R3.07 - SQL dans un langage de programmation
2023/2024

TD/TP 2 : Banque (suite)

Note: L'objectif de ce TP est de vous familiariser avec l'utilisation des structures de contrôle et de vous permettre de manipuler des curseurs en PL/SQL.
Assurez-vous de regrouper les scripts à la fin du TP sous forme d'un fichier compressé (.zip). Vous devez soumettre votre travail via l'espace de rendu dédié au TP sur Moodle, correspondant à votre groupe.

Création et manipulation des comptes :

1. Dans un script `"test_tp2.sql"`, tenter de supprimer le compte numéro `19`. Qu'observez-vous ?
2. Dans un script `"tables_bis.sql"`, modifier les contraintes pour que la suppression d'un compte entraîne également la suppression de son appartenance à un client.
3. Dans le script `"test_tp2.sql"`, tester la suppression du compte `19` et annuler-la avec un `ROLLBACK`. Est-ce que la suppression fonctionne maintenant ?
4. Dans le script `"tables_bis.sql"`, modifier les contraintes pour que la suppression d'un client entraîne également la suppression de ses appartenances.
5. Rédiger un script `"problemes.sql"` comprenant :
   a) une procédure anonyme **sans curseur** qui affiche le nombre de comptes n'ayant pas de client, s'il y en a.
   Résultat attendu :
   ```
   Attention, 2 comptes n'ont pas de client!
   ```
   b) une procédure anonyme **avec curseur** qui, en plus du nombre de comptes, affiche tous les numéros de comptes n'ayant pas de client, s'il y en a.

   Résultat attendu :
   ```
   Le compte 8 n'a pas de client!
   Le compte 20 n'a pas de client!
   Attention, 2 comptes n'ont pas de client!
   ```
6. Dans un script `"vues_bis.sql"`, créer une vue `"Compte_Client"` contenant les comptes avec les numéros de leurs clients, y compris ceux sans client.
7. Dans le script `"test_tp2.sql"` :

   a) Insérer dans la vue le compte suivant : `numCompte=21, typeCompte=EPARGNE, solde=50, numClient=1`. Qu'observez-vous ?

   b) Utiliser la vue pour supprimer les comptes n'ayant pas de numéro de client. Qu'observez-vous ?

   c) Utiliser la vue pour affecter les comptes n'ayant pas de numéro de client au client numéro `1`. Qu'observez-vous ?

Affichage avec curseur :

8. Rédiger le script `"affichage_avecCurseur.sql"` afin de créer deux procédures anonymes qui affichent :
   a) les 5 derniers retraits, du plus récent au plus ancien.
   Résultat attendu (similaire à) :
   ```
   L'opération 13 a été effectuée le 28-NOV-23.
   L'opération 5 a été effectuée le 27-NOV-23.
   L'opération 6 a été effectuée le 27-NOV-23.
   L'opération 1 a été effectuée le 26-NOV-23.
   L'opération 8 a été effectuée le 26-NOV-23.
   ```
   b) pour chaque agence, le nombre de clients n'ayant pas encore de compte épargne.
   Résultat attendu :
   ```
   L'agence 1 compte 3 client(s) sans compte épargne.
   L'agence 2 compte 4 client(s) sans compte épargne.
   L'agence 4 compte 0 client(s) sans compte épargne.
   L'agence 3 compte 6 client(s) sans compte épargne.
   ```
9. Dans le script `"problemes.sql"`, ajouter deux procédures anonymes affichant les éléments violant les deux contraintes suivantes :
   a) Une agence a nécessairement un et un seul directeur.
   Résultat attendu :
   ```
   L'agence 2 a 2 directeurs!
   L'agence 4 n'a pas de directeur!
   ```
   b) Le directeur d'une agence est mieux payé que les agents de son agence.
   Résultat attendu :
   ```
   Tout est bon!
   ```

Modification avec curseur :

10. Créer un script nommé `"augmentation.sql"` qui inclut une procédure anonyme utilisant un curseur pour mettre à jour les salaires des agents. L'augmentation est de `5 %` pour les directeurs et de `1 %` pour les autres agents.
11. Créer également un script appelé `"interets_epargne.sql"` contenant une procédure anonyme qui utilise un curseur pour obtenir les soldes globaux de tous les comptes d'épargne des clients. Ensuite, utiliser un curseur paramétré pour mettre à jour les soldes des comptes d'épargne. L'augmentation dépend du solde total, avec une augmentation de `5 %` pour un solde total supérieur ou égal à `100 000 euros`, de `3 %` pour un solde total entre `10 000` et `100 000 euros`, et de `1 %` pour un solde total inférieur à `10 000 euros`.

Exceptions/curseurs anonymes et paramétrés (solde.sql) :

12. Réécrire le script `solde.sql` du `TP1` pour inclure un bloc `PL/SQL` interactif qui permet à l'utilisateur de saisir le numéro d'un client. Ce bloc doit contenir un curseur paramétré, prenant en paramètre le numéro du client et le type de compte (avec `'EPARGNE'` comme type de compte par défaut). Ce curseur doit fournir la liste des soldes des comptes d'un type donné pour un client donné.
Le bloc doit toujours débuter en affichant le nom complet du client. Ensuite, avec un curseur anonyme, il doit extraire les types de compte. Ensuite, le bloc procédera à l'affichage des soldes des comptes de chaque type, avant d'afficher le solde total de tous les comptes. Si le numéro de compte est négatif, une exception définie par l'utilisateur doit être déclenchée.
Dans la section des exceptions, gérer le plus d'exceptions possible pour afficher des messages appropriés.

Résultat attendu (si l'on saisit 1):

```
Nom du client : Dumas Emma
----------------COURANT-------
50
3000
----------------EPARGNE-------
18500
Solde total des comptes : 21550
```

Résultat attendu (si l'on saisit -1):

```
Le numéro client ne doit pas être négatif!
```

Résultat attendu (si l'on saisit 30):

```
Aucun client ne porte ce numéro!
```

Résultat attendu (si l'on saisit a):

```
Le numéro doit être un nombre!
```