M1–Master Informatique – 2020/2021

Bases de données relationnelles

Avril 2021

Gestion de droits et authentification dans PostgreSQL

1 Avant de commencer (20mins)

Avant le TP, vous devez installer une machine virtuelle Debian.

1. Si vous ne l'avez pas déjà sur votre machine, installez le logiciel VirtualBox : `https://www.virtualbox.org/`.
2. Pour l'étape suivante, vous aurez besoin de télécharger le fichier `.iso` pour amd64 sur `https://www.debian.org/distrib/netinst`.
3. Créez la VM en suivant par exemple ce tutoriel : `https://www.numetopia.fr/comment-installer-debian-dans-virtualbox/`.
   — Vous pouvez par exemple mettre sur votre VM un disque de 8Go, 1Go de RAM, 2 processeurs (pour que l'affichage soit un peu plus fluide), laisser tout le reste par défaut.
   — Le disque de démarrage est le fichier `.iso` téléchargé au point précédent.
   — Lors de l'installation (choisissez "graphical install") conservez les paramètres par défaut (notez bien les mots de passe !). Sur la page "Sélection des logiciels", pour l'interface graphique vous pouvez prendre `xfce` par exemple, qui normalement ne prend pas trop de place. Choisir `/dev/sta` pour l'emplacement du programme de démarrage GRUB.
4. Par défaut, on ne peut pas faire de copier-coller entre la machine virtuelle et la machine hôte (i.e. la "vraie" machine). Pour remédier à cela on peut (ce n'est pas une obligation mais c'est quand même pratique) installer un additif à la machine virtuelle : Lancer la VM et dans un terminal, se connecter en tant que `root` avec la commande `su`. Exécuter :

   ```
   apt install gcc make perl dkms
   ```

   Dans le menu `Périphériques` de la VM, choisir "insérer l'image CD etc". Télécharger le fichier iso proposé. Ensuite, toujours en tant que root exécuter dans un terminal :

   ```
   sh /media/cdrom0/VBoxLinuxAdditions.run
   ```

   Puis redémarrer la machine virtuelle.

Cette machine virtuelle sera utilisée pour le TP.

2 Le TP

Documentation Postgres pour ce TP :
- Privilèges : https://www.postgresql.org/docs/13/ddl-priv.html
- Rôles : https://www.postgresql.org/docs/13/user-manag.html
- Connexion et authentification : https://www.postgresql.org/docs/9.1/client-authentication.html
- Schémas : https://www.postgresql.org/docs/13/ddl-schemas.html

2.1 Installation de PostgreSQL

Installez les paquets postgresql et postgresql-client avec la commande (à lancer en tant que root [1]) `apt install postgresql postgresql-client`. Contrairement à ce qu'indique l'installation, celle-ci démare déjà le serveur (ce qui peut se voir avec la commande `pg_lsclusters`), et donc il n'y a pas besoin d'exécuter la commande `pg_ctlcluster 11 main start`.

Question 2.1 : Lors de l'installation, un superutilisateur ainsi qu'une base de données lui appartenant sont créés, tous deux nommés `postgres`. Le serveur utilise le fichier `pg_hba.conf` qui se situe dans `/etc/postgresql/11/main/`. Inspectez son contenu. Essayez de vous connecter à la base en utilisant la commande (en tant que root, ou avec l'utilisateur normal) `psql -U postgres -d postgres`. Que se passe-t-il, et pourquoi ? Passez maintenant en tant que l'utilisateur `postgres` de la VM avec `su postgres`, puis essayez de vous connecter à la base avec la commande `psql`. Que se passe-t-il, pourquoi ?

Question 2.2 : Les deux premières lignes du fichier `pg_hba.conf` indiquent que la méthode d'authentification `peer` est utilisée pour les connexions locales. Cela implique que, lorsque l'on créera de nouveaux utilisateurs de la base pour le TP, il faudra également créer des utilisateur de la VM avec le même nom. C'est pénible. Supprimez les deux premières lignes et rajoutez la ligne `local all all trust` (faites une sauvegarde du fichier avant cela, par exemple `cp pg_hba.conf pg_hba.conf.backup`). La modification n'est pas prise en compte directement ; éteignez le serveur avec `systemctl stop postgresql` puis rallumez-le avec `systemctl start postgresql`. Retentez la commande `psql -U postgres -d postgres` en tant que root (ou avec l'utilisateur normal).

2.2 Création d'un rôle administrateur et de la base

Nous reprendrons en partie le scénario du TP sur l'indexation, à savoir une base de données de disquaire. Nous allons imaginer que ce disquaire : vend ses produits dans une boutique physique, dispose de quelques vendeurs, d'un administrateur de la base, d'un comptable, et d'un site web qui permet à n'importe qui de consulter les articles qu'il propose (mais pas de les acheter en ligne). Prenez connaissance du schéma de la base (fichier `music-schema.sql`).

1. `sudo` n'est pas configuré par défaut. Pour passer root, vous pouvez utiliser `su`.

Question 2.3 : Gérer une base de données directement avec un superutilisateur (comme `postgres`) est généralement un mauvaise idée, tout comme utiliser root dans un système UNIX pour des tâches anodines. Créez un utilisateur (rôle avec LOGIN) `admin` avec un mot de passe, qui aura le droit de créer des bases de données et de créer des rôles. Connectez-vous avec `admin` et créez une base de données, qu'on appellera `labase`. Vous pouvez vous connecter à cette base directement depuis psql avec la commande `\c labase` (ou alors, sortir de psql puis `psql -U admin -d labase`). Par défaut, le privilège `CONNECT` est donné à `PUBLIC` (un rôle spécial dont tout le monde est membre) sur toute base nouvellement créée. Révoquez ce droit.

Remarque : vous aurez constaté que le mot de passe n'est pas demandé lors de la connexion, et c'est normal au vu du contenu de `pg_hba.conf`. Le mot de passe nous servira dans la section suivante.

Question 2.4 : Lire la documentation sur les schémas. Constater avec la commande `\dn` que la base contient un schéma `public`. Exécutez la commande `\dn+` : qui est le propriétaire de ce schéma (pourquoi ?) et quels sont les droits associés à ce schéma ? Arrangez-vous pour supprimer ce schéma, puis créez (en tant qu'`admin`) un nouveau schéma que nous appellerons `music`. Donnez le privilège `USAGE` sur ce schéma à tous les utilisateurs (utilisez le mot clé `PUBLIC`). Vérifiez les droits avec `\dn+`.

Question 2.5 : Assignez le chemin de recherche au schéma nouvellement créé.
Créez les tables dans ce schéma avec le fichier `music-schema.sql`. Pour cela, soit vous disposez du copier-coller (cf la section sur l'installation de la VM), soit vous devez recopier le fichier sur votre VM et exécuter dans psql l'instruction `\i music-schema.sql`. Pour copier un fichier, vous pouvez utiliser le gestionnaire de fichier de votre VM (menu "Machines") dans lequel la création d'une session vous permettra d'avoir accès à vos 2 systèmes de fichier. Vérifiez avec `\dn` que les tables sont bien là. Vous pouvez créer quelques lignes dans chaque table afin de pouvoir tester les questions suivantes.

2.3 Autres rôles et privilèges

Occupons-nous maintenant de mettre en place les autres rôles avec leurs privilèges.

Question 2.6 : Créez un rôle `vendeurs` qui représentera le groupe des vendeurs de la boutique, et créez deux utilisateurs membres de ce groupe, `vendeur1` et `vendeur2`, avec mots de passe. Assignez les privilèges de telle sorte que les vendeurs puissent lire, insérer, supprimer et mettre à jour les tables `factures`, `lignes_factures` et `clients`. Connectez-vous à `labase` avec `vendeur1` puis testez ces privilèges.

Question 2.7 : Créez un utilisateur `comptable` (avec mot de passe) qui pourra voir les tables `factures`, `lignes_factures` et `employes` (mais pas les modifier). Créez de même un utilisateur `serveurweb` (avec mot de passe) qui pourra voir les tables `albums` et `produits`. Testez.

2.4 Connexion « à distance »

Depuis la question 2.2, toutes les tentatives de connexion locales ne demandent pas de mot de passe (mode `trust`). Dans notre scénario fictif, nous supposons que la machine sur laquelle est installée le serveur PostgreSQL est sécurisée et que seules des personnes de confiance y ont accès, donc nous conserverons la ligne en question de `pg_hba.conf`. Nous souhaitons maintenant donner la possibilité aux divers utilisateurs de se connecter à la base de données depuis leurs postes de travail. On ne le fera que pour l'utilisateur `vendeur1`. Nous supposerons pour cela que le poste de travail de cet utilisateur est votre machine à vous (tandis que le serveur Postgres tourne sur la VM).

Question 2.8 : Un détail technique. Par défaut, la VM est configurée avec un réseau de type NAT (voir la section 6.3 de https://www.virtualbox.org/manual/ch06.html). Pour avoir accès au serveur Postgres de la VM depuis votre machine, nous allons configurer une redirection de port (section 6.3.1). Éteignez la VM puis, depuis l'interface graphique de VirtualBox, ajoutez une redirection d'un port inutilisé de votre machine (5555 par exemple) vers le port 5432 de la VM (port sur lequel le serveur Postgres écoute par défaut). Cela se trouve dans le menu `Paramètres`, `Réseau`, onglet `Adapter1`, cliquer sur `Avancé`, puis `Redirection de ports`. Le port hôte est celui de la « vraie » machine, i.e., 5555, et le port invité celui de la VM, donc 5432.

Question 2.9 : Rallumez la VM, puis tentez de vous connecter à la base depuis votre machine avec la commande
`psql -U vendeur1 -d labase -h localhost -p 5555.`[2] Si tout va bien, cela ne fonctionne pas et vous indique que la connexion a été fermée de manière impromptue. En fait, par défaut le serveur PostgreSQL ignore toutes les tentatives de connexion ne venant pas de la machine sur laquelle il tourne, voir le paramètre `listen_addresses` dans https://www.postgresql.org/docs/9.1/runtime-config-connection.html. Éteignez le serveur Postgres, puis éditez le fichier `/etc/postgresql/11/main/postgresql.conf` pour transformer la ligne
`#listen_addresses = 'localhost'`
en
`listen_addresses = '*'`.
(soyons généreux). Redémarez le serveur, puis retentez
`psql -U vendeur1 -d labase -h localhost -p 5555`
depuis votre machine. Que vous raconte psql ? Éditez `pg_hba.conf` pour que cela fonctionne enfin.

2. Si on ne précise pas `-h localhost`, psql va essayer de se connecter avec les sockets Unix (donc sur votre propre machine), ce qui n'est pas ce qu'on veut.