Université Gaston Berger
UFR SAT
Séction Mathématiques appliquées

Projet tutoré Base de Données

26 Mai 2021

Membres : Seyni KANE
Ramatoulaye DIALLO

Table de matiere

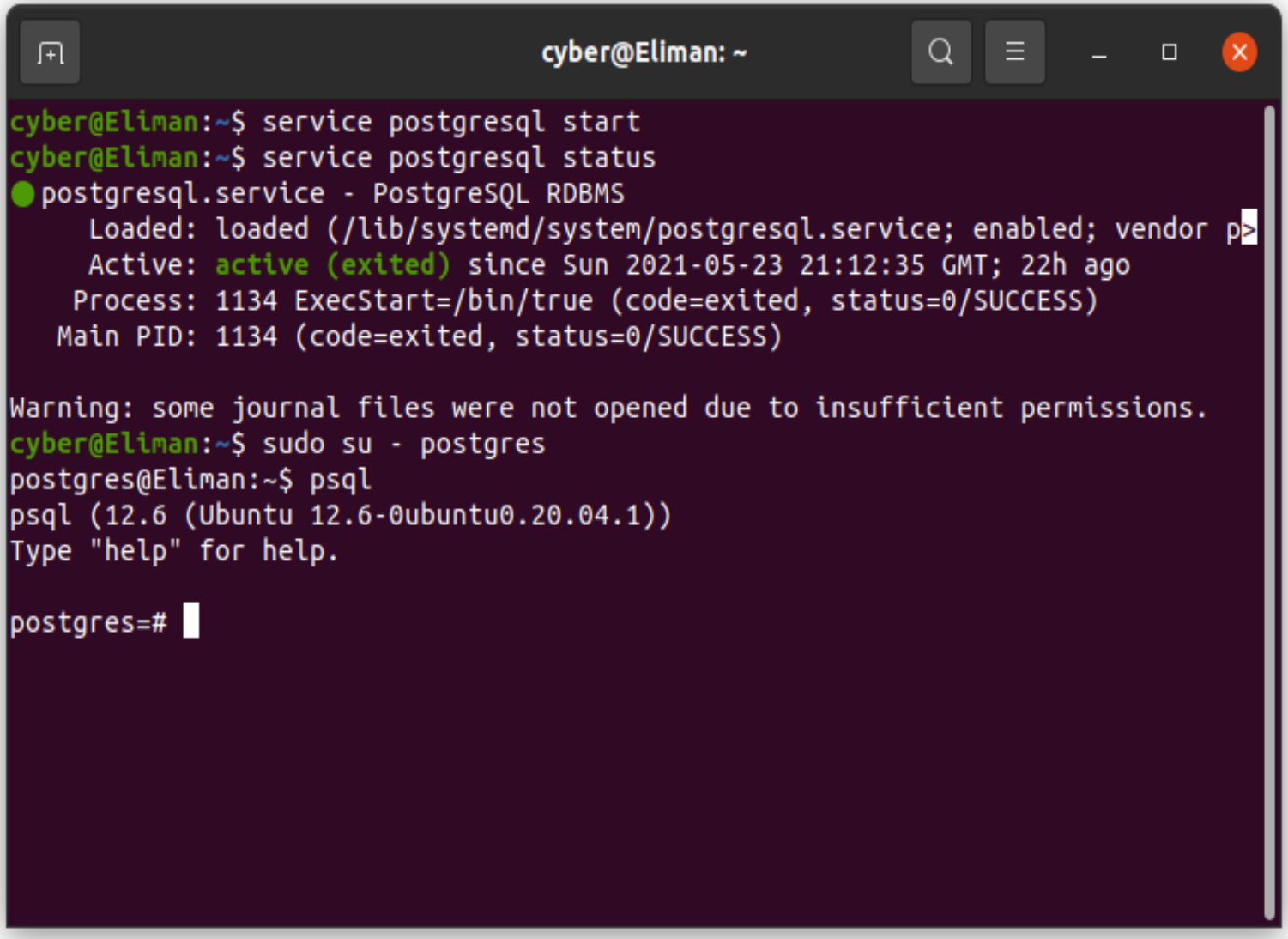
Introduction

En informatique, une base de données relationnelle est une base de données òu l'information est organisée dans des tableaux à deux dimensions appelés des relations ou tables.
Ici nous allons essayer de traiter l'exercice qui nous ait proposer.

Démarrage de postgres

Voici les differents étapes suivit depuis le demarrage du serveur postgres *à* la connection *à* la base de données

- On demmarre le serveur postgres avec la commade
  $ service postgresql start
- On teste si le serveur postgres a bien demarrer avec la commade
  $ service postgresql status
- On se connect a l'utilisateur postgress avec la commande
  $ sudo su - postgres
- On accéde au au shell psql avec la commande
  $ psql


cyber@Eliman: ~
cyber@Eliman:~$ service postgresql start
cyber@Eliman:~$ service postgresql status
postgresql.service - PostgreSQL RDBMS
Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor p
Active: active (exited) since Sun 2021-05-23 21:12:35 GMT; 22h ago
Process: 1134 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
Main PID: 1134 (code=exited, status=0/SUCCESS)
Warning: some journal files were not opened due to insufficient permissions.
cyber@Eliman:~$ sudo su - postgres
postgres@Eliman:~$ psql
psql (12.6 (Ubuntu 12.6-0ubuntu0.20.04.1))
Type "help" for help.
postgres=#

Création de la base de données

Voici les differents étapes suivit pour la création de notre base de données *projet_base_donnees*

- On s'assure de bien supprimer la base si el existe déja avec l'istruction
  # DROP DATABASE IF EXISTS projet_base_donnees ;
- On crée notre base avec l'instruction
  # CREATE DATABASE projet_base_donnees ;

Création de notre utilisateur

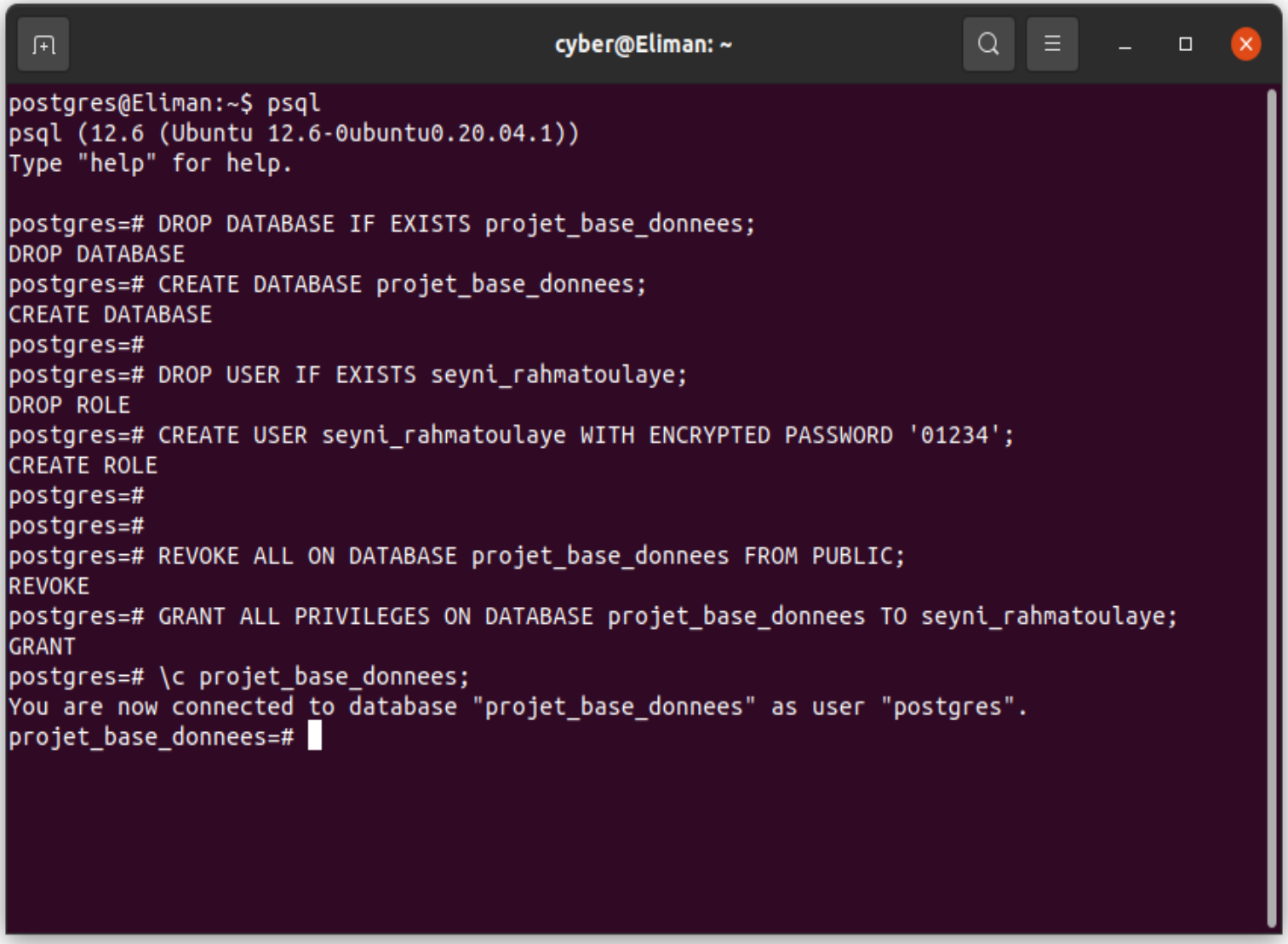
Voici les differents étapes suivit pour la création de notre utilisateur *seyni_rahmatoulaye*

- On s'assure de bien supprimer l'utilisateur si el existe déja avec l'istruction
  # DROP USER IF EXISTS seyni_rahmatoulaye ;
- On crée notre utilisateur avec le mot de passe *'01234'*, comme suit
  # CREATE USER seyni_rahmatoulaye WITH ENCRYPTED PASSWORD '01234' ;
  La capture ci-dessus est une illustration du resultat aprés exécution

Ajout des droits et connection à la base de données

- On s'assure de bien révoquer les droits
  # REVOKE ALL ON DATABASE projet_base_donnees FROM PUBLIC ;
- On donne tous les droit à notre utilisateur sur notre base de données, avec l'instruction
  # GRANT ALL PRIVILEGES ON DATABASE projet_base_donnees TO seyni_rahmatoulaye ;
- On se connect a notre base avec l'instruction
  # \c projet_base_donnees ;

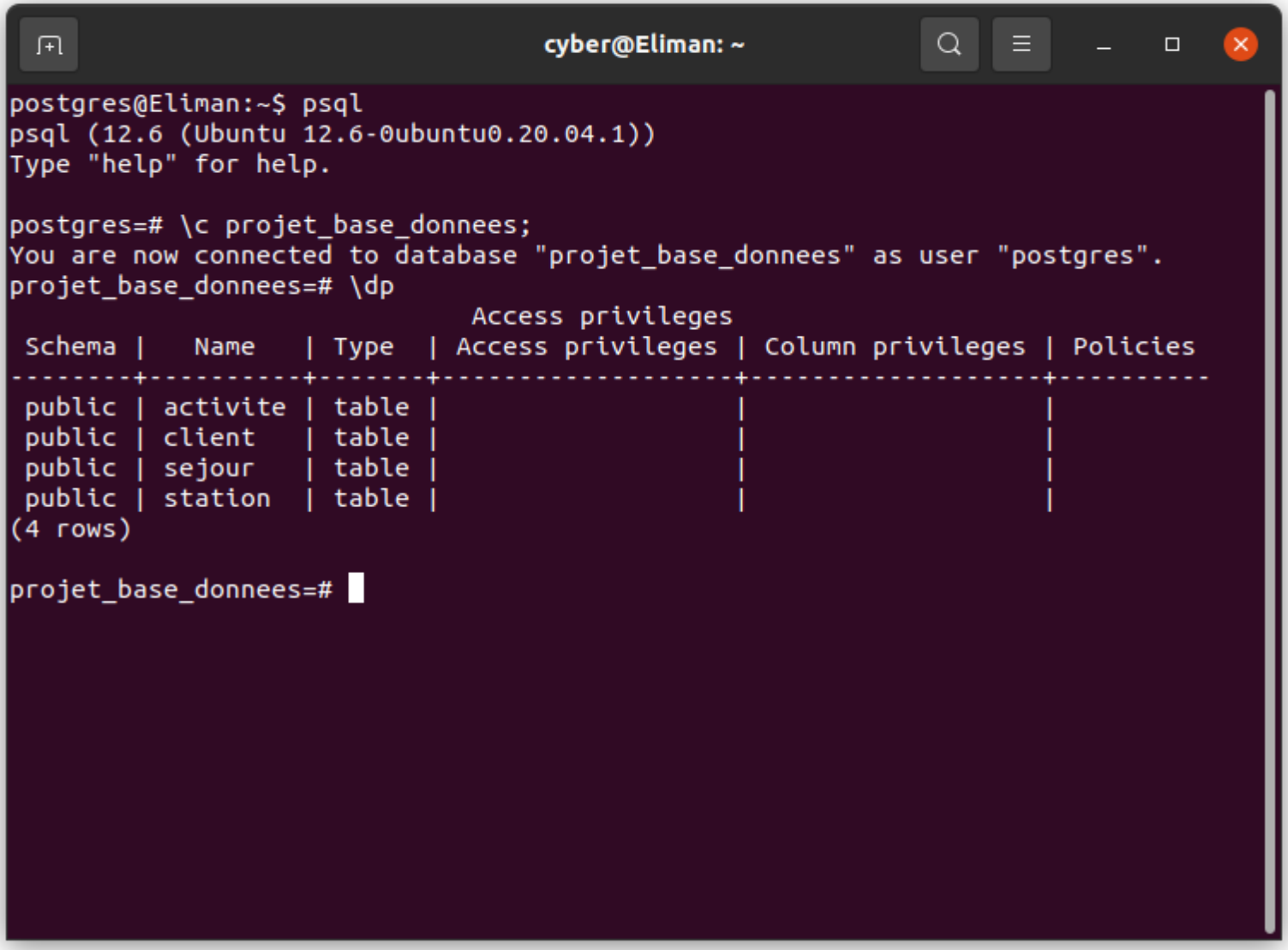
La capture ci-dessous est une illustration de tous ce qu'on vient de developper aprés exécution


cyber@Eliman: ~
postgres@Eliman:~$ psql
psql (12.6 (Ubuntu 12.6-0ubuntu0.20.04.1))
Type "help" for help.
postgres=# DROP DATABASE IF EXISTS projet_base_donnees;
DROP DATABASE
postgres=# CREATE DATABASE projet_base_donnees;
CREATE DATABASE
postgres=#
postgres=# DROP USER IF EXISTS seyni_rahmatoulaye;
DROP ROLE
postgres=# CREATE USER seyni_rahmatoulaye WITH ENCRYPTED PASSWORD '01234';
CREATE ROLE
postgres=#
postgres=#
postgres=# REVOKE ALL ON DATABASE projet_base_donnees FROM PUBLIC;
REVOKE
postgres=# GRANT ALL PRIVILEGES ON DATABASE projet_base_donnees TO seyni_rahmatoulaye;
GRANT
postgres=# \c projet_base_donnees;
You are now connected to database "projet_base_donnees" as user "postgres".
projet_base_donnees=#

Dans cette partie nous allons procédé a la création des table et des contraintes d'integritées

Création des tables

Pour le code de la création des tables et des contraintes voir le script ci-joint. Ce pendant ici nous allons lister les table que nous venons de créer avec la commande :
$\backslash dn$
La capture ci-dessous est une ullistration de ce que nous venons de développer.


cyber@Eliman: ~
postgres@Eliman:~$ psql
psql (12.6 (Ubuntu 12.6-0ubuntu0.20.04.1))
Type "help" for help.
postgres=# \c projet_base_donnees;
You are now connected to database "projet_base_donnees" as user "postgres".
projet_base_donnees=# \dp
Access privileges
Schema | Name | Type | Access privileges | Column privileges | Policies
public | activite | table |
public | client | table |
public | sejour | table |
public | station | table |
(4 rows)
projet_base_donnees=#

Reponses

1. Pour les données qui doivent toujours être connut on utilise la contrainte *NOT NULL* comme suit :
   Exemple : *capacite INTEGER NOT NULL*
2. Pour les motants qui ont une valeur par defaut on utilise la clause *DEFAULT* comme suit :
   Exemple : *tarif INTEGER DEFAULT 0*
3. Ici on utilise la contrainte *UNIQUE* sur les attributs *lieu et region* dans la table *Station* comme suit :
   CONSTRAINT Unique_station UNIQUE (lieu, region)
4. Ici on utilise la contrainte *CHECK* sur l'attributs *region* dans la table *Station* comme suit :
   CHECK (region IN ('Ocean Indien','Antille','Europe','Amerique', 'Extreme Orient'))

Reponses

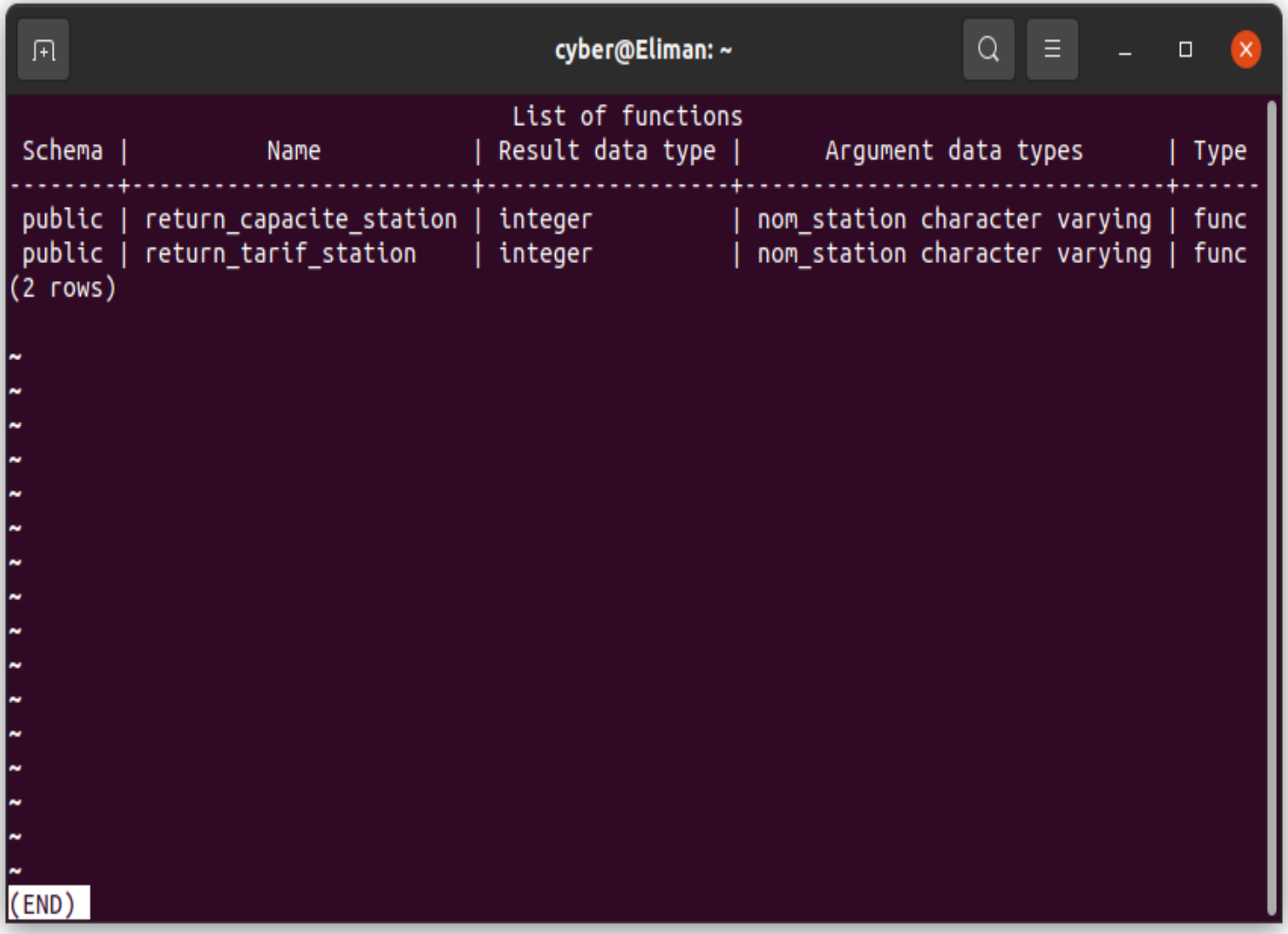
5 Ici on repercute la destruction d'une *Station* avec *ON DELETE CASCADE* sur la clef secondaire *nom_station* dans la table *Activite* et *station* dans la table *Sejour* comme suit :
Exemple : *CONSTRAINT FK_activite FOREIGN KEY (nom_station) REFERENCES Station(nom_station) ON DELETE CASCADE*

6 D'abord on a crée une fonction intituler *return_tarif_station* qui prend en entrée le nom de la station et retourn le *tarif* de la station.
Voici la signature de la fonction :
CREATE OR REPLACE FUNCTION return_tarif_station(station VARCHAR (50)) RETURNS INTEGER AS.
Ensuite on utilise la contrainte *CHECK* avec la fonction *return_tarif_station* sur l'attributs *prix* dans la table *Ativite* comme suit :
CONSTRAINT CH_prix CHECK (prix > 0 AND prix < return_tarif_station(nom_station))

Reponses

7 Ici comme dans la question *6* on utilise la contrainte *CHECK* avec la fonction *return_capcite_station* que nous avons crée sur l'attributs *debut* dans la table *Sejour* comme suit :
CONSTRAINT CH_prix CHECK (nb_place < return_capacite_station(station))

NB : Pour plus de details sur les fonctions *return_tarif_station* et *return_capcite_station* voir le script ci-joint.
La capture ci-dessous liste les fonctions presentent dans notre base de données.


cyber@Eliman: ~
List of functions
Schema | Name | Result data type | Argument data types | Type
public | return_capacite_station | integer | nom_station character varying | func
public | return_tarif_station | integer | nom_station character varying | func
(2 rows)
(END)

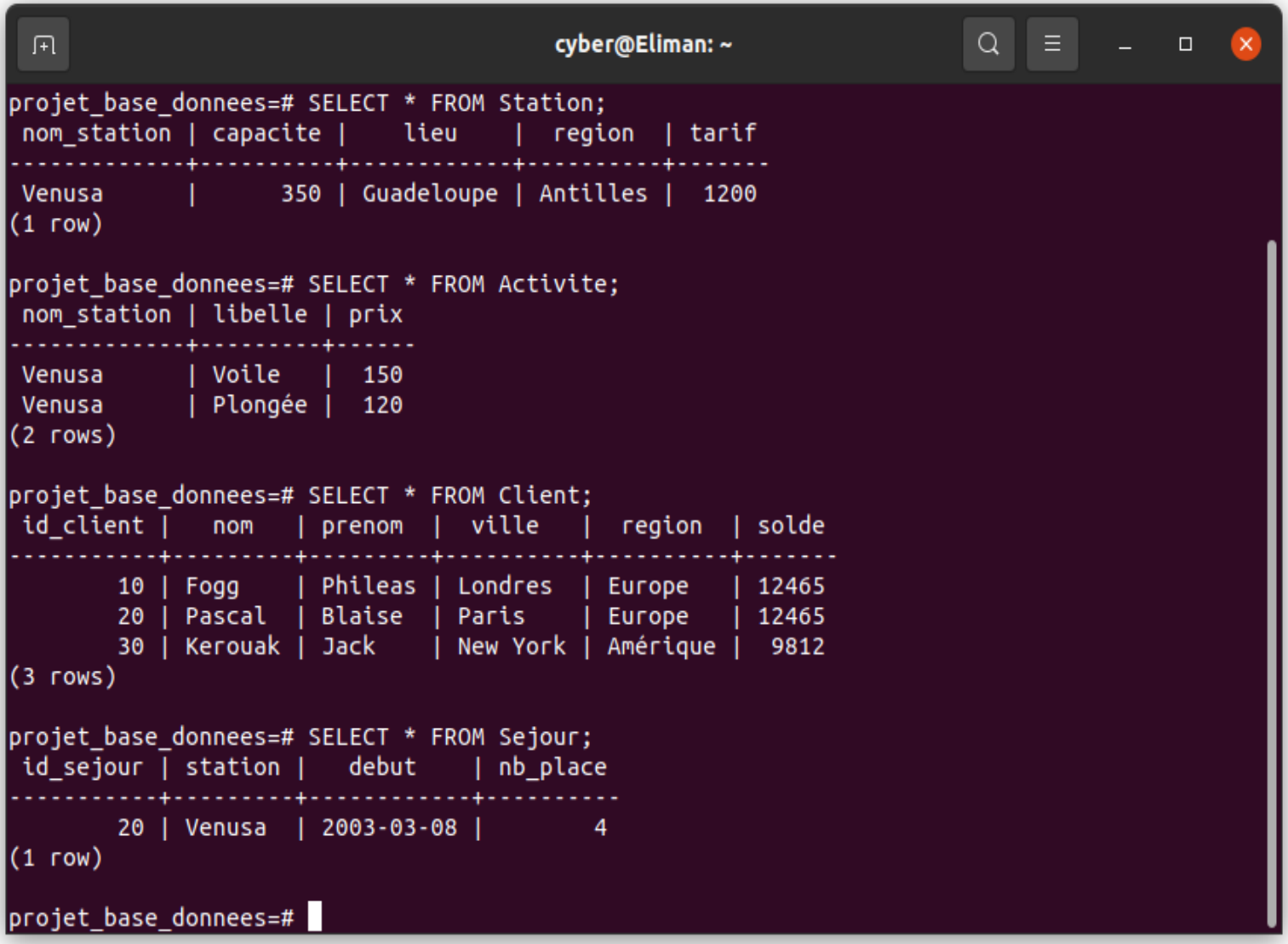
Dans cette partie nous allons proceder a l'insertion des données dans nos table. Mais d'abord nous allons nous assurer que nos tables ne contiennent aucune données.

Suppression des données

On vide les table avec l'instruction *DELETE FROM*
Exemple : *DELETE FROM Station ;* pour la table *Station*. C'est a partire de la qu'on commence l'insertion.

Insertion des données

L'insertion se fait avec l'instruction *INSERT INTO ... VALUES*
Exemple : *INSERT INTO Station (nom_station, capacite, lieu, region, tarif) VALUES ('Venusa', 350, 'Guadeloupe', 'Antilles', 1200) ;* pour l'insertion de la premiere ligne de la table *Station*.
La capture ci dessus est une ullistration des tables aprés insertion ds données.


cyber@Eliman: ~
projet_base_donnees=# SELECT * FROM Station;
 nom_station | capacite | lieu | region | tarif
-------------+----------+------------+----------+-------
 Venusa | 350 | Guadeloupe | Antilles | 1200
(1 row)
projet_base_donnees=# SELECT * FROM Activite;
 nom_station | libelle | prix
-------------+---------+------
 Venusa | Voile | 150
 Venusa | Plongée | 120
(2 rows)
projet_base_donnees=# SELECT * FROM Client;
 id_client | nom | prenom | ville | region | solde
-----------+---------+---------+----------+----------+-------
        10 | Fogg | Phileas | Londres | Europe | 12465
        20 | Pascal | Blaise | Paris | Europe | 12465
        30 | Kerouak | Jack | New York | Amérique | 9812
(3 rows)
projet_base_donnees=# SELECT * FROM Sejour;
 id_sejour | station | debut | nb_place
-----------+---------+------------+----------
        20 | Venusa | 2003-03-08 | 4
(1 row)
projet_base_donnees=#

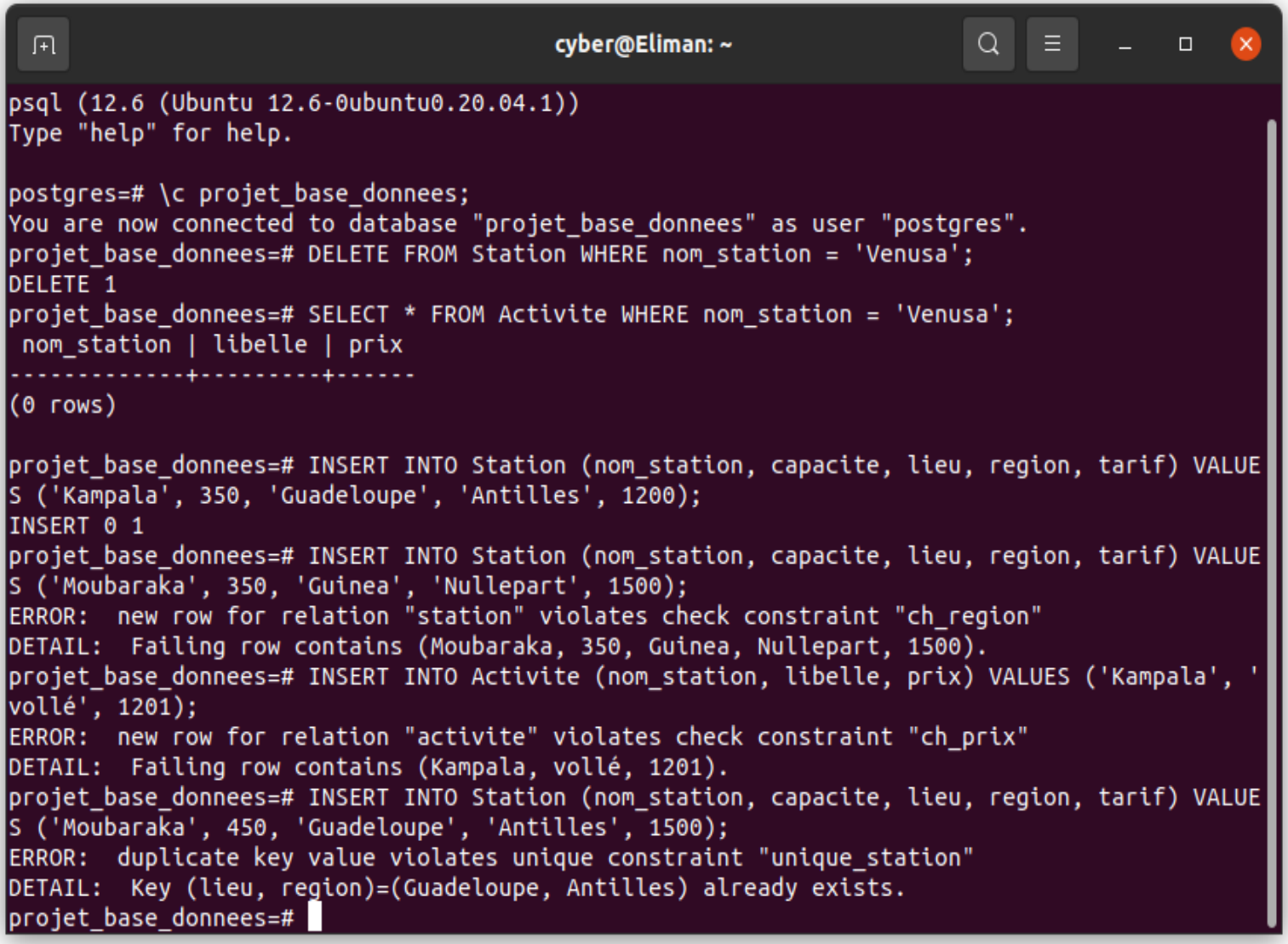
Dans cette section nous allons tester quelques unes de nos contraintes.

Tests

- On detruit la station *Venusa* avec la commande :
  DELETE FROM Station WHERE nom_station = 'Venusa';
- Verification sur la table *Activite* avec la commande :
  *SELECT * FROM Activite WHERE nom_station = 'Venusa';*
  On constate en effet que les activiées que nous avons entrées on disparuts (voir capture ci-dessous).
- Insertion d'une station dans une région *'Nullepart'* avec la commande :
  INSERT INTO Station (nom_station, capacite, lieu, region, tarif) VALUES ('Moubaraka', 350, 'Guinea', 'Nullepart', 1500);
  On constate que l'insertion ne s'est pas bien derouler car la contrainte *CH_region* a été violer (voir capture ci-dessous). En effet *'Nullepart'* ne fait pas partie des regions autorisées.

Tests

- Test de la contrainte *CH_prix* par insertion d'une nouvelle activité ayant un prix superieur au tarif de la station avec la commande :
  INSERT INTO Activite (nom_station, libelle, prix) VALUES ('Kampala', 'vollé', 1201) ;
  On constate que l'insertion ne s'est pas bien derouler car la contrainte *CH_prix* a été violer (voir capture ci-dessous). En effet ici le prix est superieur au tarif de la station.
- Test de la contrainte *Unique_station* insertion d'une nouvelle station dans le même lieu et la même region qu'une autre station avec la commande :
  INSERT INTO Station (nom_station, capacite, lieu, region, tarif) VALUES ('Moubaraka', 450, 'Guadeloupe', 'Antilles', 1500) ;
  On constate que l'insertion ne s'est pas bien derouler car la contrainte *Unique_station* a été violer (voir capture ci-dessous). En effet ici il ne peut pas existé deux stations sur même *lieu* et la même *region*.


cyber@Eliman: ~
psql (12.6 (Ubuntu 12.6-0ubuntu0.20.04.1))
Type "help" for help.
postgres=# \c projet_base_donnees;
You are now connected to database "projet_base_donnees" as user "postgres".
projet_base_donnees=# DELETE FROM Station WHERE nom_station = 'Venusa';
DELETE 1
projet_base_donnees=# SELECT * FROM Activite WHERE nom_station = 'Venusa';
nom_station | libelle | prix
(0 rows)
projet_base_donnees=# INSERT INTO Station (nom_station, capacite, lieu, region, tarif) VALUES ('Kampala', 350, 'Guadeloupe', 'Antilles', 1200);
INSERT 0 1
projet_base_donnees=# INSERT INTO Station (nom_station, capacite, lieu, region, tarif) VALUES ('Moubaraka', 350, 'Guinea', 'Nullepart', 1500);
ERROR: new row for relation "station" violates check constraint "ch_region"
DETAIL: Failing row contains (Moubaraka, 350, Guinea, Nullepart, 1500).
projet_base_donnees=# INSERT INTO Activite (nom_station, libelle, prix) VALUES ('Kampala', 'vollé', 1201);
ERROR: new row for relation "activite" violates check constraint "ch_prix"
DETAIL: Failing row contains (Kampala, vollé, 1201).
projet_base_donnees=# INSERT INTO Station (nom_station, capacite, lieu, region, tarif) VALUES ('Moubaraka', 450, 'Guadeloupe', 'Antilles', 1500);
ERROR: duplicate key value violates unique constraint "unique_station"
DETAIL: Key (lieu, region)=(Guadeloupe, Antilles) already exists.
projet_base_donnees=#

Conclusion

On a essayé de donner des reponses aux questions qui sont posées, en definissant nos tables et les contraintes specifiées dans l'exercice. Aprés test on constate que tout fonctionnent correctement. Les captures inserer dans le documents en temoignes ainsi que le script ci-joint.
La realisation du projet a été une experience trés enréchisssante en terme d'apprentissage.

Sources

https://fr.wikipedia.org/wiki/
https://stackoverflow.com/questions/23237471/
postgresql-check-constraint-for-foreign-key-condition?rq=1/

Bibliographie

- *Systéme d'Information - Base de données Relationnelle*
  Par Cheikh BA
- *Documentation PostgreSQL 10.3*
  Par The PostgreSQL Global Development Group