Polytech Paris-Sud App. 4ème année 5 février 2014
Bases de Données

TP 1 : Prise en main de l'environnement / pagination

Préliminaires

Les fichiers nécessaires au TP se trouvent à l'adresse :

`https://www.lri.fr/~kn/files/bd_app4_tp1.tar.gz`

Remarque : ce TP peut servir de modèle pour les TP notées. Les réponses doivent être rédigées soigneusement (et commentées) dans des fichiers. Les codes sources doivent être commentés, et les fichiers doivent avoir le nom de l'auteur.

Exercice 1 : SQL, contraintes

On souhaite travailler sur une base de données de films. Pour cette dernière, les ordres de création de tables sont donnés dans le fichier `create_movies.pg_sql` (avec une syntaxe spécifique à Postgresql) et les ordres de remplissage dans `insert_movies.sql` (en SQL standard).

1. (mise en place) Se connecter au serveur Postgresql au moyen de la ligne de commande :

   ```
   psql -h s11 -U login_a
   ```

   où `login` est votre login Unix (du compte). Le mot de passe est le même que le mot de passe Unix par défaut. Le mot de passe spécifique à postgresql peut être modifié au moyens de l'interface Web (accessible uniquement depuis les machines du PUIO) :

   `http://s19.dep-informatique.u-psud.fr/etudiant/index.php`

   Une fois connecté, exécuter les deux scripts, soit en faisant un copier/collé de leur contenu, soit en exécutant la commande

   ```
   \i create_movies.pg_sql;
   \i insert_movies.sql;
   ```

   les deux fichiers de scripts doivent se trouver dans le repertoire courant.

2. Lister (en regardant l'ordre de création de table) toutes les contraintes et pour chacune d'elle, donner un ordre SQL qui viole la contrainte. Vérifiez en tappant l'ordre et en constant que vous obtenez une erreur

Corrigé:

`PEOPLE` :
- `pid INTEGER PRIMARY KEY` : contrainte de clé primaire, violée par `INSERT INTO PEOPLE VALUES (2, 'foo', 'bar');`.

`MOVIE` :
- `mid INTEGER PRIMARY KEY` : contrainte de clé primaire, violée par `INSERT INTO MOVIE VALUES (1000, 'foo', 10, 10, 10);`.

— `title VARCHAR(80) NOT NULL` : contrainte de non-nullité du titre, violée par `INSERT INTO MOVIE VALUES (10000, NULL, 10, 10, 10);`. Idem pour `year`, `runtime`, `integer`.

`ROLE` :

— `mid INTEGER REFERENCES MOVIE` : contrainte de clé étrangère, violée par `INSERT INTO ROLE VALUES (100000, 10, 'foo');`. Idem pour `pid`.

— `UNIQUE(mid,pid)` : unicité des paires `mid,pid` violée par `INSERT INTO MOVIE VALUES (2169,671,'foo');`

`DIRECTOR` Comme pour `ROLE`.

3. Donner le code SQL pour chacune des requêtes suivantes :
   (a) Renvoyer tous les titres de films
   (b) Renvoyer tous les acteurs (sans doublons)
   (c) Compter le nombre de réalisateurs uniques

Exercice 2 : SQL, requêtes

Donner le code SQL permettant d'évaluer chacune des requêtes suivantes. On ne cherchera pas forcément à écrire la requête de la manière la plus efficace possible.

1. Selectionne tous les titres de films
2. Selectionne toutes les personnes sans doublons
3. Selectionne tous les acteurs (i.e. toutes les personnes qui ont un role dans un film), sans doublons
4. Compte le nombre d'acteurs dans la base
5. Selectionne les films dans lesquel Clint eastwood est acteur
6. Selectionne les films dans lesquel Clint Eastwood est acteur mais pas réalisateur (en utilisant EXCEPT)
7. Selectionne les films dans lesquel Clint Eastwood et réalisateur (Sans utiliser INTERSECT)
8. Les réalisateurs qui ne sont pas des acteurs
9. Compter tous les films sortis entre 1999 et 2005
10. Le nombre films par année (pour chaque année)
11. L'année qui a eu le plus de film (si plusieurs années ont le même nombre de film, une quelconque de ces années)
12. Pour chaque acteur, une table donnant le prenom et nom de l'acteur et son nombre de films.
13. Une liste donnant pour chaque acteur les films dans lesquels il a joué, tiré par prenom puis nom puis titre de film.
14. Pour chaque acteur (prenom et nom), la durée entre son film le plus récent et son film le plus ancien,
15. Tous les gens (prenom,nom) qui ont joué avec Al Pacino (sans doublons)

16. La liste des acteurs (prenom,nom) ayant joué pour deux réalisateurs différents (sans doublons)

> **Corrigé**: Voir le fichier `exo_req.sql`

Exercice 3 : Motifs d'accès en lecture écriture

Remarque Pour cet exercice, il est important que les fichiers que vous créez soient dans le répertoire `/var/tmp/` de la machine. En effet, il ne faut **surtout pas** les créer dans votre *home directory*, ce dernier se trouvant sur une partition réseau, vous risquer *(i)* de fortement dégrader les performances de votre machine et *(ii)* d'exploser votre quota de stockage. Les fichiers crées dans `/var/tmp` seront effacés lors du redémarage de votre poste.

1. Exécuter les instructions suivantes :
   (a) Placez vous dans le répertoire `/var/tmp` (`cd /var/tmp`)
   (b) Créez un fichier de 512Mo ne contenant que des octets nuls. Pour celà effectuez la commande : `dd if=/dev/zero of=file1.bin bs=1M count=512` (voir ci-dessous pour l'explication)
   (c) Exécutez la commande `sudo /18_64/drop_caches` (voir ci-dessous pour l'explication, vous devez rentrer votre mot de passe Unix)
   (d) Effectuez une première fois la commande : `dd if=file1.bin of=/dev/null` et notez la vitesse de transfert
   (e) Exécutez une deuxième fois la commande ci-dessus. Pourquoi la vitesse de transfert a-t-elle augmenté ?

> **Corrigé**: Après avoir exécuté `drop_cache` plus aucun fichier ne reside en cache dans la mémoire. La première lecture se fait depuis le disque. Lors de la deuxième lecture, les pages du fichier sont en cache en mémoire, les données sont lues depuis la mémoire sans accès disque.

2. Exécuter les instructions suivantes :
   (a) Placez vous dans le répertoire `/var/tmp` (`cd /var/tmp`)
   (b) Créez un fichier de 512Mo en utilisant le script fournit dans l'archive du TP. En supposant que vous avez placé le répertoire contenu dans l'archive à la racine de votre *home directory*, `~/bd_app4_tp1/create_file.sh 512`
   (c) Exécutez la commande `sudo /18_64/drop_caches` (voir ci-dessous pour l'explication, vous devez rentrer votre mot de passe Unix)
   (d) Effectuez une première fois la commande : `dd if=file_512.bin of=/dev/null` et notez la vitesse de transfert
   (e) Répétez les points *(c)* et *(d)* ci-dessus plusieurs fois et calculez la moyenne des vitesses de transferts. Comparez à la vitesse obtenu précédemment et proposer un scenario qui explique la différence (alors que les deux fichiers lu font la même taille). Une lecture du code du script `create_file.sh` peut s'avérer utile.

Corrigé: Le script crée le fichier cible par petit bout de 1Mo, mais crée un fichier temporaire de 100Mo sur le disque et en force l'écriture entre chaque bouts de 1Mo. On peut donc supposer que les données du fichier final ne sont pas organisée de manière continue sur le disque. Une fois avoir vidé les caches, une lecture du fichier depuis le disque est effectuée. La vitesse moindre s'explique par le fait que le disque doit faire plus de déplacement pour passer de morceau en morceau.

Indications

La commande `dd` permet de prendre un fichier source (`if=...`) et un fichier cible (`of=...`) et de copier le contenu du fichier source dans le fichier de sortie, créant ce dernier au besoin.

Le fichier `/dev/zero` est un fichier virtuel sous linux qui se comporte comme un fichier de taille infinie contenant uniquement l'octet 00.

Le fichier `/dev/null` est un fichier virtuel sous linux qui se comporte comme un « puit ». Tout ce qui est écrit dedans est perdu.

La commande `drop_caches` (qui nécessite certains privilèges administrateurs donnés par la commande `sudo`) vide complètement les caches disques se trouvant en mémoire (*i.e.* toutes les pages des fichiers *bufferisées* en mémoire sont écrite sur le disque si besoin et la mémoire est libérée).