Le Langage SQL

Traitement des données

Le Langage de Manipulation de Données

L'ajout, la suppression et la modification de données

Introduction

- **Les ordres SQL :**
  - **INSERT**
  - **UPDATE**
  - **DELETE**
- **Ces instructions SQL influent directement sur les données, contrairement à l'instruction SELECT**

L'insertion de données (1)

La syntaxe générale

INSERT INTO nom_de_la_table_cible **[(**liste_des_colonnes_visées**)]** **VALUES** **(**liste_des_valeurs**)**

[] ***optionnel dans la requête***

3 types d'insertion

- **Insertion explicite**
- **Insertion multiple**
- **Insertion à base de sous requête SELECT**

L'insertion de données (2)

- **Insertion simple explicite**

  On insère directement les valeurs dans les colonnes de la table,

  !! insertion d'une seule ligne !!

  Exemple :

  insertion d'une ligne dans la table T_MODE_PAIEMENT,
  elle contient deux colonnes (code et libelle)

```
INSERT INTO T_MODE_PAIEMENT (CODE, LIBELLE)
                    VALUES('VIR','virement bancaire');
```

OU

```
INSERT INTO T_MODE_PAIEMENT
                    VALUES('VIR','virement bancaire');
```

L'insertion de données (3)

- **Insertion multiple**

  On insère plusieurs lignes dans la table cible,

  Exemple :
  insertion de trois lignes dans la table T_MODE_PAIEMENT, elle contient deux colonnes (code et libelle)

```
INSERT INTO T_MODE_PAIEMENT (CODE, LIBELLE)
            VALUES      ('VIR','virement bancaire'),
                        ('CB', 'Carte Bancaire'),
                        ('CHQ','Chéque');
```

L'insertion de données (4)

Insertion partiellement explicite

Exemple de création de table

```
CREATE TABLE  CONNEXION
    (
        C_USER     VARCHAR(128)                NOT NULL  DEFAULT 'Administrateur',
        DATE_HEURE  TIMESTAMP     NOT NULL  DEFAULT CURRENT_TIMESTAMP
    );
```

```
INSERT INTO CONNEXION (C_USER, DATE_HEURE)  VALUES('Dupont', DEFAULT);
```

c_user	date_heure
Dupont	2015-02-05 11:18:22.654

```
INSERT INTO CONNEXION    VALUES('Durant', DEFAULT);
```

c_user	date_heure
Dupont	2015-02-05 11:18:22.654
Durant	2015-02-05 11:19:37.743

```
INSERT INTO CONNEXION (C_USER)  VALUES('Duchemol');
```

c_user	date_heure
Dupont	2015-02-05 11:18:22.654
Durant	2015-02-05 11:19:37.743
Duchemol	2015-02-05 11:20:33.544

```
INSERT INTO CONNEXION  VALUES ( DEFAULT, DEFAULT);
```

c_user	date_heure
Dupont	2015-02-05 11:18:22.654
Durant	2015-02-05 11:19:37.743
Duchemol	2015-02-05 11:20:33.544
Administrateur	2015-02-05 11:21:29.957

L'insertion de données (5)

Insertion à base de sous requêtes

```
-- sous requête qui retourne une seule colonne et une seule ligne
        INSERT INTO CONNEXION
                VALUES( (select NOM from CLIENT where CLI_ID=4), default);

-- sous requête qui retourne plus d'une ligne
        INSERT INTO   CLIENT (CLI_ID, CODE, NOM, PRENOM)
                SELECT   PRP_ID, PRP_CODE_TITRE, PRP_NOM, PRP_PRENOM
                FROM   T_PROSPECT ;
```

La suppression de données (1)

- **La syntaxe générale**

```
DELETE FROM nom_table_cible
[WHERE condition]
[RETURNING  *]
```

[] *optionnel dans la requête*

- **les types de suppression**
  - **Suppression totale**
  - **Suppression conditionnelle**
  - **Suppression et sous requête**

La suppression de données (2)

- **Suppression totale**

```
DELETE FROM PROSPECT;
```

- **Suppression conditionnelle**

```
DELETE FROM PROSPECT
WHERE  prenom LIKE '%A%' ;
```

- **Suppression et sous requête**

```
DELETE FROM CLIENT
WHERE cli_id IN (SELECT prp_id FROM PROSPECT);
```

- **Avec retour des lignes supprimées**

```
DELETE FROM CLIENT
WHERE cli_id IN (SELECT prp_id FROM PROSPECT)
RETURNING  *; -- * Pour toute les colonnes ou sinon lister les colonnes
```

DELETE FROM table1 USING table2 WHERE condition RETURNING returning_columns;
DELETE FROM table1 USING table2 WHERE condition RETURNING returning_columns;

La suppression de données avec la clause USING(3)

- **La Clause USING permet de faire une pseudo jointure avec une autre table.**

```
DELETE FROM table1
USING table2
WHERE condition
RETURNING returning_columns;
```

- **Exemple :**

```
DELETE FROM circuit C
USING pilote P
WHERE C.idpilote = P.idpilote;
```

Ici on supprime les lignes de la table circuit quand il existe les pilotes dans la table pilote

La suppression de données avec la clause USING(3)

- **Autre exemple**

```
DELETE FROM grandprix G
USING circuit C
WHERE G.id_circuit= C.id_circuit
and payscircuit = 'Autriche';
```

Ici on supprime les lignes de la table grandprix quand le circuit sur lequel il se déroule et en Autriche

La suppression de données (3)

- **Autre instruction SQL**

  TRUNCATE supprime rapidement toutes les lignes d'un ensemble de tables. Elle a le même effet qu'un DELETE sans condition, mais comme elle ne parcourt pas la table, elle est plus rapide.

  - **Syntaxe :** **TRUNCATE** nomtable

  - **Exemple :**

    Vide la table personne : **TRUNCATE** personne **;**

    Vide la table personne et adresse : **TRUNCATE** personne**,** adresse **;**

    Vide les tables en cascade : **TRUNCATE** cereale **CASCADE;**

    suppression en cascade de toutes les tables qui référencent cereale via des contraintes de clés étrangères

La modification de données (1)

- **La syntaxe générale**

```
UPDATE nom_table_cible
SET colonne = valeur
[, colonne2 = valeur2 ...]
[WHERE condition]
[RETURNING *];
```

[] ***optionnel dans la requête***

La modification de données (2)

Mise à jour sans condition

Attention modification de ou des colonnes spécifiées sur toutes les lignes de la table

```
// mise à jour du tarif
UPDATE TARIF
      SET    PETIT_DEJEUNE = 10 ;


//majoration de 15% du tarif
UPDATE TARIF
      SET    PETIT_DEJEUNE = PETIT_DEJEUNE * 1.15;
```

La modification de données (3)

Mise à jour sans condition

Attention modification de ou des colonnes spécifiées sur toutes les lignes de la table

```
UPDATE CLIENT
    SET     nom= UPPER(nom),
            prenom   = UPPER(prenom),
            enseigne = UPPER(enseigne);
```

Ou

```
UPDATE CLIENT
    SET (nom, prenom, enseigne) =( UPPER(nom), UPPER(prenom), UPPER(enseigne)) ;
```

La modification de données (4)

- **Mise à jour conditionnelle**

```
UPDATE TARIF
SET    PETIT_DEJEUNE = PETIT_DEJEUNE * 1.15
WHERE  EXTRACT(YEAR FROM DATE_DEBUT) > 1999 ;
```

La modification de données (5)

Mise à jour et les sous requêtes

```
UPDATE PROSPECT
SET    nom = 'bis_'  || nom
WHERE  prp_id IN (SELECT cli_id  FROM CLIENT)
RETURNING nom , prenom , email;
```

prenom	nom	email
MARY	bis_SMITH	MARY.SMITH@sakilacustomer.org
PATRICIA	bis_JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org

Modifie le nom si le l'identifiant des prospects est dans celui de client
Retourne le nom et prénom des prospects modifiés

La modification de données (6)

- **Mise à jour avec sous requête corrélée**

```
UPDATE PROSPECT  AS P
SET    nom= ( SELECT nom
              FROM CLIENT
              WHERE id_cli=P.prp_id);
```

Le nom du prospect est égal à celui du client dont l'identifiant est égal à celui du prospect courant

La modification de données (7)

- **Il est possible faire une "jointure" avec une autre table, la syntaxe est une peu différente de la jointure classique**

```
UPDATE courir C
SET positionarrivee = 'A'
    FROM pilote P
    WHERE C.idpilote = P.idpilote
    AND nompilote='Alonso';
```

FIN