Experiment 9: Recovery Techniques

Prerequisites

Knowledge of transactions, ACID properties, and basic concurrency control concepts

---

Lab Objective

To understand database recovery mechanisms and simulate how database systems restore consistency after system failures using log-based recovery techniques.

Learning Outcomes

By completing this experiment, students will be able to:

- Understand the need for recovery mechanisms in DBMS.
- Explain log-based recovery concepts (Undo and Redo).
- Understand checkpointing mechanisms.
- Simulate transaction failure and system crash scenarios.
- Analyze how recovery techniques maintain database consistency.

Introduction

In real-world database systems, failures such as system crashes, power outages, or transaction errors may occur. Recovery techniques ensure that the database returns to a consistent state after such failures.

Database recovery is based on maintaining logs that record every transaction. Using these logs, the system can:

- Undo incomplete transactions

- Redo committed transactions
- Restore database consistency

This experiment demonstrates how recovery mechanisms work using simulated failures.

Key Concepts Covered

- **Transaction Log:** Record of all database modifications.
- **Write-Ahead Logging (WAL):** Log entry is written before actual data modification.
- **Undo Operation:** Reverses changes of uncommitted transactions.
- **Redo Operation:** Reapplies committed changes after a crash.
- **Checkpointing:** Saves database state periodically to reduce recovery time.
- **Crash Recovery:** Restoring database after system failure.

Experiment Steps

---

1. Simulate Transaction Failure (Undo Operation)

Start a transaction and perform updates.

```
START TRANSACTION;
UPDATE Students SET age = age + 1 WHERE student_id = 1;
```

Simulate failure by rolling back.

```
ROLLBACK;
```

Observation:

- Changes are undone.
- Database returns to previous consistent state.

---

2. Simulate Committed Transaction (Redo Concept)

START TRANSACTION;
UPDATE Students SET age = age + 2 WHERE student_id = 2;
COMMIT;

Observation:

- Changes are permanently stored.
- If system crashes after commit, recovery process redoes committed operations.

---

3. Demonstrate Checkpoint Concept

Discuss or simulate:

- Periodic checkpoint saves database state.
- After checkpoint, recovery process only checks transactions after the checkpoint.

Example conceptual command (DB-specific):

CHECKPOINT;

Students should understand how checkpoints reduce recovery time.

---

4. Simulate Crash Scenario (Conceptual Demonstration)

Scenario:

1. Transaction T1 updates record but does not commit.
2. Transaction T2 updates record and commits.
3. System crashes.

Recovery process:

- Undo T1 (uncommitted).
- Redo T2 (committed).

Students should describe expected database state after recovery.

---

5. Analyze Log-Based Recovery Process

Discuss how transaction logs contain:

- Transaction ID
- Operation performed
- Old value (for Undo)
- New value (for Redo)

Explain how Write-Ahead Logging ensures safe recovery.

How It Works

- Every database modification is recorded in a log.
- If a failure occurs:
    - Uncommitted transactions are undone.
    - Committed transactions are redone.
- Checkpoints reduce the amount of log scanning required.
- Recovery mechanisms guarantee Atomicity and Durability (ACID properties).

Submission Instructions

Students must submit:

- Screenshot demonstrating rollback (Undo).
- Screenshot showing committed transaction.
- Explanation of crash recovery scenario.
- Short note explaining Undo and Redo operations.

Conclusion

This experiment helps students understand how database systems maintain consistency and reliability during failures. Recovery mechanisms are essential in enterprise systems such as banking, healthcare, and e-commerce applications where data integrity is critical.