info710 : Compléments de bases de données
TP 3 : gestion d'un catalogue dans une librairie

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

Remarque : la documentation de PostgreSQL se trouve en ligne sur
`http://www.postgresqlfr.org/`
et le livre "Practical PostgreSQL" se trouve en ligne sur
`http ://www.faqs.org/docs/ppbook/book1.htm`

Ce TP est la première partie du cycle de 3 TPs qui sera noté. Faites donc attention, car certaines décisions prises lors de ce TP pourront se révéler handicapantes par la suite...

Paradoxalement, peu de points seront vraiment donnés sur ce TP : il s'agit uniquement de poser les bases...

Allez lire les remarques à la fin du sujet.

Exercice 0 : le problème

Vous recevez la lettre suivante :

> Cher Mademoiselle (ou madame, ou monsieur),
>
> je suis le gérant d'une grosse librairie et je voudrais passer à l'informatique pour faciliter la gestion des stock, des clients et des commandes.
>
> Vous savez, comme à la FNAC : ils regardent sur leur ordinateur, et toute l'information est là.
>
> J'aimerais faire pareil et M Hyvernat m'a conseillé de m'adresser à vous.
>
> Voici quelques exemples de choses qui pourraient m'être utile :
> - pour chaque livre, savoir si je l'ai en stock, ou s'il est en commande, ou bien épuisé,
> - pour chaque livre, connaître le prix et le fournisseur (si jamais je dois le commander)
> - avoir les informations sur mes fournisseurs (adresse, nom de mon interlocuteur, etc.)
> - une liste de mes clients fidèles (je distribue des cartes de fidélité, et je note les goûts de chacun pour pouvoir les conseiller)
> - pouvoir avoir facilement accès aux livres selon des critères de prix, d'auteur, de genre, de type (poche), de langue etc.
> - j'aimerais aussi pouvoir mettre où sont stockés les livres que j'ai en rayon
> - il faudrait aussi garder le numéro ISBN pour chaque livre. Je peux vous expliquer si vous le souhaitez...
> - comme je fais régulièrement une devanture "meilleures ventes", il faudrait savoir combien d'exemplaires de chaque livre a été vendu
>
> Je tenais aussi à préciser que ni moi, ni mes employés ne sommes informaticiens : si vous pouviez faire quelque chose de relativement simple à utiliser, cela serait encore mieux.
>
> Merci d'avance
>
> Victor Dictionnaire

Question 0. Comme vous pouvez le constater, le demande du client est assez vague. La première étape sera donc de spécifier ce que vous allez faire, en essayant de satisfaire au mieux M Dictionnaire.

Commencez donc par décrire l'architecture de la base de données que vous allez proposer à V. Dictionnaire, et vérifiez avec lui que cela convient.

Exercice 1 : création de la base, contraintes

Question 1. Chacun de vous est "propriétaire" d'une base de données appelée `tp710-login` sur le serveur `eco.univ-savoie.fr`. Loggez vous sur le serveur, et à l'aide de `psql`, créez les tables adéquates.

Pensez en particulier à mettre toutes les contraintes possibles sur les tables (un livre doit avoir au moins un auteur, un auteur doit avoir écrit des livres, un éditeur doit exister, etc.)

Remarque : utilisez un éditeur de texte pour copier/coller vos commandes. Vous pourrez ainsi facilement recommencer si vous effacez une table par erreur.

Question 2. Peuplez vos tables avec quelques éléments afin de pouvoir effectuer des tests.

Remarque : si une contrainte de clé étrangère est déclarée avec l'adjectif "`DEFERRABLE`", on peut reporter son application à un moment ultérieur :

```
BEGIN ;
SET CONSTRAINTS constraint_name DEFERRED ;
...
-- ici, la contrainte n'est pas vérifiée
...
SET CONSTRAINTS constraint_name IMMEDIATE ;
COMMIT ;
```

Question 3. Vous pouvez facilement ajouter, supprimer ou modifier une contrainte nommée en utilisant

```
ALTER TABLE nom table [DROP,ADD,RENAME,MODIFY] CONSTRAINT ... ;
```

Testez ces commandes en rajoutant, supprimant ou modifiant certaines contraintes.

Exercice 2 : les transactions

Il est parfois utile d'effectuer des requêtes "à la suite", en garantissant que rien ne se passe entre les différentes étapes : c'est une *transaction*. On commence une transaction avec l'instruction "`BEGIN ;`", et on enregistre les changements avec l'instruction "`COMMIT ;`".

Parfois, il est utile de revenir en arrière, parce que quelque chose c'est mal passé : on peut mettre un "point de sauvegarde" dans une transaction avec l'instruction "`SAVEPOINT nom ;`". Pour revenir en arrière, on utilise l'instruction "`ROLEBACK TO SAVEPOINT nom ;`". L'instruction "`ROLLBACK ;`" permet de revenir au début de la transaction en cours, càd de l'annuler.

Question 1. Commencez une transaction et ajoutez des valeurs dans vos tables.

Rajoutez une valeur qui viole une des contraintes. Essayez de faire un `SELECT`. Que se passe-t'il ?

Terminez la transaction ; faite un `SELECT`. Que se passe-t'il ?

Question 2. Commencez une transaction et ajoutez des éléments ; ajoutez un point de sauvegarde.

Ajoutez un élément qui viole une transaction.

Revenez en arrière puis supprimez un élément.

Revenez en arrière puis terminez la transaction. Que se passe-t'il ?

Exercice 3 : les fonctions SQL (section 33.4 du manuel)

Il est parfois utile de définir des fonctions pour faire des requêtes SQL que l'on utilise souvent.

Pour créer une fonction qui prend des arguments de type `t1`, `t2`, ... et renvoie un résultat de type `t`, on utilise la syntaxe

```
CREATE OR REPLACE FUNCTION f (t1,t2,...) RETURNS t AS
 '...' language SQL ;
```

Pour faire référence au premier argument dans le corps de la fonction, on utilise `$1` ; pour le deuxième, on utilise `$2` ; etc.

Allez voir l'aide pour plus de détails !

Question 1. Créez une fonction `livre_par_auteur` qui prend deux arguments :
- un nom d'auteur
- et un début de titre

et renvoie les vingt premiers livres (auteur, titre, prix et état des stocks) de cet auteur dont le titre commence par le deuxième argument.

Remarque : pour les comparaisons de chaînes de caractères, allez voir la section 9.7 du manuel.

Question 2. Améliorez la fonction précédente pour que la recherche du titre puisse se faire avec un mot apparaissant n'importe où dans le titre, et que l'on n'est pas sûr de l'orthographe du nom de l'auteur.

Question 3. Écrivez les fonctions suivantes :
- `nouveau_livre` qui permet de rajouter un livre dans la base (quels arguments ?)
- une fonction `nouveau_livre_auteur` qui permet de créer un nouveau livre avec un nouvel auteur
- une fonction `vente_livre` que l'on utilisera quand un livre sera vendu

Quelles autres fonctions pourriez-vous avoir envie de rajouter ?

Remarques :

Le sujet de ce TP est volontairement très elliptique : c'est à vous de prendre des décisions. De même, plusieurs concepts ne sont pas expliqués. Vous devrez chercher l'information où elle se trouve (documentation de PostgreSQL, internet, ...) Là aussi, c'est volontaire. Vous devez devenir autonomes !

La note finale sera basée sur plusieurs choses :
- des fichiers SQL comprenant vos définitions, ainsi qu'une base d'exemples pour faire des essais,
- un fichier *texte* (pas de Word, de pdf ou autre format binaire) comportant un petit compte-rendu. Ce rapport (quelques pages au plus) devra m'expliquer ce que vous avez fait, quelles décisions vous avez prises et pourquoi, les problèmes rencontrés et les améliorations envisageables si vous aviez le temps,
- un fichier *texte* qui explique à V. Dictionnaire comment se servir de l'outil que vous lui proposez. (Ce document est donc destiné à des non-informaticiens...)