DBMS [Day – 3]

UID: 24MCI10204

Name: Rahul Saxena

Branch: 24MCA – AI & ML

Question 1: You are managing a database for an online learning platform with the following tables:

- Students(student_id, name, email, enrollment_date)
- Courses(course_id, title, instructor)
- Enrollments(enrollment_id, student_id, course_id, grade)

The platform experiences performance issues with frequent grade reports and course analysis.

Tasks:

1. Subqueries:
    - Write a correlated subquery to list all students who scored above the average grade in the course they are enrolled in.
    - Write a multi-row subquery to find the names of all instructors who teach courses where at least one student received a grade below 50.
2. Views:
    - Create a view, high_achievers that shows student names, course titles, and grades for all students who scored above 85.
    - Update the view to include only courses that have "Data" in their title (case-insensitive).
    - Drop the view when no longer needed.
3. Indexes:
    - Explain the difference between clustered and non-clustered indexes using the Enrollments table.
    - Create an index on Enrollments(grade) and justify how it helps improve performance in frequent grade-based filtering.

Code:

Table Creation

```
CREATE TABLE Students (
   student_id INT PRIMARY KEY,
   name VARCHAR(100),
   email VARCHAR(100),
   enrollment_date DATE
);
CREATE TABLE Courses (
   course_id INT PRIMARY KEY,
   title VARCHAR(100),
   instructor VARCHAR(100)
);
CREATE TABLE Enrollments (
```

```
    enrollment_id INT PRIMARY KEY,
    student_id INT,
    course_id INT,
    grade INT,
    FOREIGN KEY (student_id) REFERENCES Students(student_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);
```

Data Insertion

```
INSERT INTO Students VALUES
(1, 'Rahul Saxena', 'rahul@gmail.com', '2023-01-10'),
(2, 'Ram Kumar', 'ram@gmail.com', '2023-02-15'),
(3, 'Atul Pardhi', 'atul12@gmail.com', '2023-03-05'),
(4, 'Lakshya Singh', 'Lsingh@outlook.com', '2023-04-20');

INSERT INTO Courses VALUES
(101, 'Artificial Intelligent', 'Dr. Arun Singh'),
(102, 'Web Development', 'Dr. Disha'),
(103, 'Data Analysis', 'Dr. Rekha');

INSERT INTO Enrollments VALUES
(1001, 1, 101, 90),
(1002, 2, 101, 70),
(1003, 3, 102, 45),
(1004, 1, 103, 88),
(1005, 2, 103, 55),
(1006, 4, 102, 75),
(1007, 3, 103, 49);
```

Subqueries

A)
```
SELECT s.name, e.grade, c.title
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id
WHERE e.grade > (
    SELECT AVG(e2.grade)
    FROM Enrollments e2
    WHERE e2.course_id = e.course_id
);
```

B)
```
SELECT DISTINCT instructor
FROM Courses
WHERE course_id IN (
    SELECT course_id
    FROM Enrollments
    WHERE grade < 50
);
```

Views Creation

A) CREATE VIEW high_achievers AS
SELECT s.name AS student_name, c.title AS course_title, e.grade
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON c.course_id = e.course_id
WHERE e.grade > 85;

B) CREATE OR REPLACE VIEW high_achievers AS
SELECT s.name AS student_name, c.title AS course_title, e.grade
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON c.course_id = e.course_id
WHERE e.grade > 85 AND LOWER(c.title) LIKE '%data%';

C) DROP VIEW IF EXISTS high_achievers;

Indexes

CREATE INDEX idx_grade ON Enrollments(grade);

Question 2: You are given a poorly designed single table called EmployeeInfo with the following structure:

EmployeeInfo(emp_id, emp_name, dept_id, dept_name, project1, project2, salary)
Tasks:

1. Normalization:
    - Identify the insertion, update, and deletion anomalies present in the current design.
    - Decompose EmployeeInfo into 3NF, showing each resulting relation and specifying the Primary Key and Functional Dependencies used at each stage (1NF → 2NF → 3NF).
    - Explain whether this design satisfies BCNF, and if not, perform further decomposition.
2. Denormalization:
    - Under what circumstances would denormalizing the normalized schema be beneficial in this case?
    - Propose a denormalized version of the schema optimized for reporting employee projects per department, and explain what trade-offs are involved.
3. Real-World Connection:
    - Discuss a real-world scenario (e.g., in analytics, dashboards, or data warehousing) where views or denormalized tables are preferred over a fully normalized structure.

Solution

Step 1 – 1NF (First Normal Form): Remove Repeating Groups

Employee(emp_id, emp_name, dept_id, dept_name, salary)

ProjectAssignment(emp_id, project_name)

Step 2 – 2NF (Second Normal Form): Remove Partial Dependencies

Department(dept_id, dept_name)

Employee(emp_id, emp_name, dept_id, salary)

Step 3 – 3NF (Third Normal Form): Remove Transitive Dependencies

Employee(emp_id, emp_name, dept_id, salary)

Department(dept_id, dept_name)

ProjectAssignment(emp_id, project_name)

Denormalization Suggestion

EmployeeReport(emp_id, emp_name, dept_name, project1, project2, salary)