UNIT -II

RELATIONAL MODEL AND SQL

Joins, Indexes, Sequences, and Triggers – 16 Marks

1. Joins in SQL

A **join** is used to retrieve data from two or more tables based on related columns. Joins combine rows of tables to produce meaningful information.

1.1 Inner Join

- Returns only the **matching rows** from both tables.
- Non-matching rows are discarded.

Syntax

```
SELECT columns
FROM A INNER JOIN B
ON A.key = B.key;
```

Example

```
SELECT Student.Name, Dept.DeptName
FROM Student INNER JOIN Dept
ON Student.DeptID = Dept.DeptID;
```

Only students with a department assigned will appear.

1.2 Outer Joins

Outer joins include **matching rows** and also **non-matching rows**.

A. Left Outer Join

- Returns all rows from **left** table + matching rows from right table.
- Non-matching rows in right table appear as **NULL**.

```
SELECT Name, DeptName
FROM Student LEFT JOIN Dept
ON Student.DeptID = Dept.DeptID;
```

B. Right Outer Join

- Returns all rows from **right** table + matching rows from left.
- Non-matching rows from left become **NULL**.

```
SELECT Name, DeptName
FROM Student RIGHT JOIN Dept
ON Student.DeptID = Dept.DeptID;
```

C. Full Outer Join

- Returns **all rows** from both tables.
- Non-matching rows are padded with NULLs on corresponding sides.

```
SELECT Name, DeptName
FROM Student FULL OUTER JOIN Dept
ON Student.DeptID = Dept.DeptID;
```

1.3 Self Join

A table is joined with **itself**.
Used in hierarchical data (e.g., employees and managers).

Example

```
SELECT E.Name AS Employee, M.Name AS Manager
FROM Employee E
JOIN Employee M
ON E.ManagerID = M.EmpID;
```

---

2. Indexes

An **index** is a database object that improves the **speed of data retrieval**.
It works like a book index to locate records quickly.

Characteristics

- Improves **SELECT** query performance.
- Slows down **INSERT / UPDATE / DELETE** because index must be updated.
- Typically implemented using **B-tree** structures.

Syntax

```
CREATE INDEX idx_student_name
ON Student(Name);
```

Unique Index

Prevents duplicate values.

```
CREATE UNIQUE INDEX idx_rollno
ON Student(RollNo);
```

---

3. Sequences

A **sequence** is an object that generates **automatic numeric values**, commonly used for primary keys.

Characteristics

- Auto-incremented numbers.
- Supports NEXTVAL and CURRVAL operations.

Create Sequence

```
CREATE SEQUENCE stud_seq
START WITH 1
INCREMENT BY 1;
```

Use in Insert

```
INSERT INTO Student(RollNo, Name)
VALUES(stud_seq.NEXTVAL, 'Arun');
```

---

4. Triggers

A **trigger** is a stored program that automatically executes in response to certain events such as **INSERT, UPDATE, or DELETE**.

Uses of Triggers

- Enforcing business rules
- Auditing changes
- Automatic updates
- Maintaining logs

Types

- **BEFORE Trigger**
- **AFTER Trigger**
- **ROW-level / STATEMENT-level**

Example Trigger

Automatically log deleted rows:

```
CREATE TRIGGER log_delete
AFTER DELETE ON Student
FOR EACH ROW
BEGIN
    INSERT INTO Student_Log(RollNo, Name, DeletedOn)
    VALUES(OLD.RollNo, OLD.Name, SYSDATE);
END;
```
