Hands On Lab: InnoDB Engine Deep Dive

LAB 3.1: Transactions and ACID Properties

Objective: Understand ACID properties and transaction handling in InnoDB.

Prerequisites:

- MySQL Server with InnoDB engine
- Two MySQL client connections

Step-by-Step Instructions:

1. **Create Test Database and Tables**

```
CREATE DATABASE innodb_transactions;
USE innodb_transactions;

CREATE TABLE accounts (
   account_id INT PRIMARY KEY AUTO_INCREMENT,
   account_name VARCHAR(100),
   balance DECIMAL(10, 2)
) ENGINE=InnoDB;

INSERT INTO accounts (account_name, balance) VALUES
   ('Alice Account', 1000.00),
   ('Bob Account', 500.00),
   ('Charlie Account', 750.00);
```

2. **Open Two MySQL Connections (Terminal 1 and Terminal 2)**

Connection 1:

```
mysql -u root -p innodb_transactions
```

Connection 2:

```
mysql -u root -p innodb_transactions
```

3. **Connection 1: Start Transaction**

   ```
   BEGIN;
   -- Or: START TRANSACTION;


   SELECT * FROM accounts;
   ```

4. **Connection 1: Perform Update (Don't commit yet)**

   ```
   UPDATE accounts SET balance = balance - 100 WHERE account_id = 1;
   UPDATE accounts SET balance = balance + 100 WHERE account_id = 2;


   SELECT * FROM accounts;
   ```

5. **Connection 2: Try to Read Updated Data (Isolation test)**

   ```
   SELECT * FROM accounts;
   -- Should see original values due to transaction isolation
   ```

6. **Connection 1: Commit Transaction**

   ```
   COMMIT;
   ```

7. **Connection 2: Read Data Again**

   ```
   SELECT * FROM accounts;
   -- Should see updated values now
   ```

8. **Connection 1: Test Rollback**

   ```
   BEGIN;
   UPDATE accounts SET balance = balance - 50 WHERE account_id = 3;
   SELECT * FROM accounts;
   ROLLBACK;
   SELECT * FROM accounts;
   ```

9. **Check Transaction Isolation Level**

   ```
   SHOW VARIABLES LIKE 'transaction_isolation';
   SELECT @@transaction_isolation;
   ```

10. **Set Different Isolation Levels and Test**

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

Hands-on Tasks:

- Perform multiple concurrent transactions
- Test dirty reads, non-repeatable reads, and phantom reads
- Observe transaction locks: SHOW OPEN TABLES WHERE In_use > 0;
- Monitor transaction status: SHOW PROCESSLIST;
- Document the differences between isolation levels

LAB 3.2: InnoDB Tablespace Configuration

Objective: Configure and manage InnoDB tablespaces.

Step-by-Step Instructions:

1. **Check Default Tablespace Configuration**

```
SHOW VARIABLES LIKE 'innodb_data_file_path';

SHOW VARIABLES LIKE 'innodb_undo_tablespaces';

SELECT * FROM INFORMATION_SCHEMA.INNODB_TABLESPACES;
```

2. **View Tablespace Information**

```
SELECT
   SPACE,
   NAME,
   FILE_TYPE,
   ENGINE,
   EXTENT_SIZE,
   INITIAL_SIZE,
   CURRENT_SIZE
FROM INFORMATION_SCHEMA.INNODB_TABLESPACES
LIMIT 10\G
```

3. **Create Database and Tables**

```
CREATE DATABASE tablespace_demo;
USE tablespace_demo;

CREATE TABLE table1 (
   id INT PRIMARY KEY AUTO_INCREMENT,
   data VARCHAR(255)
) ENGINE=InnoDB;

CREATE TABLE table2 (
   id INT PRIMARY KEY AUTO_INCREMENT,
   content LONGTEXT
) ENGINE=InnoDB;
```

4. **Insert Sample Data**

```
-- Insert 10,000 rows into table1
INSERT INTO table1 (data) VALUES ('Sample data');
-- Repeat with a loop or script

-- For bash script:
bash -c 'for i in {1..10000}; do
  mysql innodb_transactions -e "INSERT INTO table1 (data) VALUES (\"Test data $i\");"
done'
```

5. **Monitor Table Size**

```
SELECT
    TABLE_SCHEMA,
    TABLE_NAME,
    ROUND(((DATA_LENGTH + INDEX_LENGTH) / 1024 / 1024), 2) AS size_mb,
    TABLE_ROWS
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'tablespace_demo';
```

6. **View InnoDB Buffer Pool Usage**

```
SHOW ENGINE INNODB STATUS\G
-- Look for "Buffer pool" section
```

7. **Configure InnoDB File-Per-Table Tablespaces**

```
SHOW VARIABLES LIKE 'innodb_file_per_table';
-- It should be ON by default in MySQL 5.7+
```

8. **Create Separate Tablespace for Large Table**

```
CREATE TABLE large_table (
    id INT PRIMARY KEY AUTO_INCREMENT,
    big_data LONGBLOB
) ENGINE=InnoDB
TABLESPACE = innodb_file_per_table;
```

9. **Reclaim Tablespace**

```
DELETE FROM table1 WHERE id > 5000;
OPTIMIZE TABLE table1;
```

Hands-on Tasks:

- Monitor tablespace growth as data is inserted
- Observe the relationship between row count and tablespace size
- Test the effect of OPTIMIZE TABLE on disk space
- Document tablespace file locations: ls -lh /var/lib/mysql/tablespace_demo/

LAB 3.3: Monitoring InnoDB with SHOW ENGINE INNODB STATUS

Objective: Interpret InnoDB status output for performance monitoring.

Step-by-Step Instructions:

1. **Generate InnoDB Status Report**

   ```
   SHOW ENGINE INNODB STATUS;
   ```

2. **Understand Key Sections**

a) **Background Threads Section**

-- Example output:

-- I/O thread 0 state: waiting for i/o request

b) **Pending I/O Reads and Writes**

-- Monitor active I/O operations

c) **Buffer Pool Efficiency**

-- Look for: Buffer pool hit rate (should be > 99%)

d) **Transaction Information**

-- Current transactions, locks, and conflicts

3. **Create Activity for Monitoring**

   ```
   -- Terminal 1: Start long-running transaction

   USE tablespace_demo;

   BEGIN;

   SELECT * FROM table1 WHERE id < 100;


   -- Terminal 2: Try to update locked rows

   UPDATE table1 SET data = 'Updated' WHERE id = 50;
   ```

4. **Monitor Lock Information**

   ```
   SHOW PROCESSLIST;

   SELECT * FROM INFORMATION_SCHEMA.INNODB_LOCKS\G

   SELECT * FROM INFORMATION_SCHEMA.INNODB_LOCK_WAITS\G
   ```

5. **Check Buffer Pool Details**

```
SELECT
   POOL_ID,
   POOL_SIZE,
   FREE_BUFFERS,
   DATABASE_PAGES,
   PAGES_MADE_YOUNG,
   PAGES_NOT_MADE_YOUNG
FROM INFORMATION_SCHEMA.INNODB_BUFFER_POOL_STATS\G
```

6. **Monitor Transaction Throughput**

```
SHOW STATUS LIKE 'Innodb_rows_%';
SHOW STATUS LIKE 'Innodb_data_%';
SHOW STATUS LIKE 'Innodb_buffer_pool_%';
```

Hands-on Tasks:

- Capture InnoDB status before and after workload
- Identify buffer pool hit rate
- Observe how transactions appear in status output
- Document thread activity and I/O metrics