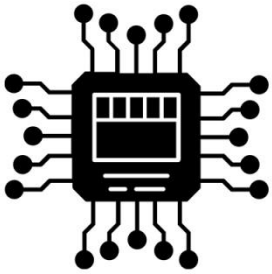


# قسم علوم الذكاء الاصطناعي
# Department of Artificial Intelligence

## Subject:

# Database

## Class:

**second**

## Lecturer:

M.Sc. Ali Haider Alazam

Noor Hassan Aldulimi

# Laboratory : (1- 2)

## Introduction to Database

## 1. Foundational Concepts

**Data** is raw. It consists of unorganized facts, figures, and signals that have no context on their own. In a computer system, "data" is simply the bits and bytes (0s and 1s) stored on a hard drive.

- *Example:* 105, Red, True.

**Information** is data that has been processed, structured, or organized to provide meaning. It answers the questions "who, what, where, and when".

- *Example:* "Student ID **105** is wearing a **Red** shirt".

**Metadata** is "data about data." It describes the structure, constraints, format, and characteristics of the actual data, rather than the content itself.

## 2. The Practical Environment

To perform the practical labs in this course, you need to understand the infrastructure where the database lives. A database does not float in the cloud; it sits on a server with specific network configurations.

### A. Network & Connectivity

The **Network** is the bridge between you (the Client) and the Database (the Server).

- **Client-Server Architecture:**
    - **The Client:** This is your workstation or the application (like a website or the SQL Management Studio tool). It sends requests.
    - **The Server:** This is the powerful computer running the Database Management System (DBMS). It processes the request and sends back data.

### B. IP Addressing & Ports

To connect your client to the server, you need an address.

- **Localhost (127.0.0.1):** This is a special "loopback" address that tells your computer to talk to itself. You will use this when your database is installed directly on your own laptop.
- **Remote IP (e.g., 192.168.1.50):** This is used when the database is on a different machine.
- **Static IP:** A database server should **always** have a "Static IP" (one that doesn't change). If the server used a dynamic IP (DHCP) and the address changed after a reboot, every application trying to connect to it would fail.

**C. System Resources (Virtual Memory)**

Databases are **memory-hungry**. They try to load as much data as possible into the fast RAM to answer queries quickly.

- **The Risk:** If the server runs out of physical RAM, the Operating System will start using **Virtual Memory** (fake RAM stored on the slow hard drive).
- **The Result:** The database performance will crash (high latency), potentially causing a Denial of Service (DoS) state where no one can access data.

---

## 3. The Database System (RDBMS)

Now that we have the environment, let's look at the software.

**A. DBMS vs. RDBMS**

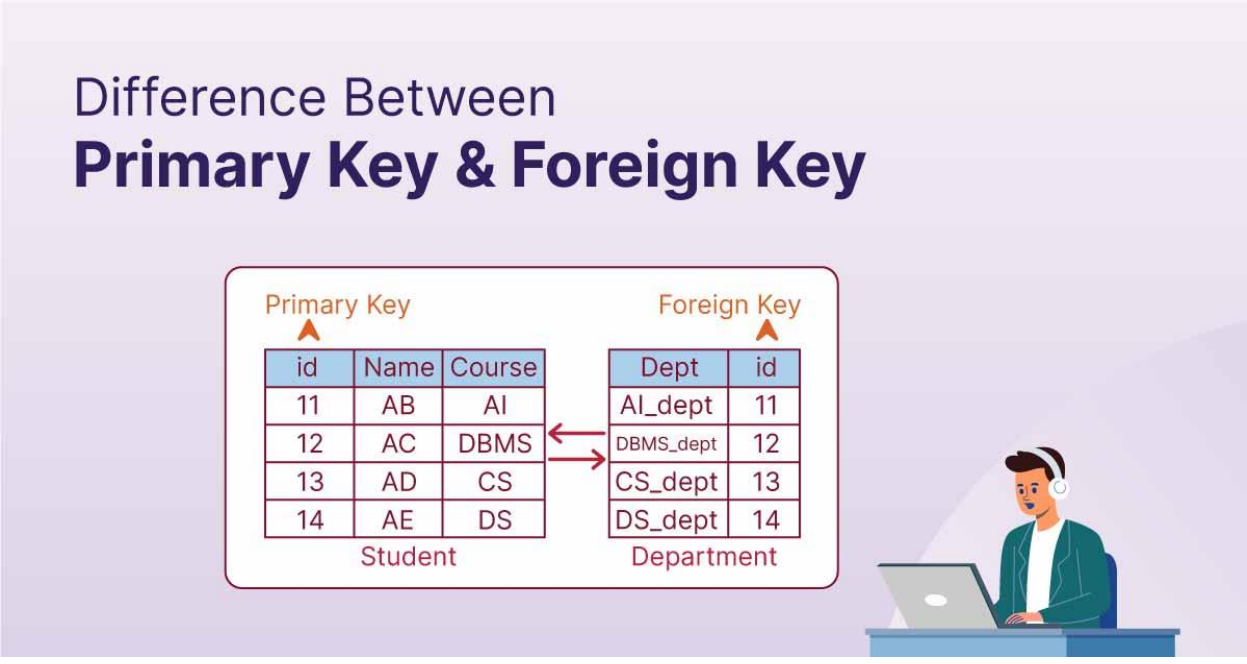
A **Database Management System (DBMS)** is the general software used to **store**, **retrieve**, and **manage** data. However, in modern enterprise and security contexts, we almost exclusively use a **Relational Database Management System (RDBMS)**.

- **RDBMS Definition:** Software that stores data in a structured format using **Rows** and **Columns**, similar to a spreadsheet.
- **The "Relation":** The true power of an RDBMS is that it splits data into many small, efficient tables that are **related** (linked) to each other.

**B. The Relational Model: Keys**

How do we link these separate tables together? We use **Keys**.

1. **Primary Key (PK):**
    - This is a column that uniquely identifies **every single row** in a table. It acts like a digital fingerprint.
    - *Rule:* It cannot be NULL and must be unique.
    - *Security Note:* If Primary Keys are predictable (e.g., User 1, User 2, User 3), attackers can use **IDOR (Insecure Direct Object Reference)** attacks to simply guess the ID of an admin account.
2. **Foreign Key (FK):**
    - This is a column that points to the Primary Key of *another* table. This creates the link.
    - *Example:* Your UserID is a Primary Key in the **Users** table, but it appears as a Foreign Key in the **Orders** table to link a purchase to you.



## 4. Real World Applications (SQL in Action)

To understand why this matters, let's look at how the apps you use every day rely on SQL and Databases.

### Example 1: Instagram (Logging In)

- **The Scenario:** You open Instagram and type in your username (cool_cat_99) and password (secret123).
- **What SQL Does:** The app asks the database: "Do we have a user with this name AND this password?" .
- **The Code:**

```
SQL

SELECT * FROM Users
WHERE username = 'cool_cat_99'
AND password = 'secret123';
```

*Result:* If the database finds a match, you get in. If it returns nothing, you get "Incorrect Password".

### Example 2: Amazon (Filtering Products)

- **The Scenario:** You are shopping for **Nike** shoes but have a budget of **$100**.
- **What SQL Does:** Amazon's database has millions of items. SQL filters out the expensive boots and cheap socks so you only see what you asked for.
- **The Code:**

```
SQL

SELECT product_name, price, image
FROM Products
WHERE brand = 'Nike'
AND price < 100;
```

### Example 3: Banking App (Checking Balance)

- **The Scenario:** You check your balance after the weekend.

- **What SQL Does:** The bank doesn't just store one number for your balance. It calculates it fresh every time by adding up all your deposits and subtracting your withdrawals .
- **The Code:**

```
SQL
```

```sql
SELECT SUM(amount)
FROM Transactions
WHERE account_id = '12345';
```

## 5. Data Safety & Integrity

### A. The ACID Properties

For a database to be considered "safe" (especially for banking or healthcare), it must guarantee four properties, known as **ACID**:

1. **A - Atomicity ("All or Nothing"):**
    - *Concept:* A transaction must fully happen or not happen at all.
    - *Example:* If a bank transfer fails halfway, the database **rolls back** to the start so no money is lost. No partial data is saved.
2. **C - Consistency:** The database ensures data follows all rules (e.g., you cannot save a text string into a field meant for dates).
3. **I - Isolation:** Multiple users can access the database at the same time without corrupting each other's data.
4. **D - Durability:** Once the database says "Saved," that data is written to the physical disk and will survive a total power failure.

### B. The Danger of NULL

We need to address a specific concept that confuses many beginners: **NULL**.

- **What it is:** NULL represents **missing**, **unknown**, or **undefined** data.
- **What it is NOT:** It is *not* the number 0, and it is *not* an empty text string (""). 0 is a value; NULL is the *absence* of a value.

- **Security Risk:** Attackers often try to force NULL values into login forms. If the programmer didn't plan for NULL, the application might crash or, worse, default to "True" and let the attacker in.

## 6. Introduction to SQL

**SQL (Structured Query Language)** is the standard language we use to talk to the RDBMS. It is a "domain-specific" language, meaning it is designed for one job only: managing data.

A. The Four Subsets of SQL

1. **DQL (Data Query Language):** Used to ask questions and retrieve data. (Main Command: SELECT) .
2. **DML (Data Manipulation Language):** Used to change data. (Commands: INSERT, UPDATE, DELETE) .
3. **DDL (Data Definition Language):** Used to define the structure, like creating tables. (Commands: CREATE, DROP) .
4. **DCL (Data Control Language):** Used to handle permissions and access control. (Commands: GRANT, REVOKE) . *This is the most critical subset for security policy.*

**B. Basic Syntax & Comments**

SQL is designed to read like English. A standard query looks like this:

```
SQL
```

```
SELECT username, email      -- Columns you want to see
FROM users                  -- The table where they live
WHERE role = 'admin';       -- The filter condition
```

**Practical Tip: Comments**

Just like in Python or C++, you can leave notes in your code that the computer ignores using two dashes --.

- *Why this matters:* This is the primary tool used in **SQL Injection**. Attackers add -- to the end of their malicious input to "comment out" the password check in your code, effectively tricking the database into ignoring the security rules.