info710 : compléments de bases de données

TP 1 : prise en main de PostgreSQL

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 22
téléphone : 04 79 75 94 22
email : Pierre.Hyvernat@univ-savoie.fr
www : http://www.lama.univ-savoie.fr/~hyvernat/

Remarque : la documentation de PostreSQL se trouve en ligne sur
`http://www.postgresqlfr.org/`
section "Doc v.8.1x fr" dans la colonne de gauche. Je vous conseille de garder un navigateur web ouvert sur cette page pendant les TP...

Exercice 1 : le système Unix (Linux)

Par soucis pédagogique (et non pas de contradiction), nous allons utiliser Linux plutôt que Windows... Redémarrez donc votre ordinateur en choisissant Linux au début.

Question 1 : Une fois loggé(e), démarrez si nécessaire un environnement graphique avec la commande "`> startx`". En utilisant les menus ou l'icône de la barre des taches, lancez un terminal.
Nous allons nous connecter à une machine hébergeant un serveur de bases de données. Cette machine se trouve dans le bâtiment Mont Blanc et a comme url `eco.univ-savoie.fr`. Vous devriez avoir un compte sur cette machine...
Pour vous connecter dessus, utilisez la commande "`> ssh login@eco.univ-savoie.fr`" et utilisez votre mot de passe habituel (celui du portail).

Exercice 2 : premiers pas avec PostreSQL

Question 1 : Cette machine héberge un serveur. Pour se connecter à la base `tp1` sur ce serveur, il faut utiliser la commande "`> psql tp1`". Le mot de passe de votre utilisateur sur le serveur est égal à votre login. La première chose à faire est de modifier ce mot de passe en utilisant la commande suivante : (dans l'environnement PostreSQL)

```
> ALTER ROLE login WITH PASSWORD 'mot-de-passe' ;
```

Comme `psql` garde un historique des commandes, il est conseillé de quitter l'environnement `psql` et de faire un "`> rm .psql_history`" afin de supprimer le fichier où apparaît votre nouveau mot de passe.

Question 2 : Reloggez-vous sur la base `tp1` du serveur ("`> psql tp1`") ; vous pouvez obtenir des informations sur les tables présentes dans cette base avec la commande

```
> \dt
```

Le résultat est

```
           List of relations
 Schema |   Name    | Type  | Owner
--------+-----------+-------+-------
 public | cours     | table | phyve
 public | cours_etu | table | phyve
 public | etu       | table | phyve
(3 rows)
```

ce qui indique la présence de trois tables nommées `cours`, `cours_etu` et `etu`. Pour obtenir des informations sur une table particulière, il faut utiliser "`> \d nom_de_table`".

⋆ Quelles sont les attributs de chacune de ces tables ? Quelles sont les clés principales ?

⋆ Quels sont leur types et comment interprétez-vous la mention `not null` dans la troisième colonne ?

Question 3 : D'autres commandes sont disponibles :
- "`\l`" pour obtenir la liste des bases de données sur le serveur ;
- "`\du`" la liste des utilisateurs sur le serveur ;
- "`\?`" pour obtenir la liste des commandes possibles ;
- "`\h COMMANDE`" pour obtenir de l'aide sur une commande SQL.

⋆ Testez certaines de ces commandes et essayer d'interpréter les résultats.

Exercice 3 : premières requêtes SQL

Question 1 : pour lister les lignes d'une table, il suffit d'utiliser

```
> SELECT * FROM nom_de_table ;
```

Remarque : toutes les requêtes doivent se terminer par un point virgule " ;".

⋆ Trouvez la liste des lignes (éléments) de chacune des tables de la base `tp1`.

Question 2 : on peut projeter une table sur des attributs en utilisant

```
> SELECT Attribut1, Attribut2, ... FROM nom_de_table ;
```

⋆ Comment pouvez-vous obtenir la liste des tous les prénoms des étudiants ?

Question 3 : l'insertion d'un élément dans une table se fait de la manière suivante :

```
> INSERT INTO nom_table VALUES (v1, v2, ...) ;
```

où `v1`, `v2`, ... sont les valeurs des attributs `Att1`, `Att2`, ... de la la table `nom_table`.
Pour spécifier une chaîne de caractère, il faut utiliser des guillemets :

```
> INSERT INTO cours VALUES ( 'Hyvernat', ...) ;
```

On peut donner les valeurs de attributs dans un ordre différent, à condition de spécifier les attributs de la manière suivante :

```
> INSERT INTO nom_table (B1, B2, ... ) VALUES (val1, val2, ...) ;
```

Il faut que `B1`, `B2`, ... soient des attributs de la table `nom_table` et que les valeurs `val1`, `val2`, ... soient des valeurs pour les attributs `B1`, `B2`, ...

⋆ Chacun devra, en utilisant les deux méthodes (sans spécifier l'ordre des attributs et en spécifiant l'ordre des attributs) insérer au moins deux éléments dans chacune des tables.

Si vous donnez l'ordre des attributs, vous n'êtes pas (forcément) obligé de spécifier tous les arguments.

⋆ Testez l'insertion d'éléments en ne donnant les valeurs que de quelques attributs.

⋆ En vous rappelant du contenu des tables au départ, que pensez-vous que le contenu des tables soit devenu ? Vérifiez-le.

Question 4 : les conventions sont d'utiliser des majuscules pour les mot clés SQL. Ceci n'est pas obligatoire car PostreSQL ne fait pas de différence entre les deux. (Sauf dans les chaînes de caractères...)
Testez les commandes suivantes
- `> SELECT * FROM etu ;`
- `> select * from ETU ;`
- `> SeLeCt * fRoM eTU ;`

Essayer maintenant les commandes suivantes
- `> SELECT * FROM "etu" ;`
- `> SELECT * fRoM "eTu" ;`
- `> SELECT * "FROM" eTu ;`

Qu'en pensez-vous ? (Quelle est la différence entre `FROM` et `etu` ?)

Question 5 : que pensez-vous que la commande suivante donne comme résultat ?

```
SELECT Prenom, Nom FROM Etu, Cours_Etu WHERE
         Etu.NEtu = Cours_Etu.NEtu
    AND Cours_Etu.NCours = 'info-710' ;
```

Testez et comparez avec votre prévision.

En suivant un schéma similaire, essayer d'écrire des requêtes "complexes". (Vous pouvez au besoin rajouter des éléments dans chacune des tables...)