Experiment 7: Transaction Control Commands

Prerequisites

Knowledge of SQL queries and basic understanding of database transactions

---

Lab Objective

To understand and implement transaction control commands in SQL and analyze how transactions maintain data consistency and integrity in a database system.

Learning Outcomes

By completing this experiment, students will be able to:

- Understand the concept of database transactions.
- Explain ACID properties (Atomicity, Consistency, Isolation, Durability).
- Use transaction control commands such as COMMIT, ROLLBACK, and SAVEPOINT.
- Manage auto-commit behavior.
- Demonstrate how transactions ensure data reliability.

Introduction

A transaction is a sequence of one or more SQL operations executed as a single logical unit of work. Transactions ensure that database changes are processed reliably and consistently.

Transaction Control Language (TCL) commands allow users to manage changes made by DML statements (INSERT, UPDATE, DELETE). Proper transaction handling is essential in systems such as banking, e-commerce, and reservation systems.

Key Concepts Covered

- **Transaction:** A logical unit of work performed on the database.
- **ACID Properties:**
    - Atomicity – All operations succeed or none do.
    - Consistency – Database remains in a valid state.
    - Isolation – Transactions do not interfere with each other.
    - Durability – Committed changes are permanent.
- **COMMIT:** Permanently saves changes.
- **ROLLBACK:** Undoes changes made during the transaction.
- **SAVEPOINT:** Creates a rollback point within a transaction.
- **Auto-Commit Mode:** Automatically commits each statement.

Experiment Steps

---

1. Start a Transaction

Disable auto-commit (if enabled).

SET autocommit = 0;

Begin a transaction.

START TRANSACTION;

---

2. Perform Data Modification

Insert or update records.

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

Verify the change before committing.

---

3. Use COMMIT

Save changes permanently.

COMMIT;

Confirm that changes persist.

---

4. Use ROLLBACK

Start another transaction and perform an update.

START TRANSACTION;

UPDATE Students
SET age = age + 2
WHERE student_id = 2;

Undo changes.

ROLLBACK;

Verify that original data is restored.

---

5. Use SAVEPOINT

Create a savepoint within a transaction.

START TRANSACTION;

UPDATE Students SET age = age + 1 WHERE student_id = 3;

SAVEPOINT sp1;

UPDATE Students SET age = age + 2 WHERE student_id = 4;

Rollback to savepoint.

ROLLBACK TO sp1;

Commit remaining changes.

COMMIT;

---

6. Demonstrate ACID Properties

- Show atomic behavior using rollback.
- Demonstrate durability by verifying committed changes remain after session restart.
- Discuss isolation conceptually (with multiple sessions if possible).

How It Works

- A transaction groups multiple SQL statements into a single logical unit.
- Changes are temporary until committed.
- ROLLBACK restores database to previous consistent state.
- SAVEPOINT allows partial rollback within a transaction.
- ACID properties ensure reliable and secure database operations.

Submission Instructions

Students must submit:

- Screenshot showing transaction start.
- Output after performing UPDATE before commit.
- Screenshot after COMMIT.

- Screenshot demonstrating ROLLBACK behavior.
- Example showing SAVEPOINT usage.

Conclusion

This experiment demonstrates how transaction control commands maintain database consistency and reliability. Understanding transaction management is essential for designing secure and fault-tolerant database systems used in real-world applications.