Experiment 10: Parallel Query Processing

Prerequisites

Knowledge of SQL queries, query optimization, indexing, and basic distributed database concepts

---

Lab Objective

To understand the concept of parallel query processing in database systems and analyze how parallelism improves query performance and scalability.

Learning Outcomes

By completing this experiment, students will be able to:

- Understand the need for parallel query processing.
- Explain different types of parallelism in DBMS.
- Analyze how queries are divided into parallel tasks.
- Compare performance between serial and parallel execution.
- Understand advantages and limitations of parallel processing.

Introduction

In large-scale database systems handling massive datasets, executing queries sequentially (serial execution) can lead to high response times. To improve performance, modern database systems use **parallel query processing**, where multiple operations are executed simultaneously using multiple CPUs or processors.

Parallel query processing is commonly used in data warehousing, big data analytics, and high-performance computing environments.

This experiment demonstrates how parallel execution reduces query response time and improves throughput.

Key Concepts Covered

- **Parallelism:** Executing multiple operations simultaneously.
- **Types of Parallelism:**
  - Intra-Query Parallelism
  - Inter-Query Parallelism
- **Data Partitioning:** Dividing data across multiple processors.
- **Pipeline Parallelism:** Overlapping execution of operations.
- **Parallel Join Algorithms**
- **Speedup and Scalability**

Experiment Steps

---

1. Understand Serial Query Execution

Execute a complex query normally and observe:

- Execution time
- CPU usage
- Rows processed

Example:

```
EXPLAIN SELECT *
FROM Enrollments e
JOIN Students s ON e.student_id = s.student_id
JOIN Courses c ON e.course_id = c.course_id;
```

Record execution plan details.

---

2. Enable Parallel Execution (Database-Specific)

Depending on DBMS, enable parallel query processing.

Example (conceptual):

SET max_parallel_workers_per_gather = 4;

Re-run the same query and compare execution plans.

---

3. Analyze Parallel Execution Plan

Observe:

- Number of workers used
- Parallel sequential scan
- Parallel hash join
- Reduced execution time

Students should compare:

- Cost
- Rows processed
- Execution time

---

4. Study Types of Parallelism

Intra-Query Parallelism

Single query divided into smaller tasks executed in parallel.

Inter-Query Parallelism

Multiple independent queries executed simultaneously.

Discuss real-world examples such as reporting dashboards handling multiple user queries.

---

5. Measure Speedup

Speedup formula:

Speedup = Time (Serial Execution) / Time (Parallel Execution)

Students should calculate and analyze performance gain.

How It Works

- Large tables are divided across multiple processors.
- Query operations (scan, join, aggregation) run concurrently.
- Results from parallel tasks are combined.
- Parallel processing reduces execution time significantly for large datasets.

However:

- It requires more CPU resources.
- Overhead of coordination may reduce benefit for small queries.

Submission Instructions

Students must submit:

- Screenshot of serial execution plan.
- Screenshot of parallel execution plan.
- Execution time comparison.
- Speedup calculation.
- Brief analysis explaining observed improvements.

Conclusion

Parallel query processing improves performance and scalability in modern database systems. It is widely used in data warehouses and large enterprise systems to handle massive data

efficiently. Understanding parallel execution prepares students for working with high-performance and distributed database environments.