Tutorial de connexion à postgreSQL sur nivose

Master 1 d'Informatique 2020-2021

Requêtes simples en SQL

---

1 Connexion au serveur PostgreSQL

Les outils de gestion de bases de données que nous allons utiliser dans ce cours - PostgreSQL `https://www.postgresql.org/`, "the world's most advanced open source database" - se trouvent sur un serveur distant.

Dans tout ce qui suit il faut toujours remplacer le mot « login » par le login que vous utilisez dans les salles de TP.

Vous pouvez accéder à la base de données depuis de chez vous en vous connectant en SSH. Sous Linux, OSX (Mac), il vous suffit de lancer un terminal :

1. Connectez vous sur un poste de travail (éventuellement en tant qu'invité en salle de TP).
2. Lancez une fenêtre terminal.
3. Dans cette fenêtre, tapez

   ```
   ssh login@nivose.informatique.univ-paris-diderot.fr
   ```

   ou bien si vous êtes sur une machine de salle TP

   ```
   ssh nivose
   ```

   (en remplaçant `login` par votre login, bien sûr) afin de vous connecter au serveur.
4. Entrez ensuite votre mot de passe lorsque le serveur vous le demande et validez avec la touche Entrée. Vous êtes désormais connecté au serveur.

Sous Windows, il faut installer un logiciel capable de se connecter en SSH au serveur : PuTTY `http://www.putty.org/`. Après avoir téléchargé et exécuté PuTTY, procédez aux réglages suivants :

1. Dans la case "Host Name (or IP address)", inscrivez :

   `login@nivose.informatique.univ-paris-diderot.fr`

   (toujours en remplaçant `login` par votre login).
2. Vérifiez que le champ "Port" contient bien 22 et que SSH est sélectionné dans le champ "Connection type".
3. Cliquez sur le bouton "Save" pour ne pas avoir à retaper ces informations à chaque fois.
4. Cliquez enfin sur "Open". Une fenêtre noire s'affiche et vous demande votre mot de passe. Entrez votre mot de passe. Vous êtes désormais connecté au serveur.

2 Travailler sur le serveur

Sur la machine `nivose.informatique.univ-paris-diderot.fr` vous disposez du *client* `psql` pour interagir avec le *serveur* de la base de données. [1] Si votre configuration n'est pas à jour, il est possible qu'il vous faille ajouter `/usr/local/pgsql/bin` à votre `PATH` :

— Dans le cas où vous n'utilisez pas `bash` par défaut, vous pouvez le lancer avec `bash`.

— Modifiez votre profil :
`cat >> .bash_profile`
`export PATH=/usr/local/pgsql/bin:$PATH`
suivi de `Ctrl-D` pour sortir de `cat`.

— Saisissez `source ./.bash_profile` pour que la modification prenne effet.

Ici le serveur est *local*, il écoute sur port *officiel*, il suffit de préciser le catalogue sur lequel vous souhaitez travailler (`base_m1`), en exécutant la commande :

```
psql -d base_m1 login
```

(« `base_m1` » est le nom de la base de données qui est déjà installé et qui commune pour vous tous).

Remplacez évidemment `login` par votre login (le même que celui avec lequel vous avez accédé à nivose). Vous devez ensuite saisir le mot de passe par défaut : `urquinaona`. Vous êtes désormais connecté à la base de données. Vous avez chacun un nom d'utilisateur (votre login) et un schéma pour travailler (`tp_login`) (encore une fois, remplacer le mot `login` par votre login).

Lors de la première connexion vous devez changer votre mot de passe. Tapez, en remplaçant `login` par votre login et `mdp` par votre nouveau mot de passe :

```
ALTER USER login WITH PASSWORD 'mdp';
```

Il est nécessaire de changer le schéma par défaut au début de chaque connexion à la base. En effet à la connexion vous êtes sur le schéma "public" de la base, accessible à tous. Mais vous avez chacun un schéma à vous, qui porte le nom `tp_login` (vous devez déjà savoir ce qu'il faut mettre à la place de `login`).

Pour passer dans votre schéma : `SET search_path TO tp_login;`

Pour connaître le schéma courant : `SHOW search_path;`

Apres la commande `SET search_path TO tp_login;` toute nouvelle table sera créé dans ce schéma.

`PostgreSQL` n'est (à priori) pas sensible à la casse des lettres (le fait que ce soit une majuscule ou une minuscule) dans les commandes. Vous pouvez utiliser la tabulation pour obtenir une complétion automatique. Voici quelques commandes `psql` utiles :

`\help` pour obtenir de l'aide

`\?` donne toutes les commandes

`\h` ou `\h mot_cle` donne une aide sur les commandes SQL

`\i commandes.sql` permet d'importer un fichier de commandes SQL. Ce script doit bien sûr se trouver dans le même répertoire que celui dans lequel vous étiez lorsque vous avez lancé PostgreSQL. Nous vous encourageons à écrire toutes vos requêtes dans un fichier pour garder une trace de votre travail.

`\q` pour quitter `PostgreSQL`

`\d` pour obtenir de l'information sur vos tables (leur définition)

`\d matable` pour obtenir de l'information sur une table particulière

1. Sur votre machine personnelle, vous pouvez également installer le client `pgcli`, qui ajoute par-dessus `psql` des fonctions un peu plus sympathiques : complétion automatique, coloration de la syntaxe etc. Cependant, `psql` étant le client officiel, il vous sera plus facile de trouver de l'aide sur internet concernant ce client.

`\echo message` permet d'afficher un message, par exemple le numéro de la question à laquelle va répondre la requête ;

`\e` édite la dernière commande (mode vi ou emacs)

`\! ls` effectuer la commande `ls` (marche pour toute commande Unix)

`\cd` change de répertoire

`fleche haut` revenir à la commande précédente ;

`CTRL-a` aller au début de la ligne ;

`CTRL-e` aller à la fin de la ligne ;

`CTRL-d` effacer le caractère suivant ;

`CTRL-k` effacer tout jusqu'à la fin de la ligne.

Mise en place

Pour chaque TP, créez un répertoire BD/TPi. Placez-vous dans ce répertoire puis lancez `psql`. Toujours dans ce répertoire, créez et éditez un script TP$i$sol.sql qui contient les commandes demandées, plutôt que de taper les commandes directement sous psql. Pour chaque requête, mettre par exemple `\! echo Requête <<i>>:` suivi de la requête SQL. Soignez bien l'indentation des requêtes pour qu'elles soient lisibles. En travaillant ainsi, vous gagnerez du temps, et aurez une solution de votre TP qui pourra vous être utile au moment des révisions.

3 Alternatives

Il peut être pratique d'installer PostgreSQL sur vos machines personnelles. Il existe plusieurs tutorials sur Internet ; en voici un en français plutôt clair, pour GNU/Linux : `https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-20-04-fr`

Pour celles et ceux qui seraient sous Windows, il y a plusieurs solutions, soit natives, soit en passant par Linux via le WSL sur Windows 10 ou via une machine virtuelle. Vous pouvez consulter cette page, mais attention les guides sont un peu obsolètes : `https://www.postgresql.fr/guidedemarragerapide`.

Finalement, pour celles et ceux qui n'ont pas de machine personnelle, ou n'arrivent pas à faire fonctionner correctement PostgreSQL, il existe des solutions en ligne, comme `https://sqliteonline.com/` qui gère la syntaxe PostgreSQL (mais avec des limitations). Il en existe d'autres mais peu qui supportent PostgreSQL, comme `https://repl.it/` qui utilise une syntaxe SQLite.