Programmation – Curseurs

Nous avons vu qu'il était possible de sélectionner des données dans une procédure stockée, à condition que le résultat consiste en un seul enregistrement.

Pour gérer plusieurs enregistrements, nous allons utiliser un curseur.

Programmation – Curseurs

Pour déclarer un curseur, nous utiliserons la syntaxe suivante :

```
DECLARE nom_du_curseur CURSOR FOR requête_sql;
```

La requête SQL doit être une requête de sélection.

Les curseurs doivent être déclarés avant les gestionnaires d'erreur.

Programmation – Curseurs

Structure des déclarations :

```
-- Déclaration des variables locales

-- Déclaration des conditions

-- Déclaration des curseurs

-- Déclaration des gestionnaires d'erreur

-- Début du traitement
```

Programmation – Curseurs

Exemple pour sélectionner tous les produits:

```
DECLARE curseur_produits CURSOR
FOR
   SELECT * FROM produit;
```

Programmation – Curseurs

Procédure complète (script chap_3c_1.sql) :

```
DROP PROCEDURE IF EXISTS test_curseur;
DELIMITER |
CREATE PROCEDURE test_curseur()
BEGIN
  DECLARE curseur_produits CURSOR
  FOR
    SELECT * FROM produit;

  SELECT 'START PROCEDURE test_curseur';

  SELECT 'END PROCEDURE test_curseur';
END|

DELIMITER ;

CALL test_curseur();
```

Programmation – Curseurs

Tel quel, le curseur n'est pas utilisable car il doit d'abord être ouvert :

```
OPEN nom_du_curseur;
```

Bien sûr, tout curseur ouvert doit être fermé :

```
CLOSE nom_du_curseur;
```

Programmation – Curseurs

Entre ces deux requêtes, le curseur peut être parcouru pour accéder aux différents enregistrements retournés par la requête de sélection :

```
FETCH nom_du_curseur INTO var1[,var2,var3,…];
```

Le nombre de variables doit correspondre au nombre des colonnes retournées.

Cette requête retourne le résultat suivant dans l'ensemble des enregistrements sélectionnés, et uniquement séquentiellement.

Par conséquent, pour parcourir tous les enregistrements, nous avons besoin d'une boucle.

Programmation – Curseurs

Boucle sur le curseur (script chap_3c_2.sql) :

```
DROP PROCEDURE IF EXISTS test_curseur;
DELIMITER |
CREATE PROCEDURE test_curseur()
BEGIN
  -- Déclaration des variables
  DECLARE l_idproduit  CHAR(10);
  DECLARE l_libelle    CHAR(50);
  DECLARE l_prix       DECIMAL(9,2);
  DECLARE l_qstock     MEDIUMINT(8);

  -- Déclaration des curseurs
  DECLARE curseur_produits CURSOR
  FOR
    SELECT * FROM produit;

  -- Début du traitement
  SELECT 'START PROCEDURE test_curseur';
  OPEN curseur_produits;

  LOOP
    FETCH curseur_produits INTO l_idproduit, l_libelle, l_prix, l_qstock;
    SELECT l_idproduit, l_libelle, l_prix, l_qstock;
  END LOOP;

  CLOSE curseur_produits;
  SELECT 'END PROCEDURE test_curseur';
END|
DELIMITER ;
```

Programmation – Curseurs

L'exécution se termine sur une erreur :

```
ERROR 1329 (02000): No data - zero rows fetched, selected, or processed
```

Pour déterminer la fin de la boucle, nous allons utiliser cette erreur

Programmation – Curseurs

Nous devons donc déclarer un gestionnaire d'erreur et une condition :

```
-- Déclaration des variables
DECLARE l_fin_de_boucle BOOLEAN DEFAULT FALSE;

-- Déclaration des conditions
DECLARE plus_d_enregistrement CONDITION FOR 1329;

-- Déclaration des gestionnaires d'erreur
DECLARE CONTINUE HANDLER FOR plus_d_enregistrement
  BEGIN
    SET l_fin_de_boucle:=TRUE;
  END;
```

Programmation – Curseurs

Code de la boucle (script chap_3c_3.sql) :

```
OPEN curseur_produits;

REPEAT
  FETCH curseur_produits INTO l_idproduit, l_libelle, l_prix, l_qstock;

  SELECT l_idproduit, l_libelle, l_prix, l_qstock;
UNTIL l_fin_de_boucle>0
END REPEAT;

CLOSE curseur_produits;
```

Programmation – Curseurs

Nous pouvons également utiliser un gestionnaire d'erreur basé sur la classe `NOT FOUND ('02')` (script chap_3c_4.sql) :

```
-- Déclaration des variables
DECLARE l_fin_de_boucle BOOLEAN DEFAULT FALSE;

-- Déclaration des conditions

-- Déclaration des gestionnaires d'erreur
DECLARE CONTINUE HANDLER FOR NOT FOUND
  BEGIN
    SET l_fin_de_boucle:=TRUE;
  END;
```

Programmation – Curseurs

Une autre alternative consiste en l'utilisation d'une variable de « comptage » des enregistrements (script chap_3c_4b.sql) :

```
-- Déclaration des variables
DECLARE l_n_produits INT;

  …
SELECT COUNT(*) INTO l_n_produits FROM produit ;

REPEAT
   FETCH curseur_produits INTO l_idproduit, l_libelle, l_prix,
l_qstock;
   SELECT l_idproduit, l_libelle, l_prix, l_qstock;
   SET l_n_produits := l_n_produits - 1;
UNTIL l_n_produits <= 0
END REPEAT;
```

Programmation – Curseurs

Si nous voulons mettre à jour les données au sein de la boucle d'exploitation du curseur, nous devons utiliser l'option `FOR UPDATE` avec le `SELECT` (script chap_3c_5.sql):

```
-- Déclaration des curseurs
DECLARE curseur_produits CURSOR
FOR
  SELECT * FROM produit
  FOR UPDATE;
…

CLOSE curseur_produits;
COMMIT;
```

Programmation – Curseurs

Exercices 1

1. Créez une procédure stockée pour modifier les prix des produits en fonction d'une valeur de TVA passée en paramètre
2. Créez une procédure stockée pour mettre en minuscules tous les noms et prénoms dans la table **client**
3. Créez une fonction stockée qui retourne la catégorie à laquelle un client appartient en fonction de ses achats :
   - 1 achat ou moins : A
   - 2 achats : B
   - 3 achats ou plus : C
   - Moins de 1000€ : 0
   - De 1000 à 15000€ : 1
   - Plus de 15000€ : 2

Programmation – Curseurs

Exercices 1

4. Utilisez la fonction créée au point 3 pour modifier les catégories des clients grâce à une nouvelle procédure stockée
5. Ajoutez une colonne « numéro » à la table client. Créez une procédure stockée qui permet de séparer le contenu de la colonne adresse en deux valeurs :
   - Le numéro dans la colonne « numéro »
   - Le reste dans la colonne « adresse »

Programmation – Curseurs

Exercices 2

À l'aide de la DB sakila, vous devez implémenter :

1. Une fonction qui permet de calculer le prix payé ou à payer (amende éventuelle comprise) pour une location passée en paramètre (si date de retour à NULL alors égale à l'instant d'exécution de la procédure)
2. Une procédure qui permet de déterminer si un film est en stock dans une vidéothèque donnée

Programmation – Curseurs

Exercices 2

À l'aide de la DB sakila, vous devez implémenter :

3. Une fonction qui détermine à quelle catégorie le client appartient selon le nombre de ses locations :
   - 0 : A
   - 1-20 : B
   - 21-30 : C
   - 31-40 : D
   - 41 et plus : E
4. Une procédure qui utilise la fonction précédente pour, après avoir ajouté une colonne category à la table customer, mettre à jour les valeurs de la nouvelle colonne