Experiment No.: 03

Title: To implement database for relational model in Experiment no. 2 using DDL statements.

Batch: B1 **Roll No.: 16010420133** **Experiment No.: 03**

Aim: To implement database for relational model in experiment no. 2 using DDL statements (Virtual Lab).

Resources needed: PostgreSQL PgAdmin3

Theory:

The Data Definition Language (DDL) is used to create and modify the relational schema. Also it is used to add various constraints to the table like the primary key, foreign key, check constraint, not null constraint and unique constraint.

The DDL statements are:

CREATE

DROP

ALTER

PostgreSQL supports the standard SQL types int, smallint, real, double precision, char(N), varchar(N), date, time, timestamp, and interval for creating tables.

Procedure:

Create Database and use it:

$ createdb mydb

$ psql mydb

Delete a database: $

dropdb mydb

Create table:

```
CREATE TABLE my_first_table (
first_column text,
second_column integer
);

CREATE TABLE products (
product_no integer,
name text, price numeric);
```

Drop Table:

```
DROP TABLE my_first_table;
DROP TABLE products;
```

Default Value:

```
CREATE TABLE products (
product_no integer,
name text,
price numeric DEFAULT 9.99 );
```

Constraints:

1. **Primary Key**

```
CREATE TABLE products (
product_no integer PRIMARY KEY,
name text,
price numeric );
```

Primary keys can also constrain more than one column.

```
CREATE TABLE example (
a integer,
b integer,
c integer,
PRIMARY KEY (a, c)
);
```

2. **Check Constraint**

```
CREATE TABLE products (
product_no integer,
name text,
price numeric CHECK (price > 0) );
```

3. **Not Null Constraint**

```
CREATE TABLE products (
product_no integer NOT NULL,
name text NOT NULL,
price numeric );
```

4. **Unique Constraint**

```
CREATE TABLE products (
product_no integer UNIQUE,
name text,
price numeric );
```

5. **Foreign Key Constarint**

```
CREATE TABLE products (
product_no integer PRIMARY KEY,
name text,
price numeric );

CREATE TABLE orders (
order_id integer PRIMARY KEY,
product_no integer REFERENCES products (product_no),
quantity integer );
```

Here a foreign key constraint in the order table references the products table.

Modifying table:

Adding column
ALTER TABLE products ADD COLUMN description text;

Removing column
ALTER TABLE products DROP COLUMN description;

Adding Constraint
ALTER TABLE products ADD CONSTRAINT some_name UNIQUE (product_no); ALTER TABLE products ADD FOREIGN KEY (product_group_id) REFERENCES product_groups;

Removing Constraint
ALTER TABLE products DROP CONSTRAINT some_name;

Adding Not Null Constraint
ALTER TABLE products ALTER COLUMN product_no SET NOT NULL;

Removing Not Null Constraint
ALTER TABLE products ALTER COLUMN product_no DROP NOT NULL;

Results: (Queries printout with output)

CONTRAINTS:

1. Primary key:

```
CREATE TABLE employee (
ELoginID text PRIMARY KEY,
Name text UNIQUE,
Password text
);
INSERT into employee values('22', 'Soumen','abcdef');
INSERT into employee values('23', 'Mihir','abcde');
INSERT into employee values('24', 'Deva','abcd');
INSERT into employee values('25', 'Dev','abc');
SELECT * from employee;
```

Output:

	eloginid [PK] text	name text	password text
1	22	Soumen	abcdef
2	23	Mihir	abcde
3	24	Deva	abcd
4	25	Dev	abc

2. Check constraint:

```
CREATE TABLE building (
BName text PRIMARY KEY,
Photos text NOT NULL,
no_of_flats integer CHECK (no_of_flats > 200)
);
INSERT into building values('Galaxy2', 'Link2', 216);
INSERT into building values('Galaxy3', 'Link3', 316);
INSERT into building values('Galaxy1', 'Link1', 116);
```

Output:

```
ERROR:  new row for relation "building" violates check constraint "building_no_of_flats_check"
DETAIL:  Failing row contains (Galaxy1, Link1, 116).
SQL state: 23514
```

---

3. Not NULL constraint:

```
CREATE TABLE building (
BName text PRIMARY KEY,
Photos text NOT NULL,
no_of_flats integer CHECK (no_of_flats > 200)
);
INSERT into building values('Galaxy1',NULL, 116);
```

Output:

Data Output | Explain | Messages | Notifications

```
ERROR:  null value in column "photos" violates not-null constraint
DETAIL:  Failing row contains (Galaxy1, null, 116).
SQL state: 23502
```

4. Unique constraint:

```
CREATE TABLE employee (
ELoginID text PRIMARY KEY,
Name text UNIQUE,
Password text
);
INSERT into employee values('22', 'Soumen',’abcdef');
INSERT into employee values('23', 'Mihir','abcde');
INSERT into employee values('24', 'Deva','abcd');
INSERT into employee values('25', 'Dev','abc');
INSERT into employee values('21', 'Soumen','abcdefg');
```

Output:

```
ERROR:  duplicate key value violates unique constraint "employee_name_key"
DETAIL:  Key (name)=(Soumen) already exists.
SQL state: 23505
```

5. Foreign key constraint:

```
CREATE TABLE employee (
ELoginID text PRIMARY KEY,
Name text UNIQUE,
Password text
);
INSERT into employee values('22', 'Soumen','abcdef');
INSERT into employee values('23', 'Mihir','abcde');
INSERT into employee values('24', 'Deva','abcd');
INSERT into employee values('25', 'Dev','abc');
CREATE TABLE building (
BName text PRIMARY KEY,
Photos text NOT NULL,
no_of_flats integer CHECK (no_of_flats > 200)
);
INSERT into building values('Galaxy2', 'Link2', 216);
INSERT into building values('Galaxy3', 'Link3', 316);
CREATE TABLE builder (
BLoginID text PRIMARY KEY,
Name text,
Password text,
E_LoginID text REFERENCES employee (ELoginID),
B_name text REFERENCES building (Bname)
);
INSERT into builder values('11', 'Kevin','abcdefg', '20');
```

Output:

Data Output | Explain | Messages | Notifications

```
ERROR:  insert or update on table "builder" violates foreign key constraint "builder_e_loginid_fkey"
DETAIL:  Key (e_loginid)=(20) is not present in table "employee".
SQL state: 23503
```

MODIFYING TABLE:

1. Adding column:

```
CREATE TABLE employee (
ELoginID text PRIMARY KEY,
Name text UNIQUE,
Password text
);
INSERT into employee values('22', 'Soumen','abcdef');
INSERT into employee values('23', 'Mihir','abcde');
INSERT into employee values('24', 'Deva','abcd');
INSERT into employee values('25', 'Dev','abc');
ALTER TABLE employee ADD COLUMN Gender text;
SELECT * from employee;
```

Output:

	eloginid [PK] text	name text	password text	gender text
1	22	Soumen	abcdef	[null]
2	23	Mihir	abcde	[null]
3	24	Deva	abcd	[null]
4	25	Dev	abc	[null]

2. Removing column:

```
ALTER TABLE employee DROP COLUMN Gender;
SELECT * from employee;
```

Output:

	eloginid [PK] text	name text	password text
1	22	Soumen	abcdef
2	23	Mihir	abcde
3	24	Deva	abcd
4	25	Dev	abc

3. Adding constraint:

```
CREATE TABLE employee (
ELoginID text PRIMARY KEY,
Name text UNIQUE,
Password text
);
INSERT into employee values('22', 'Soumen','abcdef');
INSERT into employee values('23', 'Mihir','abcde');
INSERT into employee values('24', 'Deva','abcd');
INSERT into employee values('25', 'Dev','abc');
ALTER TABLE employee ADD CONSTRAINT password_should_be_unique UNIQUE
(Password);
INSERT into employee values('26', 'Kevin','abcdef');
```

Output:

```
ERROR:  duplicate key value violates unique constraint "password_should_be_unique"
DETAIL:  Key (password)=(abcdef) already exists.
SQL state: 23505
```

4. Adding Not Null constraint:

```
CREATE TABLE employee (
ELoginID text PRIMARY KEY,
Name text UNIQUE,
Password text
);
ALTER TABLE employee ALTER COLUMN Password SET NOT NULL;
INSERT into employee values('26', 'Kevin');
```

Output:

```
ERROR:  null value in column "password" violates not-null constraint
DETAIL:  Failing row contains (26, Kevin, null).
SQL state: 23502
```

5. Removing Not Null constraint:

```
ALTER TABLE employee ALTER COLUMN Password DROP NOT NULL;
INSERT into employee values('26', 'Kevin');
SELECT * from employee;
```

Output:

	eloginid [PK] text	name text	password text
1	22	Soumen	abcdef
2	23	Mihir	abcde
3	24	Deva	abcd
4	25	Dev	abc
5	26	Kevin	[null]

Questions:

Q1) What is difference between Truncate, Drop and delete? Explain with example

Ans)

The **DELETE** command is used to remove some or all rows from a table. A WHERE clause can be used to only remove some rows. If no WHERE condition is specified, all rows will be removed.

TRUNCATE removes all rows from a table. The operation cannot be rolled back and no triggers will be fired. As such, TRUNCATE is faster and doesn't use as much undo space as a DELETE.

The **DROP** command removes a table from the database. All the tables' rows, indexes and privileges will also be removed. The operation cannot be rolled back.

Initially:-

id	name	price
1	milk	2.40
2	bread	3.68
3	butter	5.55
4	sugar	2.88

```
DELETE FROM product WHERE price<2.90;
```

id	name	price
2	bread	3.68
3	butter	5.55

```
TRUNCATE TABLE product;
```

deletes all records stored in the table `product`

```
DROP TABLE product;
```

removes all data in the table `product` and the structure of the table.

Outcomes:

CO3: Illustrate the concept of security, Query processing, indexing and Normalization for Relational database.

Conclusion:

This experiment helps us convert relational model to tables in sql using postgresql software. All the commands given in the write up have been executed, attached a screenshot and thus we have a better understanding of these commands.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

Reference books:

1. Elmasri and Navathe, “Fundamentals of Database Systems”, 6th Edition, Pearson

Education
2. Korth, Slberchatz,Sudarshan, :"Database System Concepts", 6th
3. Edition, McGraw – Hill.

WebSite:

1. http://www.tutorialspoint.com/postgresql/
2. http://sage.virtual-labs.ac.in/home/pub/21/