TP4 : Requêtes SQL
Création d'un schéma relationnel

15 mars 2010

Assurez-vous d'avoir pris connaissance du petit guide SQLPlus.

Durant ce TP, nous allons définir un schéma de base de données, y ajouter des contrainre, des vues, des déclencheurs (*triggers*), ainsi qu'un certain nombre d'informations.

La base de données s'intitule *Agence de voyages*, et propose de décrire les activités de différents client lors d'un séjour dans une station balnéaire.

1 Création des tables

Voici le schéma *Agence de voyages* :

- STATION (nomStation,capacité, lieu, région, tarif)
- ACTIVITE (nomStation, libellé, prix)
- CLIENT (id, nom, prénom, ville, région, solde)
- SEJOUR (id, station, début, nbPlaces)

Voici les autres contraintes portant sur ces tables.

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', 'Ameriques' et 'Extreme Orient'.
5. La destruction d'une station doit entraîner la destruction de ses activités et de ses séjours.

Créer les tables du schéma *Agence de voyages*. On veillera à bien définir les clés primaires et étrangères, et à donner des noms explicites au contraintes `check`, `primary key` et `foreign key`.

2 Insertion de données

Insérer dans la base les données ci-dessous :

STATION

NomStation	Capacité	Lieu	Région	Tarif
Venusa	350	Guadeloupe	Antilles	1200

ACTIVITES

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	6763
30	Kerouac	Jack	New York	Amérique	9812

SEJOUR

idClient	station	début	nbPlaces
20	Venusa	1998-08-03	4

Tester ensuite les contraintes avec quelques ordres SQL. Par exemple : détruire la station et vérifier que les activités ont disparu ; insérer une autre station en (Guadeloupe, Antilles) ; insérer une station dans une région 'Nullepart', etc.

3 Vues

1. Créer les vues suivantes sur le schéma précédent.
   (a) Une vue `ActivitesModiques (Station, Activite)` donnant le nom des stations et des activités dont le prix est inférieur à 140 euros. Toute ligne insérée dans cette vue doit apparaître dans la vue ensuite.

   (b) Une vue `ActivitesCheres`, de même schéma, avec prix supérieur à 140 euros, et la même contrainte d'insertion.

   (c) Une vue `StationFrancs (Nom, Capacite, Lieu, TarifFrancs)` donnant le nom d'une station, sa capacité, le lieu et le tarif en francs (un euro=6,58 FF).
   (d) Une vue `Tarifs (Station, Tarif, OptionMin, OptionMax)` donnant, pour chaque station, le tarif et les prix min et max des activités.
2. Consulter ensuite le contenu de ces vues. On pourra insérer quelques lignes supplémentaires dans les tables et constater qu'elles sont prises en compte dans les vues.
3. Dans quelles vues peut-on insérer, détruire et mettre-à-jour ? Essayer les opérations suivantes :
   (a) Insérer une activité 'Kayac' pour la station 'Venusa' dans `ActivitesCheres` et `ActivitesModiques`. Le contrôle sur l'insertion est-il utile dans ce cas ?
   (b) Peut-on insérer dans StationFrancs ? Sous quelle condition ? Faire l'essai.
   (c) Détruire une ligne de StationFrancs.

4 Déclencheurs

Les *triggers* sont des ordres de déclenchement d'opérations quand un événement survient sur une table. Ils sont souvent utilisés pour assurer la cohérence des données dans la base, en réalisant des contraintes qui doivent porter sur plusieurs tables. La syntaxe est la suivante :

Le *trigger* désigne une fonction funcname qui doit se déclencher avant ou après un événement sur un enregistrement d'une table. Les événements susceptibles de déclencher un trigger sont `INSERT`, `UPDATE` et `DELETE`.

Dans le cas *FOR EACH ROW*, si plusieurs enregistrements sont touchés par une insertion suppression ou modification, alors le *trigger* sera exécuté autant de fois qu'il y a d'enregistrements concernés. Dans le cas *FOR EACH STATMENT* c'est une seule exécution pour tout le groupe d'enregistrements concernés qui sera réalisée.

Si le *trigger* doit se déclencher après l'événement, le *trigger* aura accès à plusieurs informations. Sinon, il aura la possibilité d'annuler l'opération qui a déclenché l'événement.

Attention, la création de *triggers* peut réaliser des boucles infinies très facilement, (si l'insertion d'un enregistrement provoque l'insertion d'un autre enregistrement...).

Indication importante : la dernière ligne de création d'un *trigger* doit être un `END ;`, suivie d'une ligne contenant seulement `/`.

1. Implantez par un *trigger* la règle suivante : si le prix d'une activité baisse, alors le tarif de la station doit augmenter de la différence.
   Indication : le *trigger* doit se déclencher sur une modification, et tester pour chaque ligne que la nouvelle valeur est plus grande que l'ancienne. Si ce n'est pas le cas, faire un `UPDATE` de la station pour ajouter la différence entre l'ancienne et la nouvelle valeur.

2. Faites l'expérience : diminuez le prix d'une activité, et regardez ce qui se passe pour la station.
3. On veut disposer de l'information `nbActivites` dans la table `Station`. Pour cela :
   (a) Ajouter la colonne `nbActivites` avec pour valeur par défaut `0`.
   (b) Créer un *trigger* qui maintient cette information.