,

name: "Artist"



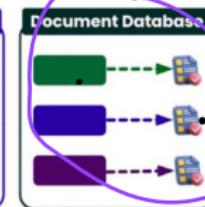
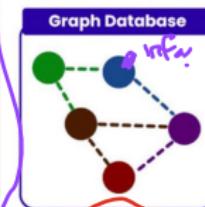
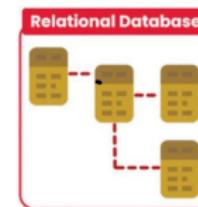
# Types of Database

TEST → PC

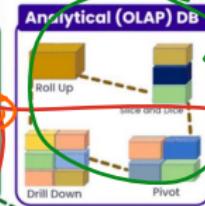
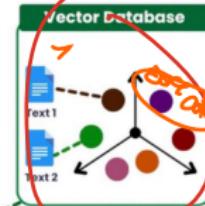
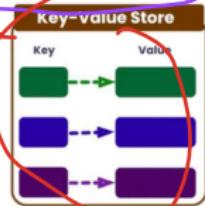


## JSON → BSON

## How Many Types of Database Do You Know?



四



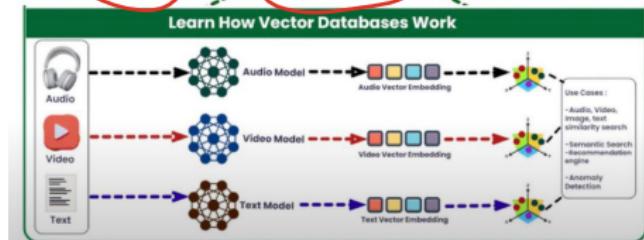
Overlap

4

100

Today 90

```
class b {  
    int d;  
    char name;  
    char password;  
}
```



# DataBases Models



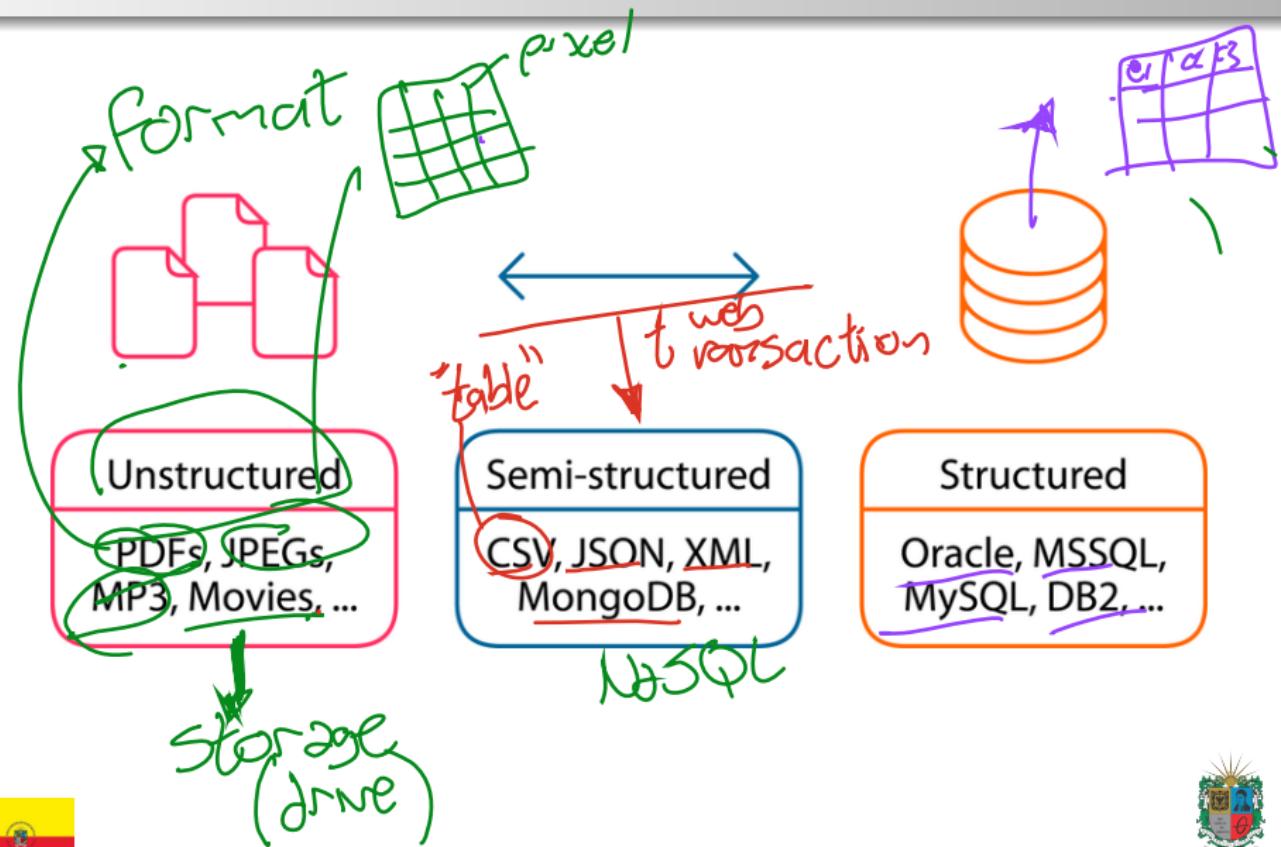
**DataBases Models** are the way to organize and store data in a database.

There are some common models:

- Hierarchical
- Network
- Relational
- Object-Oriented
- Document-Based
- Graph-Based
- ... ?



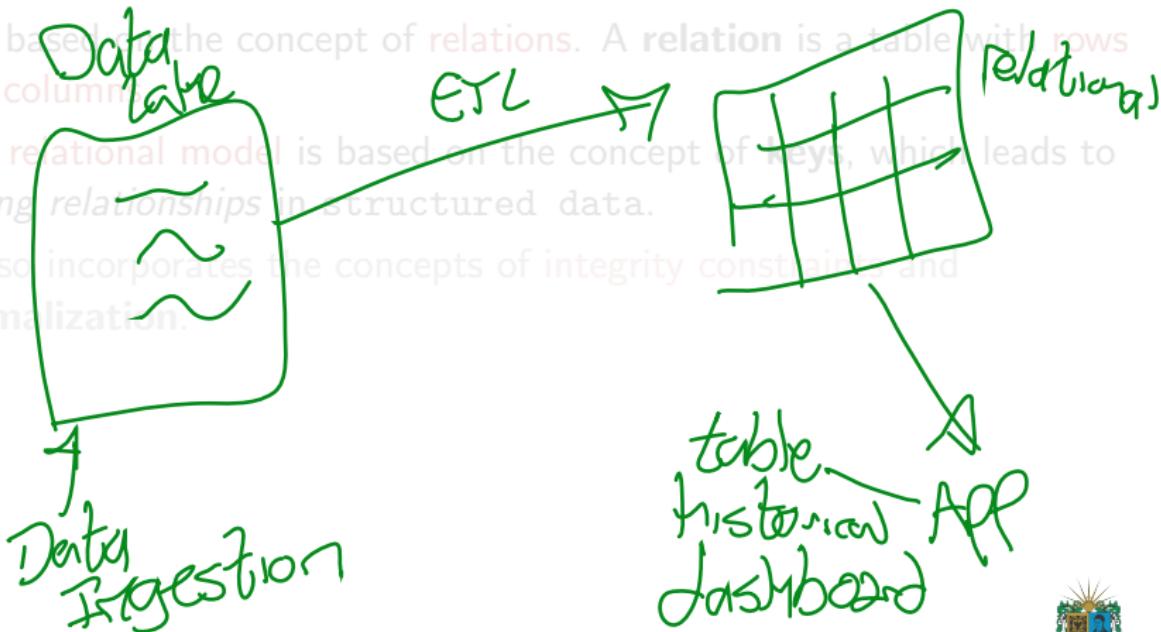
## Semi-Structured Data



# Relational Model

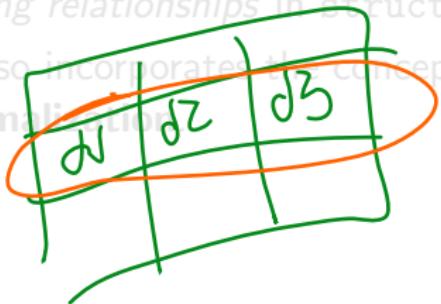
- The **relational model** is the **most common** and widely used model today.

- It is based on the concept of **relations**. A relation is a table with **rows** and **columns**.
- The relational model is based on the concept of **keys**, which leads to **strong relationships** in structured data.
- It also incorporates the concepts of **integrity constraints** and **normalization**.



## Relational Model

- The **relational model** is the **most common** and widely used model today.
  - It is based on the concept of **relations**. A **relation** is a table with **rows** and **columns**.
  - The **relational model** is based on the concept of **keys**, which leads to **strong relationships** in structured data.
  - It also incorporates the concept of **constraints** and **relations**.

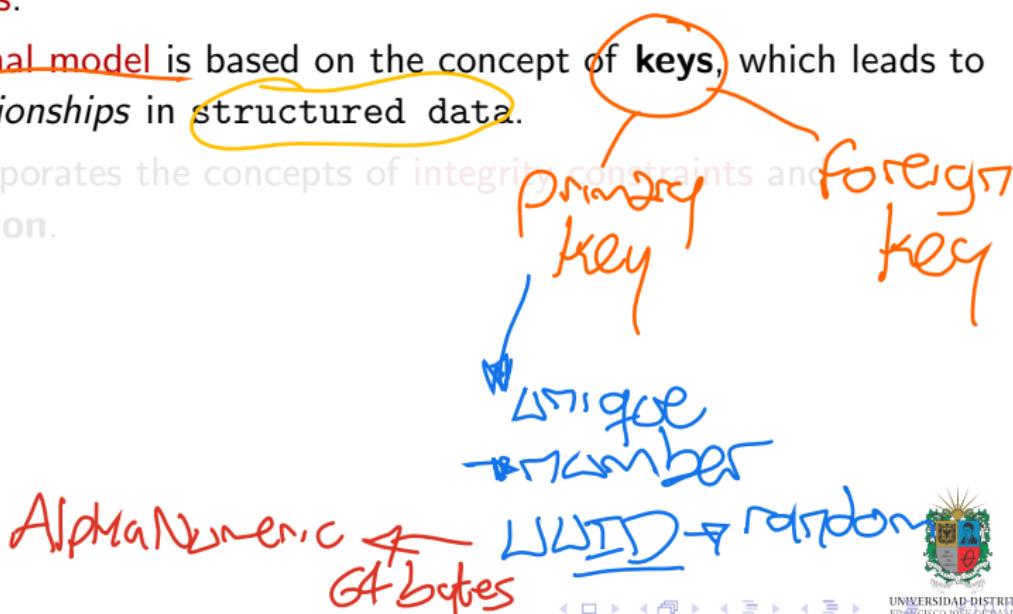


*relation*



# Relational Model

- The **relational model** is the **most common** and widely used model today.
- It is based on the concept of **relations**. A **relation** is a table with **rows** and **columns**.
- The **relational model** is based on the concept of **keys**, which leads to **strong relationships** in **structured data**.
- It also incorporates the concepts of **integrity constraints** and **normalization**.



# Relational Model

- The **relational model** is the **most common** and widely used model today.
- It is based on the concept of **relations**. A **relation** is a table with **rows** and **columns**.
- The **relational model** is based on the concept of **keys**, which leads to *strong relationships* in structured data.
- It also incorporates the concepts of **integrity constraints** and **normalization**.

→ FN  
↓  
↓ FN

→ standardize

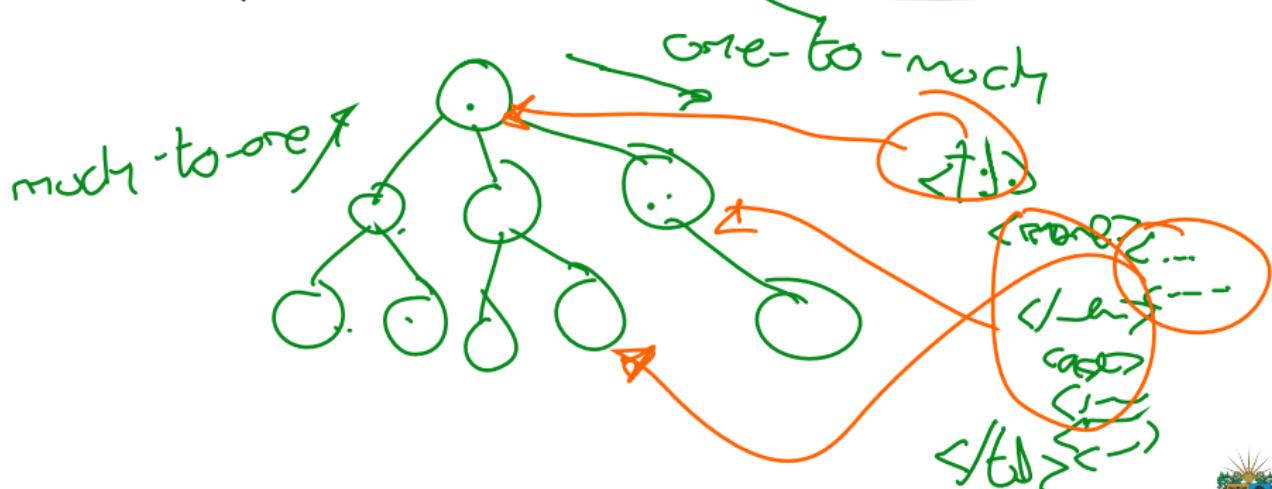
→ ↗ no redundancy

→ ↗ no m-m relations

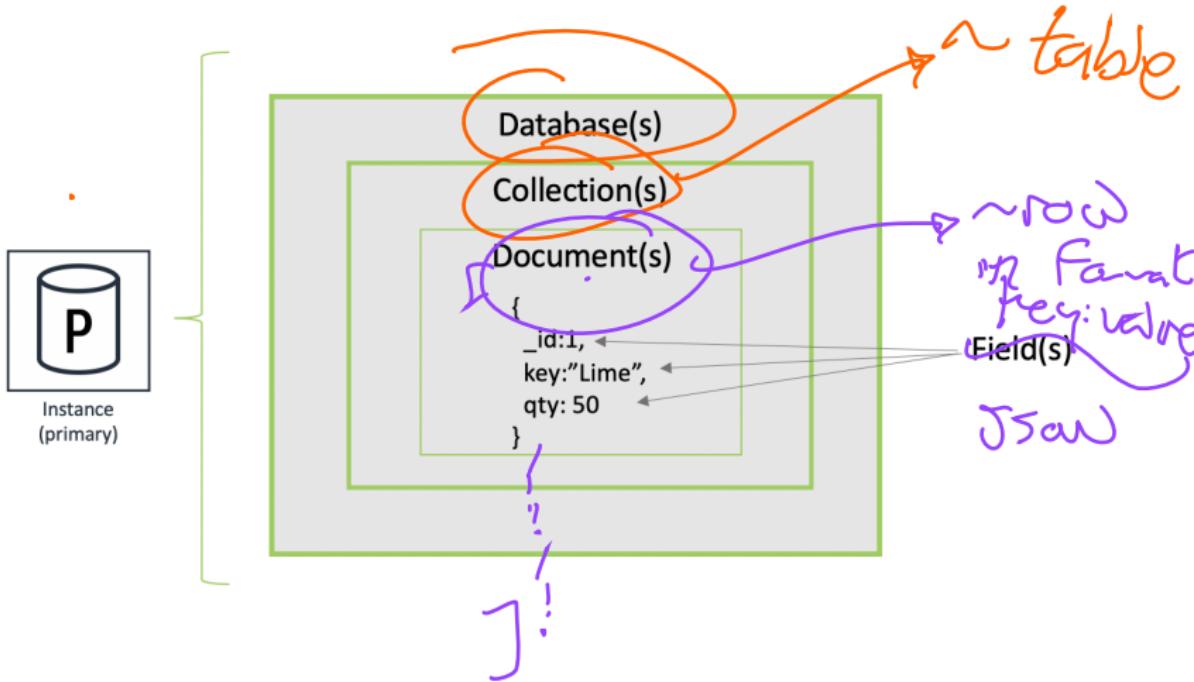


# Hierarchical Model

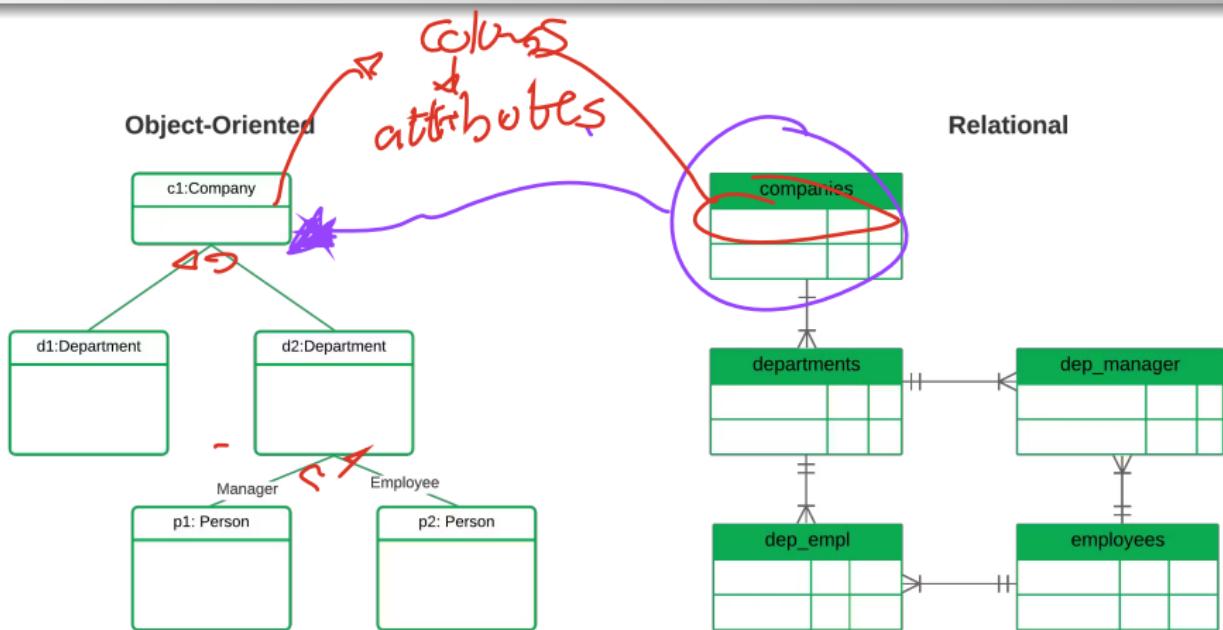
- The **hierarchical model** organizes data in a **tree-like structure**.
- It is based on the concept of **parent-child relationships**, meaning **one-to-many** relationships.
- An example of a **hierarchical model** is the **XML format**.



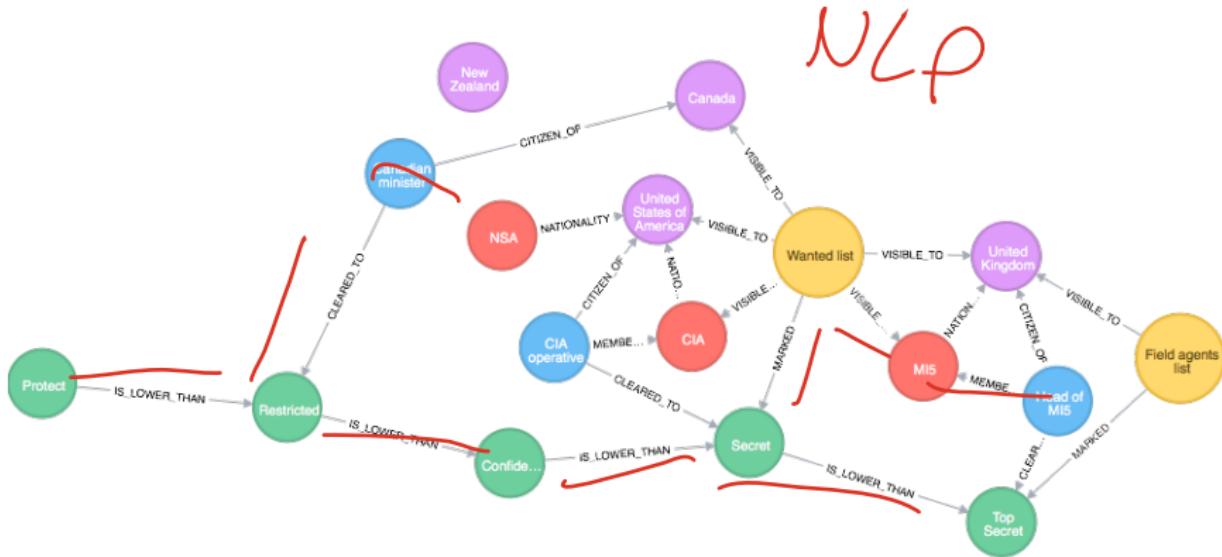
# Document-Based Model



# Object-Oriented Model

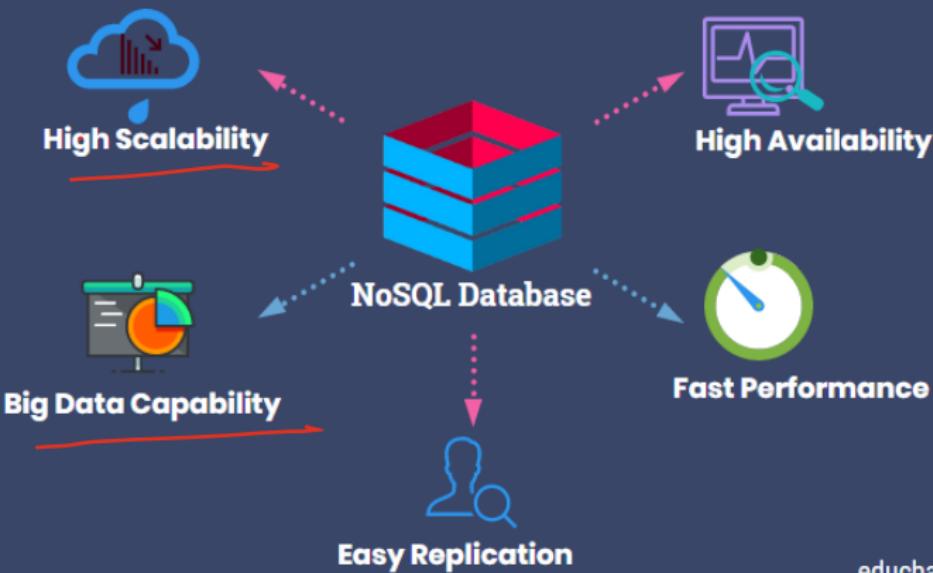


# Graph-Based Model



# NoSQL Model

## What is NoSQL Database



educba.com



# Outline

1 Software Components and Applications

2 Glosary

3 DataBase Classification

4 Relational Database Design

5 Entity-Relation Model (MER)

6 DataBase Management Systems — DBMS

7 DataBases Infrastructure

8 DevOps

9 Data Engineering





# Database Design Foundations

- In the context of **databases**, **designing** a database is the process of producing a **detailed data model**.
- This **data model** encompasses all the required **logical and physical design decisions**, as well as **physical storage parameters**, to generate a design in a *data definition language* that can subsequently be used to create the database.
- A **fully attributed data model** contains detailed attributes for **each entity**.
- Relational data models avoid redundancy and inconsistency by ensuring that data is **normalized**.



# Database Design Foundations

- In the context of **databases**, **designing** a database is the process of producing a **detailed data model**.
- This **data model** encompasses all the required **logical and physical design decisions**, as well as **physical storage parameters**, to generate a design in a **data definition language** that can subsequently be used to create the database.
- A **fully attributed data model** contains detailed attributes for each entity.
- Relational data models avoid redundancy and **inconsistency** by ensuring that data is **normalized**.



# Database Design Foundations

- In the context of **databases**, **designing** a database is the process of producing a **detailed data model**.
- This **data model** encompasses all the required **logical and physical design decisions**, as well as **physical storage parameters**, to generate a design in a *data definition language* that can subsequently be used to create the database.
- A **fully attributed data model** contains **detailed attributes** for **each entity**.
- Relational data models avoid redundancy and **inconsistency** by ensuring that data is **normalized**.



# Database Design Foundations

- In the context of **databases**, **designing** a database is the process of producing a **detailed data model**.
- This **data model** encompasses all the required **logical and physical design decisions**, as well as **physical storage parameters**, to generate a design in a *data definition language* that can subsequently be used to create the database.
- A **fully attributed data model** contains detailed attributes for **each entity**.
- **Relational data models** avoid **redundancy** and **inconsistency** by ensuring that data is **normalized**.



# Set Theory in Databases

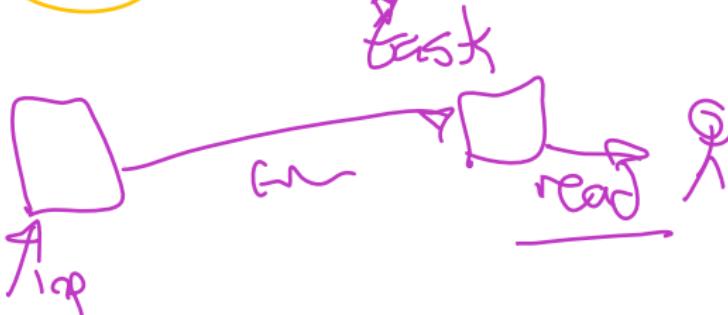
- The set theory is a branch of mathematical logic that studies sets, which are collections of objects.
- The set theory is applied in databases to define the relational model and the relational algebra.
- The relational model is a mathematical model of data for large shared data banks and it has a solid theoretical foundation.
- The relational algebra is a procedural query language, which takes relations as input and produces relations as output.



# Set Theory in Databases



- The **set theory** is a branch of **mathematical logic** that studies sets, which are **collections of objects**.
- The **set theory** is applied in **databases** to define the **relational model** and the **relational algebra**.
- The **relational model** is a **mathematical model** of data for large shared **data banks** and it has a **solid theoretical foundation**.
- The **relational algebra** is a **procedural query language**, which takes relations as **input** and produces relations as **output**.



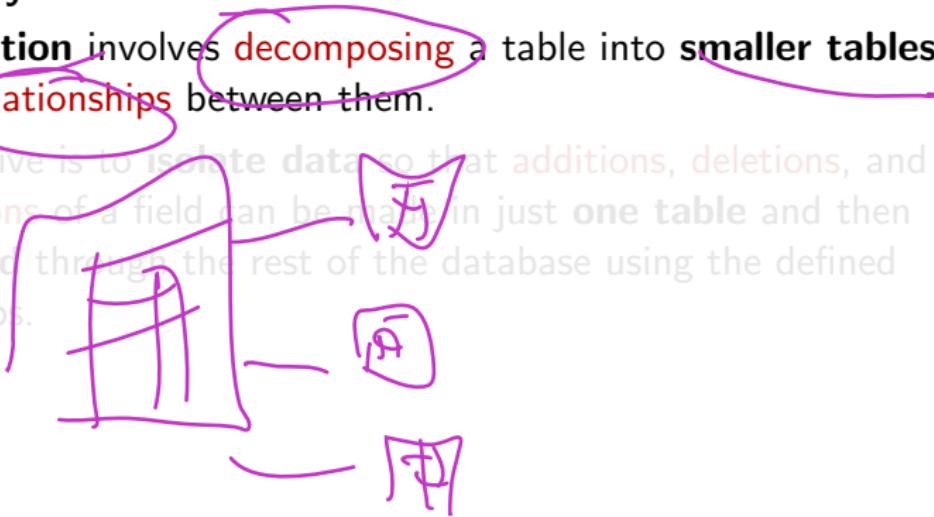
# Normalization in Databases

- **Normalization** is the process of **organizing** the **columns** (attributes) and **tables** (relations) of a relational database to **minimize data redundancy**.
- Normalization involves decomposing a table into smaller **tables** and defining **relationships** between them.
- The objective is to **isolate data** so that **additions**, **deletions**, and **modifications** of a field can be made in just **one table** and then **propagated** through the rest of the database using the defined relationships.



# Normalization in Databases

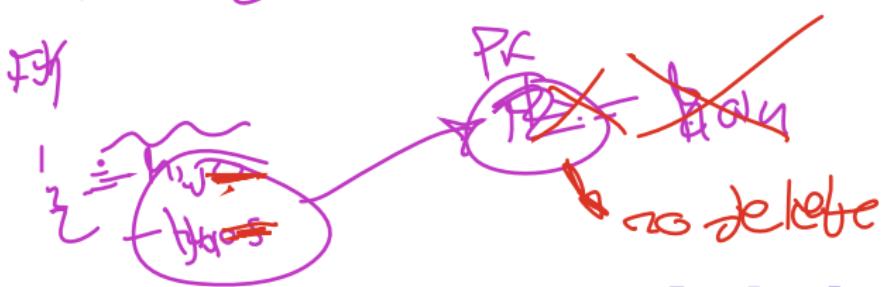
- **Normalization** is the process of **organizing** the **columns** (attributes) and **tables** (relations) of a relational database to **minimize data redundancy**.
- **Normalization** involves **decomposing** a table into **smaller tables** and defining **relationships** between them.
- The objective is to **isolate data** so that **additions**, **deletions**, and **modifications** of a field can be made in just **one table** and then **propagated** through the rest of the database using the defined relationships.



# Normalization in Databases

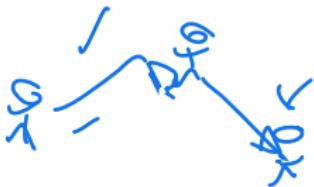
- **Normalization** is the process of **organizing** the **columns** (attributes) and **tables** (relations) of a relational database to **minimize data redundancy**.
- **Normalization** involves **decomposing** a table into **smaller tables** and defining **relationships** between them.
- The objective is to **isolate data** so that **additions**, **deletions**, and **modifications** of a field can be made in just **one table** and then **propagated** through the rest of the database using the defined **relationships**.

→ Cascade



# Normal Levels

- ① **First normal form (1NF)**: The table is a two-dimensional table with rows and columns. Each column contains atomic values, and there are no repeating groups or arrays.
- ② **Second normal form (2NF)**: The table is in first normal form and all the non-key attributes are fully functionally dependent on the primary key.
- ③ **Third normal form (3NF)**: The table is in second normal form and all the non-key attributes are non-transitively dependent on the primary key.
- ④ **Fourth normal form (4NF)**: The table is in third normal form and there are no multi-valued dependencies.



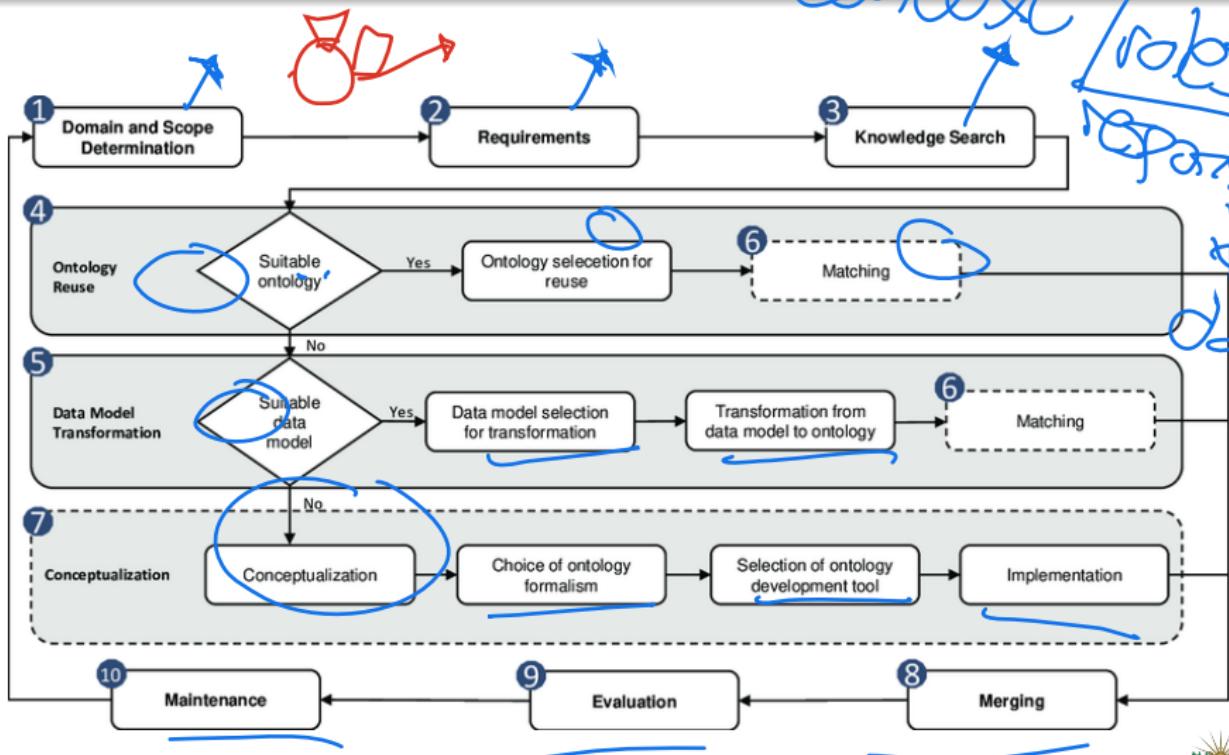
# Ontologies

*web semantics*

- An **ontology** is a formal naming and definition of the types, properties, and interrelationships of the entities that really or fundamentally exist for a particular domain of discourse.
- **Ontologies** are used in databases to define the schema of the database.
- The **schema** of a database is a formal definition of the structure of the database: the types of data that are stored, the relationships between the data, and the constraints on the data.



# Ontology Workflow



# Outline

- 1 Software Components and Applications
- 2 Glosary
- 3 DataBase Classification
- 4 Relational Database Design
- 5 Entity-Relation Model (MER)
- 6 DataBase Management Systems — DBMS
- 7 DataBases Infrastructure
- 8 DevOps
- 9 Data Engineering



# Definitions

- **Entity:** A thing or object in the real world that is distinguishable from other objects.
- **Attribute:** A property or characteristic of an entity.
- **Relationship:** An association between entities.
- **Cardinality:** The number of instances of an entity that can be associated with another entity.
- **Degree:** The number of entities that participate in a relationship.



List  
of  
rows



col  
List  
of  
objects



# Definitions

- **Entity:** A thing or object in the real world that is **distinguishable** from other **objects**.
- **Attribute:** A **property** or characteristic of an **entity**.
- Relationship: An association between entities.
- Cardinality: The number of instances of an entity that can be associated with another entity.
- Degree: The number of entities that participate in a relationship.

**4 Data type**  
constraints

source → API \$

web scraping  
errors... low quality

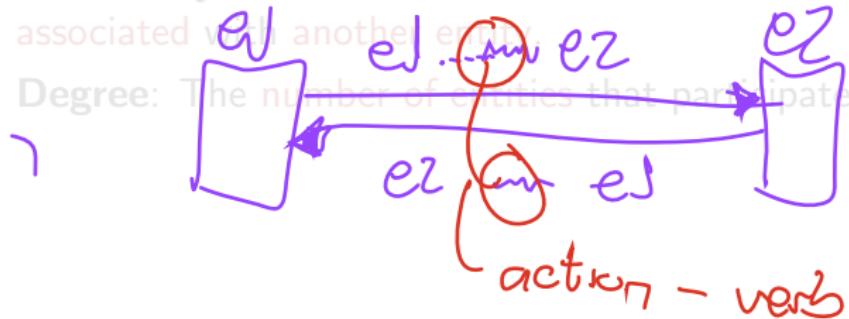
HTML → tags

metatag → JSON



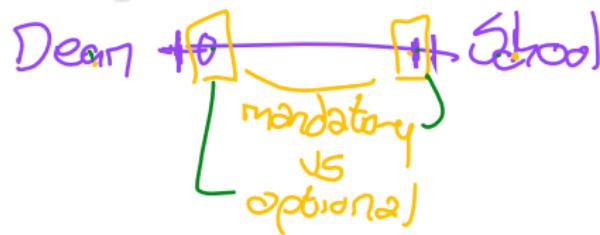
# Definitions

- **Entity:** A thing or object in the real world that is **distinguishable** from other **objects**.
- **Attribute:** A **property** or characteristic of an **entity**.
- **Relationship:** An **association** between entities.
- **Cardinality:** The number of **instances** of an **entity** that can be associated with another entity.
- **Degree:** The number of **entities** that participate in a **relationship**.



# Definitions

- **Entity:** A thing or object in the real world that is **distinguishable** from other **objects**.
- **Attribute:** A **property** or characteristic of an **entity**.
- **Relationship:** An **association** between entities.
- **Cardinality:** The number of **instances** of an **entity** that can be **associated** with another entity.
- **Degree:** The number of entities that participate in a relationship.

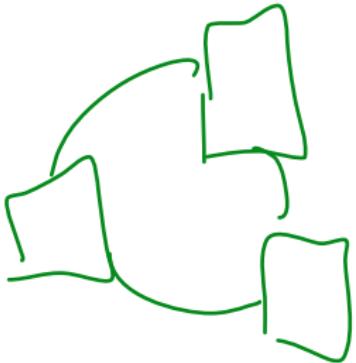


# Definitions

- **Entity:** A thing or object in the real world that is **distinguishable** from other **objects**.
- **Attribute:** A **property** or characteristic of an **entity**.
- **Relationship:** An **association** between entities.
- **Cardinality:** The number of **instances** of an **entity** that can be associated with another entity.
- **Degree:** The **number of entities** that participate in a **relationship**.



---



# Entity-Relation Model

- The **Entity-Relation Model** is a graphical representation of the **entities** and their **relationships** in a **database**.
- The Entity-Relation Model is used to design the schema of a database and to communicate the design to stakeholders.
- Following a process based on a **ontology** it is easy to define the entities, attributes, and relationships of the database.



# Entity-Relation Model

- The **Entity-Relation Model** is a graphical representation of the entities and their relationships in a database.
- The **Entity-Relation Model** is used to **design** the **schema** of a database and to **communicate** the **design** to **stakeholders**.
- Following a process based on a ontology it is easy to define the entities, attributes, and relationships of the database.

**Database**

↳ Schemas

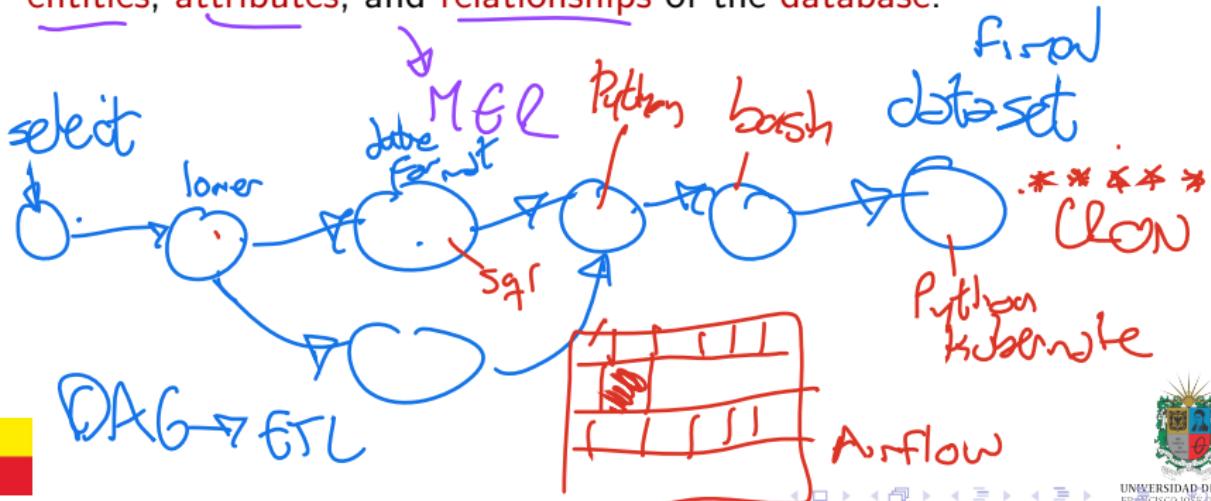
↳ Tables

↳ Rows

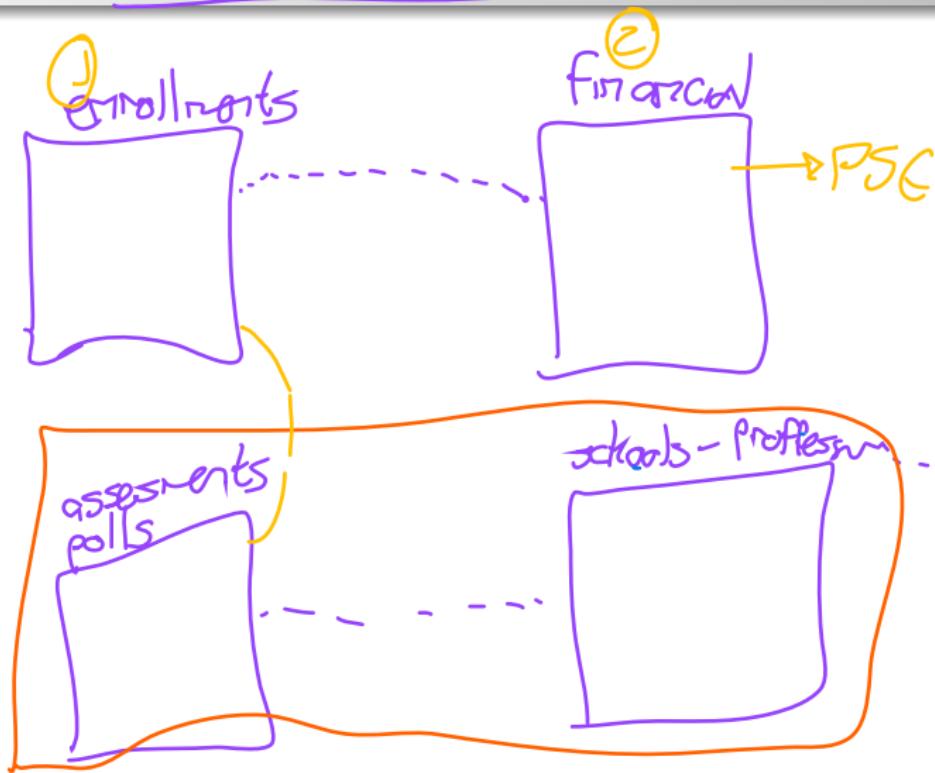


# Entity-Relation Model

- The **Entity-Relation Model** is a **graphical representation** of the **entities** and their **relationships** in a **database**.
- The **Entity-Relation Model** is used to **design** the **schema** of a **database** and to **communicate** the **design** to **stakeholders**.
- Following a process based on a **ontology** it is easy to **define** the **entities**, **attributes**, and **relationships** of the **database**.



# Step 1. Define Components



## Step 2. Define Entities



e- student

et Professor

# FINANCIAC

## c7. Payment

ets-Bark

e-Services

## Don ell - Payment Method

e3-Group

eb-Groß

6

e12-CourseType

## e14. Location type

domain  $\begin{cases} \text{no charge} \\ \text{simple} \end{cases}$  } 1) same description

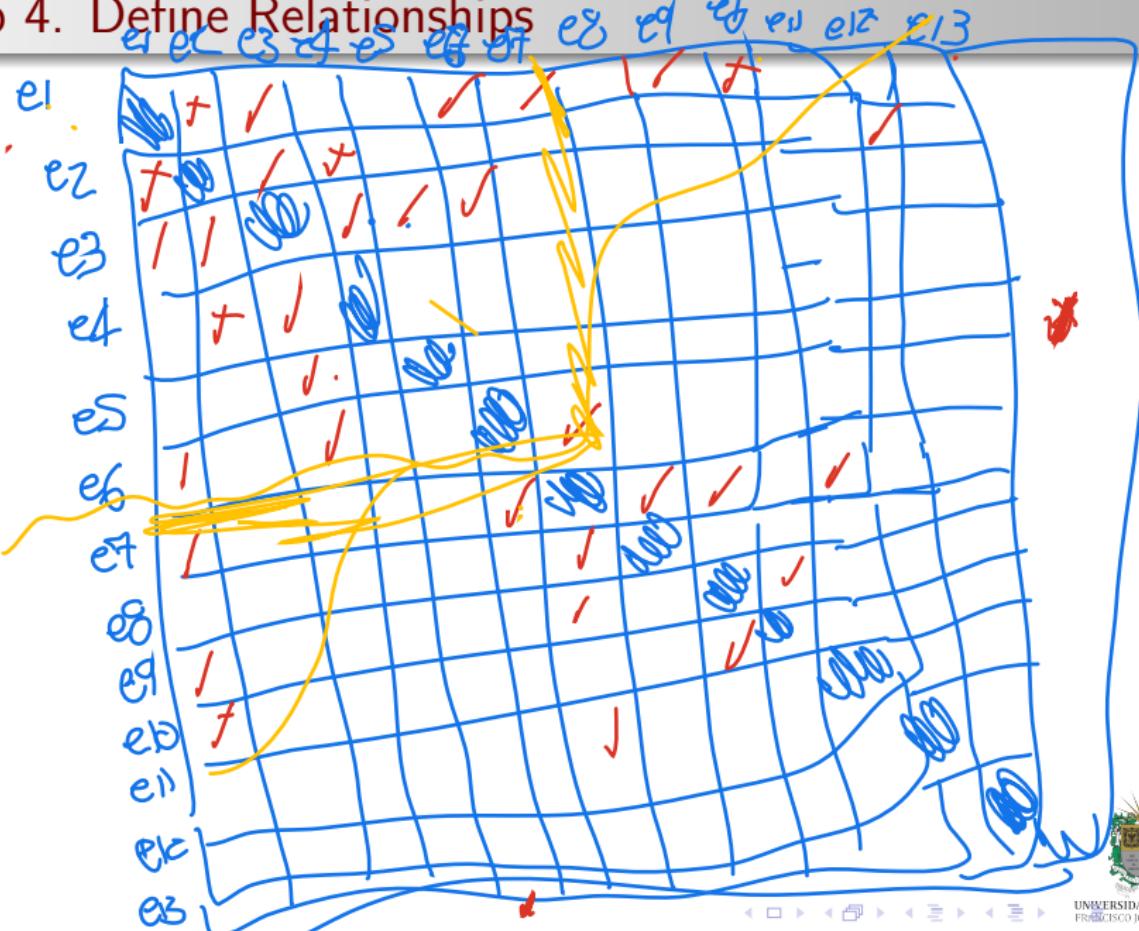


## Step 3. Define Attributes per Entity

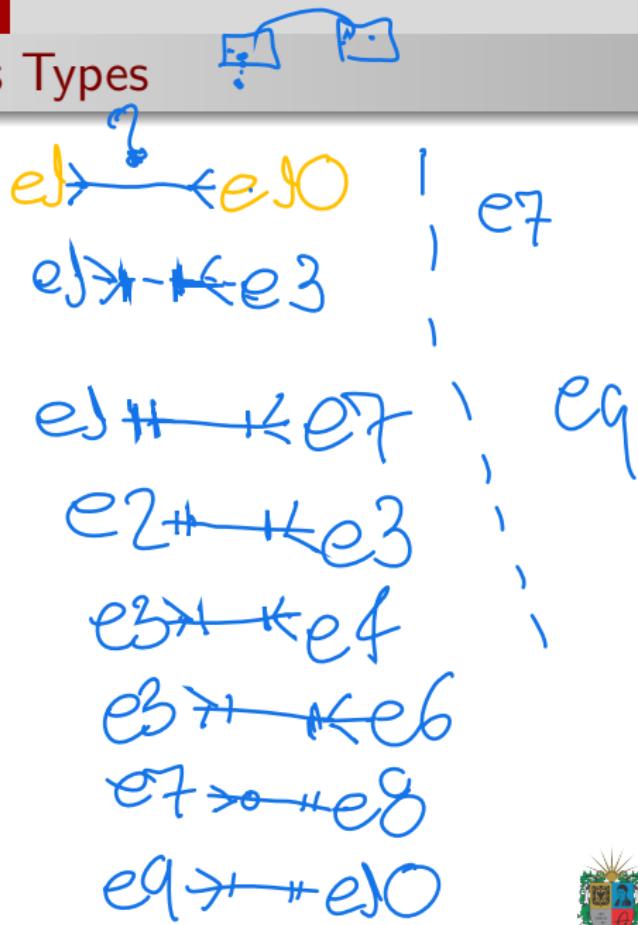
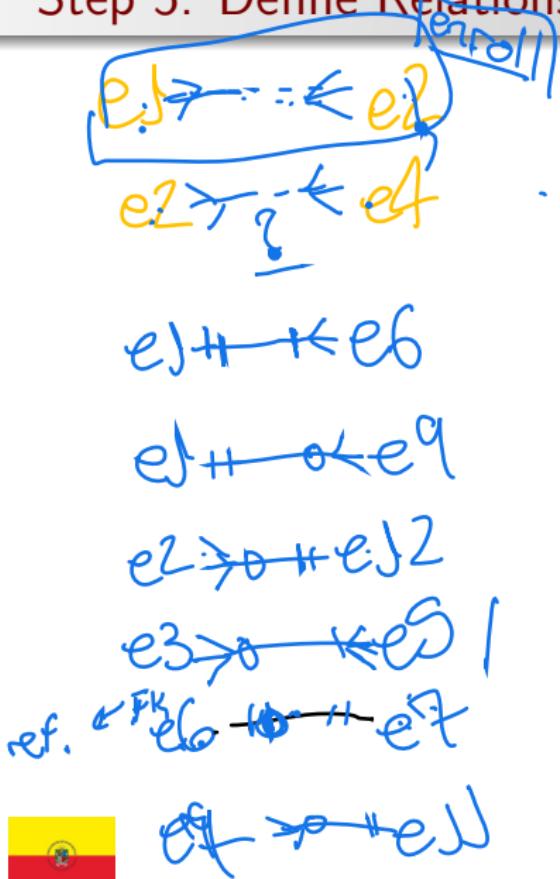
- e1. name, address, email, phone, birthday, national-id
- e2. name, credit, description, code, type
- e3. code, name, course, rooms, max-students, schedule
- e4. code, email, name, max-degree, phone
- e5. building, name, description, lat-long, type, max-occupancy
- e6. student, group, registration-date, session
- e7. value, tax, due-date, student, service, payment-method, observations, successful
- e8. name, cost, description,
- e9. name, type, number, bank, debt
- e10. Name, country, NIT
- e11. id, name, description, rate
- e12. id, name, description
- e13. id, name, description



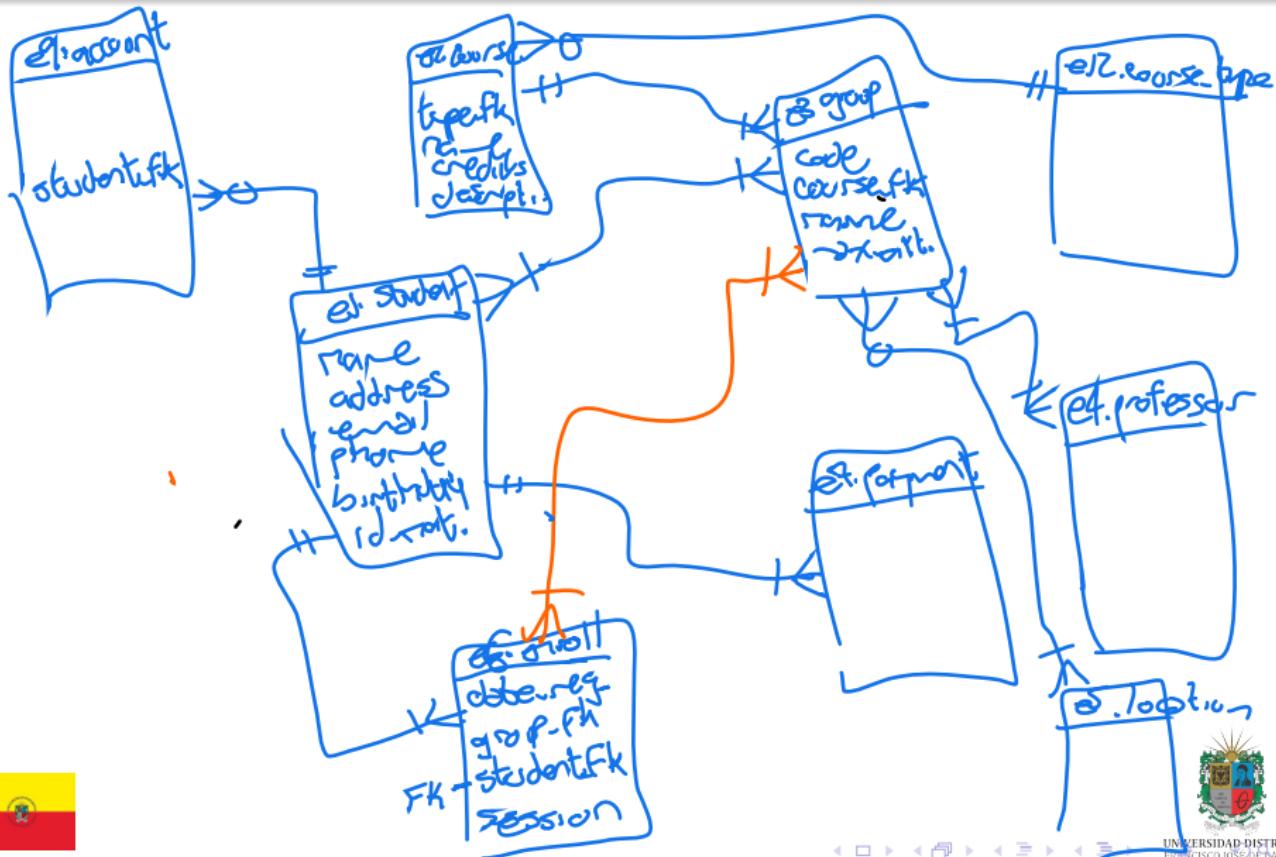
#### Step 4. Define Relationships



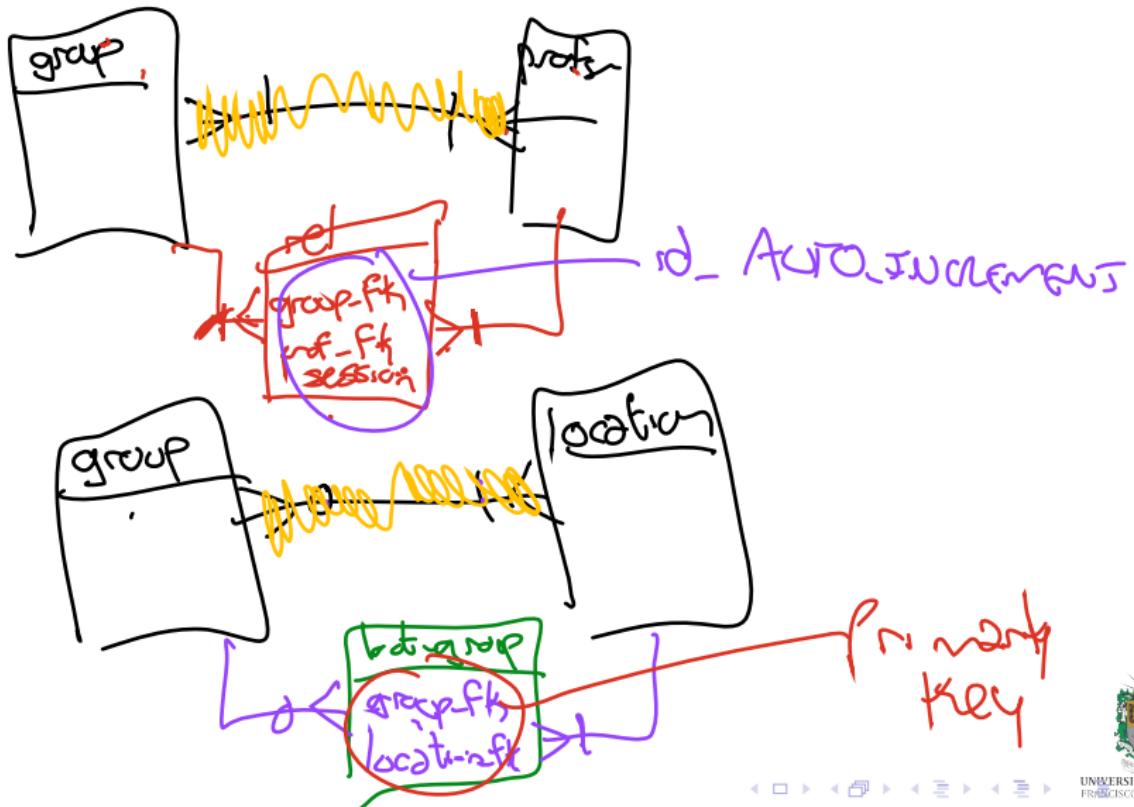
## Step 5. Define Relationships Types



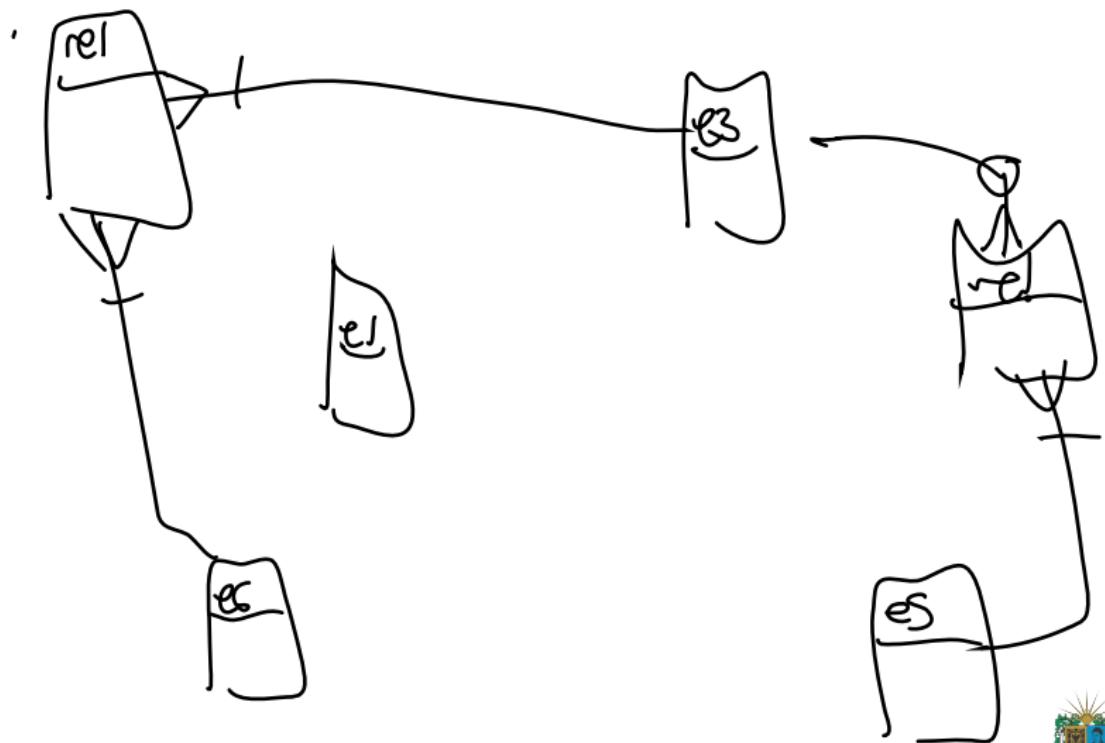
# Step 6. First Entity-Relationship Model Draw



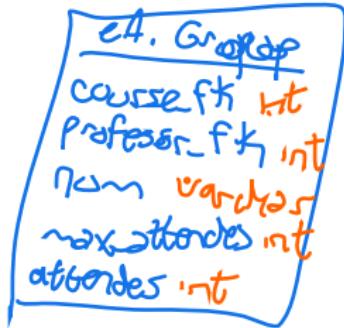
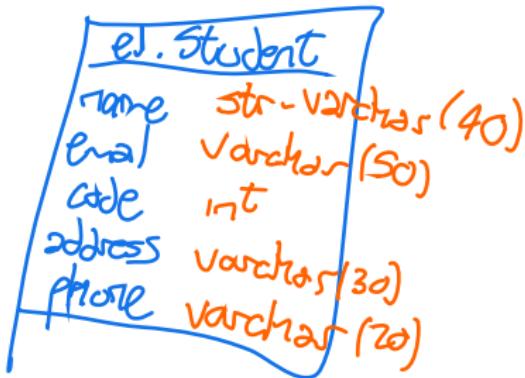
# Step 7. Split Many-to-Many Relationships



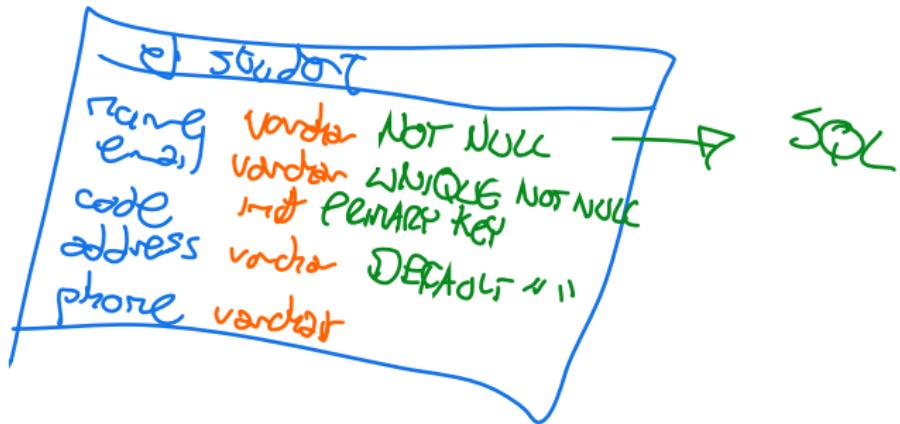
# Step 8. Second Entity-Relationship Model Draw



# Step 9. Get Data-Structure Entity-Relationship Model



# Step 10. Define Constraints and Properties of Data



```
CREATE TABLE IF NOT EXISTS student(
    name VARCHAR(40) NOT NULL,
    email VARCHAR(50) UNIQUE NOT NULL,
    code INT PRIMARY KEY,
    address VARCHAR(50) DEFAULT '',
    phone VARCHAR(30)
)
```



# Outline

- 1 Software Components and Applications
- 2 Glosary
- 3 DataBase Classification
- 4 Relational Database Design
- 5 Entity-Relation Model (MER)
- 6 DataBase Management Systems — DBMS
- 7 DataBases Infrastructure
- 8 DevOps
- 9 Data Engineering



# What is a DBMS?

- A **Database Management System (DBMS)** is a **software system** that uses a **standard** method to **store** and **organize** data.
- A **DBMS** is a **software system** that allows users to define, create, maintain, and control access to the database.
- A **DBMS** is a **software package** designed to manipulate, retrieve, and manage data in a database.



# What is a DBMS?

- A **Database Management System** (DBMS) is a **software system** that uses a **standard** method to **store** and **organize data**.
- A **DBMS** is a **software system** that allows users to **define**, **create**, **maintain**, and **control access** to the database.
- A **DBMS** is a **software package** designed to manipulate, retrieve, and manage data in a database.

transformations

integrity

Grants  
Security



# What is a DBMS?

- A **Database Management System** (DBMS) is a **software system** that uses a **standard** method to **store** and **organize** data.
- A **DBMS** is a **software system** that allows users to **define**, **create**, **Maintain**, and **control access** to the database.
- A **DBMS** is a **software package** designed to **manipulate**, **retrieve**, and **manage data** in a database.

$\hookrightarrow$  SQL  $\Rightarrow$  Declarative



# Pros & Cons of DBMS

## • Pros:

- **Data Independence:** Data is stored independently of the applications that use it.
- **Data Integrity:** Data is consistent and accurate.
- **Data Security:** Data is protected from unauthorized access.
- **Data Recovery:** Data can be recovered in case of failure.

## • Cons:

- Complexity: DBMS are complex systems.
- Cost: DBMS are expensive for bigger data volumes.
- Performance: DBMS can be slow.



# Pros & Cons of DBMS

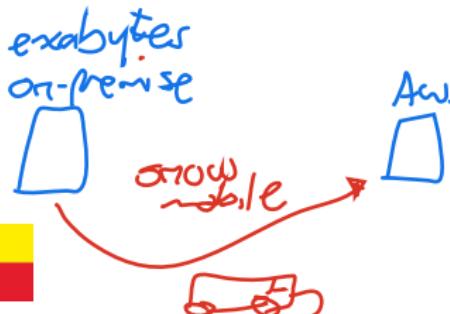
## • Pros:

- **Data Independence:** Data is **stored independently** of the applications that use it.
- **Data Integrity:** Data is **consistent** and **accurate**.
- **Data Security:** Data is **protected** from **unauthorized access**.
- **Data Recovery:** Data can be **recovered** in case of failure.

## • Cons:

- **Complexity:** DBMS are **complex systems**.
- **Cost:** DBMS are **expensive** for **bigger data volumes**.
- **Performance:** DBMS can be **slow**.

*Maintain*



# GUI Assistants

- A **Graphical User Interface (GUI)** is a type of user interface that allows users to **interact** with electronic devices using **graphical icons** and **visual indicators**.
- GUIs are **easier** to use than Command Line Interfaces (**CLI**) because they allow users to **interact** with the system using **visual elements** such as **windows**, **buttons**, and **menus**.
- GUIs are more **intuitive** and **user-friendly** than CLIs, which makes them ideal for **users** who are **not familiar** with the system.

MySQL  $\Rightarrow$  WorkBench

Postgres  $\Rightarrow$  PgAdmin



# Case of Study: DBeaver

*Agnostic*

The screenshot shows the DBeaver 22.1.3 interface with the following details:

- Database Navigator:** Shows the database structure. Under the 'system' database, there is a 'mover' schema containing tables like 'crdb\_internal', 'public', 'rides', 'user\_promo\_codes', 'users', 'vehicle\_location\_histories', and 'vehicles'. There are also 'Views', 'Indexes', 'Functions', 'Sequences', and system info.
- Project - General:** Shows a project named 'DataSource' with items: Bookmarks, ER Diagrams, and Scripts.
- Properties View:** For the 'rides' table (Object ID: 111, Owner: root).
 

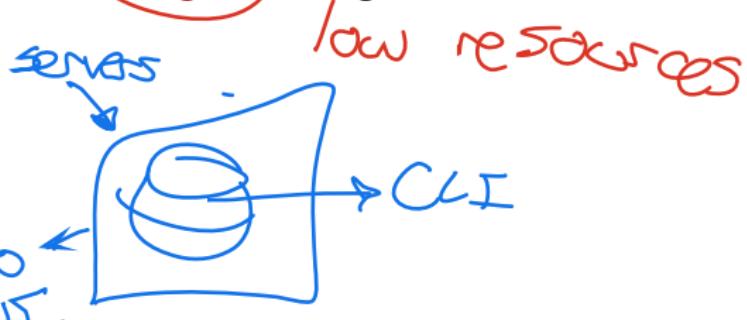
Column Name	#	Data type	Identity	Collation	Not Null	Default	Comments
id	1	uuid			[v]		
city	2	varchar	default		[v]		
vehicle_city	3	varchar	default				
rider_id	4	uuid					
vehicle_id	5	uuid					
start_address	6	varchar	default				
end_address	7	varchar	default				
start_time	8	timestamp					
end_time	9	timestamp					
revenue	10	numeric(10, 2)					



# Command Line

*Command Prompt Terminal*

- A **Command Line Interface** (CLI) is a type of user interface that allows users to interact with electronic devices using text-based commands.
- CLIs are more powerful and flexible than GUIs because they allow users to perform complex tasks using simple commands.
- CLIs are more efficient than GUIs because they do not require users to navigate through menus and windows to perform tasks.



# Case of Study: MariaDB CLI

```
arnel@arnel.com [~]# mysql -u arnel_test2 -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8643
Server version: 10.1.25-MariaDB MariaDB Server

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database      |
+-----+
| arnel_test1   |
| arnel_test2   |
| information_schema |
+-----+
3 rows in set (0.00 sec)

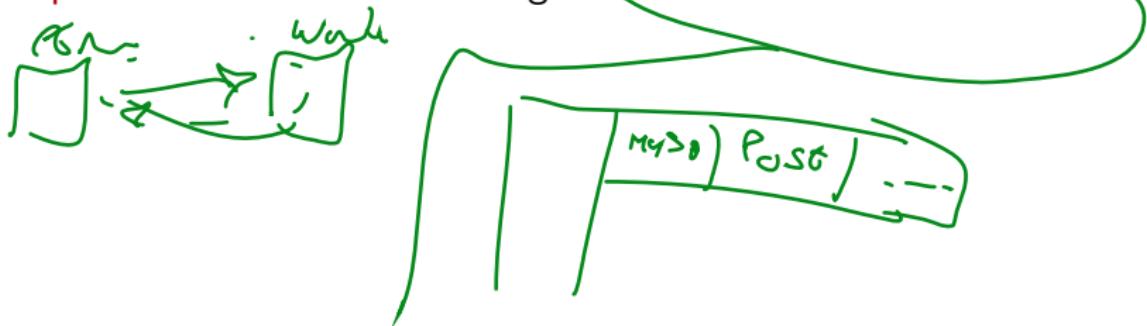
MariaDB [(none)]> use arnel_test1
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [arnel_test1]>
```



# Why use an agnostic tool?

- An **agnostic tool** is a tool that is **not tied** to a specific technology or platform.
- **Agnostic tools** are useful because they allow users to work with multiple databases without having to learn different tools.
- **Agnostic tools** are also useful because they allow users to work with multiple databases without having to switch between different tools.



# Outline

1 Software Components and Applications

2 Glosary

3 DataBase Classification

4 Relational Database Design

5 Entity-Relation Model (MER)

6 DataBase Management Systems — DBMS

7 DataBases Infrastructure

8 DevOps

9 Data Engineering



# What is the Cloud Computing?

- **Cloud computing** is the delivery of **computing services** over the **internet**.
- **Cloud computing** allows users to **access computing resources** such as **servers, storage, and databases on demand**.



# What is the Cloud Computing?

- **Cloud computing** is the delivery of **computing services** over the **internet**.
- **Cloud computing** allows users to **access computing resources** such as **servers, storage, and databases on demand**.



# What is On-Premises Computing?

- **On-premises computing** is the **traditional** way of accessing computing resources.
- On-premises computing requires users to purchase and maintain their own computing resources such as servers, storage, and databases.



# What is On-Premises Computing?

- **On-premises computing** is the **traditional** way of accessing computing resources.
- **On-premises computing** requires users to **purchase** and **Maintain** their own **computing** resources such as **servers**, **storage**, and **databases**.



# Pros & Cons of Cloud Computing

## • Pros:

- **Cost-Effective:** *Cloud computing* is a **cost-effective** way to access computing resources.
- **Scalable:** *Cloud computing* is a **scalable** way to access computing resources.
- **Flexible:** *Cloud computing* is a **flexible** way to access computing resources.

## • Cons:

- **Dependency:** One of the main cons of cloud computing is dependency on external providers. If the provider goes down or becomes unavailable, it can affect the availability of your data and services.
- **Cost:** While cloud computing can be cost-effective, it can also lead to unexpected costs if you don't manage your usage effectively. There may be hidden fees or charges for additional resources or storage.
- **Security:** Security is a major concern with cloud computing. Data stored in the cloud is vulnerable to hacking and other security threats. It's important to use strong encryption and follow best practices to protect your data.
- **Loss of Control:** When you use cloud computing, you lose some level of control over your data and how it's used. You may have less control over who has access to your data and how it's stored.
- **Regulatory Compliance:** Depending on your industry, there may be specific regulations that require you to store data in a certain location or follow specific data handling procedures. This can make it challenging to use cloud computing while still complying with regulations.



# Pros & Cons of Cloud Computing

## • Pros:

- **Cost-Effective:** Cloud computing is a cost-effective way to access computing resources.
- **Scalable:** Cloud computing is a scalable way to access computing resources.
- **Flexible:** Cloud computing is a flexible way to access computing resources.

## • Cons:

- **Security:** Cloud computing can be less secure than on-premises computing.
- **Data Loss:** Data loss can occur due to various reasons such as network issues, hardware failures, or software bugs.
- **Dependency:** Dependence on cloud providers can limit control over data and infrastructure.
- **Cost:** While initial costs may be lower, ongoing costs for storage, bandwidth, and compute can add up over time.
- **Regulatory Compliance:** Ensuring compliance with various regulations like GDPR, CCPA, and HIPAA can be challenging and costly.



# Pros & Cons of Cloud Computing

## • Pros:

- **Cost-Effective:** Cloud computing is a cost-effective way to access computing resources.
- **Scalable:** Cloud computing is a scalable way to access computing resources.
- **Flexible:** Cloud computing is a flexible way to access computing resources.

## • Cons:

- Security: Cloud computing can be less secure than on-premises computing.
- Performance: Cloud computing can be slower than on-premises computing.
- Cost: Cloud computing can be more expensive than on-premises computing.



# Pros & Cons of Cloud Computing

## • Pros:

- **Cost-Effective:** Cloud computing is a cost-effective way to access computing resources.
- **Scalable:** Cloud computing is a scalable way to access computing resources.
- **Flexible:** Cloud computing is a flexible way to access computing resources.

## • Cons:

- **Security:** Cloud computing can be less secure than on-premises computing.
- **Performance:** Cloud computing can be slower than on-premises computing.
- **Reliability:** Cloud computing can be less reliable than on-premises computing.



# Pros & Cons of Cloud Computing

## • Pros:

- **Cost-Effective:** Cloud computing is a cost-effective way to access computing resources.
- **Scalable:** Cloud computing is a scalable way to access computing resources.
- **Flexible:** Cloud computing is a flexible way to access computing resources.

## • Cons:

- **Security:** Cloud computing can be less secure than on-premises computing.
- **Performance:** Cloud computing can be slower than on-premises computing.
- **Reliability:** Cloud computing can be less reliable than on-premises computing.



# Pros & Cons of Cloud Computing

## • Pros:

- **Cost-Effective:** Cloud computing is a cost-effective way to access computing resources.
- **Scalable:** Cloud computing is a scalable way to access computing resources.
- **Flexible:** Cloud computing is a flexible way to access computing resources.

## • Cons:

- **Security:** Cloud computing can be less secure than on-premises computing.
- **Performance:** Cloud computing can be slower than on-premises computing.
- **Reliability:** Cloud computing can be less reliable than on-premises computing.



# SaaS Vs. IaaS Vs. PaaS

- **Software as a Service (SaaS)** is a **software distribution** model in which a **third-party** provider **hosts applications** and makes them available to customers over the **internet**.
- **Infrastructure as a Service (IaaS)** is a **cloud computing** model that provides **virtualized computing** resources over the **internet**.
- **Platform as a Service (PaaS)** is a **cloud computing** model that provides a **platform** for developers to build, deploy, and **manage** applications over the **internet**.



# SaaS Vs. IaaS Vs. PaaS

- **Software as a Service (SaaS)** is a **software distribution** model in which a **third-party** provider **hosts applications** and makes them available to customers over the **internet**.
- **Infrastructure as a Service (IaaS)** is a **cloud computing** model that provides **virtualized computing** resources over the **internet**.
- **Platform as a Service (PaaS)** is a **cloud computing** model that provides a **platform** for developers to build, deploy, and **manage** applications over the **internet**.

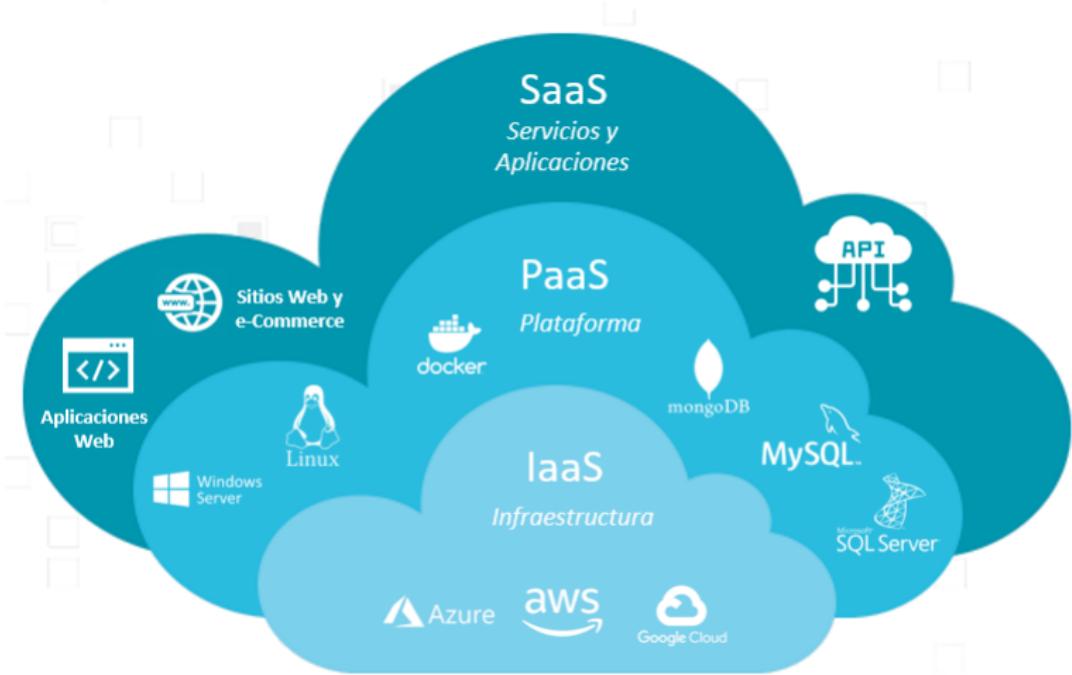


# SaaS Vs. IaaS Vs. PaaS

- **Software as a Service (SaaS)** is a **software distribution** model in which a **third-party** provider **hosts applications** and makes them available to customers over the **internet**.
- **Infrastructure as a Service (IaaS)** is a **cloud computing** model that provides **virtualized computing** resources over the **internet**.
- **Platform as a Service (PaaS)** is a **cloud computing** model that provides a **platform for developers** to build, deploy, and **manage applications** over the **internet**.



# Cloud Levels

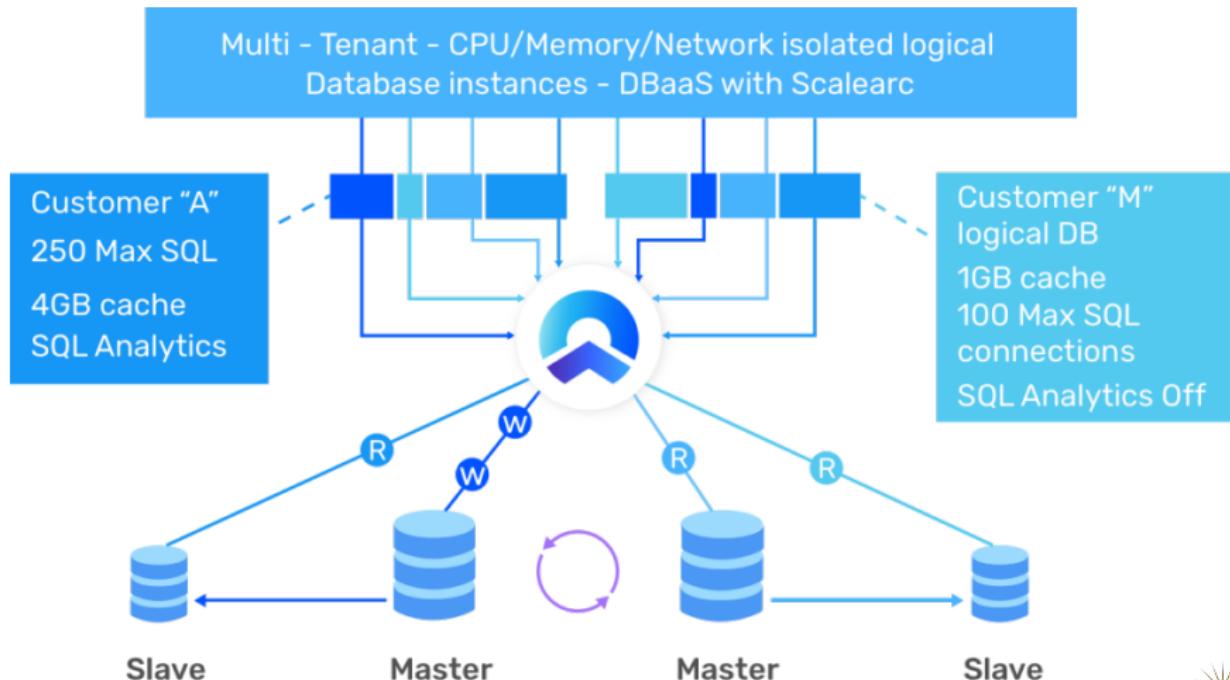


# DataBases as a Service

**Database as a Service (DBaaS)** is a **cloud computing model** that provides **database services** over the **internet**.



# Case of Study: DBaaS Custom for Clients



# Localhost

- **Localhost** is a **hostname** that refers to the **local computer** that a **program** is **running on**.
- **Localhost** is used to **access the services** that are **running on** the **local computer**.
- **Localhost** is used to **access the database services** that are **running on** the **local computer**.
- **Localhost** is used to **access the web services** that are **running on** the **local computer**.



# Localhost

- **Localhost** is a **hostname** that refers to the **local computer** that a **program** is **running on**.
- **Localhost** is used to **access the services** that are **running on** the **local computer**.
- **Localhost** is used to **access the database services** that are **running on** the **local computer**.
- **Localhost** is used to **access the web services** that are **running on** the **local computer**.



# Localhost

- **Localhost** is a **hostname** that refers to the **local computer** that a **program** is **running on**.
- **Localhost** is used to **access the services** that are **running on** the **local computer**.
- **Localhost** is used to **access the database services** that are **running on** the **local computer**.
- **Localhost** is used to **access the web services** that are **running on** the **local computer**.



# Monolithic Architecture

- **Monolithic Architecture** is a **software architecture** in which all the components of the software are **combined** into a **single program**.
- **Monolithic Architecture** is a **traditional software architecture** that was used to **build large** and **complex software systems**.
- **Monolithic Architecture** is a simple and **easy-to-understand** software architecture that is used to **build software systems** that do **not require** high scalability and flexibility.



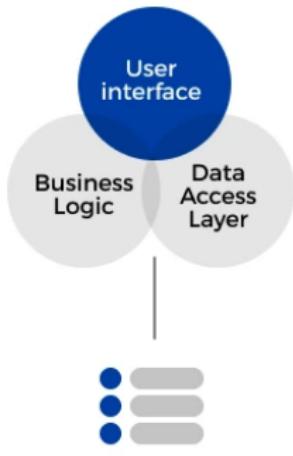
# Microservices Architecture

- **Microservices Architecture** is a **software architecture** in which the components of the software are broken down into **small, independent services**.
- **Microservices Architecture** is a **modern software architecture** that is used to **build large and complex software systems**.
- **Microservices Architecture** is a **flexible and scalable software architecture** that is used to **build software systems** that require high scalability and flexibility.

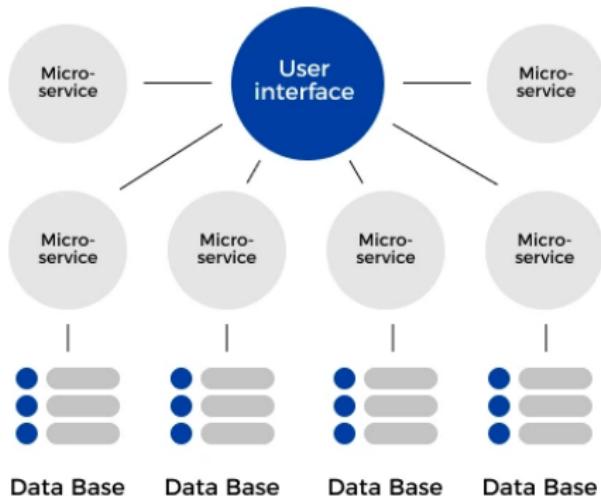


# Monolithic Architecture Schema

## MONOLITHIC ARCHITECTURE



## MICROSERVICE ARCHITECTURE



# Outline

1 Software Components and Applications

2 Glosary

3 DataBase Classification

4 Relational Database Design

5 Entity-Relation Model (MER)

6 DataBase Management Systems — DBMS

7 DataBases Infrastructure

8 DevOps

9 Data Engineering



# Continuous Integration

- **Continuous Integration** is a software development practice in which developers integrate code into a shared repository frequently.
- Continuous Integration is a software development practice that helps developers to detect and fix integration errors early.



# Continuous Integration

- **Continuous Integration** is a **software development practice** in which developers **integrate code** into a shared repository frequently.
- **Continuous Integration** is a **software development practice** that helps developers to **detect** and **fix integration errors** early.



# Continuous Deployment

- Developers **deploy code** into **production** frequently.
- **Continuous Deployment** is a software development practice that helps developers **deliver new features** to customers quickly.
- **Continuous Deployment** is also a practice that helps developers **improve the quality** of the software.



# Continuous Deployment

- Developers **deploy code** into **production** frequently.
- **Continuous Deployment** is a **software development practice** that helps developers **deliver new features** to customers **quickly**.
- **Continuous Deployment** is also a **practice** that helps developers **improve** the **quality** of the **software**.

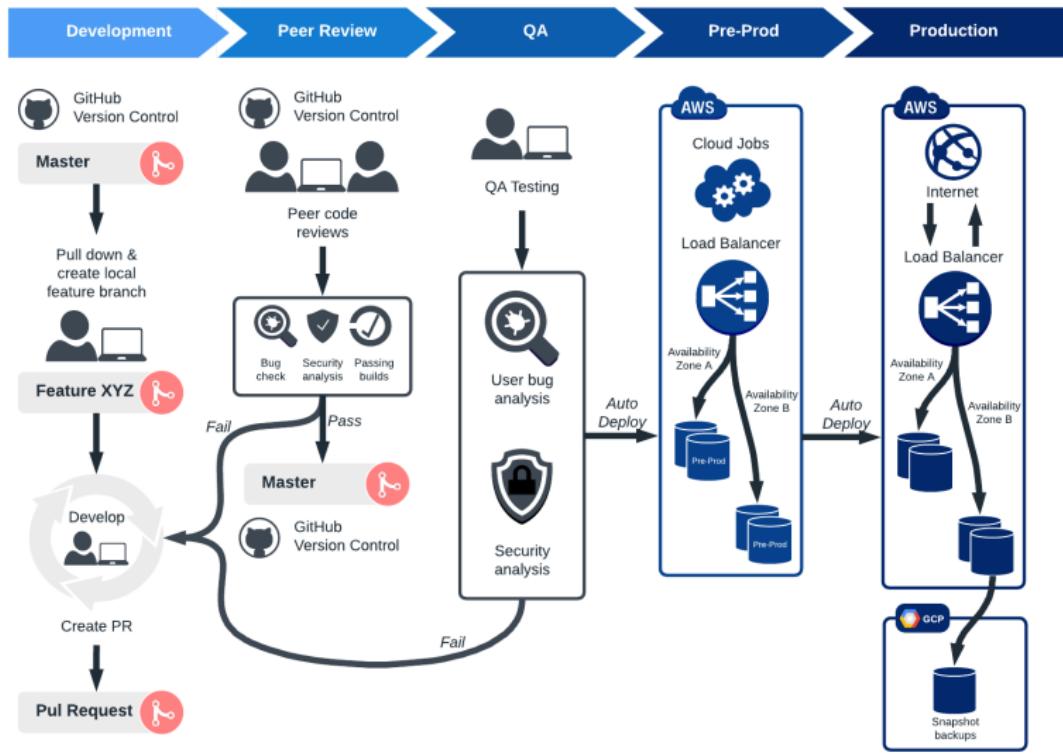


# Continuous Deployment

- Developers **deploy code** into **production** frequently.
- **Continuous Deployment** is a **software development practice** that helps developers **deliver new features** to customers **quickly**.
- **Continuous Deployment** is also a **practice** that helps developers **improve** the **quality** of the **software**.



# Development Workflow using CI/CD



# Containers and Docker

- **Containers** are a **lightweight** and **portable** way to **package software**.
- **Containers** are a method to **isolate applications** from the underlying system.
- Docker is a **platform** that allows developers to **build**, **ship**, and **run** containers.

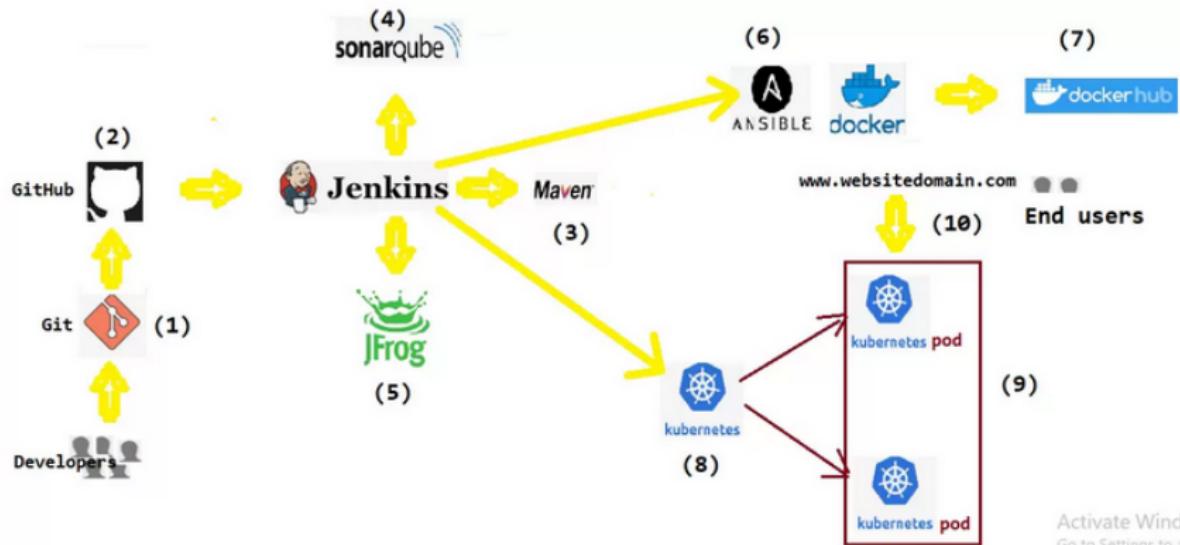


# Containers and Docker

- **Containers** are a **lightweight** and **portable** way to **package software**.
- **Containers** are a method to **isolate applications** from the underlying system.
- **Docker** is a **platform** that allows developers to **build**, **ship**, and **run** **containers**.



# From Code to Docker



# Outline

1 Software Components and Applications

2 Glosary

3 DataBase Classification

4 Relational Database Design

5 Entity-Relation Model (MER)

6 DataBase Management Systems — DBMS

7 DataBases Infrastructure

8 DevOps

9 Data Engineering



# What is Data Engineering?

- **Data Engineering** is the aspect of data science that focuses on practical applications of **data collection** and **analysis**.
- **Data Engineers** are responsible for **building** and **maintaining the architecture** that allows data scientists to perform their work.
- **Data Engineering** is a set of operations aimed at creating interfaces and mechanisms for the **flow** and **access of data**.



# What is Data Engineering?

- **Data Engineering** is the aspect of data science that focuses on practical applications of **data collection** and **analysis**.
- **Data Engineers** are responsible for **building** and **maintaining the architecture** that allows data scientists to perform their work.
- **Data Engineering** is a set of operations aimed at creating interfaces and mechanisms for the **flow** and **access of data**.



# What is Data Engineering?

- **Data Engineering** is the aspect of data science that focuses on practical applications of **data collection** and **analysis**.
- **Data Engineers** are responsible for **building** and **maintaining the architecture** that allows data scientists to perform their work.
- **Data Engineering** is a set of operations aimed at creating interfaces and mechanisms for the **flow** and **access of data**.

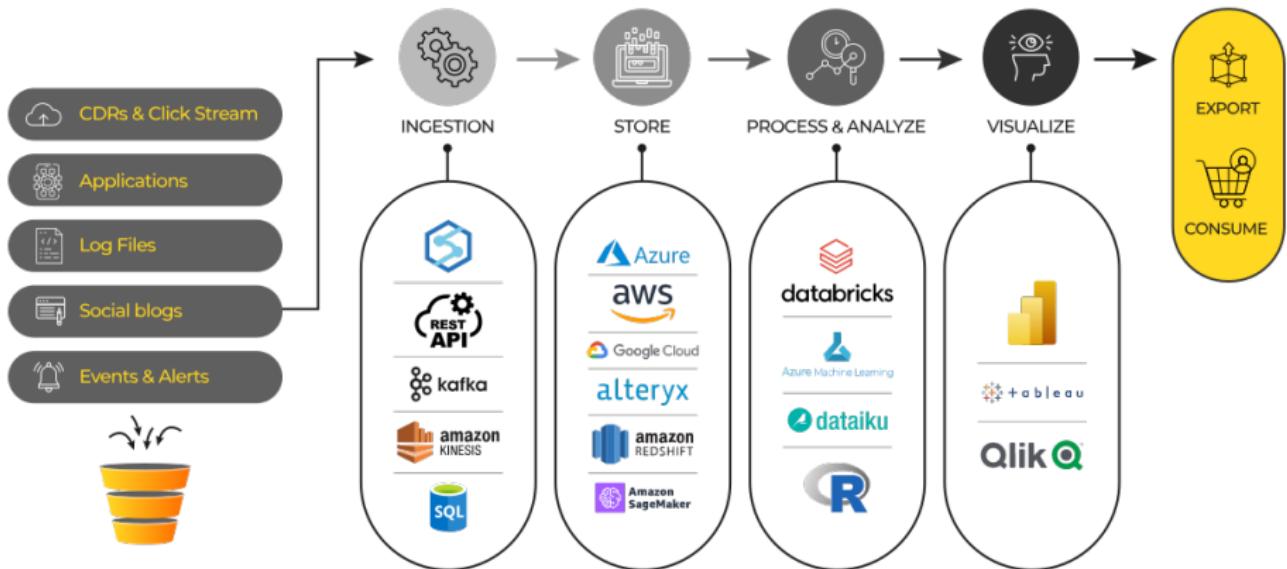


# Why is important Data Engineering?

- **Data Engineering** is the foundation of the **high-quality data** that is necessary for **effective data science**.
- **Data Engineering** is the process of **collecting, transforming, and storing data** in a way that's accessible and easy to analyze.



# Data Engineering Architecture



# Case of Study: Dashboards



# Data Science

- **Data Science** is the process of extracting knowledge from data.
- Data Science is the process of analyzing and interpreting complex digital data.
- Data Science is the process of creating models that can predict future outcomes.
- Data Science is the process of creating visualizations to help understand data.



# Data Science

- **Data Science** is the process of extracting knowledge from data.
- **Data Science** is the process of analyzing and interpreting complex digital data.
- **Data Science** is the process of creating models that can predict future outcomes.
- **Data Science** is the process of creating visualizations to help understand data.



# Data Science

- **Data Science** is the process of extracting knowledge from data.
- **Data Science** is the process of analyzing and interpreting complex digital data.
- **Data Science** is the process of creating models that can predict future outcomes.
- **Data Science** is the process of creating visualizations to help understand data.



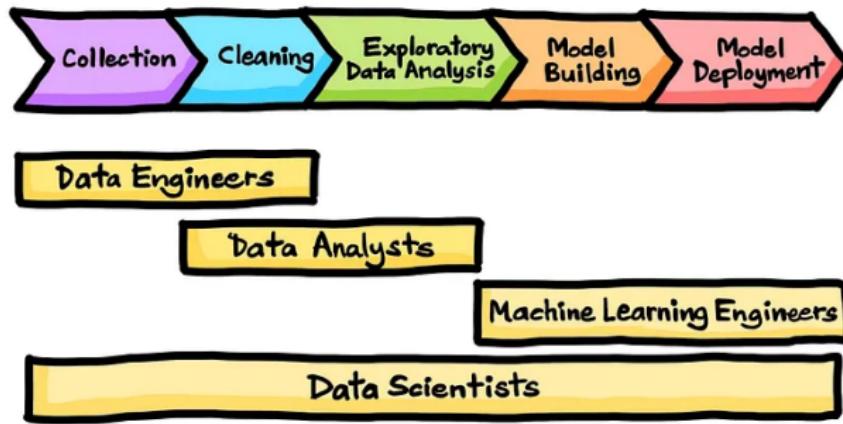
# Data Science

- **Data Science** is the process of extracting knowledge from data.
- **Data Science** is the process of analyzing and interpreting complex digital data.
- **Data Science** is the process of creating models that can predict future outcomes.
- **Data Science** is the process of creating visualizations to help understand data.



# Data Science Workflow

## THE DATA SCIENCE PROCESS



# DBOps vs Data Engineer

- **DBOps** is responsible for the **operation of the database**.
- **DBOps** is responsible for the **performance of the database**.
- **DBOps** is responsible for the **security of the database**.
- **Data Engineer** is responsible for the **data architecture**.
- **Data Engineer** is responsible for the **data quality**.
- **Data Engineer** is responsible for the **data flow**.



# DBOps vs Data Engineer

- **DBOps** is responsible for the **operation of the database**.
- **DBOps** is responsible for the **performance of the database**.
- **DBOps** is responsible for the **security of the database**.
- **Data Engineer** is responsible for the **data architecture**.
- **Data Engineer** is responsible for the **data quality**.
- **Data Engineer** is responsible for the **data flow**.



# How to improve data quality?

- **Data Quality** is the process of ensuring that **data** is **accurate**, **complete**, and **reliable**.
- **Data Quality** is the process of ensuring that **data** is **consistent** and **up-to-date**.
- **Data Quality** is the process of ensuring that **data** is **free from errors** and **inconsistencies**.
- **Data Quality** is the process of ensuring that **data** is of **high quality** and can be **trusted**.
- **Data Quality** is the process of ensuring that **data** is **fit for purpose** and can be **used effectively**.



# Outline

- 1 Software Components and Applications
- 2 Glosary
- 3 DataBase Classification
- 4 Relational Database Design
- 5 Entity-Relation Model (MER)
- 6 DataBase Management Systems — DBMS
- 7 DataBases Infrastructure
- 8 DevOps
- 9 Data Engineering



# Thanks!

# Questions?



Repo: <https://github.com/EngAndres/ud-public/tree/main/courses/databases-ii>

