cheikh.ba.sn@gmail.com

Bases de Données Relationnelles
TP - PostgreSQL

Sommaire

1 Environnement de Travail

L'objectif de cette session de TP est de manipuler un **SGBDR** standard en se basant sur un client en mode ligne de commande CLI (*Command Line Interface*). Le **SGBDR** utilisé sera **PostgreSQL**, installé sur une **machine virtuelle** dont vous disposerez.

- **Pourquoi PostgreSQL ?**
  Réponse: pourquoi pas ? Il répond à l'objectif du TP, en plus de respecter la norme SQL2.

  PostgreSQL est un Système de Gestion de Base de Données Relationnelle et Objet (SGBDRO), développé à l'origine par l'université de Berkeley. Il s'appuie sur les modèles relationnels, supportant complètement SQL, et apporte des extensions objet comme les classes, l'héritage, les types de données utilisateurs (tableaux, structures, listes..), les fonctions, et est portable sur plus de 20 environnements depuis la version 6.4.

  Ce système est concurrent d'autres Systèmes de Gestion de Base de Données, qu'ils soient libres (comme MySQL et Firebird), ou propriétaires (comme Oracle, Sybase, DB2 et Microsoft SQL Server). Comme les projets libres Apache et Linux, PostgreSQL n'est pas contrôlé par une seule entreprise, mais est fondé sur une communauté mondiale de développeurs et d'entreprises.

- **Pourquoi une machine virtuelle ?**
  PostgreSQL ne s'installait que sur les systèmes Unix/Linux (ou en mode émulation sous Windows). Il est maintenant possible, depuis la version 9, de l'installer sur Windows. Pour vous faire découvrir le principe de virtualisation, et le système d'exploitation Ubuntu (Linux), les TP se feront avec PostgreSQL installé sur un Ubuntu virtualisé (dans Windows). Vous n'avez pas besoin d'être un expert en Linux pour ce TP. Voici juste quelques commandes qui peuvent s'avérer utiles.

```
% ls -l  // Liste les fichiers du répertoire courant
% pwd  // Nom du répertoire courant
% cd rep  // Se positionne dans le répertoire 'rep'
% cd  // Ramène au répertoire de départ
% cp source cible  // Copie le fichier source vers le fichier cible.
% mv source cible  // Renomme le fichier 'source' en fichier 'cible'
% gedit &  // Lance l'éditeur de texte gedit en tâche de fond
```

  Bien sûr, toutes ce commandes peuvent être réalisées en utilisant tout simplement l'environnement graphique (explorateur de dossiers).

  La virtualisation permet de simuler, à l'aide d'un logiciel, appelé superviseur ou hyperviseur, une machine physique (disque dur, RAM, OS, ...). Il existe plusieurs logiciels pour la virtualisation (VMWare, VirtualBox, VirtualPC, ...), et nous choisirons VirtualBox (de Oracle).

1.1 Installation du superviseur Oracle VM VirtualBox:

Il vous suffit de lancer `"VirtualBox-4.1.14-77440-Win.exe"`, et de suivre les instructions.

1.2 Création d'une machine virtuelle Ubuntu.

Ce travail vous est épargné. Une machine `"Ubuntu-9.10-PostgreSQL-8.4-pgadmin3.vdi"` est déjà créée. Il ne vous reste qu'à le dire au superviseur, après avoir copié le disque dur de la machine (que l'on confond à la machine elle même) dans un répertoire (en général le répertoire contenant les disques de *VirtualBox*). Je vous montrerai la marche à suivre. Vous pouvez maintenant lancer la machine et découvrir le Système d'Exploitation Ubuntu (pour ceux qui ne l'ont jamais vu).

- Votre identifiant pour Ubuntu : `etudiant`
- Votre mot de passe : `ubuntu`

1.3 Installation du SGBDR PostgreSQL

Ceci vous a aussi été épargné. PostgreSQL est déjà installé sur la machine virtuelle, et un compte utilisateur, avec son mot de passe, et sa base de données (ensemble de relations) ont été créés.

- Votre compte PostgreSQL : `etudiant`
- Votre mot de passe PostgreSQL : `ubuntu`
- Votre base de données: `base_de_donnees`

1.4 Accéder à la base de données

Le SGBDR étant un serveur, il faudra le lancer avant de pouvoir communiquer (création de tables, insertion de données, requêtes SQL, etc.). Une fois qu'on a fini la communication, on pourra l'arrêter.

1.4.1 Lancement et arrêt du SGBDR PostgreSQL:

Ouvrir un terminal (Applications –> Accessories –> Terminal) et saisir la commande suivante[1]:

```
# sudo  /etc/init.d/postgresql-8.4  start
```

Le message suivant s'affiche à l'écran

```
* Starting PostgreSQL 8.4 database server:
```

Pour arrêter le serveur (pas maintenant !), il faut bien évidemment faire

```
# sudo  /etc/init.d/postgresql-8.4  stop
```

Pour savoir si le serveur est lancé (*"online"*) ou en arrêt (*"down"*) il faut faire

```
# sudo  /etc/init.d/postgresql-8.4  status
```

♣ **Conseil:** Il est conseillé d'ouvrir deux terminaux: l'un pour le lancement et l'arrêt du serveur PostgreSQL, l'autre pour la connexion à la base de données et pour les requêtes.

[1] Les caractères # en début de ligne ne sont pas à saisir, il s'agit du prompt

1.4.2 Connexion de *"etudiant"* à la base de données *"base_de_donnees"*

```
# psql  -U  etudiant  base_de_donnees
```

Le système vous demande le mot de passe

```
 Password for user etudiant:
```

Saisissez "ubuntu", et le système répond par

```
psql (8.4.7)
Type "help" for help.

base_de_donnees=>
```

Vous êtes maintenant connecté. Saisir la commande "help", et le système répond par

```
You are using psql, the command-line interface to PostgreSQL.
Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

base_de_donnees=>
```

Essayez maintenant les commandes suivantes:

```
# SELECT version();
```

Le système répond avec un message du genre

```
               version
-------------------------------------------------------------------------
PostgreSQL 8.4.7 on i686-pc-linux-gnu, compiled by GCC gcc (GCC)
4.1.2 20080704 (Red Hat 4.1.2-46), 32-bit
(1 row)

# SELECT current_date;
# SELECT 2 + 2;
```

Si tout s'est bien passé jusque là, alors PostgreSQL est bien installé.

2 Création d'un schéma (prédéfini) de base de données

Au départ, nous allons créer la base de données des *"Films"*, vue en cours. Les instructions de création du schéma sont regroupées dans le fichier "SchemaFilms.sql". Pour exécuter ces instructions, saisir la commande

```
# \i  SchemaFilms.sql
```

La commande "\i" de PostgreSQL permet en fait de charger des instructions SQL contenues dans le fichier fourni en argument. Vous aurez quelques messages d'erreur en retour:

```
psql:SchemaFilms.sql:10: ERROR:  table "notation" does not exist
psql:SchemaFilms.sql:11: ERROR:  table "role" does not exist
....
```

mais c'est normal: nous avons voulu nettoyer la base avant de créer un nouveau schéma.

♣ **Important:** En éditant le fichier "SchemaFilms.sql" , avec l'éditeur de texte **Gedit** par exemple (Application –> Accessoires –> Éditeur de texte gedit), vous pourrez remarquer que l'ordre de création des tables respecte le référencement entre PRIMARY KEY et FOREIGN KEY. Les tables qui sont référencées par cette dernière clause doivent être créées avant celles qui les référencent. Par exemple la table *Artiste* est créée avant la table *Film* à cause de la clé étrangère *idMES*. C'est en revanche l'ordre inverse qui est suivi pour les commandes DROP : on ne peut pas détruire une table qui est référencée par une commande FOREIGN KEY.

Notez que, en principe, on ne place pas les commandes DROP dans un script de création puisqu'on ne souhaite pas prendre le risque de détruire des données existantes. Comme il s'agit ici d'une base de test, la situation est différente.

2.1 Structure de la base de données

PostgreSQL nous permet d'étudier la structure de la base de données :

- La commande "\d" liste les relations de la base de données.
- La commande "\d nom_table" affiche la description de la table nom_table.

Utilisez ces deux commandes pour vérifier les tables présentes dans la base et leurs descriptions.

NB: la commande "\?" permet d'avoir une aide sur les commandes psql. Une fois dans l'aide, il faut taper la touche "q" pour sortir.

3 Chargement de données - Insertion de tuples dans les tables

Il s'agissait dans l'étape précédente de créer le schéma de la base de données *"Films"*. La base, dans son état actuel, ne contient aucune données. Nous allons donc procéder au chargement de quelques données dans la base. Les instructions sont dans le fichier "BaseFilms.sql". La base contient un échantillon de films avec leurs metteurs en scène, leurs acteurs et les notations de quelques internautes. Vous pouvez visualiser le fichier et voir à quoi il ressemble. Pour exécuter les instructions, il faut bien sûr faire:

```
# \i BaseFilms.sql
```

4 Interrogation avec SQL - Requêtes simples

Maintenant que votre base de données contient un échantillon de films avec leurs metteurs en scène, leurs acteurs et les notations de quelques internautes, on vous demande de concevoir, saisir et exécuter les ordres SQL correspondant aux requêtes qui suivent.

4.1 Remarques importantes

- Toutes les commandes SQL doivent se terminer par ';'. Si vous oubliez le ';', une ligne '2' vous est proposée. Dans ce cas tapez un ';' pour finir la commande.
- Les chaînes de caractères s'écrivent avec une simple quote : *'Vertigo'* et pas *"Vertigo"*.
- Les majuscules et les minuscules sont interprétés différemment. Par exemple *'Vertigo'* est considéré comme différent de *'vertigo'* ou *'VERTIGO'*. Pensez-y en faisant des sélections! Un moyen d'éviter les problèmes est d'utiliser la fonctions `UPPER` qui met tout en majuscule. Par exemple :

```
SELECT * FROM film WHERE UPPER(titre) = 'VERTIGO';
```

4.2 À propos de la date en PostGreSQL

- Vous pouvez considérer qu'il y a une colonne système ajoutée à toutes la tables par défaut. La colonne s'appelle `'current_date'` (déjà rencontrée en début de sujet), et elle contiendra, pour chaque tuple de la table, la date du jour. Un autre exemple d'utilisation:

```
SELECT email, current_date FROM internaute;
```

- Il y a beaucoup de fonctions de manipulation d'éléments de type `DATE`. Par exemple, si `maDate` est de type `DATE`, alors `EXTRACT (YEAR FROM maDate)` renvoie l'année (`YEAR`) de la date `maDate` fournie en paramètre. Par exemple, la requête suivante renvoie `2006`.

```
SELECT EXTRACT(YEAR FROM DATE '2006-01-01');
```

4.3 Sélections simples

♣ **Note:** La commande `"\s mon-fic-sauv.txt"` enregistre toutes les requêtes que vous avez saisies dans le fichier `mon-fic-sauv.txt`.

1. Tous les titres de films.
2. Nom et prénom des internautes auvergnats (habitant d'Auvergne).
3. Titre et année de tous les drames, triés par année ascendante. Donnez ensuite le tri par année descendante.
4. Nom et année de naissance des artistes nés avant 1950.

5. Titre et année de tous les films parus entre 1960 et 1980
6. Tous les genres de films (éliminez les doublons).
7. Les artistes dont le nom commence par 'H' (commande `LIKE`).
8. Titre, genre et résumé des films qui sont soit des drames, soit des westerns (utilisez la construction `IN`), et dont le résumé contient la chaîne de caractères "vie".
9. Quels sont les acteurs dont on ignore l'année de naissance ? (Valeur absente).
10. Prénom, nom et âge de chaque artiste (NB : l'âge est la différence entre l'année courante et l'année de naissance). Nommez âge la colonne obtenue (commande `AS`).

5 pgAdmin: outil graphique pour PostgreSQL

pgAdmin est la plateforme d'administration et de développement libre la plus populaire et la plus riche pour PostgreSQL, le serveur de base de données libre le plus riche en fonctionnalités. L'application est utilisable sur plusieurs plateformes (Linux, FreeBSD, OpenSUSE, Solaris, Mac OSX et Windows) pour gérer un serveur PostgreSQL. Il est conçu pour répondre aux besoins de tous les utilisateurs, de l'écriture de requêtes SQL simples aux développements de bases de données complexes. L'interface graphique supporte toutes les fonctionnalités de PostgreSQL et simplifie l'administration.

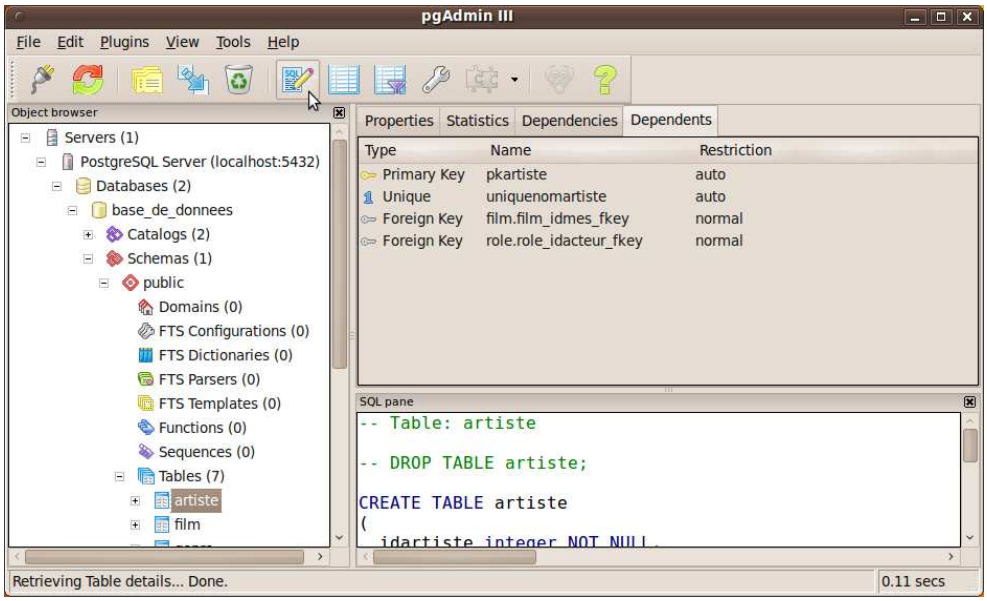

Figure 1: Vue d'ensemble de l'interface. Quelques informations sur le schéma de *artiste*

Pour lancer **pgAdmin**: "Applications –> Programming –> pgAdmin III", ou tout simplement saisir la commande suivante sur un terminal

```
# pgadmin3 &
```

Une fois l'application lancée, vous pouvez vous connecter en faisant un click droit sur "PostgreSQL Server", et en cliquant sur "Connect". Vous aurez à saisir votre mot de passe.

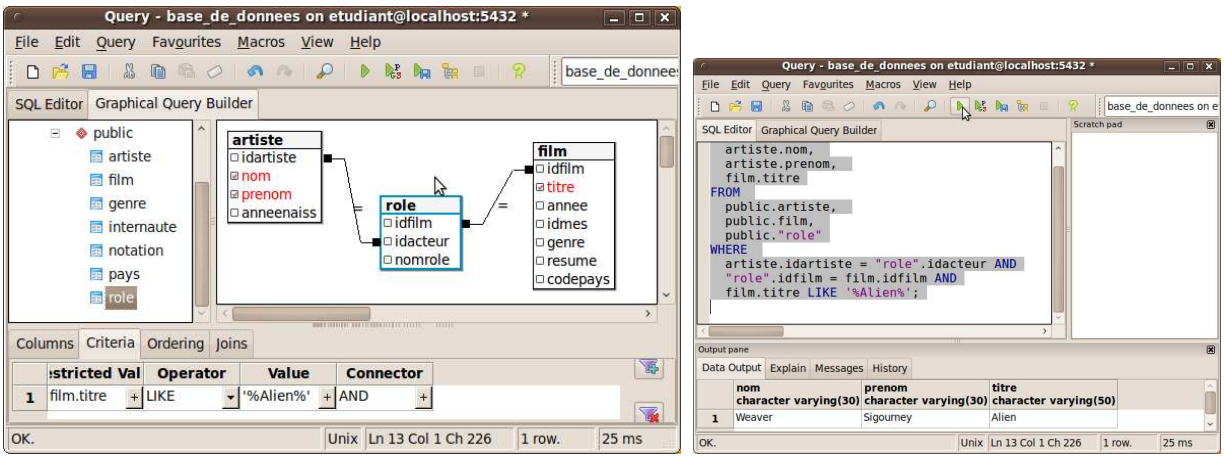

Figure 2: (a)-*Graphical Query Builder*. (b) Équivalent en SQL de la requête graphique

Les captures d'écran sur les figures 1 et 2 vous donnent une idée sur quelques fonctionnalités.

6 Interrogation avec SQL - Requêtes complexes

6.1 Jointures

1. Qui a joué le rôle de Morpheus (nom et prénom) ?
2. Qui est le réalisateur de Alien ?
3. Prénom et nom des internautes qui ont donné une note de 4 à un film (donner aussi le titre du film).
4. Quels acteurs ont joué quel rôle dans le film Vertigo ?
5. Films dont le réalisateur est Tim Burton, et un des acteurs est Johnny Depp.
6. Titre des films dans lesquels a joué Bruce Willis. Donner aussi le nom du rôle.
7. Quel metteur en scène a tourné dans ses propres films ? Donner le nom, le rôle et le titre des films.
8. Quel metteur en scène a tourné en tant qu'acteur (mais pas dans son propre film) ? Donner le nom, le rôle et le titre des films où le metteur en scène a joué.
9. Dans quels films le metteur en scène a-t-il le même prénom que l'un des interprètes ? (titre, nom du metteur en scène, nom de l'interprète). Le metteur en scène et l'interprète ne doivent pas être la même personne.

6.2 Requêtes imbriquées

Les requêtes suivantes peuvent s'exprimer avec une imbrication des clauses `SELECT`, mais on peut également utiliser des jointures "à plat". Si le cœur vous en dit, essayez les deux versions.

1. Donnez les nom et prénom des artistes qui on mis en scène un film.
2. Donnez le titre et l'année des films qui ont le même genre que Matrix.
3. Donnez le nom des internautes qui ont noté le film Alien. Donnez également la note.

6.3 Négation

Là encore, il existe souvent plusieurs manières d'exprimer la même requête.

1. Les films sans rôle.
2. Nom et prénom des acteurs qui n'ont jamais mis en scène de film.
3. Les internautes qui n'ont pas noté de film paru en 1999.

6.4 Fonctions de groupe

1. Quelle est le nombre de films notés par l'internaute rigaux@cnam.fr, quelle est la moyenne des notes données, la note minimale et la note maximale ?
2. Combien de fois Bruce Willis a-t-il joué le rôle de McClane ?
3. Année du film le plus ancien et du film le plus récent.
4. id, Nom et prénom des réalisateurs, et nombre de films qu'ils ont tournés.

7 Création d'un schéma relationnel

Il s'agit de définir un schéma de base de données, d'y intégrer des contraintes et d'y insérer quelques informations. Vérifiez le comportement des contraintes en essayant de les mettre en défaut.

7.1 Création des tables

Créez les tables du schéma *'Agence de voyages'*, vues en cours, et rappelées ci-dessous.

Station (nomStation, capacité, lieu, région, tarif)

Activité (nomStation, libellé, prix)

Client (id, nom, prénom, ville, région, solde)

Séjour (id, station, début[2], nbPlaces)

Attention à bien définir les clés primaires et étrangères. Voici les autres contraintes.

[2]Utiliser le type DATE

1. Les données *capacité*, *lieu*, *nom*, *ville*, *solde* et *nbPlaces* doivent toujours être connues.
2. Les montants (*prix*, *tarif* et *solde*) ont une valeur par défaut à 0.
3. Il ne peut pas y avoir deux stations dans le même lieu et la même région.
4. Les régions autorisées sont : 'Ocean Indien', 'Antilles', 'Europe', 'Amériques' et 'Extrême Orient'.
5. La destruction d'une station doit entraîner la destruction de ses activités et de ses séjours.
6. Le prix d'une activité doit être inférieur au tarif de la station et supérieur à 0.
7. Pour une date de début donnée, le nombre total de places réservées dans une station doit être inférieur à la capacité de la station.

♣ **Conseil:** donnez des noms à vos contraintes `PRIMARY KEY`, `FOREIGN KEY` et `CHECK` avec la clause `CONSTRAINT`.

7.2 Insertion de données

Insérez dans la base les données de la figure 3 avec des ordres `INSERT`. Attention, l'ordre des `INSERT` est important (pourquoi ?).

Vous pouvez ensuite tester les contraintes avec quelques ordres SQL. Par exemple : détruisez la station et vérifiez que les activités ont disparu ; insérez une autre station en (Guadeloupe, Antilles); insérez une station dans une région 'Nullepart', etc.

Station

NomStation	Capacité	Lieu	Région	Tarif
Venusa	350	Guadeloupe	Antilles	1200

Activité

NomStation	Libellé	Prix
Venusa	Voile	150
Venusa	Plongée	120

Client

id	Nom	Prénom	Ville	Région	Solde
10	Fogg	Phileas	Londres	Europe	12465
20	Pascal	Blaise	Paris	Europe	12465
30	Kerouak	Jack	New York	Amérique	9812

Séjour

idClient	Station	début	nbPlaces
20	Venusa	03-08-2003	4

Figure 3: La base "Agence"