class9|cs] DBMS and RDBMS

DBMS (Database Management System):

- What: A software system that allows you to store, manage, and retrive data.
- Why: Without DBMS, you would need to manage files manually, which is slow and messy.

How: provides an interface to insert, update, delete, and query data.

Example: Think of a notebook where you write all names, phone, and addresses. You can search but it takes time. That's like a basic DBMS.

RDBMS (Relational Database Management system):

What: A type of DBMS where data is stored in tables (row and columns) and related using keys.

Why: Makes data structured, consistent, and easier to query.

How: Uses SQL to defines relations between tables.

Example: MySQL, PostgreSQL, Oracle.

In a school system, you have:
→ student Table (studentID, Name, class)
→ Course Table (courseID, courseName)
→ Enrollments (studentID, courseID - link both)

This relation between table = RDBMS power.

Client - Server - Database:

→ Client: The user or application requesting data. (e.g. mobile app, web browsers)

→ Server: Process the request, runs queries, and sends back results.

→ Database: Where actual data is stored.

→ Real-life example:

- You (client) order food using Foodpanda
- Foodpanda app server connects to the

database to check restaurants, menu, and orders status.
- Database stores all restaurants, orders, customers info.

SQL vs NoSQL

1. SQL (structured Query Language):

- What: Language for managing relational (table-based) databases.
- Use Case: When data is structured (like banking, HR, e-commerce)
- Faster Case: SQL is faster when dealing with complex joining and structured data
- Example: Banking system where customer → account → transaction are strictly related.

2. NoSQL (Not only SQL):

- What: Non-relational database (document, key-value, graph, column).
- Use Case: When data is unstructured, scalable, flexible (like social media post, IOT)

- Faster case: NoSQL is faster when dealing with large unstructured data and real-time applications.
- Example: Facebook post, YouTube comments, chat messages.
- Real-life analogy:
  → SQL: like a structured Excel sheet with fixed row and columns.
  → NoSQL: like a diary where each page (document) can have different info.

Entity, Attributes, Keys, Rows, columns.

1. Entity: Real-world ~~project~~ object → stored as a table (like student)
2. Attributes: Properties of entity → stored as a columns.
   Example: studentID, Name, Email
3. Row (Tuples): Single record of entity.
   Example:
   Row = (101, "MAHFUZ", "example@gmail.com").

4. Columns (Fields): Vertical Set of attributes.

Primary key and Foreign key:

1. Primary key (PK):
   - Unique identifiers for each row in a table.
   - Example: studentID in student table.

2. Foreign key (FK):
   - Links two table. A field in one ~~table~~ table that refers to PK in another.

Example:

courseEnrollments.studentID = students.studentID

Real-life analogy:

PK = Your National ID Number (unique)

FK = Your Bank Account references your NID to link ownership.

SQL JOINS :

Join are used to combine data from multiple tables.

* Table Example :

* students Table :

student-ID	Name
1	Mahfuz
2	Shakeer
3	Rifat

* ~~Courses Table~~

~~studentID~~

* Courses Table :

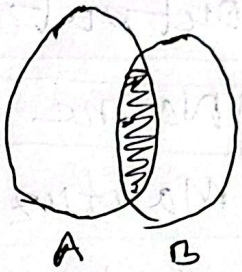
courseID.	studentID	courseName
101	1	DBMS
102	2	OOP
103	4	AI

* Inner join (common in both) :

```
Select students.Name, Courses.CourseName
From students
Inner join courses
ON students.studentID = Courses.studentID;
```

Output:

Name	courseName
Mahfuz	DBMS
Shakera	OOP

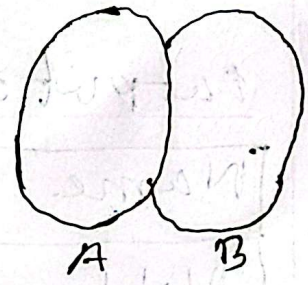


* Left Join (All from left + Matching right)

```
Select students.Name, courses.courseName
From students
Left join Courses
ON students.studentID = courses.studentID ;
```

Output:

Name	courseName
Mahfuz	DBMS
Shakera	OOP
Rifat	Null



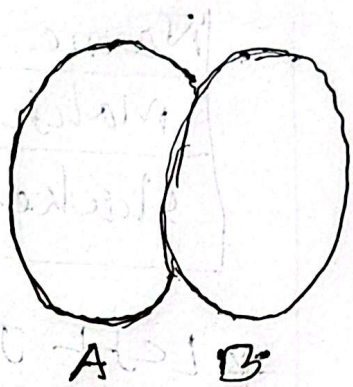
* Right Join (All from right + Matching left):

```
Select student.Name, Courses.courseName
From students
Right join Courses
ON students.studentID = Courses.studentID ;
```

↓

Output:

Name	courseName
Mahfuz	DBMS
Shakeer	OOP
Null.	AI

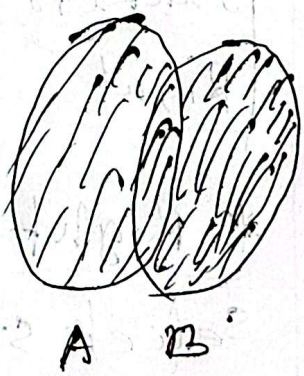


* Full Join (All records, matched or not).

```
Select student.Name, Courses.CourseName
From students
Full outer join Courses
ON students.studentID = courses.studentID;
```

Output:

Name	courseName
Mahfuz	DBMS
Shakeer	OOP
Rifat	Null
Null	AI



Summery :

- DBMS → Data management, avoid redundancy
- RDBMS → Relation between data, consistency
- SQL → Best for structured, complex queries.
- NoSQL → Best for big, unstructured, real-time data.
- Keys → Ensure uniqueness and relationship
- Join → combine multiple tables for powerful queries.