DBMS Lab Week5

Name:Chandradhar Rao

SRN:PES1UG19CS123

Topic: SQL DML- Insert (using select) Update and delete, Transactions commit rollback and savepoint.

1. **SQL DML**
   - ➔ **Create a temporary table that has the employee last name, project name, and hours per week for each employee working on a project. Insert the values into the table using insert into with select command.**

```
company=# CREATE TEMP TABLE temp1_123 AS SELECT DISTINCT e.lname,p.pname,w.hours

FROM works_on w
INNER JOIN employee e ON (e.ssn=w.essn)
INNER JOIN project p on (p.pnumber=w.pno)
;
SELECT 15
company=# SELECT * FROM temp1_123;
  lname   |      pname      | hours
----------+-----------------+-------
 Narayan | ProductZ        |  40.0
 Wong    | ProductY        |  10.0
 Wallace | Reorganization  |  15.0
 Smith   | ProductX        |  32.5
 Smith   | ProductY        |   7.5
 Jabbar  | Newbenefits     |   5.0
 Jabbar  | Computerization |  35.0
 English | ProductY        |  20.0
 Wong    | Reorganization  |  10.0
 Wong    | ProductZ        |  10.0
 Zelaya  | Computerization |  10.0
 Zelaya  | Newbenefits     |  30.0
 Wallace | Newbenefits     |  20.0
 English | ProductX        |  20.0
 Wong    | Computerization |  10.0
(15 rows)
```

All tables are shown below,we can see that temp1_123 table exists as pg_temp_4 schema.

```
company=# \d
              List of relations
  Schema   |      Name       | Type  |  Owner
-----------+-----------------+-------+----------
 pg_temp_4 | temp1_123       | table | postgres
 public    | department      | table | postgres
 public    | dependent       | table | postgres
 public    | dept_locations  | table | postgres
 public    | employee        | table | postgres
 public    | project         | table | postgres
 public    | works_on        | table | postgres
(7 rows)
```

➔ **Update the location and controlling department number of project number 10 to ‘Bellaire’ and 5, respectively.**

```
company=# update project
company-# set dnum=5,plocation='Bellaire'
company-# where pnumber=10
company-# ;
UPDATE 1
```

```
company=# select * from project;
      pname       | pnumber | plocation | dnum
------------------+---------+-----------+------
 ProductX         |       1 | Bellaire  |    5
 ProductY         |       2 | Sugarland |    5
 ProductZ         |       3 | Houston   |    5
 Reorganization   |      20 | Houston   |    1
 Newbenefits      |      30 | Stafford  |    4
 Computerization  |      10 | Bellaire  |    5
(6 rows)
```

➔ **Give all employees in the 'Research' department a 10% raise in salary.**

```
company=# UPDATE employee AS e
SET salary=salary+(salary*0.1)
FROM department AS d
WHERE d.dnumber=e.dno
AND d.dname='Research'
;
UPDATE 4
```

```
company=# SELECT * from employee as e
company-# inner join department as d
company-# on e.dno=d.dnumber
company-# and d.dname='Research';
```

```
  fname   | minit |  lname  |    ssn    |   bdate    |         address
| gender |  salary  | super_ssn | dno |  dname   | dnumber |  mgr_ssn  | mgr_sta
rt_date
----------+-------+---------+-----------+------------+-------------------------
+--------+----------+-----------+-----+----------+---------+-----------+--------
--------
 John     | B     | Smith   | 123456789 | 1965-01-09 | 731 Fondren,Houston,TX
| M      | 33000.00 | 888665555 |   5 | Research |       5 | 333445555 | 1988-05
-22
 Franklin | T     | Wong    | 333445555 | 1955-12-08 | 638 voss,Houston,TX
| M      | 44000.00 | 888665555 |   5 | Research |       5 | 333445555 | 1988-05
-22
 Ramesh   | K     | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX
| M      | 41800.00 | 333445555 |   5 | Research |       5 | 333445555 | 1988-05
-22
 Joyce    | A     | English | 453453453 | 1972-07-31 | 5631 Rice,Houston,TX
| F      | 27500.00 | 333445555 |   5 | Research |       5 | 333445555 | 1988-05
-22
(4 rows)
```

➔ **Delete employee record whose lname ='Brown'**

```
company=# DELETE FROM employee as e
WHERE e.lname='Brown'
;
DELETE 0
```

➔ **Delete all the records of the employee who doesn't have dependent. (use sub query).**

When we try to delete,due to referential integrity,it doesnt allow us to delete primary key used as foreign key.

```
company=# delete from Employee e
company-# where e.ssn not in (
company(#     select essn from dependent
company(# );
ERROR:  update or delete on table "employee" violates foreign key constraint "em
ployee_super_ssn_fkey" on table "employee"
DETAIL:  Key (ssn)=(888665555) is still referenced from table "employee".
company=#
```

Hence we should first remove the existing foreign key not null constraints so that we can delete:

```
company=# alter table Department drop constraint "department_mgr_ssn_fkey";
ALTER TABLE
company=# alter table Employee drop constraint "employee_super_ssn_fkey";
ALTER TABLE
company=# alter table WORKS_ON drop constraint "works_on_essn_fkey";
ALTER TABLE
company=# alter table DEPENDENT drop constraint "dependent_essn_fkey";
ALTER TABLE
company=#
```

Also we need to allow not null value for manager_ssn in department:

```
company=# alter table department alter COLUMN mgr_ssn drop not null;
ALTER TABLE
```

Next the constrain should be updated with the one that allows null values for the foreign keys and deletes cascading for the works_on table since its used as a primary key in combination:

```
company=# alter table Department add constraint "department_mgr_ssn_fkey" foreig
n key (mgr_ssn) references employee(ssn) on delete set null;
ALTER TABLE
company=# alter table Employee add constraint "employee_super_ssn_fkey" foreign
key (super_ssn) references employee(ssn) on delete set null;
ALTER TABLE
company=# alter table DEPENDENT add constraint "dependent_essn_fkey" foreign key
 (essn) references employee(ssn) on delete set null;
ALTER TABLE
company=# alter table WORKS_ON add constraint "works_on_essn_fkey" foreign key (
essn) references employee(ssn) on delete cascade;
ALTER TABLE
company=#
```

Now upon deleting no error is raised as we have allowed null values in the foreign key constraint:

```
company=# delete from Employee e
where e.ssn not in (
    select essn from dependent
);
DELETE 5
```

2.Transactions

Create a transaction using begin and end commands consisting of the following sql statements.

- ➔ **create a transaction consisting of a create table and multiple insert statements. After End transaction the changes should be committed and can be checked using select statement.**

  First we start a block of TRANSACTIONAL code to be written using begin.
  Then we create the table a1_123 with an int column named col_1
  Next insert 4,5
  Finally commit this permanently to the database.
  Then I display the table as a proof that its pushed permannelty.

```
company=# BEGIN;
BEGIN
company=# CREATE TABLE a1_123(col_1 int);
CREATE TABLE
company=# INSERT INTO a1_123
company-# VALUES (1),(2)
company-# ;
INSERT 0 2
company=# INSERT INTO a1_123
company-# VALUES (4),(5)
company-# ;
INSERT 0 2
company=# COMMIT;
COMMIT
company=# END;
WARNING:  there is no transaction in progress
COMMIT
company=# SELECT * FROM a1_123;
 col_1
-------
     1
     2
     4
     5
(4 rows)
```

➔ **For the above transaction introduce a roll back after inserting 2 records. The create and insert should not be reflected in the database.**

First we start a block of TRANSACTIONAL code to be written using begin.
Then we create the table a1_123 with an int column named col_1
Then we insert values 1 and 2
Then we display the table.
Then we rollback.
The table wont be saved in the db as we rolled back without commit.

```
company=# BEGIN;
BEGIN
company=# CREATE TABLE a1_123(col_1 int);
CREATE TABLE
company=# INSERT INTO a1_123
VALUES (1),(2)
;
INSERT 0 2
company=# SELECT * FROM a1_123;
 col_1
-------
     1
     2
(2 rows)

company=# rollback;
ROLLBACK
company=# select * from a1_123;
ERROR:  relation "a1_123" does not exist
LINE 1: select * from a1_123;
                      ^
company=#
```

➔ **For the first transaction introduce a save point after inserting 2 records and insert 2 more records and rollback to savepoint . They database should reflect only first 2 insert.**

First we start a block of TRANSACTIONAL code to be written using begin.
Then we create the table a1_123 with an int column named col_1
Then we insert values 1 and 2
Then we make a savepoint called s_!
Next insert 3 and 4.
Then we display the table.
Then we rollback to the checkpoint.
Then I show the table.
Finally commit this permanently to the database.

```
company=# BEGIN;
BEGIN
company=# CREATE TABLE a1_123(col_1 int);
CREATE TABLE
company=# INSERT INTO a1_123
VALUES (1),(2)
;
INSERT 0 2
company=# savepoint s_1;
SAVEPOINT
company=# INSERT INTO a1_123
VALUES (3),(4)
;
INSERT 0 2
company=# select * from a1_123;
 col_1
-------
     1
     2
     3
     4
(4 rows)
```

```
company=# rollback to s_1;
ROLLBACK
company=# select * from a1_123;
 col_1
-------
     1
     2
(2 rows)

company=# commit;
COMMIT
company=# select * from a1_123;
 col_1
-------
     1
     2
(2 rows)
```