BASE DE DONNEES ET SYSTEMES DE GESTION DE BASE DE DONNEES (SGBD)

CHAPITRE VIII : GESTION DES DONNEES

Sommaire

INTRODUCTION

Il est nécessaire d'effectuer régulièrement des modifications sur la base de données.

Le schéma conceptuel ne sera que rarement altéré (par les commandes CREATE, ALTER, DROP). Par contre, les données stockées dans les tables nécessiteront de fréquentes mises à jour. Trois opérations le permettent : INSERT, UPDATE, DELETE.

INSERT

Lorsqu'elle s'applique aux tables (son action sur les vues est expliquée au chapitre suivant), cette commande est formulée suivant la syntaxe :

```
INSERT INTO nom_table [ ( nom_col1 [ , nom_col2 ,... ] ) ]

< SELECT ... VALUES ( constante1 [ constante2 ... ] ) > ;
```

Elle permet d'insérer des tuples dans une table.

AJOUT D'UNE LIGNE

Le choix de l'option VALUES permet d'ajouter une ligne dans la table « nom_table», chaque constante correspondant à la colonne de même rang.

Ajoutons l'enregistrement d'un passager à la table RESERVATIONS, avec no_pass=27, no_vol=789, origine = 'BRU' et destin = 'CHI' :

```
INSERT INTO reservations (no_pass, no_vol, origine, destin)
    VALUES (27,789,'BRU','CHI') ;
```

Note :

Il n'est pas obligatoire d'indiquer une valeur pour chaque colonne de la table.

Les colonnes non citées recevront la valeur NULL.

Se pose alors immédiatement le problème de l'intégrité référentielle de la base de données : rien ne garantit au départ que le passager 27 existe, ni le vol 789.

Dans certains systèmes relationnels existants, ce problème n'est pas résolu. Aussi est-il nécessaire de faire attention lors d'une insertion, ou alors de placer cette insertion dans une routine qui fera la vérification de l'intégrité.

AJOUT D'UN ENSEMBLE DE LIGNES

En se servant de l'option SELECT de la commande INSERT, on peut introduire plusieurs tuples à l'aide d'une seule commande d'insertion.

Créons à cette fin une table PASSPARVOL, qui indiquera pour chaque vol, le nombre de passagers enregistrés :

```
CREATE TABLE passparvol
    (no_vol VARCHAR(13),  nbre  INTEGER);
```

On pourra y insérer les tuples voulus par :

```
INSERT INTO passparvol
    SELECT   no_vol , COUNT(*)
        FROM   vols  GROUP  BY  no_vol ;
```

Restriction :

La table citée dans la clause INTO ne peut être la même que dans la clause FROM.

AJOUT D'UN ENSEMBLE DE LIGNES (DEUXIEME SOLUTION)

Il est possible d'insérer un ensemble important de lignes dans une table en se servant de VARIABLES PROGRAMME ou variables liées (cf. le paragraphe sur les variables programme). Plutôt que de préciser des constantes comme arguments à VALUES, on y place des variables qui font chacune référence à une composante des lignes qui suivent. Dans l'exemple ci-dessous, on introduit une série de nouveaux passagers dans la table PASSAGERS.

```
INSERT  INTO  passagers (no_pass, prenom, nom, localite)
VALUES  ( :1, :2, :3, :4)
\
235, 'HENRI', 'LEGRAND', LIEGE'
236, 'ALAIN', 'VANEETVELD', 'NAMUR'
237, 'DIDIER', 'SMITH, 'GAND'
/
```

Avec MySQL, faire :

```
INSERT  INTO  passagers (no_pass, prenom, nom, localite)
        VALUES
                ('235','HENRI','LEGRAND','LIEGE'),
                ('236','ALAIN','VANEETVELD','NAMUR'),
                ('237','DIDIER','SMITH','GAND') ;
```

MISE A JOUR DE TUPLES : UPDATE

La syntaxe générale de la commande UPDATE appliquée à une table est :

```
UPDATE  nom_table  [ nom_corrélation  ]
SET  nom_col1 = <expression  NULL >
        [,   nom_col2 = <expression NULL > ,... ]
[WHERE  < condition  >] ;
```

La valeur d'une ou de plusieurs colonnes d'une table est ainsi modifiée sur la base d'une condition logique qui permet de sélectionner les tuples qui seront mis à jour.

Exemples de mises à jour :

Modifions la date de départ et d'arrivée du vol SN532 :

```
UPDATE vols SET date_dep = CAST('1988-09-25' AS DATE),
date_arr=CAST('1988-09-26' AS DATE)
 WHERE no_vol='SN532' ;
Ou
UPDATE vols SET date_dep = '1988-09-25', date_arr = '1988-09-26'
        WHERE no_vol='SN532' ;
```

Retardons de deux jours les vols à destinations de Bruxelles :

```
UPDATE vols SET date_dep = date_dep+2, date_arr = date_arr+2
        WHERE destin ='BRU' ;
```

Note :

Des problèmes d'intégrité référentielle peuvent également découler des mises à jour. On veillera donc, lors de l'exécution des commandes ci-dessus, à mettre à jour les autres tables concernées.

SUPRESSION D'UN TUPLE : DELETE

Pour supprimer un tuple dans une table, on utilise la commande DELETE dont la syntaxe est :

```
DELETE FROM nom_ table [nom_ corrélation]
        [WHERE <condition>];
```

Les lignes de la table "nom_table" qui vérifient la condition seront supprimées.

Si la clause WHERE est omise, tous les tuples de la table seront éliminés.

Exemple :

Eliminons les vols dont la date de départ est '1988-09-25 10:15:00' :

```
DELETE FROM vols WHERE date_dep = '1988-09-25 10:15:00' ;
```

Remarque :

La commande DELETE est, comme les précédentes une source potentielle de problème d'intégrité par l'incorporation de la commande dans une routine.