SQL

Ch 7 – DML
Ajouter, supprimer, maj

I. Introduction

Les ordres d'ajout de lignes, de mise à jour des valeurs de colonnes et de suppression de lignes interviennent au niveau du CONTENU des tables.
Ces ordres déclenchent

- l'APPLICATION DES CONTRAINTES D'INTEGRITE spécifiés lors de la création des tables
- les mécanismes de GESTION DE TRANSACTION afin de garantir l'intégrité des données lors d'accès concurrent aux données
- les mécanismes de JOURNALISATION permettant de garantir l'intégrité des données en cas d'erreur en cours de transaction ou en cas de nécessité de restauration d'une base à un état antérieur avec ré-application des transactions .

II. INSERT INTO : insérer des lignes

L'ordre INSERT permet l'ajout de lignes dans une table.
Dans ce cas de requête d'insertion, les listes des valeurs sont fournies par des valeurs constantes ou des expressions contenant des fonctions.
Ces listes de valeurs doivent correspondre à toutes les colonnes de la table, sinon une liste des noms de colonnes correspondantes doit être précisée.

A. Insérer une ligne complète à partir de valeurs fixes

LA VALEUR DE CHACUNE DES COLONNES DOIT ETRE DONNEE DANS LEUR ORDRE D'APPARITION DANS LA TABLE

Syntaxe générale :

```
INSERT INTO nom_table
   VALUES (valeur1, valeur2, valeur3, …)[,
   VALUES (valeur1b, valeur2b, valeur3b, …), etc.]
;
```

Par exemple : ajouter un membre

```
INSERT INTO membre
  VALUES (1,'dupont','pierre','bld Gambetta' , 'arras' ,
'20/10/1975',NULL);
```

B. Insérer une ligne partielle à partir de valeurs fixes

LES VALEURS DOIVENT ÊTRE FOURNIES DANS LE MÊME ORDRE QUE LA LISTE DES COLONNES.
Certaines colonnes resteront sans valeur (NULL, si accepté…) ou bien avec leur valeur par défaut.

Syntaxe générale :

```
INSERT INTO nom_table (col1, col2, col3, ...)
  VALUES (valeur1, valeur2, valeur3, …)
;
```

Par exemple : ajouter un membre

```
INSERT INTO membre (prenom, nom, rue, ville, datnais)
  VALUES ('jacques', 'dumoulin','bld Gambetta' , 'arras'
, '25/01/1980');
```

C. Insérer des lignes à partir d'une requête

Dans ce cas de requête d'insertion, les listes de valeurs sont fournies par une requête SELECT complète.
Ces listes de valeurs doivent correspondre à toutes les colonnes de la table, sinon une liste des noms de colonnes correspondantes doit être précisée.

Syntaxe générale :

```
INSERT INTO nom_table  [(col1, col2, col3, ...)]
  SELECT nomCol1, nomCol2, nomCol3,...
    FROM nom_table1
    [WHERE conditions]
    [GROUP BY colonnes]
    [HAVING conditions]
    [ORDER BY colonnes]
    [LIMIT nombre]
;
```

III.UPDATE : Mettre à jour des valeurs de colonnes

L'ordre UPDATE permet la mise à jour de valeurs de colonnes dans une table[1] selon la valeur d'une condition exprimée dans la clause WHERE.

[1] Certains SGBD permettent d'intégrer des jointures : « UPDATE tables jointures SET nomColonnes = nouvelleValeur X WHERE criteres ; »

ATTENTION : UN ORDRE DE MISE A JOUR SANS CLAUSE 'WHERE' AFFECTE TOUTE LES LIGNES DE LA TABLE.

Syntaxe générale :

```
UPDATE nom_table
SET col1 = valeur1 [,col2 = valeur2, col3 = valeur3,...]
[WHERE expression_condition]
;
```

La condition spécifiée peut être exprimée :
- Par rapport aux colonnes de la table mise à jour
- Par rapport à une requête imbriquée indépendante ou corrélée

Mettre à jour (corriger) le prénom d'un membre :

```
UPDATE membre
  SET prenom = 'john'
  WHERE id_memb = 10 ;
```

Augmenter le tarif des activités (de TOUTES les activités) de 10% :

```
UPDATE activite
  SET tarif = tarif * 1.1;
```

Augmenter de 10% le prix des oeuvres des artistes nés avant 1900 :

```
UPDATE t_oeuvre
  SET prix = prix * 1.1
  WHERE numartiste IN ( SELECT numartiste FROM t_artiste
    WHERE anneeNaissance < 1900 )
;
```

IV. DELETE FROM : Supprimer des lignes

L'ordre DELETE permet la suppression d'une ou plusieurs lignes d'une table selon la valeur d'une condition.

ATTENTION : UN ORDRE DE SUPPRESSION SANS CLAUSE 'WHERE' SUPPRIME TOUTE LES LIGNES DE LA TABLE[2].

Syntaxe générale :

```
DELETE FROM nom_table
  [WHERE expression_condition];
```

La condition spécifiée peut être exprimée :
- Par rapport aux colonnes de la table cible de la suppression de lignes
- Par rapport à une requête imbriquée indépendante ou corrélée

Supprimer les inscriptions du membre de numéro 10 :

```
DELETE FROM inscrire
  WHERE id_memb = 10 ;
```

[2] A moins que des contraintes d'intégrité référentielles l'en empêche

V. TRUNCATE TABLE : Vider une table

L'ordre TRUNCATE TABLE permet la suppression de TOUTES LES LIGNES D'UNE TABLE sans conservation de la trace de la suppression (SANS JOURNALISATION).

ATTENTION : LES SUPPRESSIONS DE LIGNES NE SONT PAS JOURNALISEES (A EVITER)

(un ordre ROLLBACK ne permet pas d'annuler cette suppression de lignes)

Syntaxe générale :

```
TRUNCATE TABLE nom_table;
```

L'ordre TRUNCATE TABLE a été invalidé dans les dernières versions de MySQL à cause de sa dangerosité.

VI. Validation ou abandon des mises à jour effectuées

Les mises à jour demandées au SGBD ne sont effectivement vraiment enregistrées dans la base de données que lorsque demande leur validation explicitement[3].

Par exemple, lors de l'ajout d'une commande à un client, on souhaite insérer une ligne de commande (INSERT INTO commande …) puis mettre à jour le client (UPDATE client SET … WHERE …)

A. Valider les mises à jour effectuées : COMMIT

Permet de confirmer la mise à jour des données dans la base de données (depuis le dernier ordre COMMIT)

Syntaxe générale :

```
COMMIT [TRANSACTION];
```

B. Annuler les mises à jour effectuées : ROLLBACK

Permet d'annuler la mise à jour des données dans la base de données (depuis le dernier ordre COMMIT).

Syntaxe générale :

```
ROLLBACK [TRANSACTION];
```

C. Concernant COMMIT et ROLLBACK : SET TRANSACTION

ATTENTION : dans certains cas, une instruction COMMIT est exécutée automatiquement après chaque ordre DML (on parle d'AUTO-COMMIT). C'est le cas notamment des SGBD installés dans le cadre de certains serveurs Web pour le grand public.

Dans ce cas, il ne sera plus possible d'annuler un ensemble d'ordres SQL DML représentant une transaction.

Certains SGBD proposent l'ordre SET TRANSACTION qui permet de démarrer explicitement une transaction « longue » (plusieurs ordres SQL).

[3] A moins qu'une option « auto commit » ait été définie : chaque instruction d'un groupe d'instructions est alors automatiquement validée isolément.