

INTRODUCTION TO DATABASES

Database Foundations

Author: Eng. Carlos Andrés Sierra, M.Sc.
cavirguezs@udistrital.edu.co

Full-time Adjunct Professor
Computer Engineering Program
School of Engineering
Universidad Distrital Francisco José de Caldas

2025-III



Outline

1 Software Components and Applications

2 DataBase Classification

3 Relational Database Design

4 DataBase Management Systems — DBMS

5 Data Engineering



Outline

MongoDB

1 Software Components and Applications

2 "Relational Database Classification"

3 "New Services"

4 "Old Services" : DBMS

5 Data Engineering



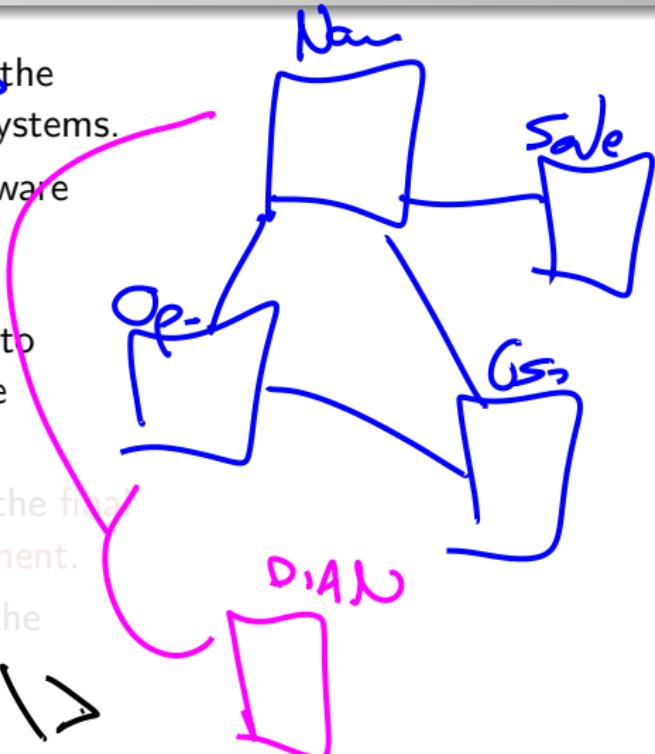
ECO \$ DIAZ
FOLKLORE



Modular Software Components

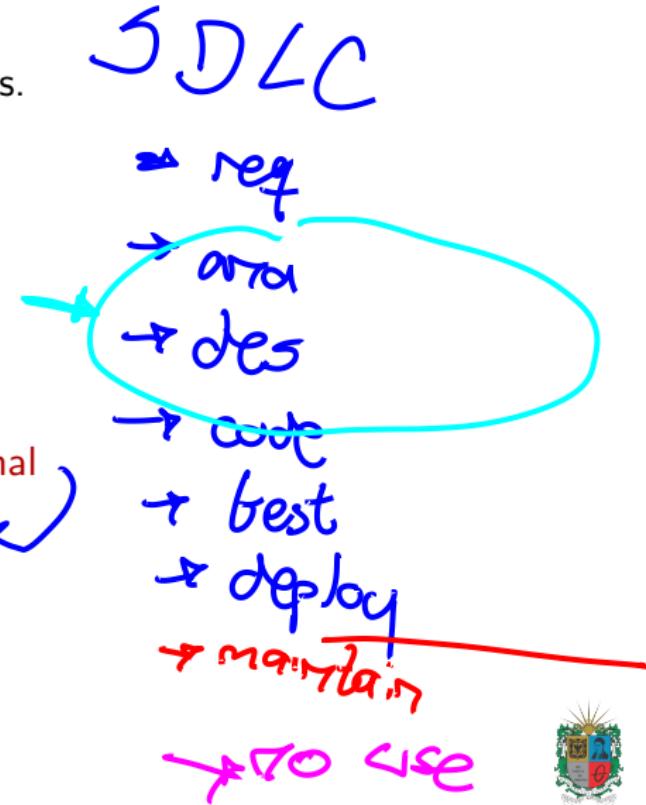
- **Software Components** are the building blocks of software systems.
- **Modular Software** is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable modules.
- **<html>** are the final product of software development.

~
{php log, q} >
<p> ~ </p>



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.

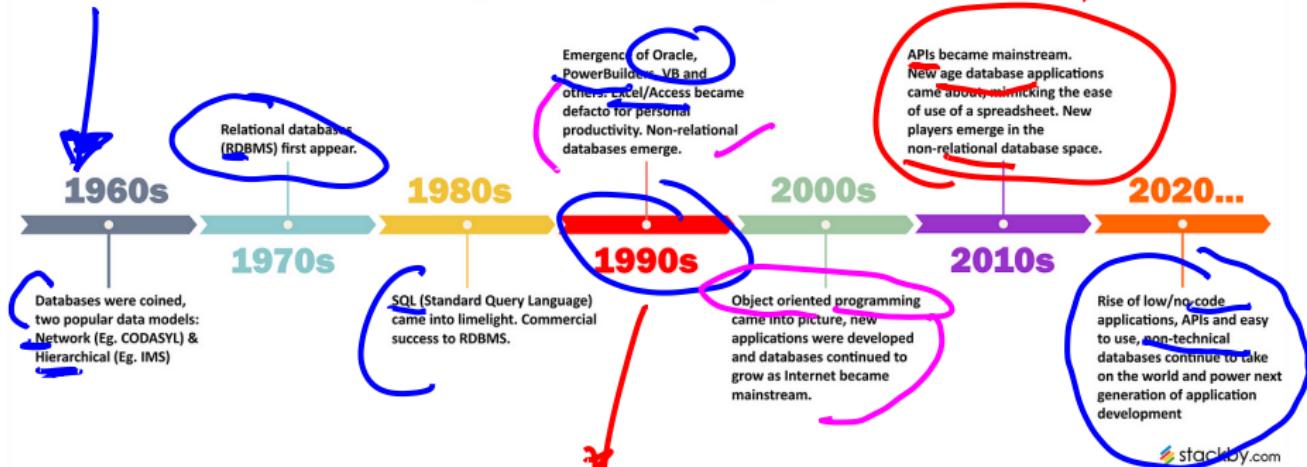


History of DataBases

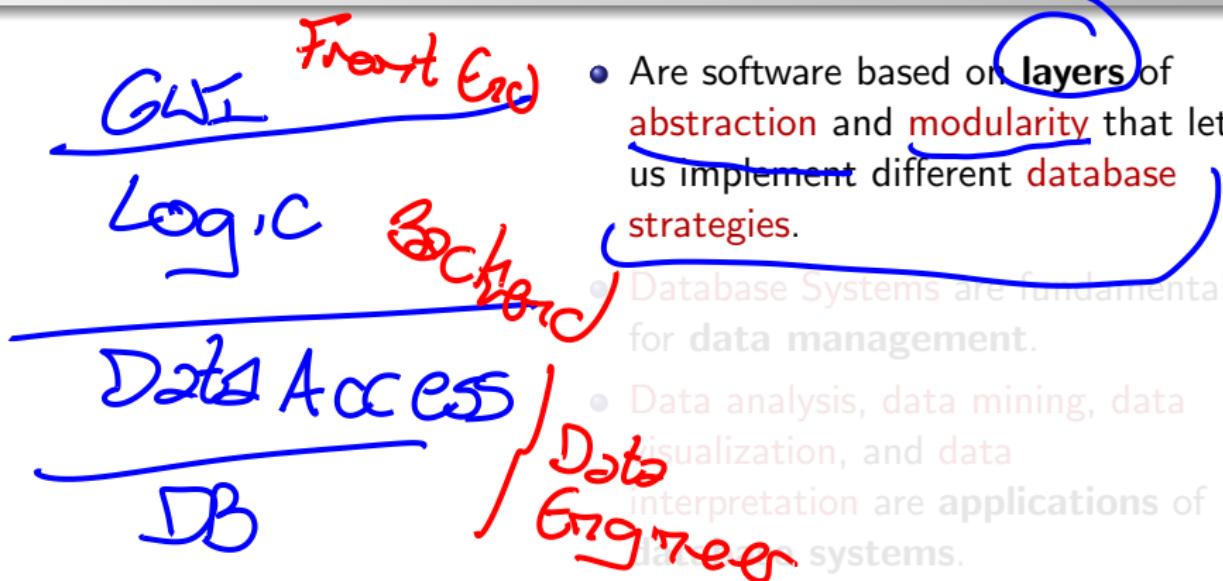
*U.b.ing
Code*

Speed

History of Databases (1960-2020)



Applications



Applications

Store - Retrieve

DB

DB

1

DB

2

DB

3

- Are software based on **layers** of abstraction and modularity that let us implement different database strategies.

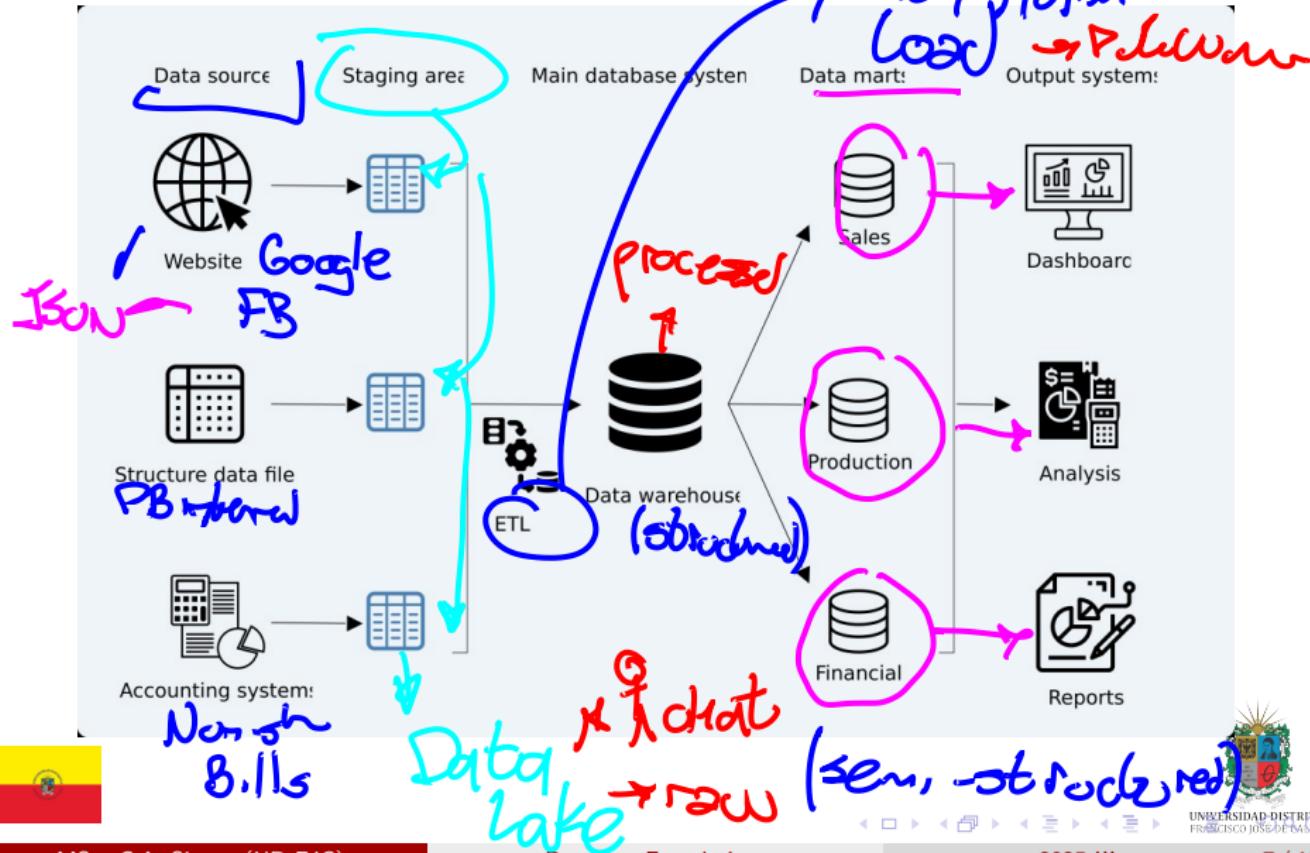
• Database Systems are fundamental for **data management**.

- Data analysis, data mining, data visualization, and data interpretation are **applications** of **database systems**.

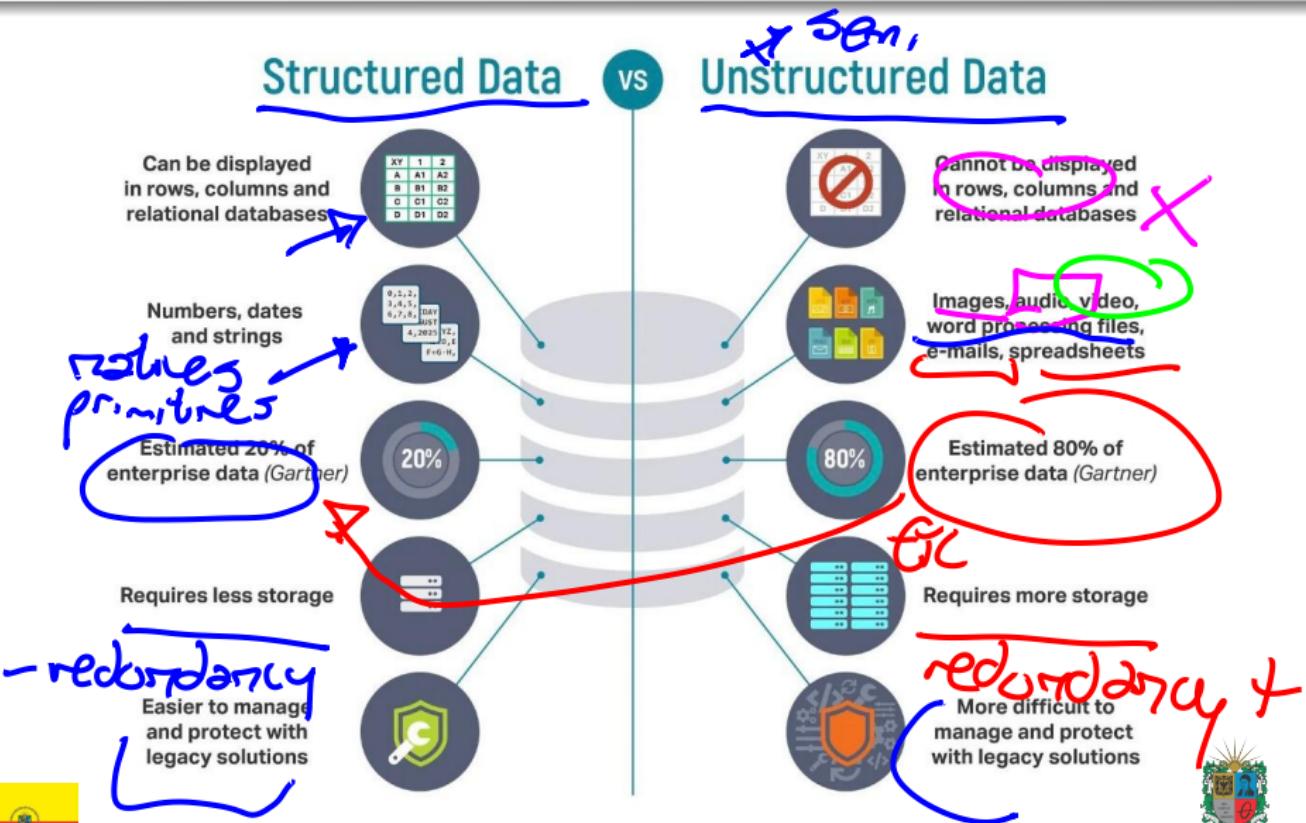


Case Study: Database System

Extract → Data
Transform
Load → Database



Structured and Unstructured Data



Tabular Structures

- **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 Key is a unique identifier for a record in a data set.
- Foreign Key

a primary key in another table.

age	name	weight
25	mp6	169

column

25

Rep-60

169

relation



Tabular Structures

87.5%



- **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 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**.

- redundancy

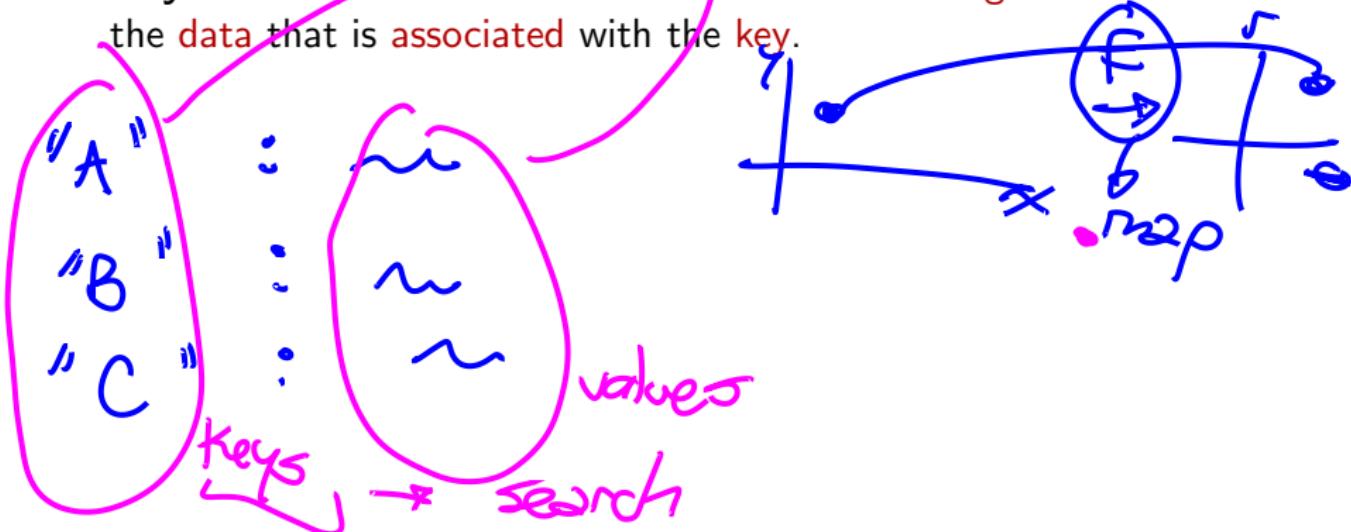
\rightarrow speed \Rightarrow index - Pk



Key-Value Data Structures

Python → Dictionary
 JS → Object

- **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.

one of these → exposure



Outline

1 Software Components and Applications

2 DataBase Classification

3 Relational Database Design

4 DataBase Management Systems — DBMS

5 Data Engineering



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.
near



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 relationships used in relational databases.

Analytics → *Reports*
↳ *Time Series*



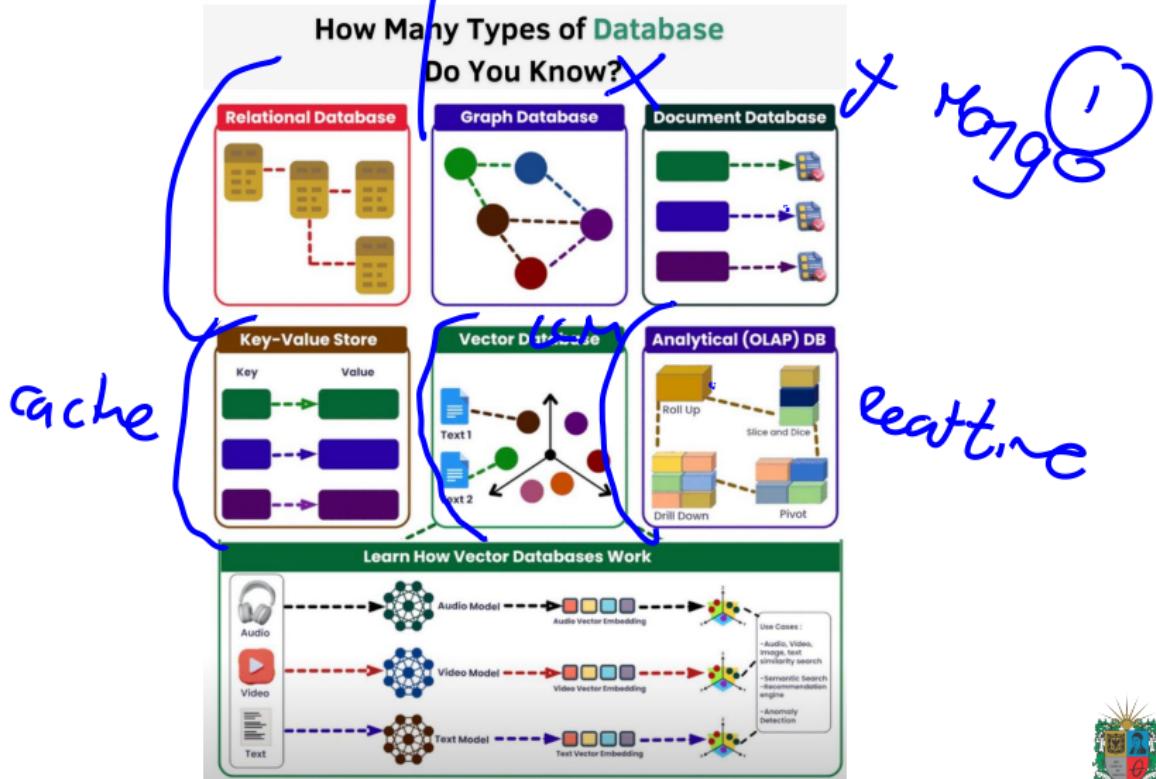
DataBase Classification

- 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.

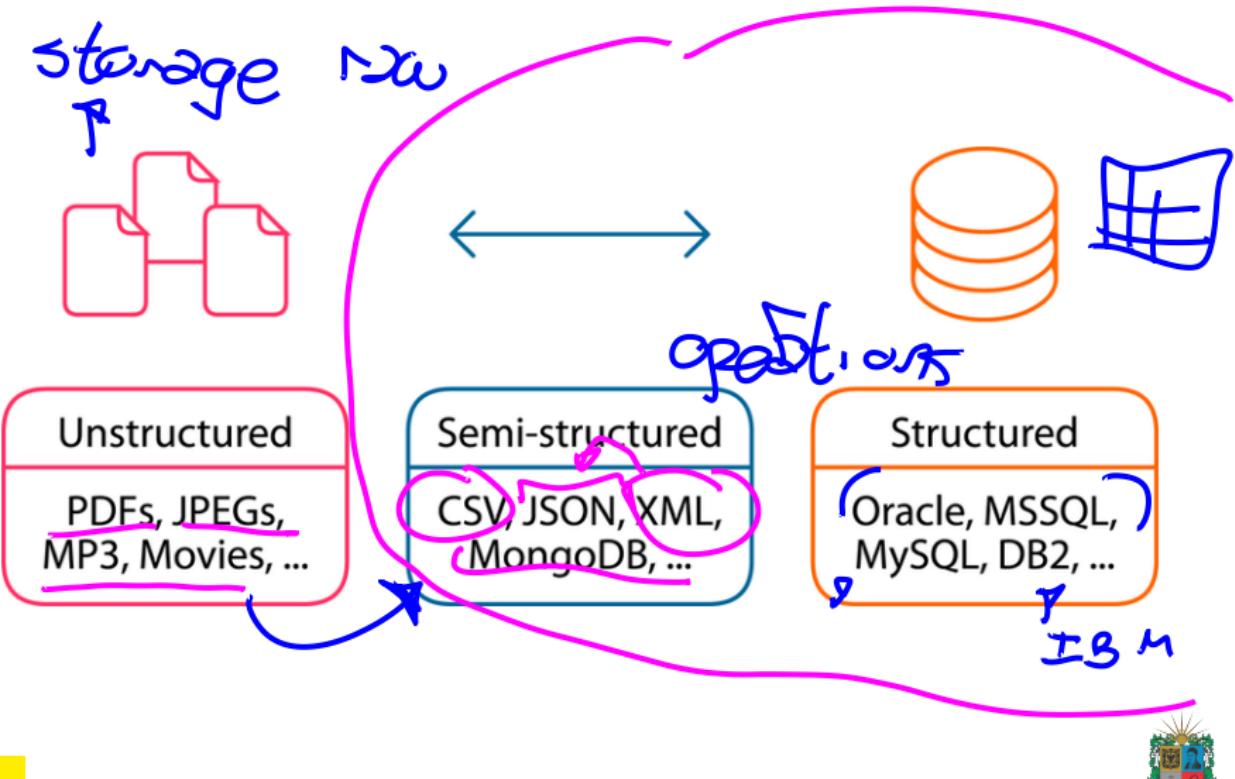
Mongo DB Function
 SQL



Types of Database



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**.

client-side



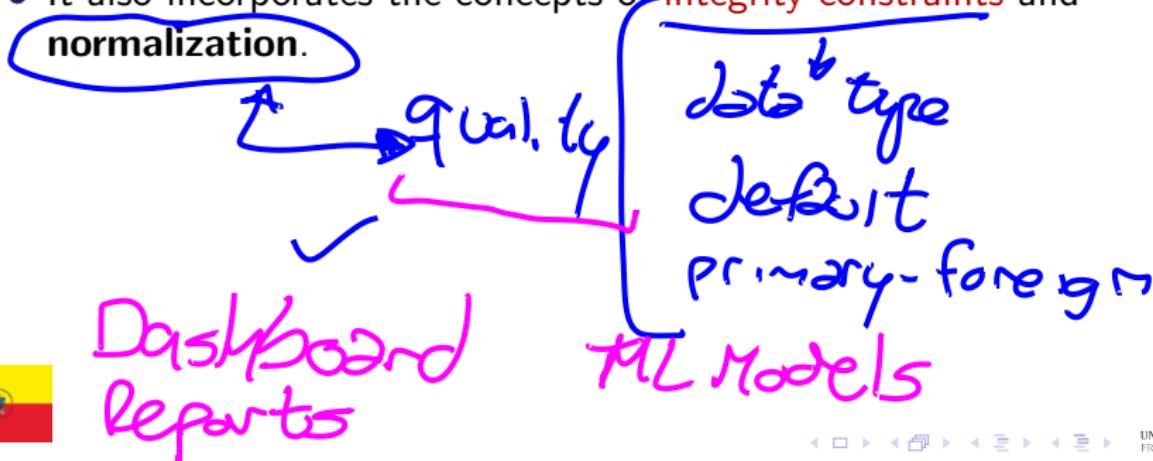
Relational Model

- 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**.



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**.

graph

```

<list>
  <student>
    <name>Repita</name>
    <code>123</code>
  </student>
  <student>
    <name>Pepito</name>
  </student>
</list>
  
```

DOM



Document-Based Model

list of
JSON



Instance
(primary)

MongoDB

list

Database(s)

Collection(s)

Document(s)

```
{
  _id:1,
  key:"Lime",
  qty: 50
}
```

Field(s)

schema 2
• mandatory
keys

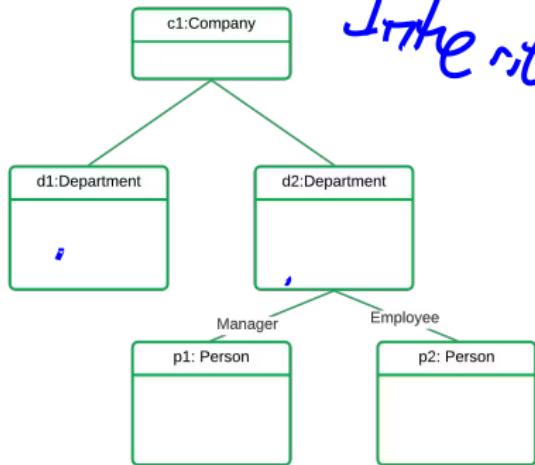
- id
key
- id
qty

Key: value (tree)

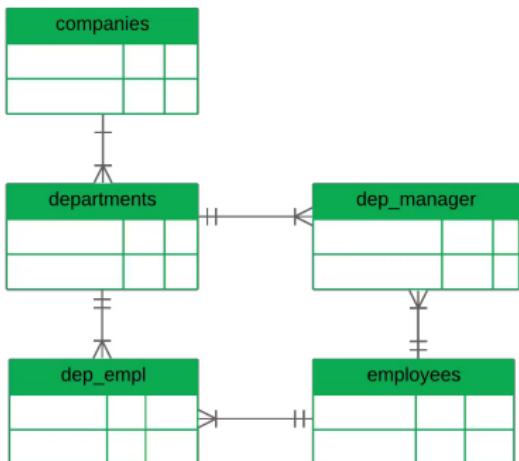


Object-Oriented Model

Object-Oriented



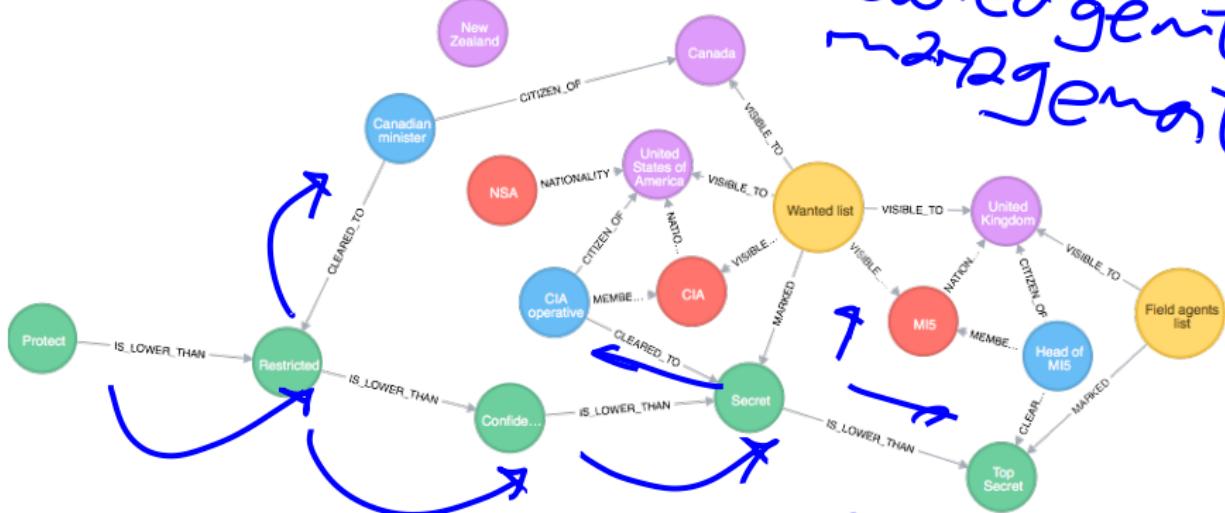
Relational



methods → store procedures



Graph-Based Model

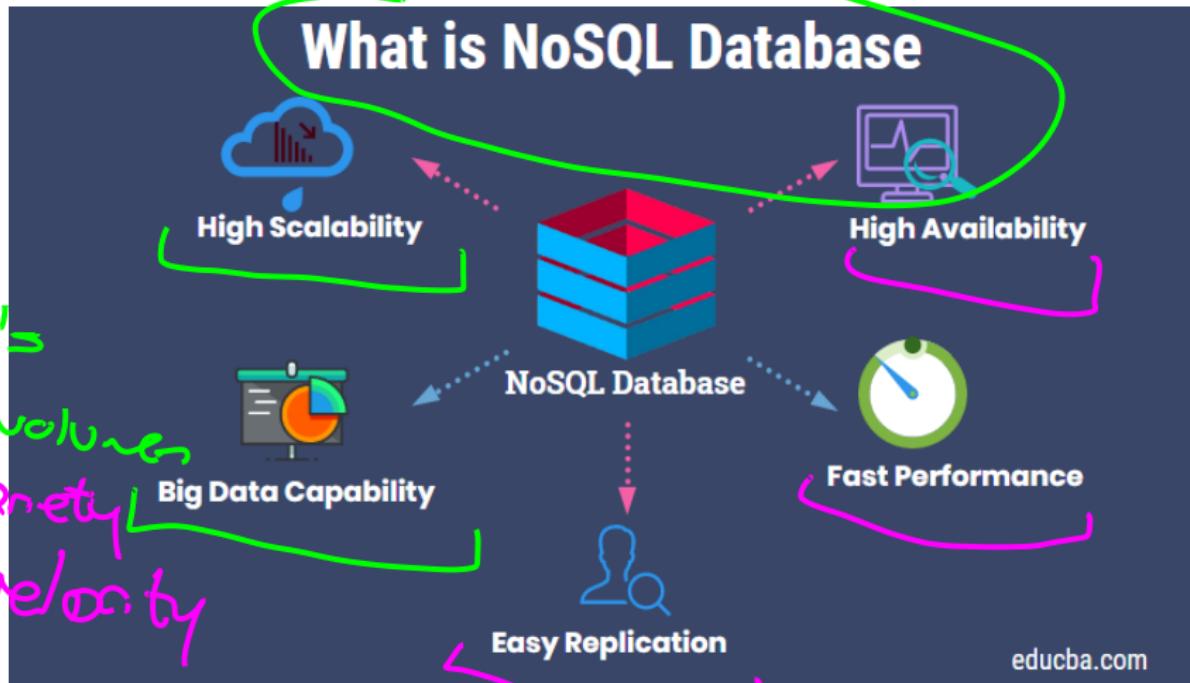


Knowledge management
IS_LOWER_THAN
CLEARED_TO

Social networks



NoSQL Model



Outline

- 1 Software Components and Applications
- 2 DataBase Classification
- 3 Relational Database Design
- 4 DataBase Management Systems — DBMS
- 5 Data Engineering



Set Theory in Databases

- The **set theory** is a branch of **mathematical logic** that studies sets, which are **collections of objects**. *Rda*
- 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**.

A

P



Set Theory in Databases

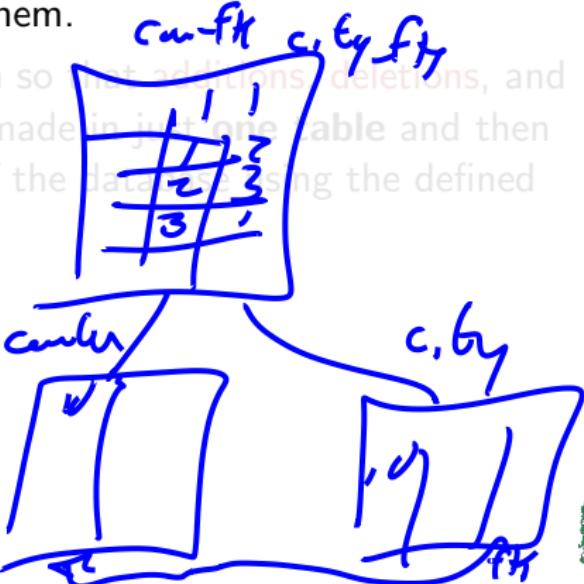
- The **set theory** is a branch of **mathematical logic** that studies sets, which are **collections of objects**.
- The **set theory** is applied in **databases** to define the **relational model** and the **relational algebra**.
- The **relational model** is a **mathematical model** of data for large 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**.

sentence → instruction



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 organize the data so that additions, deletions, and modifications of a field can be made in just one table and then propagate 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



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.
list of objects
- **Degree:** The number of entities that participate in a relationship.
list of rows



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.



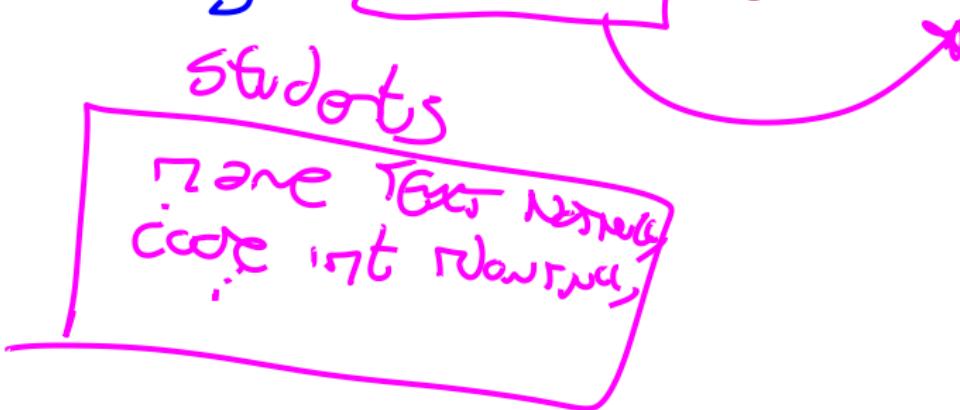
Definitions

- **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. $1-1$, $1-n$, $n-n$
- **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**.



Outline

- 1 Software Components and Applications
- 2 DataBase Classification
- 3 Relational Database Design
- 4 DataBase Management Systems — DBMS
- 5 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.



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.



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: Cost: DBMS can be expensive for large 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 can be **expensive** for **large data volumes**.
- **Performance:** DBMS can be **slow**.



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.



Case Study: DBeaver

DBeaver 22.1.3 - rides

The screenshot shows the DBeaver interface with the following details:

- Database Navigator:** Shows the database structure. Under the 'rides' schema, there are tables: promo_codes, rides, user_promo_codes, users, vehicle_location_histories, and vehicles.
- Properties Panel:** Displays properties for the 'rides' table.

Table Name:	rides	Object ID:	111
Tablespace:		Owner:	root
Extra Options:			

 Other tabs in the Properties panel include Data, ER Diagram, system, Databases, move, Schemas, public, Tables, and rides.
- Project - General Panel:** Shows a list of items:

Name	DataSource
Bookmarks	
ER Diagrams	
Scripts	

 It also shows the number of items (10 items) and various toolbar buttons like Save, Revert, Refresh, Undo, and Redo.



Command Line

- 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 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**.



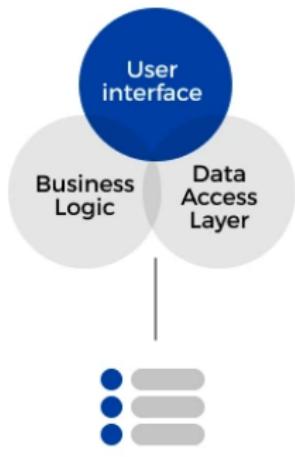
Microservices Architecture

- **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.

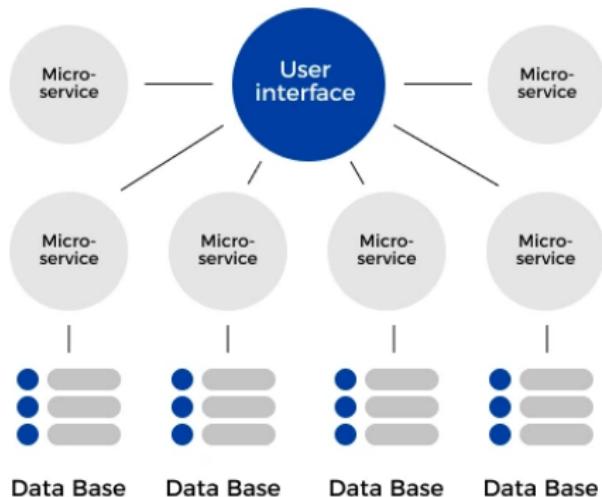


Monolithic Architecture Schema

MONOLITHIC ARCHITECTURE



MICROSERVICE ARCHITECTURE



Outline

- 1 Software Components and Applications
- 2 DataBase Classification
- 3 Relational Database Design
- 4 DataBase Management Systems — DBMS
- 5 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**.



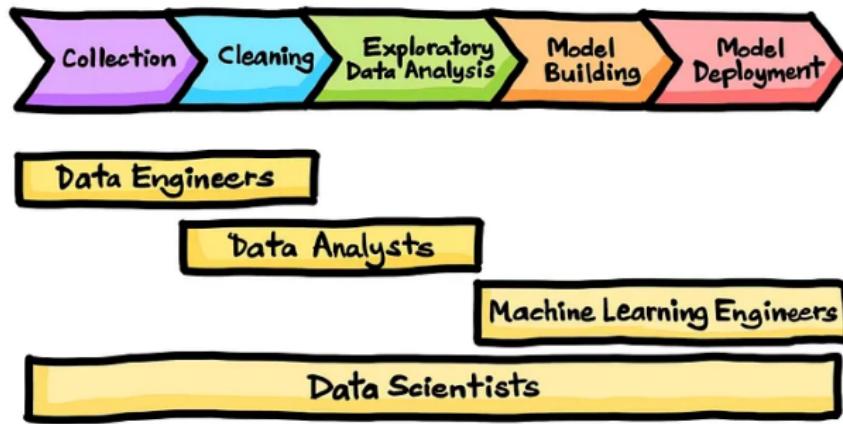
Why is Data Engineering Important?

- **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.

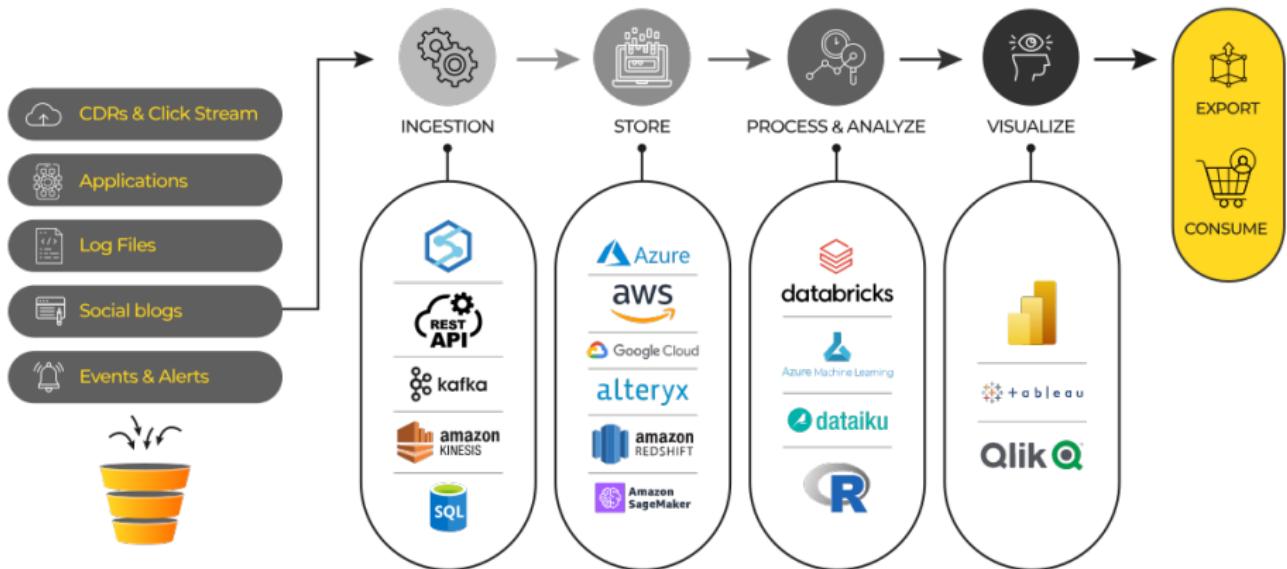


Data Science Workflow

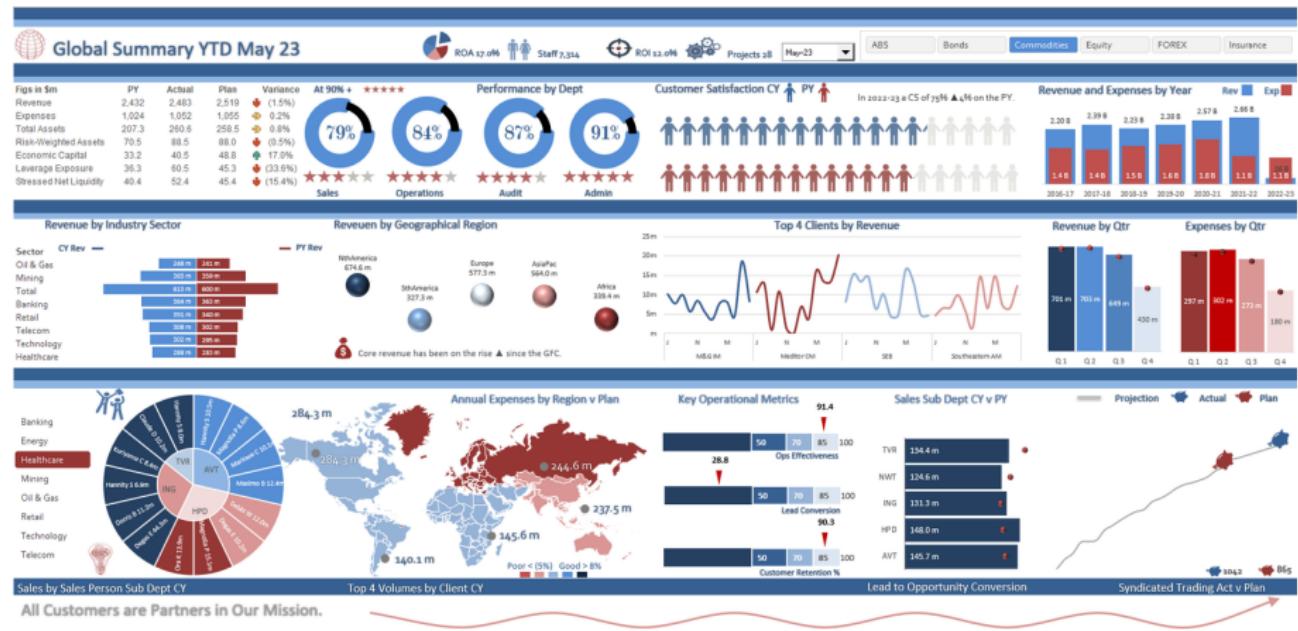
THE DATA SCIENCE PROCESS



Data Engineering Architecture



Case Study: Dashboards



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 DataBase Classification
- 3 Relational Database Design
- 4 DataBase Management Systems — DBMS
- 5 Data Engineering



Thanks!

Questions?



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

