

# DATABASES DESIGN & RELATIONAL ALGEBRA

## DataBase Foundations

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

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

2024-III



# Outline

- 1 Basic Concepts
- 2 Normalization
- 3 Relational Algebra



# Outline

- 1 Basic Concepts
- 2 Normalization
- 3 Relational Algebra



# DataBases Design Foundations

- In the context of **databases**, the **design** of a database is the process of producing a **detailed data model** of a database.
- This **data model** contains all the needed **logical and physical design choices** and **physical storage parameters** needed to generate a design in a *data definition language*, which can then be used to create a 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**.



# Outline

- 1 Basic Concepts
- 2 Normalization**
- 3 Relational Algebra



# 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.



# Ontologies

- 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.



# Normal Forms

- ❶ **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.
- ❷ **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**.
- ❸ **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.
- ❹ **Fourth normal form (4NF):** The table is in third normal form and there are **no multi-valued dependencies**.



# Outline

- 1 Basic Concepts
- 2 Normalization
- 3 Relational Algebra



# What is relational algebra?

- The **relational algebra** is a **procedural query language**, which takes **relations** as **input** and produces relations as **output**.
- The **relational algebra** is a **set of operations** that can be performed on a **relation**. Also, it is used to define the **relational model**, which is a **mathematical model** of data for large shared data banks.
- Let's take a look at the **basic operations** of the **relational algebra**. First, remember next table called **Students**:

ID	Name	Lastname	Address	Phone	Age
1	John	Doe	123 Fake St	555-1234	25
2	Jane	Smith	456 Elm St	555-5678	30
3	Mike	Johnson	789 Evergreen St	555-9012	35



# Select Operation

## Definition

**Select:**  $\sigma_{\text{condition}}(R)$ , is a unary operation that returns the rows (subset) of  $R$  that satisfy the condition.

For example, the following expression selects the students whose age is greater than 25:

$$\sigma_{\text{Age} > 25}(\text{Students})$$



# Project Operation

## Definition

**Project:**  $\pi_{\text{column\_list}}(R)$ , is a unary operation that returns the columns (subset) of  $R$  that are specified in the column list.

For example, the following expression projects the name and lastname of the students:

$$\pi_{\text{Name, Lastname}}(\text{Students})$$



# Union Operation

## Definition

**Union:**  $R \cup S$ , is a binary operation that returns the rows that are in  $R$  or in  $S$ .

For example, the following expression returns the students whose age is greater than 25 or whose lastname is Johnson:

$$\sigma_{\text{Age} > 25}(\text{Students}) \cup \sigma_{\text{Lastname} = \text{Johnson}}(\text{Students})$$





# Set Different Operation

## Definition

**Set Different:**  $R - S$ , is a binary operation that returns the rows that are in  $R$  but not in  $S$ .

For example, the following expression returns the students whose age is greater than 25 but not whose lastname is Johnson:

$$\sigma_{\text{Age} > 25}(\text{Students}) - \sigma_{\text{Lastname} = \text{Johnson}}(\text{Students})$$



# Cartesian Product Operation

## Definition

**Cartesian Product:**  $R \times S$ , is a binary operation that returns the Cartesian product of  $R$  and  $S$ . A formal definition is:

$$R \times S = \{r \cup s \mid r \in R \wedge s \in S\}$$

For example, the following expression returns the Cartesian product of the students and the courses:

Students  $\times$  Courses



# Rename Operation

## Definition

**Rename:**  $\rho_{\text{new\_name}}(R)$ , is a unary operation that returns the relation  $R$  with the name  $R$  changed to  $\text{new\_name}$ .

For example, the following expression returns the students relation with the name changed to **People**:

$$\rho_{\text{People}}(\text{Students})$$



# Exercises

- 1 Select the **students** whose **age** is **greater** than 25 and whose **lastname** is **Johnson**.
- 2 Project the **name** and **lastname** of the **students** whose **age** is **greater** than 25.
- 3 Select the **students** whose **age** is **greater** than 25 and whose **lastname** is **Johnson**, and **project** the **name** and **lastname** of the **students**, and **rename** the relation to **People**.



# Exercises

- 1 Select the **students** whose **age** is **greater** than 25 and whose **lastname** is **Johnson**.
- 2 Project the **name** and **lastname** of the **students** whose **age** is **greater** than 25.
- 3 Select the **students** whose **age** is **greater** than 25 and whose **lastname** is **Johnson**, and project the **name** and **lastname** of the **students**, and rename the relation to **People**.



# Exercises

- 1 Select the **students** whose **age** is **greater** than 25 and whose **lastname** is **Johnson**.
- 2 Project the **name** and **lastname** of the **students** whose **age** is **greater** than 25.
- 3 Select the **students** whose **age** is **greater** than 25 and whose **lastname** is **Johnson**, and **project** the **name** and **lastname** of the **students**, and **rename** the relation to **People**.



# Outline

- 1 Basic Concepts
- 2 Normalization
- 3 Relational Algebra



# Thanks!

## Questions?



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

