



**POLYTECHNIQUE
MONTRÉAL**

**UNIVERSITÉ
D'INGÉNIERIE**

INF3710 –Bases de données

Automne 2023

TP No. 1

**Groupe vendredi B2
2174916– Arman Lidder
2183376 – Hamza Boukaftane**

Soumis à : Joe Abdo

16 septembre 2023

1. Création d'une base de données

- 1.1. Afin de créer la base de données nommée «BD_TP1» avec une requête SQL, nous avons utilisé la commande «CREATE DATABASE BD_TP1 ;».

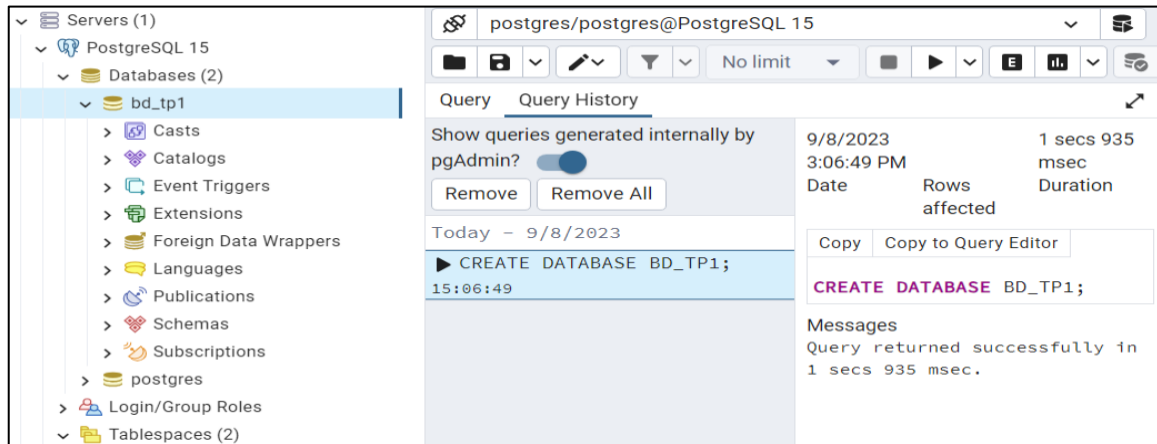


Figure 1. Commande de création de base de données.

- 1.2. Afin de créer une table nommée «ProduitCommande» avec une requête SQL, il faut utiliser la syntaxe suivante :

```
CREATE TABLE ProduitCommande (  
    NumeroProduit INT,  
    NumeroCommande INT,  
    QuantiteCommande INT,  
    PRIMARY KEY(NumeroProduit, NumeroCommande),  
    FOREIGN KEY(NumeroProduit) REFERENCES Produit(idProduit),  
    FOREIGN KEY(NumeroCommande) REFERENCES Commande(idCommande)  
);
```

2. Création des tables

- 2.1. En se basant sur le script dans le fichier «BD-TP1-schema», l'ordre d'exécution que nous avons choisi est le suivant :

1. Utilisateur
2. Adresse
3. Commande
4. Produit
5. ProduitCommande

En effet, la table Utilisateur et la table Produit ne possèdent aucune dépendance envers les autres tables. Ensuite, comme la table Adresse et la table Commande possèdent une clé étrangère (idUtilisateur) qui fait référence à la table Utilisateur, elles dépendent alors de la table Utilisateur. Ainsi, la table Utilisateur doit être exécuté avant les tables Adresse et Commande. De plus, il est important de noter que la table ProduitCommande est dépendante des tables Produit et Commande. Ainsi, elle doit être exécutée après l'exécution des table Commande et Produit. Enfin, il peut y a voir plusieurs ordres d'exécution différentes, mais la table Utilisateur doit être exécuté avant les tables Adresse et Commande et la table ProduitCommande doit être exécuté après les tables Commande et Produit.

The screenshot displays a database management interface. On the left, a tree view shows the database structure, with 'Tables (5)' expanded, listing 'adresse', 'commande', 'produit', 'produitcommande', and 'utilisateur'. On the right, a 'Query' pane shows the SQL code for creating these tables. The queries are as follows:

```

1 CREATE TABLE Utilisateur (
2   idUtilisateur INT PRIMARY KEY,
3   nomUtilisateur VARCHAR(255),
4   prenomUtilisateur VARCHAR(255),
5   courrielUtilisateur VARCHAR(255)
6 );
7
8 CREATE TABLE Adresse (
9   idAdresse INT PRIMARY KEY,
10  idUtilisateur INT,
11  numeroCivic VARCHAR(255),
12  ville VARCHAR(255),
13  province VARCHAR(255),
14  codePostal VARCHAR(20),
15  FOREIGN KEY (idUtilisateur) REFERENCES Utilisateur(idUtilisateur)
16 );
17
18 CREATE TABLE Commande (
19  idCommande INT PRIMARY KEY,
20  idUtilisateur INT,
21  dateCommande DATE,
22  totalCommande DECIMAL(10, 2),
23  FOREIGN KEY (idUtilisateur) REFERENCES Utilisateur(idUtilisateur)
24 );
25
26 CREATE TABLE Produit (
27  idProduit INT PRIMARY KEY,
28  nomProduit VARCHAR(255),
29  prixProduit DECIMAL(10, 2)
30 );
31
32 CREATE TABLE ProduitCommande (
33  NumeroProduit INT,
34  NumeroCommande INT,
35  QuantiteCommande INT,
36  PRIMARY KEY(NumeroProduit, NumeroCommande),
37  FOREIGN KEY(NumeroProduit) REFERENCES Produit(idProduit),
38  FOREIGN KEY(NumeroCommande) REFERENCES Commande(idCommande)
39 );

```

At the bottom of the right pane, a status bar indicates: 'Query returned successfully in 42 msec.'

Figure 2. Création des 5 tables avec l'ordre spécifiée.

2.2. Si l'on essaie de créer une deuxième fois la table Utilisateur avec la commande :

```
CREATE TABLE Utilisateur (  
    idUtilisateur INT PRIMARY KEY,  
    nomUtilisateur VARCHAR(255),  
    prenomUtilisateur VARCHAR(255),  
    courrielUtilisateur VARCHAR(255)  
);
```

On obtient une erreur, car une table avec la même structure existe déjà. Ainsi, par soucis de cohérence et pour éviter la redondance, le système de gestion de base de données soulève une erreur. En effet, PostgreSQL ne permet pas la création de table avec le même nom dans le même schéma. Dans ce type de situation, le mot clé manquant est «IF NOT EXISTS», car il permet de vérifier l'existence de la table avant de la créer. Par conséquent, si la table existe déjà, aucune action sera exécutée, sinon, la table sera créée.

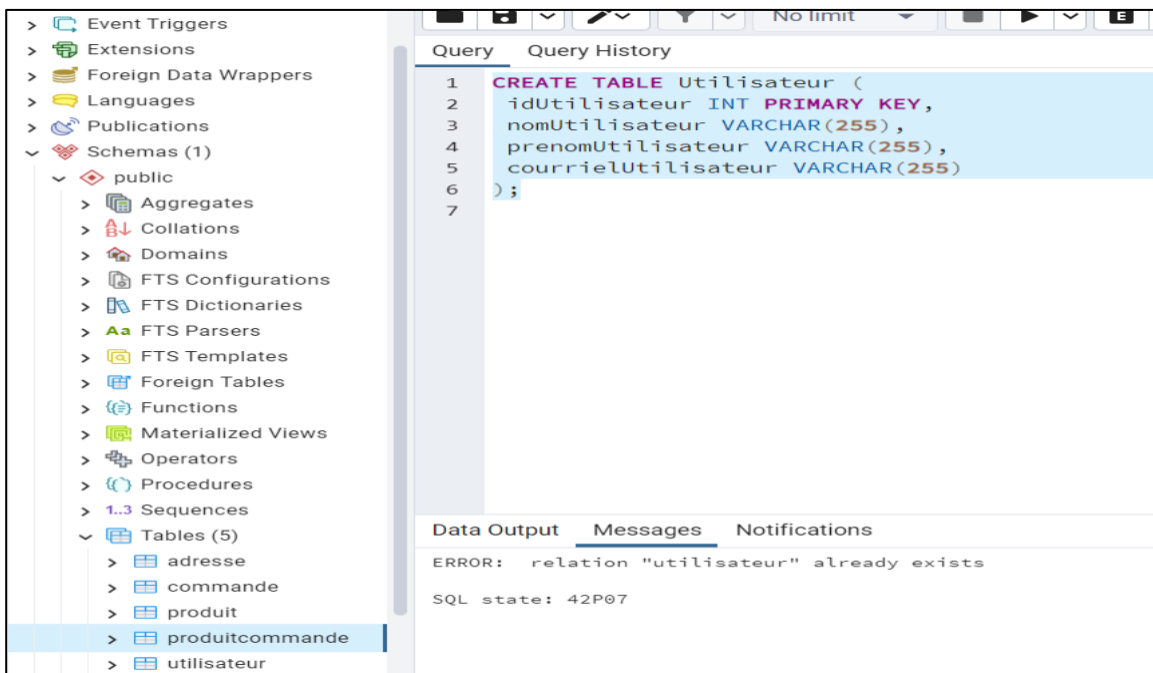


Figure 3. Erreur de création de table.

3. Peuplement des tables

3.1. Voici les requêtes SQL de peuplement de la «BD_TP1 » dans l'ordre :

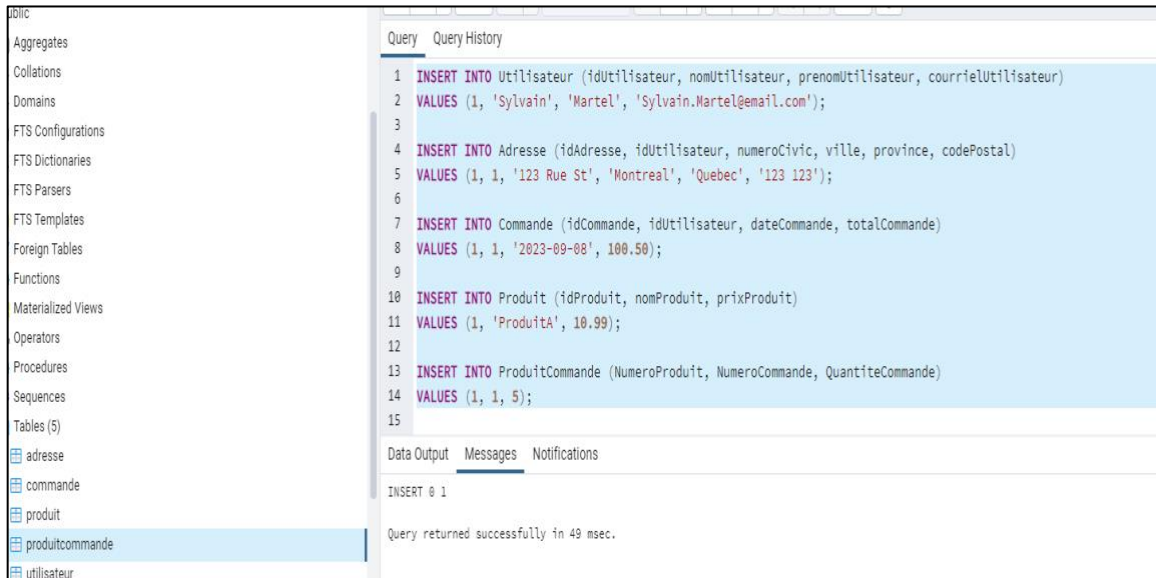


Figure 4. Ordre de peuplement de la BD_TP1.

3.2. La commande « SELECT * FROM Utilisateur; » est utilisée pour consulter et afficher l'ensemble des données contenues dans la table "Utilisateur". Ainsi, elle permet de visualiser la totalité du contenu de la table Utilisateur et effectuer des opérations de lecture sur les données.

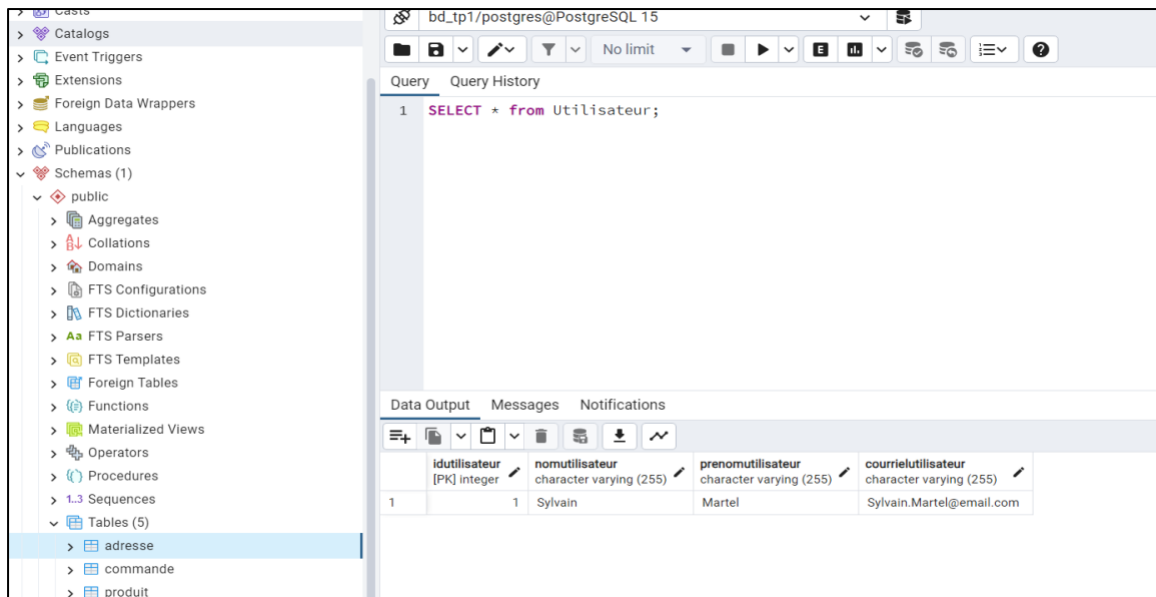
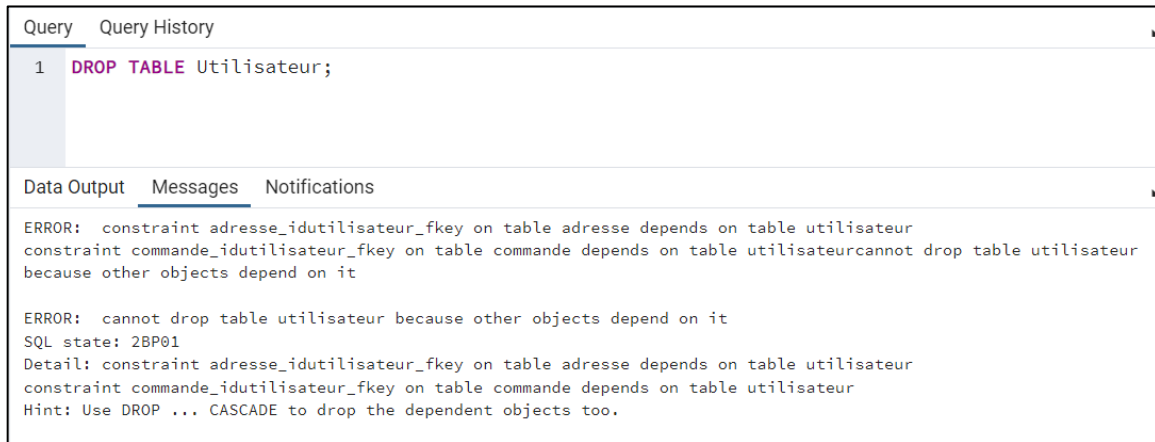


Figure 5. Commande SELECT * FROM Utilisateur.

3.3. Pour détruire une table, il faut faire `DROP TABLE "NOM"`. Cependant, la table Utilisateur a des tables dépendantes : Commande et Produit. Donc cette commande va donner une erreur :



```
Query  Query History
1 DROP TABLE Utilisateur;

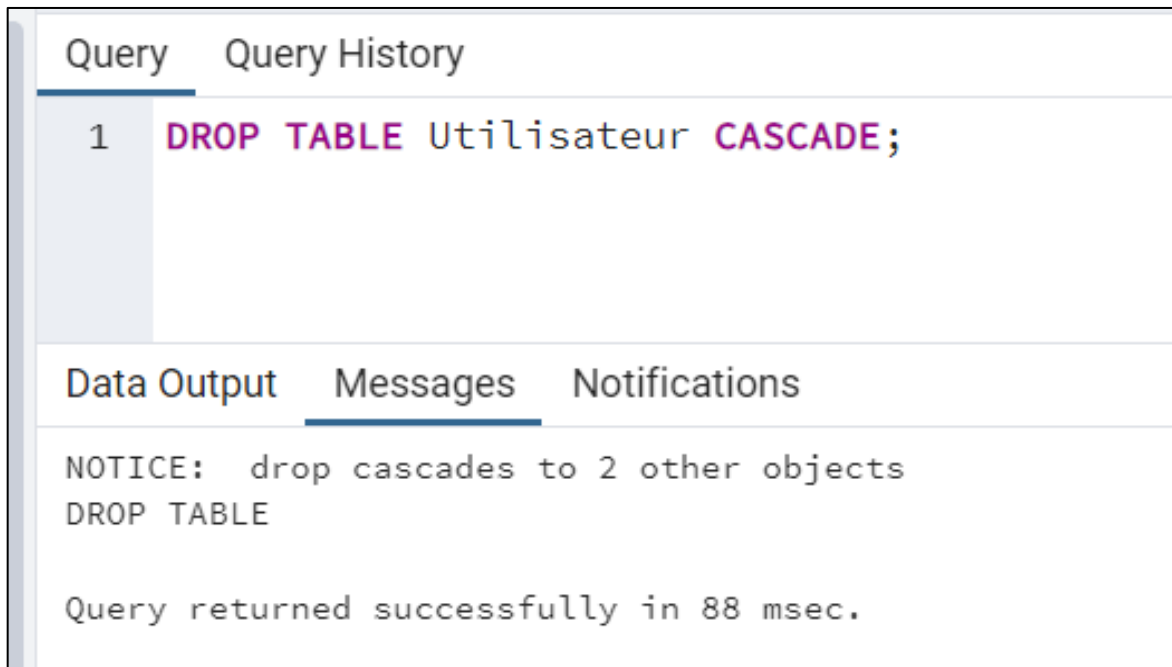
Data Output  Messages  Notifications

ERROR:  constraint adresse_idutilisateur_fkey on table adresse depends on table utilisateur
constraint commande_idutilisateur_fkey on table commande depends on table utilisateurcannot drop table utilisateur
because other objects depend on it

ERROR:  cannot drop table utilisateur because other objects depend on it
SQL state: 2BP01
Detail: constraint adresse_idutilisateur_fkey on table adresse depends on table utilisateur
constraint commande_idutilisateur_fkey on table commande depends on table utilisateur
Hint: Use DROP ... CASCADE to drop the dependent objects too.
```

Figure 6.Erreur quand `DROP TABLE Utilisateur`.

Donc, il faudra utiliser plutôt « `DROP TABLE Utilisateur CASCADE` », ce qui détruit la table, mais on conservera encore les tables qui étaient dépendantes :



```
Query  Query History
1 DROP TABLE Utilisateur CASCADE;

Data Output  Messages  Notifications

NOTICE:  drop cascades to 2 other objects
DROP TABLE

Query returned successfully in 88 msec.
```

Figure 7. Requête `DROP TABLE Utilisateur CASCADE`.

Query

Query History

1

SELECT * FROM Adresse;

Data Output

Messages

Notifications

	idadresse [PK] integer	idutilisateur integer	numerocivic character varying (255)	ville character varying (255)	province character varying (255)	codepostal character varying (20)
1	1	1	123 St	Laval	Quebec	H7W 1N9

Query

Query History

1

SELECT * FROM Commande;

Data Output

Messages

Notifications

	idcommande [PK] integer	idutilisateur integer	datecommande date	totalcommande numeric (10,2)
1	1	1	2023-09-08	100.00

Figure 8. Tables Adresse et Commande après suppression Utilisateur.

Pour détruire ProduitCommande, il suffit d'utiliser « DROP TABLE ProduitCommande », puisque les autres tables ne sont pas dépendantes de la table ProduitCommande:

Query	Query History	
1	DROP TABLE ProduitCommande;	
Data Output	Messages	Notifications
DROP TABLE		
Query returned successfully in 81 msec.		

Figure 9. Requête DROP TABLE ProduitCommande.