Découverte de PostgreSQL

Table des matières

I Contexte

Durée : 2h

Environnement de travail : psql

Pré-requis : Aucun

Lorsque vous devez implémenter une base de données relationnelle, le choix peut être complexe. De nombreux systèmes de gestion de bases de données existent : MySQL, PostgreSQL, Oracle Database, SQL Server, etc.

Comment s'y retrouver ? Chaque système a en effet ses avantages et ses inconvénients : certains sont plus performants quand le volume de données est faible, d'autres sont payants, d'autres encore proposent des fonctionnalités spécifiques à certains domaines métiers, etc.

Tout au long de ces modules, vous apprendrez à installer et à manipuler PostgreSQL, un système de gestion de bases de données libre développé depuis 1996. Il est reconnu pour sa performance, adapté pour la plupart des usages et largement adopté par les éditeurs d'applications.

II Présentation et installation de PostgreSQL

Objectif

- Installer et mettre en service PostgreSQL sur un système Linux.

Mise en situation

PostgreSQL est un des systèmes de gestion de bases de données relationnelles les plus utilisés aujourd'hui.

Si une application web comme DBFiddle permet d'essayer des SGBD comme PostgreSQL et de vérifier la validité des requêtes SQL, elle n'est pas suffisante pour une *« vraie »* application.

En effet, une application réelle communique avec le SGBD grâce à du code, ce que ne permet pas l'interface de DBFiddle.

De plus, l'installation d'un SGBD sur une de vos machines permet de le configurer finement et d'optimiser les performances. Dans ce module, vous apprendrez à installer PostgreSQL sur un système Linux.

Avantage de PostgreSQL

PostgreSQL est :

- un SGBDR,
- libre (licence BSD),
- multi-plate-formes (Unix, Linux, Windows, MacOS, etc.),
- puissant,
- très respectueux du standard SQL,
- très bien documenté.

Documentation de PostgreSQL Fondamental

La ressource de référence pour ce SGBDR mais aussi pour SQL est sa documentation :

- postgresql.org/docs[1]
- En français : docs.postgresqlfr.org[2]

Fonctionnement général

PostgreSQL comme la grande majorité des SGBD est un logiciel *client* p.34-*serveur* p.34. Sous Linux, le programme associé au serveur est `postgresql`. Il existe de nombreux autres clients, les deux plus utilisés sont le client textuel `psql` et le client graphique *PgAdmin*.

1. https://www.postgresql.org/docs/current
2. https://docs.postgresqlfr.org/current/

Attention

Une installation *a minima* de PostgreSQL **uniquement à des fins de test et d'apprentissage**, et pour un usage local est présentée ici. Pour mettre en production une base de données PostgreSQL sur le réseau, il faut suivre des directives supplémentaires, notamment pour assurer la sécurité du système, sa sauvegarde, etc. Ces thèmes ne sont pas abordés dans le cadre de ce module.

PostgreSQL est à l'origine un système de gestion de base de données conçu pour Unix ; son installation et son fonctionnement sont possibles aujourd'hui sur plusieurs OS, mais Linux reste son environnement de prédilection. C'est l'architecture PostgreSQL sur Linux qui est étudiée ici.

Installation de PostgreSQL

Fondamental

Une présentation complète de l'installation de PostgreSQL est disponible sur le document ici : https://www.postgresql.org/download/

Installation sous Debian ou Ubuntu à partir des paquets de la distribution

Exemple

Pour installer PostgreSQL sur des distribution de Linux de famille Debian comme Ubuntu, on peut exécuter la commande suivante en tant qu'administrateur :

```
1 apt-get update
2 apt install postgresql
```

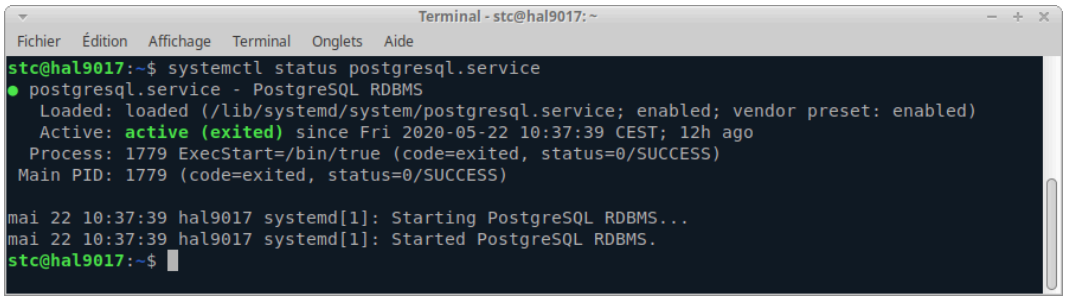
La première commande met à jour les dépôt des paquets pour `apt`.

La seconde commande télécharge et installe le méta-package `postgresql` qui :

- Installe le serveur `postgresql` et le client textuel `psql` (ainsi que le client graphique *pgAdmin*),
- Lance le service `postgresql`,
- Crée un utilisateur Linux `postgres` (que l'on peut voir dans `/etc/passwd`),
- Crée également une base de données par défaut nommée `postgres` et un utilisateur par défaut pour le SGBD nommé `postgres`.

On peut voir l'état de fonctionnement du serveur `postgresql` avec la commande :

```
1 systemctl status postgresql.service
```



Tester son installation

Méthode

```
sudo su postgres
psql
```

- `sudo su postgres` permet de devenir l'utilisateur Linux *postgres* qui a été créé lors de l'installation.
- `psql` permet d'établir la connexion par défaut entre le client *psql* et le serveur *postgresql* en utilisant l'utilisateur Linux et une base de données du même nom, donc ici *postgres*.
- Le client connecté affiche son numéro de version, propose de taper *help* et une invite de commande `postgres=#`.
- `\l` permet d'afficher la liste des base de données existantes (on retrouve la base *postgres* à laquelle on est connecté).
- `\q` permet de quitter.

Commande SHOW ALL ;

Complément

La commande `SHOW ALL;` permet de voir tous les paramètres du serveur PostgreSQL.

Par exemple `data_directory` permet de connaître le répertoire de stockage utilisé sur le disque dur.

Installer PostgreSQL sous Windows

Complément

1. Télécharger un *installer* depuis https://www.postgresql.org/download/windows[3].
2. Exécuter l'installation en validant les propositions par défaut (et sans installer les éventuels programmes complémentaires proposés à l'issue de l'installation).
3. Exécuter le client *psql* (également appelé *Shell SQL*).

3. https://www.postgresql.org/download/windows/

III Exercice : Appliquer la notion

Question 1

[solution n°1 p. 26]

Installer PostgreSQL.

Question 2

[solution n°2 p. 26]

Comment obtenir des informations sur le statut du service PostgreSQL ?

IV Le client textuel « psql »

Objectif

- Savoir dialoguer avec PostgreSQL via un terminal.

Mise en situation

Une fois PostgreSQL installé sur votre système, il y a deux manières de l'utiliser.

La première est d'écrire du code qui communique directement avec PostgreSQL, et c'est ce que vous ferez si vous développez une application web.

La seconde est de dialoguer avec PostgreSQL à l'aide d'un terminal et de commandes SQL, comme vous le feriez avec un outil comme DBFiddle.

Pour ce faire, on utilisera le *shell* PosgtreSQL, ou `psql`.

psql

Az Définition

`psql` est le client textuel de PostgreSQL. C'est une **interface en ligne de commande** ou CLI (pour *command line interface*).

Il permet une utilisation interactive de PostgreSQL.

Connexion à un serveur

Méthode

Le plus souvent, on interagit avec un serveur PostgreSQL distant. Pour initier la connexion avec *psql*, on utilise la commande suivante :

```
psql -h adresse.serveur -U user -d mydb
```

Cette commande opère la connexion avec la machine d'adresse `adresse.serveur` avec un utilisateur PostgreSQL nommé `user` à la base de données `mydb`.

Pour que cette commande fonctionne, il faut que :

- Le serveur PostgreSQL s'exécute sur la machine d'adresse `adresse.serveur` (sur le port standard 5432) ;
- L'utilisateur `user` existe sur PostgreSQL et avoir un mot de passe défini ;
- Une base de données `mydb` existe sur PostgreSQL et que l'utilisateur `user` y ait accès ;
- Le mot de passe de l'utilisateur `user` soit renseigné.

Lorsque l'on est connecté au serveur on obtient le prompt de la forme :

```
psql (9.5.21)
Type "help" for help.

user=>
```

À partir de là, on peut saisir des requêtes SQL mais aussi des commandes *psql*.

On sort de `psql` avec :

```
user=> \q
```

Écrire une instruction SQL

Syntaxe

Une requête SQL peut être directement exécutée. On veillera à ne pas oublier le point virgule « `;` » qui indique la fin de celle-ci.

```
user=> SELECT * FROM matable ;
```

Écrire une instruction SQL sur plusieurs lignes

Syntaxe

Une instruction SQL peut s'écrire sur une ou plusieurs lignes, le « *retour chariot* » n'a pas d'incidence sur la requête, c'est le « `;` » qui marque la fin de l'instruction SQL et provoque son exécution.

```
user=> SELECT *
user-> FROM matable
user-> ;
```

On notera dans `psql` la différence entre les caractères « => » et « -> » selon que l'on a ou pas effectué un retour chariot.

Commandes de base : aide

Fondamental

`\?` : Liste des commandes `psql`

`\h` : Liste des instructions SQL

`\h CREATE TABLE` : Description de l'instruction SQL `CREATE TABLE`

Commandes de base : catalogue

Fondamental

`\d` : Liste des relations (catalogue de données)

`\d maTable` : Description de la relation `maTable`

Documentation de psql

Complément

Pour obtenir toutes les options sur `psql`, on pourra se référer à la documentation de ce client ici :

https://docs.postgresql.fr/9.6/app-psql.html

pgAdmin 4

Complément

Le client *pgAdmin4* est un client graphique disponible sous Linux et sous Windows.

pgAdmin4 offre une interface graphique permettant d'effectuer les mêmes opérations qu'avec le client `psql` et offrant une navigation simplifiée et un *monitoring* facilité.

V Exercice : Appliquer la notion

Question 1

[solution n°3 p. 26]

Créer la table `tabledetest` avec pour seul attribut un `id` de type entier via *psql* en exécutant une requête SQL.

Lister les différentes tables présentes dans la base de données pour s'assurer de sa création.

Indice :

On veillera à ne pas oublier le point-virgule pour exécuter la requête.

Question 2

[solution n°4 p. 27]

Supprimer maintenant cette table.

Question 3

[solution n°5 p. 27]

Comment peut-on avoir un aperçu du tampon ayant stocké la dernière requête ?

Indice :

On parcourra la page de manuel de *psql* : https://docs.postgresql.fr/9.6/app-psql.html

VI Créer des bases de données et des utilisateurs

Objectifs

- Comprendre le concept d'utilisateur et de permissions au sens PostgreSQL.
- Savoir créer des bases de données et des utilisateurs.

Mise en situation

Une installation PostgreSQL peut gérer plusieurs bases de données. Quand a-t-on besoin de plusieurs bases de données pour une même application ? Un grand classique consiste à utiliser une base de données pour les tests, et une base de données pour la production. Qui dit base de données différente dit aussi utilisateurs différents et autorisations différentes : on peut imaginer que la base de test soit accessible à plus d'utilisateurs que la base de production.

Sur PostgreSQL, une base et un utilisateurs nommés *« postgres »* sont créés par défaut. Dans ce module, vous apprendrez à créer de nouvelles bases et de nouveaux utilisateurs avec une syntaxe proche de SQL.

Créer un utilisateur

Syntaxe

```
1 CREATE USER user1 PASSWORD 'password';
```

Créer une base de données

Syntaxe

```
1 CREATE DATABASE mydb OWNER user1;
```

La clause OWNER permet de spécifier le propriétaire (*owner*) de la base de données. Celui-ci a tous les droits sur sa base de données. Il pourra créer, modifier et détruire les tables de le base de données.

Supprimer des bases de données et des utilisateurs

Complément

```
1 DROP DATABASE mydb;
2 DROP USER user1;
```

Modifier le mot de passe d'un utilisateur

Complément

```
1 ALTER USER user1 PASSWORD 'mypassword';
```

Changer le propriétaire d'une base de données

Complément

```
1 ALTER DATABASE mydb OWNER TO user2;
```

psql : catalogues des utilisateurs et des bases de données

Syntaxe

`psql` dispose de commandes pour consulter les catalogues des utilisateurs et des bases de données :

- `\du` : liste des utilisateurs,
- `\l` : liste des bases de données.

psql : changer d'utilisateur

Syntaxe

Si on est connecté à une base de donnée avec un utilisateur et que l'on veut changer d'utilisateur et/ou de base, avec `psql` on utilise :

- `\c db user` : pour se connecter à la base *db* avec le compte *user*.

VII Exercice : Appliquer la notion

Cet exercice va permettre de découvrir la création de rôle et de *psql*.

On parcourra la page de manuel de *psql*: https://docs.postgresql.fr/9.6/app-psql.html

Question 1

[solution n°6 p. 27]

Créer un utilisateur nommé `paul` avec un mot de passe de votre choix et une base éponyme dont il est le propriétaire.

Question 2

[solution n°7 p. 27]

Lister les bases de données disponibles.

Quelle autre base de données d'un utilisateur différent apparaît ?

Question 3

[solution n°8 p. 27]

Changer le propriétaire de la base de données `paul` pour `postgres`.

VIII Exécuter des instructions SQL depuis un fichier

Objectif

- Savoir exécuter des instructions SQL depuis un fichier.

Mise en situation

Il est assez fréquent d'avoir une liste d'instructions SQL pré-définies à exécuter.

Vous pourriez recevoir le code SQL pour créer la structure d'une base de données à des fins de test, par exemple. Ou vous pourriez vouloir migrer une base de données de MySQL vers une base de données PostgreSQL, ce qui se traduira par l'exécution d'un long code SQL pour reconstruire la structure des tables et insérer les données dans PostgreSQL.

Dans ces cas, il est intéressant d'exécuter un fichier contenant une liste de requêtes SQL, plutôt que de les entrer une par une dans un client, ou de les copier-coller.

Syntaxe

Pour exécuter un fichier contenant du code SQL, utiliser la commande PostgreSQL `\i chemin/fichier.sql` :

- `chemin` désigne le répertoire dans lequel est le fichier `fichier.sql`,
- Le dossier de travail de `psql` est le dossier dans lequel il a été lancé, le script peut être lancé à partir de son dossier `home` pour en être indépendant (`~/.../fichier.sql`),
- Chaque commande doit être terminée par un `;`.

```
user=> \i chemin/fichier.sql
```

Programmer une base de données avec PostgreSQL

Méthode

Pour programmer une base de données sous PostgreSQL, voici une méthode générale :

- Rendez-vous dans un répertoire de travail : `cd /home/me/bdd1`.
- Créez ou ouvrez un fichier texte `bdd.sql` avec un éditeur de texte.
- Ouvrez un terminal et exécutez votre client *psql*.
- Écrivez votre code SQL, et testez-le au fur et à mesure : `\i bdd.sql`.

Tester votre code régulièrement

Conseil

Afin de tester régulièrement votre base de données, pensez à insérer en début de script des instructions de destruction des tables.

On supprime les tables dans l'ordre inverse de leur création, on peut ajouter la clause `IF EXISTS` afin d'éviter les erreurs lorsqu'une exécution précédente avait déjà échoué.

```
DROP TABLE IF EXISTS t2 ;
DROP TABLE IF EXISTS t1 ;
CREATE TABLE t1 (a VARCHAR PRIMARY KEY);
CREATE TABLE t2 (a VARCHAR REFERENCES t1(a));
```

IX Exercice : Appliquer la notion

On se donne les données suivantes :

nom_film	num_salle	prix	date_projection	horaire_projection
I Origins	12	7.90	20 Mars 2020	13:37
Drive	47	8.90	15 Août 2020	13:45
Mr Nobody	12	16.90	4 Mai 2020	7:20

Question 1

[solution n°9 p. 27]

Écrire la requête permettant de créer cette table.

Question 2

[solution n°10 p. 28]

Écrire les requêtes permettant d'insérer les données.

Question 3

[solution n°11 p. 28]

Mettre en place un fichier `film.sql` qui permettra d'exécuter toutes ces requêtes en même temps et de lire le contenu de la table créée à la fin.

Indice :

On veillera à supprimer la table en début de script si elle existe.

X Documentation de PostgreSQL

Objectif

- Savoir trouver des informations sur SQL et PostgreSQL.

Mise en situation

Les systèmes de gestion de bases de données sont des outils complets et complexes. Si leur configuration par défaut suffit pour la plupart des usages, il est nécessaire de la modifier pour améliorer la performance de certaines applications. Des cas typiques pourraient être un grand volume de données, ou une base avec beaucoup de lectures et peu d'écritures, etc.

Lorsque l'on a un problème ou une question, se référer à la documentation de la technologie utilisée est le meilleur point de départ. La documentation officielle de PostgreSQL est la ressource la plus complète qui existe. Elle est disponible pour toutes les versions supportées de PostgreSQL.

Documentation en ligne

Fondamental

La documentation de PostgreSQL est hébergée sur un serveur dédié :

- postgresql.org/docs[4]
- en français : docs.postgresql.fr/.

Fonction de recherche

Méthode

La fonction de recherche de la documentation en ligne permet de chercher un mot-clé particulier pour une version ou toutes les versions de PostgreSQL.

Documentation en manuel

Fondamental

Une version de la documentation peut directement être consultée via le terminal avec :

```
1 apt install PostgreSQL-doc
```

On peut y avoir accès avec la commande :

```
1 man postgres
```

Documentation pdf

Fondamental

Une version de la documentation en disponible au format PDF en téléchargement en ligne ici : https://docs.PostgreSQL.fr/12/pg12.pdf[5]

4. https://www.postgresql.org/docs/current
5. https://docs.postgresql.fr/12/pg12.pdf

Chapitres sur SQL

La partie 2 contient des chapitres sur les spécificités de SQL auxquels on peut se référer lorsque l'on débute. En particulier, on se référera :

- Au chapitre 4 sur la syntaxe SQL,
- Au chapitre 5 sur la définition des données,
- Au chapitre 6 sur la manipulation des données,
- Au chapitre 7 sur les requêtes.

Chapitres sur PostgreSQL

La partie 3 contient des chapitres sur les spécificités de PostgreSQL auxquels on peut se référer lorsque l'on débute. En particulier, on se référera :

- Au chapitre 19 sur la configuration du serveur,
- Au chapitre 20 sur l'authentification du client,
- Au chapitre 21 sur les rôles de la base de données,
- Au chapitre 22 sur l'administration de la base de données.

XI Exercice : Appliquer la notion

Question 1

[solution n°12 p. 28]

À quoi correspond le type `cidr`

Indice :

La solution se trouve dans le chapitre 8.

Question 2

[solution n°13 p. 28]

À quoi sert un tablespace ?

Indice :

La solution se trouve dans le chapitre 22.

Question 3

[solution n°14 p. 28]

Quel fichier stocke les informations pour l'authentification du client ?

Indice :

La solution se trouve dans le chapitre 20.

XII Essentiel

L'installation de PostgreSQL sous Linux est très simple : en une seule commande, vous obtenez un système prêt à être utilisé à des fins de tests.

Le shell PostgreSQL, inclus dans l'installation, est très puissant. En plus d'autoriser l'exécution des requêtes SQL classiques, contenues dans un fichier ou entrées à la main, il permet de créer de nouvelles bases de données et de nouveaux utilisateurs.

PostgreSQL intègre également un systèmes de permissions, qui indique finement ce que les utilisateurs peuvent faire ou ne pas faire pour telle ou telle base de données.

Avant de mettre un production un système PostgreSQL, il faudra s'assurer que la configuration répond bien à vos besoins, et pour ce faire une documentation très complète est disponible en ligne ou directement depuis votre terminal.

XIII Auto-évaluation

1. Exercice final

Exercice 1 : Quiz - Culture

[solution n°15 p. 28]

Exercice

Qu'est-ce qu'une CLI ?

- **A** Une interface en ligne de commande
- **B** Un type propre à PostgreSQL
- **C** Un utilisateur par défaut de PostgreSQL

Exercice

Parmi les propositions suivantes, lesquelles sont des clients de PostgreSQL ?

- **A** psql
- **B** postgresql
- **C** pgAdmin4

Exercice

Un client lourd est un client... :

- **A** Dont la taille une fois installé est plus importante que celle d'un client léger.
- **B** Qui repose sur les ressources de la machine qui l'utilise.
- **C** Est un client ancien qui n'est plus utilisé.

Exercice 5 : Quiz - Méthode

[solution n°16 p. 29]

Exercice

On crée généralement plusieurs utilisateurs... :

- **A** Pour partager l'usage du serveur entre plusieurs bases de données.
- **B** Pour avoir une gestion fine des droits.

C Parce qu'il faut nécessairement un propriétaire pour une table.

D Parce qu'il faut nécessairement un propriétaire différent pour chaque table.

Exercice

On crée plusieurs bases de données... :

A Car il en faut autant que de tables.

B Car il en faut nécessairement une par utilisateur.

C Pour permettre d'isoler différentes tables d'une application donnée.

Exercice

Les bonnes premières choses à réaliser lors de la mise en place d'un serveur pour une application sont :

A La création d'un nouvel utilisateur qui n'a pas tous les droits de l'utilisateur par défaut `postgres`.

B La création d'une nouvelle base de données pour cette application.

C Le changement du mot de passe de l'utilisateur par défaut `postgres`.

Exercice 9 : Quiz - Code

[solution n°17 p. 30]

Exercice

On exécute avec succès, via `psql`, cette instruction :

```
1 \copy fleur (variete, couleur) FROM '/tmp/orchidees.csv' WITH DELIMITER ','
```

Que peut-on en déduire ?

A Une table `fleur` a été créé par cette instruction.

B Il existe une table `fleur` avec au moins deux colonnes dans la base de données.

C Les données d'un fichier CSV `orchidees.csv` ont été importées dans la table `fleur`.

D Le nombre d'enregistrements dans la table `fleur` est égal au nombre d'enregistrements présents dans le fichier `orchidees.csv`.

Exercice

Que permet de réaliser la commande : `\timing` sous `psql` ?

- **A** Elle indique le temps d'exécution de la dernière commande.
- **B** Elle active l'affichage du temps d'exécution des prochaines commandes.
- **C** Elle donne le temps d'exécution de toutes les commandes exécutées dans la session.

Exercice

Que permet de réaliser la commande : `\e` sous `psql` ?

- **A** Elle permet d'éditer la dernière commande entrée dans la session `psql` via l'éditeur de texte par défaut.
- **B** Elle liste les paramètres du système qui ont été manuellement changés.
- **C** Elle liste tous les encodages utilisés dans la base de données courante.
- **D** Elle retourne la liste des événements importants de sécurité du SGBD.

Exercice

Que permet de réaliser la commande : `\H` sous `psql` ?

- **A** Elle fournit de l'aide sur les commandes SQL.
- **B** Elle fournit de l'aide sur les commandes psql.
- **C** Elle présente les résultats des requêtes au format HTML.

2. Exercice : Défi

On dispose des informations suivantes sur des étudiants :

pknumsecu	knumetu	nom	prenom
1800675001066	AB3937098X	Dupont	Pierre
2820475001124	XGB67668	Durand	Anne

etu

`pknumsecu` est une clé primaire qui contient exactement 13 caractères, `knumetu` est une chaîne de caractères d'au plus 20 caractères qui ne peut être nulle.

On dispose de plus des informations suivantes sur des cours :

pkcode	fketu
NF17	1800675001066
NF26	1800675001066
NF29	1800675001066

cours

`pkcode` est une chaîne de 4 caractères exactement et `fketu` est une clé étrangère référençant `pknumsecu`. Ces deux attributs forment la clé primaire.

Question 1

[solution n°18 p. 31]

Créer dans la base de données les tables `etu` et `cours`, et y ajouter les données.

Question 2

[solution n°19 p. 32]

Via `psql`, quelles sont les commandes pour accéder au catalogue et vérifier la création de la table ?

Question 3

[solution n°20 p. 32]

Utiliser deux instructions SELECT pour vérifier le contenu de la table.

Nous allons à présent réinitialiser la base avec des données contenues dans un fichier.

Question 4

[solution n°21 p. 32]

Exécuter les instructions nécessaires afin de **supprimer** les données existantes dans les tables et non les tables.

Vérifier que les tables sont vides.

Indice :

On utilisera l'instruction DELETE du SQL.

Question 5

[solution n°22 p. 32]

On dispose des deux fichiers de données suivants `gi.csv` et `P2018.csv`.

Téléchargez le fichier `gi.csv` sur votre ordinateur.
(cf. gi.csv)

Téléchargez le fichier `P2018.csv` sur votre ordinateur.
(cf. P2018.csv)

Quel est le type de ce fichier et quelles informations contiennent-ils ?

Indice :

fr.wikipedia.org/wiki/Comma-separated_values

Question 6

[solution n°23 p. 32]

Exécuter le code suivant en remplaçant `/tmp` par l'emplacement des fichiers sur notre ordinateur.

```
1 \copy etu (pknumsecu, knumetu, nom, prenom) FROM '/tmp/gi.csv' WITH CSV DELIMITER ';'
  QUOTE '"'
2 \copy cours (fketu, pkcode) FROM '/tmp/P2018.csv' WITH CSV DELIMITER ';' QUOTE '"'
```

Montrer que de nouvelles données ont été insérées dans les tables `etu` et `cours`.

Solutions des exercices

Solution n°1

[exercice p. 7]

```
apt-get update
apt install postgresql
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  postgresql-doc
  The following NEW packages will be installed:
    postgresql
    0 upgraded, 1 newly installed, 0 to remove and 93 not upgraded.
    Need to get 5,392 B of archives.
    After this operation, 59.4 kB of additional disk space will be used.
    Get:1 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 postgresql
    all 9.5+173ubuntu0.3 [5,392 B]
    Fetched 5,392 B in 0s (120 kB/s)
    debconf: delaying package configuration, since apt-utils is not installed
    Selecting previously unselected package postgresql.
    (Reading database ... 30282 files and directories currently installed.)
    Preparing to unpack .../postgresql_9.5+173ubuntu0.3_all.deb ...
    Unpacking postgresql (9.5+173ubuntu0.3) ...
    Setting up postgresql (9.5+173ubuntu0.3) ...


```

Solution n°2

[exercice p. 7]

```
systemctl status postgresql
```

```
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset:
enabled)
   Active: active (exited) since Wed 2020-05-06 12:11:15 UTC; 2s ago
  Process: 6194 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
 Main PID: 6194 (code=exited, status=0/SUCCESS)
```

Le service est bien chargé dans le système (`Loaded: loaded`) et il est bien actif (`Active: active`).

Solution n°3

[exercice p. 10]

```
CREATE TABLE tabledetest (id INTEGER);
```

On peut ensuite lister les tables avec :

```
\d
```

on a :

```
            List of relations
 Schema |    Name     | Type  |  Owner
--------+-------------+-------+----------
 public | tabledetest | table | postgres
(1 row)
```

Solution n°4

[exercice p. 10]

On supprime la table avec :

```
DROP TABLE tabledetest;
```

Solution n°5

[exercice p. 10]

On a le tampon de la dernière commande avec la commande `\p` ou `\print`.

```
postgres=# \p
DROP TABLE TableDeTest;
```

Solution n°6

[exercice p. 13]

On utilise :

```
CREATE USER paul WITH PASSWORD 'wVyHZYpiRFzKK03jnrJJJADJ2IBsk1ufQfxwAexF/1U=';
CREATE DATABASE paul OWNER paul;
```

Solution n°7

[exercice p. 13]

On utilise :

```
\l
```

On obtient :

```
                                 List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
 paul      | paul     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres
+
           |          |          |             |             | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres
+
           |          |          |             |             | postgres=CTc/postgres
(4 rows)
```

La base de données de l'utilisateur `postgres` apparaît aussi.

Solution n°8

[exercice p. 13]

On utilise :

```
ALTER DATABASE paul OWNER TO postgres;
```

Solution n°9

[exercice p. 16]

```
CREATE TABLE ticket(
nom_film VARCHAR(24) PRIMARY KEY,
num_salle INTEGER,
prix FLOAT(2),
date_projection DATE,
horaire_projection TIME
);
```

Solution n°10

[exercice p. 16]

On utilise :

```
INSERT INTO ticket VALUES('I Origins', 12, 7.90, '2020-03-20', '13:37');
INSERT INTO ticket VALUES('Drive', 47, 8.90, '2020-08-15', '13:45');
INSERT INTO ticket VALUES('Mr Nobody', 12, 16.90, '2020-05-04', '7:20');
```

Solution n°11

[exercice p. 16]

```
DROP TABLE IF EXISTS Ticket;

CREATE TABLE ticket(
nom_film VARCHAR(24) PRIMARY KEY,
num_salle INTEGER,
prix FLOAT(2),
date_projection DATE,
horaire_projection TIME
);

INSERT INTO ticket VALUES('I Origins', 12, 7.90, '2020-03-20', '13:37');
INSERT INTO ticket VALUES('Drive', 47, 8.90, '2020-08-15', '13:45');
INSERT INTO ticket VALUES('Mr Nobody', 12, 16.90, '2020-05-04', '7:20');

SELECT * FROM ticket;
```

```

 nom_film  | num_salle | prix | date_projection | horaire_projection
-----------+-----------+------+-----------------+--------------------
 I Origins |        12 |  7.9 | 2020-03-20      | 13:37:00
 Drive     |        47 |  8.9 | 2020-08-15      | 13:45:00
 Mr Nobody |        12 | 16.9 | 2020-05-04      | 07:20:00
(3 rows)

```

Solution n°12

[exercice p. 19]

C'est un type utilisé pour stocker une adresse réseau IPv4 ou IPv6.

Solution n°13

[exercice p. 19]

Un tablespace sert à spécifier emplacements sur le disque d'une installation PostgreSQL.

Solution n°14

[exercice p. 19]

Il s'agit du fichier `pg_hba.conf`.

Solution n°15

[exercice p. 21]

Exercice

Qu'est-ce qu'une CLI ?

- (A) Une interface en ligne de commande

B Un type propre à PostgreSQL

C Un utilisateur par défaut de PostgreSQL

Exercice

Parmi les propositions suivantes, lesquelles sont des clients de PostgreSQL ?

A psql

B postgresql C'est le nom du programme serveur de PostgreSQL.

C pgAdmin4

Exercice

Un client lourd est un client... :

A Dont la taille une fois installé est plus importante que celle d'un client léger.

B Qui repose sur les ressources de la machine qui l'utilise.

C Est un client ancien qui n'est plus utilisé.

Solution n°16

[exercice p. 21]

Exercice

On crée généralement plusieurs utilisateurs... :

A Pour partager l'usage du serveur entre plusieurs bases de données.

B Pour avoir une gestion fine des droits.

C Parce qu'il faut nécessairement un propriétaire pour une table.

D Parce qu'il faut nécessairement un propriétaire différent pour chaque table.

Exercice

On crée plusieurs bases de données... :

A Car il en faut autant que de tables.

B Car il en faut nécessairement une par utilisateur.

- [ ] **C** Pour permettre d'isoler différentes tables d'une application donnée.

Exercice

Les bonnes premières choses à réaliser lors de la mise en place d'un serveur pour une application sont :

- [ ] **A** La création d'un nouvel utilisateur qui n'a pas tous les droits de l'utilisateur par défaut `postgres`.
- [ ] **B** La création d'une nouvelle base de données pour cette application.
- [ ] **C** Le changement du mot de passe de l'utilisateur par défaut `postgres`.

Solution n°17

[exercice p. 22]

Exercice

On exécute avec succès, via `psql`, cette instruction :

```
1 \copy fleur (variete, couleur) FROM '/tmp/orchidees.csv' WITH DELIMITER ','
```

Que peut-on en déduire ?

- [ ] **A** Une table `fleur` a été créé par cette instruction. `\copy` ne crée pas la table, elle copie des données dans une table existante.
- [ ] **B** Il existe une table `fleur` avec au moins deux colonnes dans la base de données.
- [ ] **C** Les données d'un fichier CSV `orchidees.csv` ont été importées dans la table `fleur`.
- [ ] **D** Le nombre d'enregistrements dans la table `fleur` est égal au nombre d'enregistrements présents dans le fichier `orchidees.csv`. Il est possible que des enregistrements étaient présents avant l'ajout de données.

Exercice

Que permet de réaliser la commande : `\timing` sous `psql` ?

- [x] **A** Elle indique le temps d'exécution de la dernière commande.
- [ ] **B** Elle active l'affichage du temps d'exécution des prochaines commandes.
- [ ] **C** Elle donne le temps d'exécution de toutes les commandes exécutées dans la session.

Exercice

Que permet de réaliser la commande : `\e` sous `psql` ?

- **A** Elle permet d'éditer la dernière commande entrée dans la session `psql` via l'éditeur de texte par défaut.
- **B** Elle liste les paramètres du système qui ont été manuellement changés.
- **C** Elle liste tous les encodages utilisés dans la base de données courante.
- **D** Elle retourne la liste des événements importants de sécurité du SGBD.

Exercice

Que permet de réaliser la commande : `\H` sous `psql` ?

- **A** Elle fournit de l'aide sur les commandes SQL. C'est la commande `\h` (minuscule) qui fait cela.
- **B** Elle fournit de l'aide sur les commandes psql. C'est la commande `\?` qui fait cela.
- **C** Elle présente les résultats des requêtes au format HTML.

Solution n°18

[exercice p. 24]

```sql
CREATE TABLE etu (
  pknumsecu CHAR(13) PRIMARY KEY,
  knumetu VARCHAR(20) UNIQUE NOT NULL,
  nom VARCHAR,
  prenom VARCHAR
);
```

```sql
CREATE TABLE cours (
  pkcode CHAR(4) NOT NULL,
  fketu CHAR(13) NOT NULL,
  PRIMARY KEY (pkcode, fketu),
  FOREIGN KEY (fketu) REFERENCES etu(pknumsecu)
);
```

```sql
INSERT INTO etu (pknumsecu, knumetu, nom, prenom)
VALUES ('1800675001066', 'AB3937098X', 'Dupont', 'Pierre');

INSERT INTO etu (pknumsecu, knumetu, nom, prenom)
VALUES ('2820475001124', 'XGB67668', 'Durand', 'Anne');
```

```sql
INSERT INTO cours (pkcode, fketu)
VALUES ('NF17', '1800675001066');

INSERT INTO cours (pkcode, fketu)
VALUES ('NF26', '1800675001066');

INSERT INTO cours (pkcode, fketu)
```

```
8 VALUES ('NF29', '1800675001066');
```

Solution n°19

[exercice p. 24]

```
1 \d
2 \d etu
3 \d cours
```

Solution n°20

[exercice p. 24]

```
1 SELECT *
2 FROM etu;
```

```
1 SELECT *
2 FROM cours;
```

Solution n°21

[exercice p. 24]

```
1 DELETE FROM cours;
2 SELECT * FROM cours;
3
4 DELETE FROM etu;
5 SELECT * FROM etu;
```

Solution n°22

[exercice p. 24]

Ce sont des fichiers CSV qui contiennent respectivement des informations sur les étudiants et sur les cours.

Solution n°23

[exercice p. 25]

```
1 SELECT *
2 FROM etu;
```

```
1   pknumsecu   |  knumetu   |   nom    | prenom
2---------------+------------+----------+---------
3 1800675001066 | AB3937098X | Dupont   | Pierre
4 2820475001124 | XGB67668   | Durand   | Anne
5 1             | A          | Dupont   | Pierre
6 2             | B          | Durand   | Georges
7 3             | C          | Duchemin | Paul
8 4             | D          | Dugenou  | Alain
9 5             | E          | Dupied   | Albert
```

```
1 SELECT *
2 FROM cours;
```

```
1 pkcode |     fketu
2--------+---------------
3 NF17   | 1800675001066
4 NF26   | 1800675001066
5 NF29   | 1800675001066
6 NF17   | 1
7 NF18   | 1
8 NF19   | 1
9 NF20   | 1
10 LA13  | 1
```

```
PH01   | 1
NF17   | 2
NF18   | 2
NF19   | 2
TN01   | 2
LA14   | 2
PH01   | 2
NF17   | 3
NF18   | 3
NF19   | 3
NF21   | 3
LA14   | 3
PH01   | 3
```

Glossaire

Client

Un client est un programme informatique qui a pour fonction d'envoyer des requêtes à un autre programme informatique, appelé serveur, d'attendre le résultat de cette requête et de traiter le résultat de la requête. Notons qu'un programme peut-être client vis à vis d'un programme et serveur vis à vis d'un autre. On ne prend pas ici le terme client dans son acception matérielle, qui signifie alors un ordinateur qui a pour fonction d'héberger des programmes clients.

Serveur

Un serveur est un programme informatique qui a pour fonction de recevoir des requêtes d'un autre programme, appelé client, de traiter ces requêtes et de renvoyer en retour une réponse. Notons qu'un programme peut-être serveur vis à vis d'un programme et client vis à vis d'un autre. On ne prend pas ici le terme serveur dans son acception matérielle, qui signifie alors un ordinateur qui a pour fonction d'héberger des programmes serveurs.