IV- PL/SQL : Curseurs et Déclencheurs

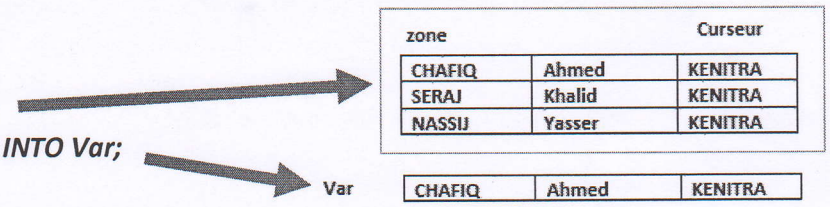
1. Les curseurs : Mécanisme, procédure d'utilisation

a. **Problématique** : Les instructions de type SELECT ... INTO ... manquent de souplesse, elles ne fonctionnent que sur des requêtes retournant une et une seule valeur. Il est donc judicieux de placer dans des variables le résultat d'une requête retournant plusieurs lignes.

b. **Définition** : Un curseur est un objet contenant le résultat d'une requête (avec 0 ou plusieurs lignes). En général, un curseur est une zone mémoire qui permet de traiter individuellement chaque ligne renvoyée par un SELECT. Il existe plusieurs manières de parcourir un curseur, comme il existe plusieurs types de curseurs à parcourir. Notez bien qu'un curseur, durant son existence (de l'ouverture à la fermeture), contient en permanence l'adresse de la ligne courante.

c. **Illustration :**

```
Procedure | Function Sous-pgm [(...)] IS
Var....
Cursor zone IS SELECT nom, prenom, Ville
                    FROM Candidat where Ville='KENITRA';
...
BEGIN
...
OPEN zone;
FETCH zone INTO Var;
.....
END
```



L'exemple illustre un curseur. En effet, le curseur est décrit dans la partie déclarative. Il est ouvert dans le code du programme pour extraire les données de la base. Le programme peut parcourir tout le curseur en récupérant les lignes une par une dans une variable locale. Le curseur est ensuite fermé.

- **Déclaration du curseur** :

  CURSOR /* nomcurseur */ IS /* requête */;

  Exemple : Récupérer tous les employés de la table EMP, on déclare le curseur Emp_Curs :

  CURSOR Emp_Curs IS SELECT * FROM EMP;

- **Ouverture :** Lors de l'ouverture d'un curseur, la requête du curseur est évaluée, et le curseur contient toutes les données retournées par la requête. On ouvre un curseur dans une section BEGIN en utilisant l'instruction : OPEN. **Exemple:**

```
DECLARE CURSOR Emp_Curs IS SELECT * FROM EMP;
BEGIN
        OPEN Emp_Curs;
                /* Utilisation du curseur */
END;
```

- **Lecture d'une ligne :** Une fois le curseur ouvert, on récupère toutes les lignes du résultat de la requête une par une en utilisant le mot-clé FETCH : ***FETCH /* nomcurseur */INTO/* list_vars*/;***

La liste de variables (***list_vars***) peut être remplacée par une structure de type **nom_curseur%ROWTYPE**. Si la lecture de la ligne échoue (Plus de ligne à lire) : L'attribut **%NOTFOUND** prend la valeur vraie.

- **Fermeture du curseur :** Après utilisation, on ferme le curseur (***CLOSE nomcurseur;***)

Exemple 1:

```
DECLARE CURSOR Emp_Curs IS
        SELECT * FROM EMP;
        ligne Emp_Curs%rowtype
BEGIN
        OPEN Emp_Curs;
        LOOP
                FETCH Emp_Curs INTO ligne;
                EXIT WHEN Emp_Curs%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE (ligne.name);
        END LOOP;
                CLOSE Emp_Curs;
END;
```

Avec While :

```
DECLARE CURSOR Emp_Curs IS
        SELECT * FROM EMP;
        ligne Emp_Curs%rowtype
BEGIN
        OPEN Emp_Curs;
        FETCH Emp_Curs INTO ligne;
        WHILE Emp_Curs %FOUND LOOP
                DBMS_OUTPUT.PUT_LINE (ligne.name);
                FETCH Emp_Curs INTO ligne;
        END LOOP;
        CLOSE Emp_Curs;
END;
```

Exemple 2 : Afficher les titres des comédies d'une BD (films).

```
DECLARE
        monNumero Film.numfilm%type;
        monTitre Film.Titre%type;
        Cursor monCurseur IS
        SELECT f.numfilm, titre
                FROM Film f, GenreFilm g
                WHERE f.numfilm=g.numfilm AND codegenre=;
BEGIN
OPEN monCurseur ;
        LOOP
        FETCH monCurseur INTO monNumero, monTitre;
        EXIT WHEN monCurseur%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE ('Numéro: '||monNumero||'Titre: '|| monTitre);
        END LOOP;
        DBMS_OUTPUT.PUT_LINE ('Total de comédies : '||monCurseur%rowCount) ;
CLOSE monCurseur;
END;
```

Utilisation de la boucle FOR : Il existe une boucle FOR se chargeant de l'ouverture, de la lecture des lignes du curseur et de sa fermeture. L'utilisation d'une boucle FOR de curseur facilite la programmation (évite les directives OPEN, FETCH et CLOSE).
La boucle s'arrête d'elle-même à la fin de l'extraction de la dernière ligne du curseur. De plus, la variable de réception du curseur est aussi automatiquement déclarée (%ROWTYPE du curseur).

- **Avec la boucle For :** (Autre Formulation de l'exemple précédent)

```
DECLARE
        cpt NUMBER: =0;
        Cursor monCurseur IS
        SELECT f.numfilm, titre
                        FROM Film f, GenreFilm g
                        WHERE f.numfilm=g.numfilm AND codegenre=1;
BEGIN
For ligneCurseur IN monCurseur
        LOOP
        DBMS_OUTPUT.PUT_LINE ('Numéro: '||ligneCurseur.numFilm ||'Titre: '||
                                        ligneCurseur.titre);
        cpt: =cpt+1;
        END LOOP;
        DBMS_OUTPUT.PUT_LINE ('Total de comédies: '||cpt);
END;
```

d. Curseurs paramétrés

Un curseur paramétré peut servir plusieurs fois avec des valeurs des paramètres différentes. On doit fermer le curseur entre chaque utilisation de paramètres différents si on n'utilise pas la boucle FOR :

```
DECLARE
        Cursor monCurseur (Gen integer) IS
        SELECT f.numfilm, titre, codegenre
                        FROM Film f, GenreFilm g
                        WHERE f.numfilm=g.numfilm AND codegenre= Gen;
BEGIN
For ligneCurseur IN monCurseur (10)
        LOOP
        DBMS_OUTPUT.PUT_LINE ('Numéro: '||ligneCurseur.numFilm ||'Titre: '||
                                        ligneCurseur.titre);
        END LOOP;
For ligneCurseur IN monCurseur (20)
        LOOP
        DBMS_OUTPUT.PUT_LINE ('Numéro: '||ligneCurseur.numFilm ||'Titre: '||
                                        ligneCurseur.titre);
        END LOOP;
END;
```

e. Curseurs anonymes : sans déclarer le curseur

On spécifie le select directement dans le for. La variable de contrôle **Var** déclarée automatiquement du type du résultat de select.

```
BEGIN
FOR Var in (SELECT num_emp, name_emp, salary_emp FROM employee)
LOOP
DBMS_OUTPUT.PUT_LINE ('Num = ' || Var.num_emp || ', Nom = ' ||
        Var.name_emp ||', Salaire = '|| Var.salary_emp);
END LOOP;
END;
```

f. Curseurs et accès concurrents

Afin de verrouiller les lignes d'une table interrogée par un curseur dans le but de mettre à jour la table, sans qu'un autre utilisateur ne la modifie en même temps, nous utilisons la clause FOR UPDATE.
Elle s'utilise lors de la déclaration du curseur et verrouille les lignes concernées lorsque le curseur est ouvert. Les verrous sont libérés à la fin de la transaction. La déclaration d'un curseur FOR UPDATE, qu'on peut qualifier de « modifiable », est la suivante :

```
CURSOR Nom_Curseur[(paramétres)] IS
        SELECT ... FROM {nomTable | nomVue} WHERE ...
        FOR UPDATE [OF [[schéma.] {nomTable | nomVue }.]colonne [, ...]
        [NOWAIT | WAIT entier]
```

- La directive **OF** permet de connaître les colonnes à verrouiller. Sinon, toutes les colonnes issues de la requête seront verrouillées.
- **NOWAIT** précise de ne pas faire attendre le programme si les lignes demandées sont verrouillées par une autre session.
- **WAIT** spécifie le nombre de secondes à attendre au maximum avant que les lignes soient déverrouillées par une autre session. Sans NOWAIT et WAIT, le programme attend que les lignes soient disponibles.
- Il n'est pas possible de déclarer un curseur **FOR UPDATE** en utilisant dans la requête les directives **DISTINCT** ou **GROUP BY**, un **opérateur ensembliste, ou une fonction d'agrégat**.

Maintenant, pour pouvoir modifier la ligne courante, il faut utiliser la clause :

WHERE CURRENT OF NOM_Curseur

```
UPDATE nom_table SET VALUES ...
WHERE CURRENT OF NOM_Curseur;
```

Exemple :

```
DECLARE
Cursor monCurseur IS SELECT * FROM Film FOR UPDATE OF FILM_STATUS NOWAIT;
BEGIN
For ligne IN monCurseur LOOP
        IF ligne.Titre= 'NA' THEN
          Update Film SET FILM_STATUS= 'NOT ASSIGNED' WHERE CURRENT OF monCurseur;
        ELSIF ligne.Titre= 'RET' THEN
          Update Film SET FILM_STATUS= 'RETIRED' WHERE CURRENT OF monCurseur;
        ELSE
          Update Film SET FILM_STATUS= 'AUTHORIZED' WHERE CURRENT OF monCurseur;
        END IF;
END LOOP;
COMMIT;
END;
```

g. Variables de type curseurs

Une variable curseur ou référence à un curseur (REF CURSOR) définit un curseur dynamique qui n'est pas associé à aucune requête fixe (contrairement à un curseur classique dit statique). Ceci permettra au curseur d'évoluer tout au long du programme

Une variable « curseur » est déclarée en deux étapes :

```
TYPE TYP_Curs_Dyna IS REF CURSOR [RETURN TYPE_RETOUR];
Nom_Curs_Dyna  TYP_Curs_Dyna ;
```

- En général, TYPE_RETOUR (Facultatif) représente la structure d'un enregistrement d'une table (ROWTYPE).
- SI le curseur dynamique inclut le type de retour, on parle de curseur dynamique « Typé ».
- Sinon, il s'agira d'un curseur dynamique « Non typé » et ceci permettra une grande flexibilité puisque toute requête peut être associé.

Pour pouvoir utiliser une variable curseur, on affecte une requête à cette variable de curseur dans le corps du programme. Ensuite, on utilisera cette variable comme n'importe quel autre curseur. Toutefois, pour changer la requête affectée à un curseur dynamique, il faut fermer le curseur s'il est ouvert.

- OPEN NOM_Curs_Dyna FOR requête : Permet d'ouvrir le curseur ;
- FETCH : permet de parcourir le curseur ;
- CLOSE : permet de fermer le curseur ;

Exemples :

EXEMPLE1 : Variables de type Curseurs Typés

```
DECLARE
        TYPE mon_TYP_Curseur IS REF CURSOR RETURN FILM%ROWTYPE;
        monCurseur mon_TYP_Curseur ;
        var FILM%ROWTYPE;
BEGIN
        OPEN monCurseur FOR SELECT * FROM FILM WHERE CODE_GENRE='COM';
        FETCH monCurseur INTO var ;

        WHILE (monCurseur%FOUND) LOOP
                DBMS_OUTPUT.PUT_LINE ('TITRE :' || var.TITRE || 'GENRE = COMEDIE');
                FETCH monCurseur INTO var ;
        END LOOP;

        IF monCurseur%ISOPEN THEN
                CLOSE monCurseur;
        END IF;

        OPEN monCurseur FOR SELECT * FROM FILM WHERE CODE_GENRE='DRA';
        FETCH monCurseur INTO var ;
        WHILE (monCurseur%FOUND) LOOP
                DBMS_OUTPUT.PUT_LINE ('TITRE :' || var.TITRE || 'STATUS:'|| var.STATUS
                ||'GENRE = DRAMA');
                FETCH monCurseur INTO var2 ;
        END LOOP;
        CLOSE monCurseur;
END;
```

EXEMPLE2-3 : Variables de type Curseurs NON Typés

```
DECLARE
  req VARCHAR2(100);
  TYPE mon_TYP_Curseur IS REF CURSOR;
  monCurseur mon_TYP_Curseur ;
  v_maVariable maTable%ROWTYPE;
BEGIN
  req := 'SELECT * FROM maTable';
  OPEN monCurseur FOR req;
  LOOP
  FETCH monCurseur INTO v_maVariable;
  EXIT WHEN monCurseur%NOTFOUND;
   -- mes instructions SQL
 END LOOP;
 CLOSE monCurseur;
END;
```

```
DECLARE
        TYPE mon_TYP_Curseur IS REF CURSOR;
         monCurseur mon_TYP_Curseur ;
         var1 FILM.TITRE%TYPE;
         var2 FILM.STATUS%TYPE;
BEGIN
        OPEN monCurseur FOR SELECT TITRE FROM FILM WHERE
                                CODE_GENRE='COM';
        FETCH monCurseur INTO var1 ;
        WHILE (monCurseur%FOUND) LOOP
                DBMS_OUTPUT.PUT_LINE ('TITRE :' || var1 || 'GENRE = COMEDIE');
                FETCH monCurseur INTO var1 ;
        END LOOP;

        IF monCurseur%ISOPEN THEN
                CLOSE monCurseur;
        END IF;

        OPEN monCurseur FOR SELECT TITRE, STATUS FROM FILM WHERE
                                CODE_GENRE='DRA';
        FETCH monCurseur INTO var1, var2 ;
        WHILE (monCurseur%FOUND) LOOP
                DBMS_OUTPUT.PUT_LINE ('TITRE :' || var1 ||'STATUS: ' ||
                var2||'GENRE = DRAMA');
                FETCH monCurseur INTO var1, var2 ;
        END LOOP;
        CLOSE monCurseur;
END;
```