



**ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR**

Membre de 
HONORIS UNITED UNIVERSITIES

École Marocaine des Sciences de l'Ingénieur

Filière: IIR

Les Bases de Données

Prof. Zakaria KHATAR

Les curseurs en PL/SQL

4) LES curseurs en PL/SQL:

Un curseur en PL/SQL est un mécanisme qui **permet de traiter les lignes retournées par une requête SQL une par une (ligne par ligne)**. Il fonctionne comme un pointeur qui se déplace à travers les lignes de résultats, permettant de lire, manipuler ou traiter chaque ligne individuellement dans un bloc PL/SQL.

Il existe deux types de curseurs :

- **Les curseurs implicites** : sont automatiquement gérés par Oracle pour les requêtes simples qui retournent **une unique ligne de résultat**, sans intervention explicite du développeur.
- **Les curseurs explicites** : nécessitent une définition et une gestion explicites par le développeur pour les requêtes renvoyant plusieurs lignes, permettant ainsi un **traitement individuel** de chaque ligne de résultats.

4) Les curseurs en PL/SQL:

L'application d'un curseur en PL/SQL se déroule en trois étapes principales :

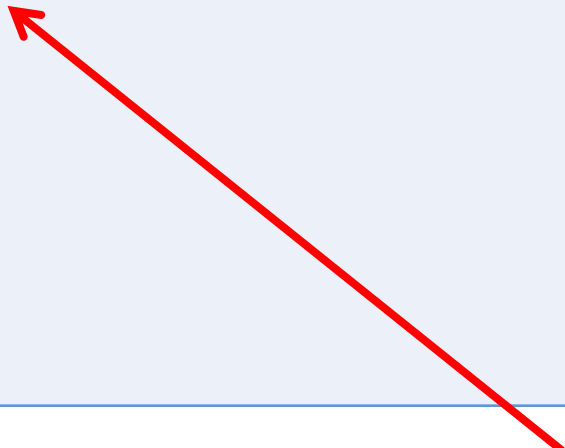
1. Déclaration du curseur
2. Ouverture du curseur
3. Traitement et fermeture du curseur

4-1) Déclaration du curseur :

Déclaration du curseur : se fait dans la partie DECLARE.

Syntaxe :

```
DECLARE  
    CURSOR nom_curseur IS  
    Requête d'interrogation;  
    v_variable type;  
    ...  
BEGIN  
    ....  
END;
```



Les requêtes d'interrogation sont celles
qui commencent par la clause **SELECT**

4-1) Déclaration du curseur :

Exemple :

DECLARE

CURSOR cur_emp **IS**

SELECT last_name, first_name, salary

FROM employees

WHERE salary > 15000;

v_last_name employees.last_name%type;

v_first_name employees.first_name%type;

v_salary employees.salary%type;

...

BEGIN

....

END;

4-2) Ouverture du curseur :

L'ouverture du curseur lance l'exécution de la requête SELECT associé au curseur.
L'ouverture se fait dans la section **BEGIN** du bloc.

Syntaxe :

```
DECLARE  
    CURSOR nom_curseur IS  
    Requête d'interrogation;  
    v_variable type;  
    ...  
BEGIN  
    OPEN nom_curseur;  
    ....  
END;
```

Pour ouvrir le curseur, écrivez **OPEN** suivi du nom du curseur déclaré.

4-2) Traitement et fermeture du curseur :

Après l'exécution du **SELECT**, les lignes ramenées sont traitées une par une, la valeur de chaque colonne du **SELECT** doit être stockée dans une variable réceptrice.

Syntaxe :

```
BEGIN  
    OPEN nom_curseur;  
    LOOP  
        FETCH nom_curseur INTO v_variable1, v_variable2;  
        EXIT WHEN nom_curseur%NOTFOUND;  
        Instructions  
        ...  
    END LOOP;  
    CLOSE nom_curseur;  
END;
```

à **chaque itération** de la boucle, le curseur pointe sur **la ligne suivante**, et la clause **FETCH** prend les valeurs de cette ligne et les **affecte aux variables**.

4-2) Traitement et fermeture du curseur :

- ❖ **OPEN nom_curseur** : ouvre le curseur et exécute la requête SELECT associée.
- ❖ **FETCH nom_curseur INTO v_variable1, v_variable2, ...** : récupère la ligne suivante du curseur et affecte les valeurs de chaque colonne dans les variables indiquées.
- ❖ **nom_curseur%NOTFOUND** : une variable booléenne qui vaut "TRUE" si le curseur n'a plus de lignes à récupérer, et "FALSE" sinon.
- ❖ **CLOSE nom_curseur**: ferme le curseur et libère les ressources associées.

4-2) Traitement et fermeture du curseur :

Les variables de test qui pourraient être utilisées avec les curseurs :

VARIABLES	DESCRIPTION
nom_curseur%FOUND	Variable booléenne qui vaut TRUE si la dernière instruction exécutée a affecté au moins une ligne, FALSE dans le cas contraire.
nom_curseur%NOTFOUND	Variable booléenne qui vaut TRUE si la dernière instruction exécutée n'a affecté aucune ligne, FALSE dans le cas contraire.
nom_curseur%ROWCOUNT	Variable numérique qui contient le nombre de lignes affectées par la dernière instruction exécutée.

4-2) Traitement et fermeture du curseur :

Exemple :

```
DECLARE
    CURSOR cursor1 IS
        SELECT last_name, salary FROM employees ORDER BY salary DESC;
    v_last_name employees.last_name%type;
    v_salary employees.salary%type;
BEGIN
    OPEN cursor1;
    LOOP
        FETCH cursor1 INTO v_last_name, v_salary;
        EXIT WHEN cursor1%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(v_last_name || ', ' || v_salary);
    END LOOP;
    CLOSE cursor1;
END;
```

4-2) Traitement et fermeture du curseur :

Exemple : Le même exemple avec %ROWTYPE

```
DECLARE
    CURSOR cursor1 IS
        SELECT * FROM employees ORDER BY salary DESC;
        v_row employees%ROWTYPE;
BEGIN
    OPEN cursor1;
    LOOP
        FETCH cursor1 INTO v_row;
        EXIT WHEN cursor1%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(v_row.last_name || ', ' || v_row.salary);
    END LOOP;
    CLOSE cursor1;
END;
```

4-2) Traitement et fermeture du curseur : Explication de l'exemple précédent

- ❖ L'exemple précédent déclare un curseur (**cursor1**) qui sélectionne **les noms** et **les salaires** de tous les employés de la table "**employees**", triés par salaire **décroissant**.
- ❖ Le curseur est ensuite ouvert, et une boucle est utilisée pour itérer à travers chaque ligne du résultat.
- ❖ Pour chaque itération de la boucle, la méthode **FETCH** est utilisée pour récupérer la ligne suivante du curseur et l'assigner à des variables (**v_last_name** et **v_salary**).
- ❖ Si le curseur n'a pas atteint la fin des résultats (**cursor1%NOTFOUND** est false), la boucle s'arrête. Sinon, **le nom** et **le salaire** sont concaténés en une chaîne de caractères et affichés à l'aide de la méthode **DBMS_OUTPUT.PUT_LINE**.
- ❖ Enfin, une fois que la boucle a terminé de parcourir toutes les lignes du curseur, le curseur est fermé à l'aide de la méthode **CLOSE**.