# Experiment 5: Indexing Techniques

## Prerequisites

Knowledge of SQL queries and basic understanding of query execution

---

# Lab Objective

To understand the concept of indexing in relational databases and analyze how indexes improve query performance.

---

# Learning Outcomes

By completing this experiment, students will be able to:

- Understand the purpose and importance of indexes.
- Create and manage different types of indexes.
- Compare query performance before and after indexing.
- Identify when indexing is beneficial.
- Understand the trade-offs of indexing (storage and maintenance cost).

---

# Introduction

Indexes are special data structures used by database systems to speed up data retrieval operations. Without indexes, the database performs a **full table scan** to locate records, which can be time-consuming for large datasets.

Indexing improves search efficiency but also introduces additional storage requirements and maintenance overhead during INSERT, UPDATE, and DELETE operations. This experiment focuses on studying different indexing techniques and evaluating their impact on performance.

---

# Key Concepts Covered

- **Primary Index:** Automatically created for primary key columns.
- **Unique Index:** Ensures uniqueness of column values.
- **Composite Index:** Index created on multiple columns.
- **Clustered Index:** Determines the physical order of data storage.
- **Non-Clustered Index:** Maintains a separate structure for quick lookups.
- **B-Tree Structure:** Common indexing structure used in relational databases.

---

# Experiment Steps

---

## 1. Observe Query Performance Without Index

Execute queries on a large table and observe performance.

### Example:

EXPLAIN SELECT * FROM Students WHERE age = 21;

Record:

- Access type (ALL indicates full table scan)
- Rows examined
- Execution time

---

## 2. Create Different Types of Indexes

### Single-Column Index

CREATE INDEX idx_age ON Students(age);

### Unique Index

CREATE UNIQUE INDEX idx_email ON Students(email);

### Composite Index

CREATE INDEX idx_course_marks ON Enrollments(course_id, marks);

Students should verify indexes using:

SHOW INDEX FROM Students;

---

# 3. Analyze Performance After Indexing

Re-run the same query after creating an index.

EXPLAIN SELECT * FROM Students WHERE age = 21;

Compare:

- Access type (range/index instead of ALL)
- Number of rows scanned
- Estimated cost
- Execution time

Document performance improvements.

---

# 4. Study Index Maintenance

Insert and update records after indexing.

Example:

INSERT INTO Students VALUES (6, 'Rohan', 22);

Observe:

- Slight delay due to index update
- Increased storage usage

Discuss how indexes affect data modification speed.

---

## 5. Compare Clustered vs Non-Clustered Behavior

- Observe how clustered indexes affect data storage order.
- Compare retrieval speed for indexed vs non-indexed columns.
- Analyze differences in execution plan output.

Students should explain which type performs better under specific conditions.

---

# How It Works

- Indexes create a structured lookup mechanism (typically using B-Tree).
- Instead of scanning entire tables, the database navigates the index structure.
- This reduces search time significantly for large datasets.
- However, indexes require additional storage and maintenance during updates.

---

# Submission Instructions

Students must submit:

- Screenshot of table structure before indexing.
- Screenshot showing created indexes.
- Execution plan comparison (before and after indexing).
- Execution time comparison.

- Brief analysis explaining performance improvement observed.

---

# Conclusion

This experiment helps students understand how indexing improves database performance and when it should be applied. Proper indexing is a critical skill for optimizing real-world database applications and ensuring efficient data retrieval in large-scale systems.