**Files, Records, and Fields:**

* **File**: A collection of related records. Example: A file containing all student records.
* **Record**: A collection of related fields. Example: A single student’s record with fields like **Name, Roll Number, and Grade.**
* **Field**: A single piece of data. Example: The **Name** field in a student record.

**Data Independence:**

Data independence means that the way data is stored (physical level) or organized (logical level) does not affect how users interact with it. There are two types:

* **Logical Data Independence**: Changes in the logical structure (e.g., adding a new table) do not affect the user’s view.
  + Example: Adding a new column **Email** to the **Students** table does not break existing applications.
* **Physical Data Independence**: Changes in the physical storage (e.g., moving data to a new disk) do not affect the logical structure.
  + Example: Changing the file format of the database does not require rewriting queries.

**Three-Level Architecture (ANSI-SPARC):**

This architecture divides the database system into three levels to achieve data abstraction and independence.

**a) External Level (View Level):**

* Represents how users see the data.
* Example: A student sees their grades, while a teacher sees all students’ grades.

**b) Conceptual Level (Logical Level):**

* Represents the overall structure of the database.
* Represent what data is stored in the database
* Example: A unified view of all students, courses, and grades.

**c) Internal Level (Physical Level):**

* Represents how data is stored physically.
* Example: Data is stored in files on a disk, with indexes for faster access.

**Objectives of Three-Level Architecture:**

* **All users access the same data**: A student and a teacher see consistent data.
* **User views are immune to changes**: Adding a new field does not break existing views.
* **Users don’t need to know physical storage details**: Users don’t care if data is stored on a hard drive or in the cloud.

**Schemas and Instances:**

* **Schema**: The blueprint or structure of the database.
  + Example: A **Students** table with columns **StudentID**, **Name**, and **Age**.
* **Instance**: The actual data in the database at a specific time.
  + Example: The **Students** table contains rows like **(1, "Alice", 20)** and **(2, "Bob", 22)**

**Database Schema:**

* A **schema** describes the structure of the database.
* Example: A schema diagram shows tables like **Students**, **Courses**, and their relationships.
* **Schema Constructs**: Objects in the schema, such as tables, columns, and relationships.

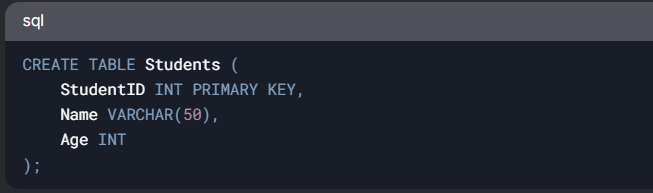
**Database State/Instance:**

* The **database state** is the data in the database at a particular moment.
* Example: At 10:00 AM, the **Students** table has 100 rows. At **11:00 AM**, after a new student is added, it has 101 rows.

**Database Languages:**

**a) Data Definition Language (DDL):**

* Used to define the structure of the database.
* Example: Create a table.



**b) Data Manipulation Language (DML):**

* Used to manipulate data in the database.
* ***Procedural DML****:* Specifies how to retrieve data.
  + Example: Write a loop to fetch all students.
* ***Non-Procedural DM*L**: Specifies what data to retrieve.
  + Example: SELECT \* FROM Students WHERE Age > 20.