

CURSORS | CURSEURS

CHAP 5.0 H.CHARARA

SELECTION MULTI – LIGNE : PRINCIPE DES CURSEURS

UN CURSEUR EST UN MOYEN DE PARCOURIR LES RESULTATS D'UNE REQUETE SELECT ET D'EN TRAITER LES VALEURS UNE PAR UNE QUEL QUE SOIT LE NOMBRE DE LIGNES QUI ONT ETE RECUPEREES DANS LA REQUETE SELECT.

Il permet à un programme d'avoir accès à chaque ligne du résultat d'une requête et non à la totalité des lignes retournées par une instruction select.

- Trait d'union entre l'orientation ensembliste d'une SGBD et la programmation séquentielle
 - Possibilité de suppression ou modification d'une ligne dans une table à partir de la position du curseur.
 - Le curseur contient en permanence l'@ de la ligne courante
 - Zone mémoire (SGA : Share Global Area) partagée pour stocker les résultats
-
- ☐ Declare : déclaration du curseur pour le traitement
 - ☐ Open : ouverture du curseur
 - ☐ Fetch : extraction de la ligne dans le curseur pour traitement.
 - ☐ Répétition du processus d'extraction jusqu'à son terme ou son abandon.
 - ☐ Close : fermeture du curseur.
 - ☐ Deallocate : désallocation du curseur.

Déclaration du curseur

- Sémantique d'une instruction SQL – Ordre SQL sans exécution I
- La déclaration d'un curseur ne peut se faire qu'à l'intérieur d'un bloc BEGIN/END, par exemple dans une procédure stockée.

```
DECLARE nom_curseur CURSOR  
FOR SELECT * FROM utilisateur;
```

Ouverture du curseur

- SQL 'monte' les lignes sélectionnées en SGA
- Verrouillage préventif possible (voir + loin)

```
Open nom_curseur
```

Extraction | Selection de lignes à partir d'un curseur ouvert

- Chaque FETCH ramène une ligne dans le programme client
- Tant que ligne en SGA : FETCH

```
Fetch nom_curseur into liste_fetch_target.
```

```
/* tant qu'il y a des lignes */
```

```
/* remarque : 0 extraction réussie, 1 erreur , 2 plus de données à extraire */
```

```
While @@sqlstatus != 2
```

```
begin
```

```
Fetch nom_curseur into liste_fetch_target.
```

```
End
```

Fermeture du curseur

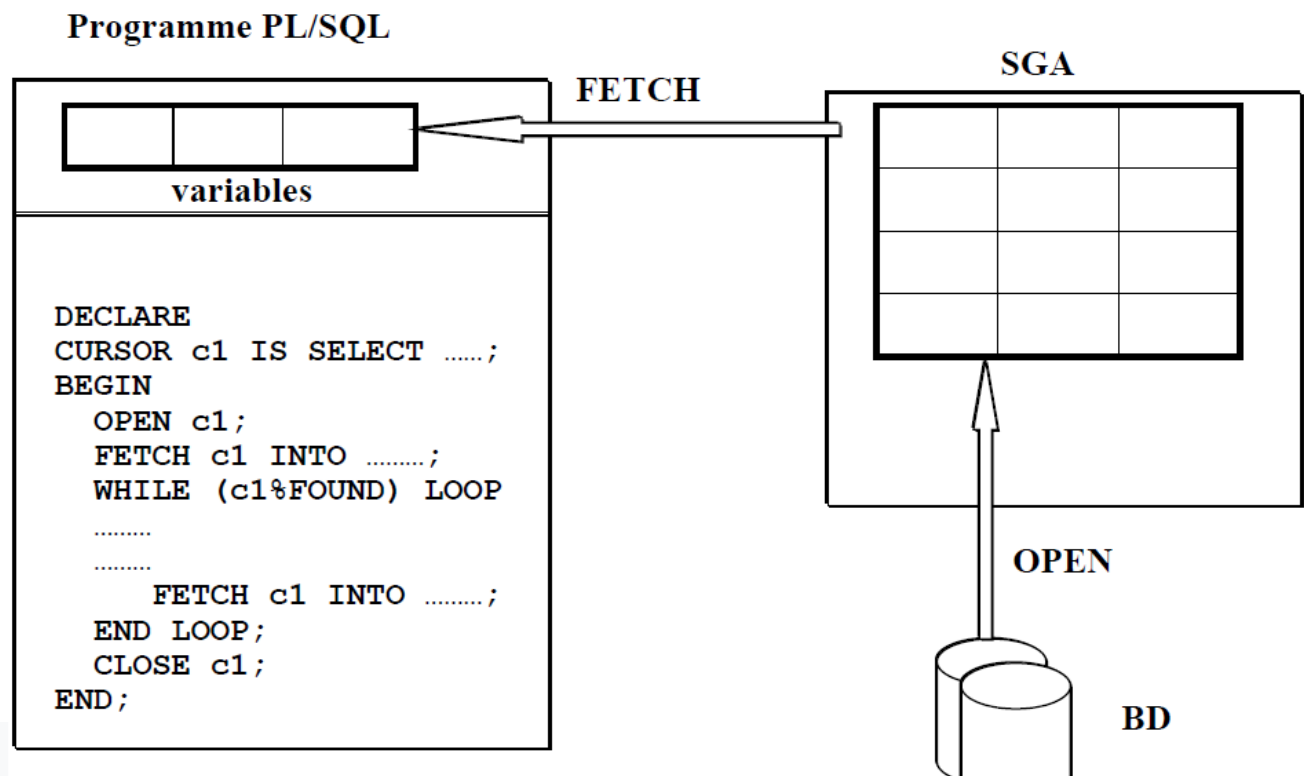
- Récupération de l'espace mémoire en SGA

```
CLOSE
```

Exemple:

```
DECLARE authors_cursor CURSOR FOR
SELECT au_lname, au_fname FROM authors
OPEN authors_cursor
FETCH NEXT FROM authors_cursor
WHILE @@FETCH_STATUS = 0
BEGIN
FETCH NEXT FROM authors_cursor
END
CLOSE authors_cursor
DEALLOCATE authors_cursor
```

1. On déclare le curseur (avec son SELECT)
2. On l'ouvre (on n'est pas obligé de l'ouvrir tout de suite)
3. On en récupère les lignes une à une, en commençant par la première
4. On le referme



```
USE Base1
declare @Nom varchar(20)
DECLARE curseur1 CURSOR FOR SELECT Prenom FROM Table1
OPEN curseur1

/* Premier enregistrement de la sélection */
FETCH NEXT FROM curseur1 into @Nom
print 'Salut ' + @Nom

/* Traitement de tous les autres enregistrements dans une boucle */
while @@FETCH_STATUS = 0
    begin
        FETCH NEXT FROM curseur1 into @Nom
        print 'Salut ' + @Nom
    end

CLOSE curseur1;
DEALLOCATE curseur1;
```

Gestion 'classique' d'un curseur

```
DECLARE
CURSOR c1 IS SELECT nom,moyenne FROM etudiant ORDER BY 1;
vnom          etudiant.nom%TYPE;
vmoyenne      etudiant.moyenne%TYPE;
e1 ,e2 NUMBER;
BEGIN
    OPEN c1;
    FETCH c1 INTO vnom,vmoyenne;
    WHILE c1%FOUND LOOP
        IF vmoyenne < 10 THEN e1:=e1+1;
            INSERT INTO liste_refus VALUES (vnom) ;
        ELSE e2:=e2+1;
            INSERT INTO liste_refus VALUES (vnom) ;
        END IF;
        FETCH c1 INTO vnom,vmoyenne;
    END LOOP;
    CLOSE c1;
    DBMS_OUTPUT.PUT_LINE (TO_CHAR(e2) || 'Reçus ');
    DBMS_OUTPUT.PUT_LINE (TO_CHAR(e1) || 'Collés ');
    COMMIT;
END;
```

Les variables système des Curseurs

- **Curseur%FOUND**
 - Variable booléenne
 - Curseur toujours 'ouvert' (encore des lignes)
- **Curseur%NOTFOUND**
 - Opposé au précédent
 - Curseur 'fermé' (plus de lignes)
- **Curseur%COUNT**
 - Variable number
 - Nombre de lignes déjà retournées
- **Curseur%ISOPEN**
 - Booléen : curseur ouvert ?

`%ISOPEN`

```
IF NOT moncurseur%ISOPEN THEN
OPEN moncurseur;
END IF;
```

`%FOUND`

```
OPEN moncurseur;
LOOP
  FETCH moncurseur INTO variable1,variable2;
  EXIT WHEN NOT moncurseur%FOUND;
END LOOP;
CLOSE moncurseur;
```

Avec un WHILE :

```
OPEN moncurseur;
FETCH moncurseur INTO variable1,variable2;
WHILE moncurseur%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(variable1)||TO_CHAR(variable2));
    FETCH moncurseur INTO variable1,variable2;
END LOOP;
CLOSE moncurseur;
```

%NOTFOUND

```
OPEN moncurseur;
LOOP
    FETCH moncurseur INTO variable1,variable2;
    EXIT WHEN moncurseur%NOTFOUND;
END LOOP;
CLOSE moncurseur;
```

%ROWCOUNT

```
SET SERVEROUTPUT ON
DECLARE
    CURSOR moncurseur IS
        SELECT champ1,champ2 FROM table;
        variable1 table.champ1%TYPE;
        variable2 table.champ2%TYPE;
BEGIN
    OPEN moncurseur;
    LOOP
        FETCH moncurseur INTO variable1,variable2;
        EXIT WHEN NOT moncurseur%FOUND;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(moncurseur%ROWCOUNT));
    CLOSE moncurseur;
END;
```

FOR UPDATE

Si l'on utilise un curseur pour faire des mises à jour, il est préférable d'utiliser la syntaxe suivante :

```
DECLARE  
CURSOR moncurseur IS  
SELECT champ1,champ2 FROM table  
FOR UPDATE;
```

Cette déclaration verrouille les lignes de la table sélectionnée. Personne d'autre ne pourra mettre à jour ces lignes pendant le traitement de cette procédure.

On n'est pas obligé de verrouiller tous les champs de la ligne. On peut choisir les champs comme ceci :

```
DECLARE  
CURSOR moncurseur IS  
SELECT champ1,champ2 FROM table  
FOR UPDATE OF champ1,champ2;
```

WHERE CURRENT OF moncurseur

La clause `WHERE CURRENT OF <nom_du_curseur>` est utile pour les UPDATE et les DELETE : elle permet de préciser que la mise à jour ou la destruction s'applique à la ligne en cours récupérée par le curseur cité : pas besoin d'une condition WHERE sur la valeur d'un champ.

Exemple de UPDATE :

```
DECLARE
  CURSOR moncurseur IS
  SELECT NUMERO FROM FASCICULE
  WHERE ABONNEMENT = 2500
  FOR UPDATE;
BEGIN
  FOR monfascicule IN moncurseur LOOP
    UPDATE FASCICULE
    SET NUMERO = NUMERO + 1
    WHERE CURRENT OF moncurseur;
  END LOOP;
COMMIT;
END;
```

Ce programme récupère le champ NUMERO de la table des fascicules, pour les fascicules liés à l'abonnement 2500. Il met à jour ce champ pour chaque fascicule.

Gestion 'automatique' des curseurs

```
DECLARE
CURSOR c1 IS SELECT nom,moyenne FROM etudiant ORDER BY 1;
-- PAS DE DECLARATION DE VARIABLE DE RECEPTION
e1 ,e2 NUMBER;
BEGIN
    --PAS D'OUVERTURE DE CURSEUR
    --PAS DE FETCH
    FOR c1_ligne IN c1 LOOP
        IF c1_ligne.moyenne < 10 THEN e1:=e1+1;
        INSERT INTO liste_refus VALUES(c1_ligne.nom);
        ELSE    e2:=e2+1;
        INSERT INTO liste_refus VALUES(c1_ligne.nom);
        END IF;
    END LOOP;
    --PAS DE CLOSE
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(e2)||'Reçus ');
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(e1)||'Collés ');
    COMMIT;
END;
```

