AY: 2024-25

Class:	**SE**	**Semester:**	**IV**
Course Code:	**CSL402**	**Course Name:**	**DBMS Lab**

Name of Student:	**Shravani Sandeep Raut**
Roll No. :	**48**
Experiment No.:	**7**
Title of the Experiment:	**Perform DCL and TCL commands**
Date of Performance:	**19/03/2025**
Date of Submission:	**26/03/2025**

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Ms. Neha Raut

Signature :

Date:

Aim :- Write a query to implement Data Control Language(DCL) and Transaction Control Language(TCL) commands

Objective :- To learn DCL commands like Grant and Revoke privileges to the user and TCL commands to commit the transactions and recover it using rollback and save points.

Theory:

Data Control Language:

DCL commands are used to grant and take back authority from any database user.

- Grant
- Revoke

a. Grant: It is used to give user access privileges to a database.

Example

```
1. GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER,
   ANOTHER_USER;
```

b. Revoke: It is used to take back permissions from the user.

Example

```
1. REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;
```

Transaction Control Language

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

- COMMIT
- ROLLBACK
- SAVEPOINT

a. Commit: Commit command is used to save all the transactions to the database.

Syntax:

1. COMMIT;

Example:

1. DELETE FROM CUSTOMERS
2. WHERE AGE = 25;
3. COMMIT;

b. Rollback: Rollback command is used to undo transactions that have not already been saved to the database.

Syntax:

1. ROLLBACK;

Example:

1. DELETE FROM CUSTOMERS
2. WHERE AGE = 25;
3. ROLLBACK;

c. SAVEPOINT: It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Syntax:

2. SAVEPOINT SAVEPOINT_NAME;

Implementation:

1. Data Control Language :

```
265 •  CREATE USER 'student_admin'@'localhost' IDENTIFIED BY 'admin123';
266 •  CREATE USER 'student_reader'@'localhost' IDENTIFIED BY 'reader123';
267
268 •  GRANT ALL PRIVILEGES ON student.* TO 'student_admin'@'localhost';
269 •  FLUSH PRIVILEGES;
270 •  GRANT SELECT ON student.* TO 'student_reader'@'localhost';
271 •  GRANT insert ON student.* TO 'student_admin'@'localhost';
272 •  INSERT INTO STUDENT_INFORMATION (STUDENT_ID, STUDENT_NAME, STUDENT_AGE, STUDENT_GENDER, ADDRESS, Student_CONTACT)
273    VALUES (5, 'Emma Brown', 20, 'Female', '202 Maple St, Leeds', '555-3344');
274 •  REVOKE INSERT ON student.STUDENT_INFORMATION FROM 'student_admin'@'localhost';
275
```

Context Help

Output

Action Output

#	Time	Action	Message
57	18:43:57	CREATE TABLE ENROLLMENT (ENROLLMENT_ID INT PRIMARY KEY, Student_ID INT, QUANTITY INT, FOREIGN KEY (Student_ID) ...	0 row(s) affected
58	18:44:15	INSERT INTO ENROLLMENT (ENROLLMENT_ID, Student_ID, QUANTITY) VALUES (1, 1, 5), -- John Doe is enrolled in 5 activities/courses (2, 2, 2),...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0
59	18:44:31	SELECT s.STUDENT_NAME, s.STUDENT_AGE, s.STUDENT_GENDER, e.QUANTITY FROM STUDENT_INFORMATION s INNER JOI...	3 row(s) returned
60	18:54:02	CREATE USER 'student_admin'@'localhost' IDENTIFIED BY 'admin123'	0 row(s) affected
61	18:54:02	CREATE USER 'student_reader'@'localhost' IDENTIFIED BY 'reader123'	0 row(s) affected
62	18:54:37	GRANT ALL PRIVILEGES ON student.* TO 'student_admin'@'localhost'	0 row(s) affected

2. Transaction Control Language:

```
276 •  START TRANSACTION;
277 •  UPDATE STUDENT_INFORMATION
278    SET STUDENT_AGE = 21
279    WHERE STUDENT_ID = 2;
280 •  GRANT INSERT ON student.STUDENT_INFORMATION TO 'student_reader'@'localhost';
281 •  INSERT INTO Student_INFORMATION (STUDENT_ID, STUDENT_NAME, STUDENT_AGE, STUDENT_GENDER, ADDRESS, Student_CONTACT)
282    VALUES (9, 'adam Taylor', 23, 'Male', '303 Birch Rd, Bristol', '555-4455');
283 •  COMMIT;
284 •  SELECT * FROM STUDENT_INFORMATION;
285 •  SELECT * FROM ENROLLMENT;
286
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

STUDENT_ID	STUDENT_NAME	STUDENT_AGE	STUDENT_GENDER	ADDRESS	STUDENT_CONTACT	EMAIL
5	Emma Brown	20	Female	202 Maple St, Leeds	555-3344	NULL
6	Mark Taylor	23	Male	303 Birch Rd, Bristol	555-4455	NULL
8	adam Taylor	23	Male	303 Birch Rd, Bristol	555-4455	NULL
9	adam Taylor	23	Male	303 Birch Rd, Bristol	555-4455	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid | Apply | Context Help

STUDENT_INFORMATION 35 × ENROLLMENT 36

Output

Action Output

#	Time	Action	Message
7	19:26:34	UPDATE STUDENT_INFORMATION SET STUDENT_AGE = 21 WHERE STUDENT_ID = 2	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
8	19:26:34	GRANT INSERT ON student.STUDENT_INFORMATION TO 'student_reader'@'localhost'	0 row(s) affected
9	19:26:34	INSERT INTO Student_INFORMATION (STUDENT_ID, STUDENT_NAME, STUDENT_AGE, STUDENT_GENDER, ADDRESS, Student_CONTACT...	1 row(s) affected
10	19:26:34	COMMIT	0 row(s) affected
11	19:26:34	SELECT * FROM STUDENT_INFORMATION LIMIT 0, 1000	7 row(s) returned
12	19:26:34	SELECT * FROM ENROLLMENT LIMIT 0, 1000	4 row(s) returned

```
START TRANSACTION;
UPDATE STUDENT_INFORMATION
SET STUDENT_AGE = 21
WHERE STUDENT_ID = 2;
GRANT INSERT ON student.STUDENT_INFORMATION TO 'student_reader'@'localhost';
INSERT INTO Student_INFORMATION (STUDENT_ID, STUDENT_NAME, STUDENT_AGE, STUDENT_GENDER, ADDRESS, Student_CONTACT)
VALUES (9, 'adam Taylor', 23, 'Male', '303 Birch Rd, Bristol', '555-4455');
COMMIT;
SELECT * FROM STUDENT_INFORMATION;
SELECT * FROM ENROLLMENT;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

STUDENT_ID	STUDENT_NAME	STUDENT_AGE	STUDENT_GENDER	ADDRESS	STUDENT_CONTACT	EMAIL
5	Emma Brown	20	Female	202 Maple St, Leeds	555-3344	NULL
6	Mark Taylor	23	Male	303 Birch Rd, Bristol	555-4455	NULL
8	adam Taylor	23	Male	303 Birch Rd, Bristol	555-4455	NULL
9	adam Taylor	23	Male	303 Birch Rd, Bristol	555-4455	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

STUDENT_INFORMATION 35 | ENROLLMENT 36

Output

Action Output

#	Time	Action	Message
7	19:26:34	UPDATE STUDENT_INFORMATION SET STUDENT_AGE = 21 WHERE STUDENT_ID = 2	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
8	19:26:34	GRANT INSERT ON student.STUDENT_INFORMATION TO 'student_reader'@'localhost'	0 row(s) affected
9	19:26:34	INSERT INTO Student_INFORMATION (STUDENT_ID, STUDENT_NAME, STUDENT_AGE, STUDENT_GENDER, ADDRESS, Student_CONTACT...	1 row(s) affected
10	19:26:34	COMMIT	0 row(s) affected
11	19:26:34	SELECT * FROM STUDENT_INFORMATION LIMIT 0, 1000	7 row(s) returned
12	19:26:34	SELECT * FROM ENROLLMENT LIMIT 0, 1000	4 row(s) returned

Conclusion:

A) Explain about issues faced during rollback in mysql and how it got resolved.

Rollback issues in MySQL often arise due to autocommit being enabled, non-transactional storage engines like MyISAM, or improper transaction handling. When a rollback is attempted under these conditions, changes may not revert, causing data inconsistency. This was resolved by ensuring the use of transactional engines like InnoDB, explicitly disabling autocommit, and enclosing SQL operations within `START TRANSACTION` and `COMMIT/ROLLBACK`. Proper error handling and constraints were also implemented to manage failures efficiently. These measures ensured reliable rollback and data integrity during unexpected failures or manual intervention.

B) Explain how to create a user in sql.

To create a user in SQL, the `CREATE USER` statement is used. Issues may arise if the user already exists or if proper privileges are missing. The syntax is:

CREATE USER 'username'@'host' IDENTIFIED BY 'password';

For example:

CREATE USER 'shravani'@'localhost' IDENTIFIED BY 'pass123';

This command creates a new user 'shravani' who can connect from localhost. To resolve errors like "access denied," make sure the admin account has enough privileges. After creation, use GRANT to assign permissions, ensuring the user can access and modify data as needed.