CS3120 Database Management Systems Laboratory

Assignment 6

Mayank Singla
111901030

Using VSCode to write the queries in a file and then execute that file from the terminal using "`\i file.sql`"

Schema Creation

Q1. Create a University database with the name univ_db.

```
solution.sql U ×
dbms-lab > labwork > assignment_6 > solution.sql > ...
Run on active connection | Select block
-- ==========================================================
/* Schema Creation */
/*****************************************************************
***************/
/* Q1: Create a University database with name univ db. */
Run SQL
CREATE DATABASE univ_db;
```

Traditional command to create a database

```
mayank=# \i solution.sql
CREATE DATABASE
mayank=# \l
                              List of databases
   Name    |  Owner   | Encoding | Collate |  Ctype  |   Access privileges
-----------+----------+----------+---------+---------+-----------------------
 dvdrental | mayank   | UTF8     | C.UTF-8 | C.UTF-8 | =Tc/mayank           +
           |          |          |         |         | mayank=CTc/mayank
 mayank    | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
 postgres  | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
 template0 | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
           |          |          |         |         | postgres=CTc/postgres
 template1 | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
           |          |          |         |         | postgres=CTc/postgres
 univ_db   | mayank   | UTF8     | C.UTF-8 | C.UTF-8 |
(6 rows)

mayank=#
```

Listing all the databases

Q2. Create an Student table with 4 columns(stud id,stud name,dept name,tot cred) with 10 records and Department table with 4 columns(dept id,dept name,building,budget). Choose a unique column as a primary key and also other constraints as per your understanding. **Note** : Domain of dept name = (”Mtech DS”,”Mtech SOCD”,”Mtech COM” and ”Mtech Geo”).

solution.sql U

dbms-lab > labwork > assignment_6 > solution.sql > ...

```sql
CREATE TABLE departments(
    dept_id SERIAL PRIMARY KEY,
    dept_name VARCHAR(50) UNIQUE NOT NULL CHECK(
        dept_name IN (
            'Mtech DS',
            'Mtech SOCD',
            'Mtech COM',
            'Mtech Geo'
        )
    ),
    building VARCHAR(50) NOT NULL,
    budget INT NOT NULL CHECK(budget > 0)
);
```

▶ Run SQL

```sql
CREATE TABLE students (
    stud_id SERIAL PRIMARY KEY,
    stud_name VARCHAR(30) NOT NULL,
    dept_id INT NOT NULL,
    tot_cred INT NOT NULL CHECK(tot_cred >= 0),
    FOREIGN KEY(dept_id) REFERENCES departments(dept_id)
);
```

Creating both tables with suitable constraints on each column. Creating the `dept_id` as the foreign key in the students’ table.

```
univ_db=# \i solution.sql
CREATE TABLE
CREATE TABLE
univ_db=# \dt
           List of relations
 Schema |     Name      | Type  | Owner
--------+---------------+-------+--------
 public | departments | table | mayank
 public | students    | table | mayank
(2 rows)
```

solution.sql U ×

dbms-lab > labwork > assignment_6 > solution.sql > ...

```sql
INSERT INTO departments(dept_name, building, budget)
VALUES ('Mtech DS', 'Watson', 90000),
    ('Mtech SOCD', 'Taylor', 100000),
    ('Mtech COM', 'Painter', 85000),
    ('Mtech Geo', 'Packard', 120000);
```

Inserting the departments for each department in the domain into the departments' table.

```
univ_db=# \i solution.sql
INSERT 0 4
univ_db=# SELECT * FROM departments;
 dept_id | dept_name  | building | budget
---------+------------+----------+--------
       1 | Mtech DS   | Watson   |  90000
       2 | Mtech SOCD | Taylor   | 100000
       3 | Mtech COM  | Painter  |  85000
       4 | Mtech Geo  | Packard  | 120000
(4 rows)
```

solution.sql U ×

dbms-lab > labwork > assignment_6 > solution.sql > ...

```sql
INSERT INTO students(stud_name, dept_id, tot_cred)
VALUES ('Satyam', 1, 3),
    ('Amish', 2, 4),
    ('Neel', 3, 5),
    ('Aditya', 4, 2),
    ('Harsh', 1, 3),
    ('Mayank', 2, 4),
    ('Naren', 3, 5),
    ('Anurag', 4, 2),
    ('Jerry', 1, 3),
    ('Hrishi', 2, 4);
```

Inserting 10 records into the students' table

```
univ_db=# \i solution.sql
INSERT 0 10
univ_db=# SELECT * FROM students;
 stud_id | stud_name | dept_id | tot_cred
---------+-----------+---------+----------
       1 | Satyam    |       1 |        3
       2 | Amish     |       2 |        4
       3 | Neel      |       3 |        5
       4 | Aditya    |       4 |        2
       5 | Harsh     |       1 |        3
       6 | Mayank    |       2 |        4
       7 | Naren     |       3 |        5
       8 | Anurag    |       4 |        2
       9 | Jerry     |       1 |        3
      10 | Hrishi    |       2 |        4
(10 rows)
```

Transactions

Q1. Insert/Update the data on the univ_db using the following clauses without violating the foreign key constraint. At least 5 records must be present for each department.

(a) BEGIN-COMMIT
(b) BEGIN-ROLLBACK
(c) BEGIN-SAVEPOINT-COMMIT

solution.sql U ×

dbms-lab > labwork > assignment_6 > solution.sql > ...

```sql
BEGIN TRANSACTION;
▶ Run SQL
INSERT INTO students(stud_name, dept_id, tot_cred)
VALUES ('Jasir', 1, 3),
       ('Yagnesh', 2, 4),
       ('Abraham', 3, 5);
▶ Run SQL
COMMIT;
```

Inserting some data into the students' table using BEGIN-COMMIT block

```
univ_db=# \i solution.sql
BEGIN
INSERT 0 3
COMMIT
univ_db=#
```

```
univ_db=# SELECT * FROM students;
 stud_id | stud_name | dept_id | tot_cred
---------+-----------+---------+----------
       1 | Satyam    |       1 |        3
       2 | Amish     |       2 |        4
       3 | Neel      |       3 |        5
       4 | Aditya    |       4 |        2
       5 | Harsh     |       1 |        3
       6 | Mayank    |       2 |        4
       7 | Naren     |       3 |        5
       8 | Anurag    |       4 |        2
       9 | Jerry     |       1 |        3
      10 | Hrishi    |       2 |        4
      11 | Jasir     |       1 |        3
      12 | Yagnesh   |       2 |        4
      13 | Abraham   |       3 |        5
(13 rows)
```

Current output after the first block

solution.sql U ×

dbms-lab > labwork > assignment_6 > solution.sql > ...

```sql
BEGIN TRANSACTION;
▶ Run SQL
INSERT INTO students(stud_name, dept_id, tot_cred)
VALUES ('Jasir', 2, 4),
       ('Yagnesh', 3, 5),
       ('Abraham', 1, 3);
▶ Run SQL
ROLLBACK;
```

Inserting same students again and therefore rolling back.

```
univ_db=# \i solution.sql
BEGIN
INSERT 0 3
ROLLBACK
univ_db=# SELECT * FROM students;
 stud_id | stud_name | dept_id | tot_cred
---------+-----------+---------+----------
       1 | Satyam    |       1 |        3
       2 | Amish     |       2 |        4
       3 | Neel      |       3 |        5
       4 | Aditya    |       4 |        2
       5 | Harsh     |       1 |        3
       6 | Mayank    |       2 |        4
       7 | Naren     |       3 |        5
       8 | Anurag    |       4 |        2
       9 | Jerry     |       1 |        3
      10 | Hrishi    |       2 |        4
      11 | Jasir     |       1 |        3
      12 | Yagnesh   |       2 |        4
      13 | Abraham   |       3 |        5
(13 rows)

univ_db=#
```

The state of the table is still the same as before.

solution.sql U

dbms-lab > labwork > assignment_6 > solution.sql > ...

```sql
BEGIN TRANSACTION;
▶ Run SQL
INSERT INTO students(stud_name, dept_id, tot_cred)
VALUES ('Saurabh', 1, 3),
    ('Shubham', 2, 4),
    ('Rupesh', 3, 5),
    ('Kundan', 4, 2),
    ('Sahil', 3, 5);
▶ Run SQL
SAVEPOINT first_savepoint;
▶ Run SQL
INSERT INTO students(stud_name, dept_id, tot_cred)
VALUES ('Mayank', 1, 3),
    ('Aditya', 2, 4),
    ('Kundan', 3, 5);
▶ Run SQL
ROLLBACK TO first_savepoint;
▶ Run SQL
INSERT INTO students(stud_name, dept_id, tot_cred)
VALUES ('Joel', 4, 2),
    ('Ishwar', 4, 2);
▶ Run SQL
COMMIT;
```

Inserting more data by creating savepoints and rolling back when inserted wrong data to the created savepoint.

```
univ_db=# \i solution.sql
BEGIN
INSERT 0 5
SAVEPOINT
INSERT 0 3
ROLLBACK
INSERT 0 2
COMMIT
```

```
univ_db=# SELECT * FROM students;
 stud_id | stud_name | dept_id | tot_cred
---------+-----------+---------+----------
       1 | Satyam    |       1 |        3
       2 | Amish     |       2 |        4
       3 | Neel      |       3 |        5
       4 | Aditya    |       4 |        2
       5 | Harsh     |       1 |        3
       6 | Mayank    |       2 |        4
       7 | Naren     |       3 |        5
       8 | Anurag    |       4 |        2
       9 | Jerry     |       1 |        3
      10 | Hrishi    |       2 |        4
      11 | Jasir     |       1 |        3
      12 | Yagnesh   |       2 |        4
      13 | Abraham   |       3 |        5
      17 | Saurabh   |       1 |        3
      18 | Shubham   |       2 |        4
      19 | Rupesh    |       3 |        5
      20 | Kundan    |       4 |        2
      21 | Sahil     |       3 |        5
      25 | Joel      |       4 |        2
      26 | Ishwar    |       4 |        2
(20 rows)
```

Q2. Why is it necessary to mention the COMMIT or ROLLBACK keyword followed by BEGIN keyword? If you do not use the COMMIT or ROLLBACK keyword followed by BEGIN, What error can be generated? Explain with example scenarios.

If we will not mention the COMMIT or ROLLBACK keyword followed by BEGIN keyword, then the transaction will still be running and holding locks for the tables, and when we will close the connection from the database, the transaction will be rolled back and then terminated. It can be thought of as a system failure that happens in between a transaction and the transaction gets rolled back.

Below, I am trying to insert a row within the transaction in the left session (can be identified by *# symbol), which adds a row with stud_id 27 into it. Now, in the second session, it will not see that effect and is paused right there itself because I am trying to insert a row with the same stud_id, which is completely valid for the second session. This can be problematic as the transaction is still holding the lock on the table and we are not able to proceed with the second session.

PS: The below snapshot of the table is after I have done the Python code question 4.

```
INSERT 0 1
univ_db=*# select * from students;
 stud_id | stud_name | dept_id | tot_cred
---------+-----------+---------+----------
       1 | Satyam    |       1 |        3
       3 | Neel      |       3 |        5
       4 | Aditya    |       4 |        2
       5 | Harsh     |       1 |        3
       7 | Naren     |       3 |        5
       8 | Anurag    |       4 |        2
       9 | Jerry     |       1 |        3
      11 | Jasir     |       1 |        3
      13 | Abraham   |       3 |        5
      17 | Saurabh   |       1 |        3
      19 | Rupesh    |       3 |        5
      20 | Kundan    |       4 |        2
      21 | Sahil     |       3 |        5
      25 | Joel      |       4 |        2
      26 | Ishwar    |       4 |        2
       2 | Amish     |       1 |        4
       6 | Mayank    |       1 |        4
      10 | Hrishi    |       1 |        4
      12 | Yagnesh   |       1 |        4
      18 | Shubham   |       1 |        4
      27 | Ram       |       3 |        4
(21 rows)

univ_db=*#
```

```
univ_db=# insert into students(stud_id, stud_name, dept_id, tot_c
red)
values (27, 'Ramu', 3, 4);
```

I got the below error for the above case.

```
ERROR:  canceling statement due to user request
CONTEXT:  while inserting index tuple (0,35) in relation "student
s_pkey"
```