Les Curseurs

➢ Définition

Un **curseur** en PL/SQL est une structure utilisée pour gérer et parcourir les résultats d'une requête SQL qui renvoie **plusieurs lignes**. Il Permet de manipuler les données ligne par ligne.

Curseur est créé dès qu'on exécute une instruction SQL.

C'est une zone de travail de l'environnement utilisateur qui contient les informations relatives à l'instruction SQL :

- Le texte source de l'ordre SQL
- Le texte «compilé» de l'ordre SQL
- Un tampon pour une ligne du résultat
- Le statut (cursor status)
- Des informations de travail et de contrôle

Les Curseurs

Pourquoi utiliser des curseurs ?

1. **Gestion des résultats multiples** :

   Les commandes SQL comme SELECT INTO ne permettent de récupérer qu'une seule ligne. Les curseurs permettent de gérer **plusieurs lignes** efficacement.

2. **Contrôle ligne par ligne** :

   Les curseurs permettent de **parcourir les données une ligne à la fois**, ce qui est utile pour des traitements personnalisés.

3. **Amélioration de la lisibilité et de la gestion du code** :

   Ils rendent le code plus lisible lorsqu'il faut traiter des ensembles de données complexes.

Mode de fonctionnement des curseurs

1. **Créer une zone de contexte :**

   Lorsqu'une requête SQL (par exemple, un SELECT) est exécutée, Oracle crée une **zone de contexte** en mémoire. Cette zone de contexte contient :

   - Les résultats de la requête (les lignes renvoyées par SELECT).
   - Des métadonnées, comme le nombre de colonnes, les types de données, etc.

2. **Nommer la zone de contexte avec un curseur :** Le curseur agit comme un **pointeur** vers cette zone de contexte. Cela permet de donner un **nom** à la zone et d'accéder aux résultats ou aux métadonnées.
3. **Accéder aux informations :** Le curseur donne accès aux lignes de résultats **une par une**, en utilisant des instructions comme FETCH.

   Il permet également d'exécuter la commande SQL associée et de parcourir les résultats dans un ordre défini.

Les Curseurs

➢ Types

- **Curseurs implicites**: On ne déclare pas de curseur, ni on l'ouvre ni on le ferme. Créés automatiquement par Oracle pour gérer les requêtes SQL simples (plus facile à utiliser)

- **Curseurs explicites**: Déclarés et contrôlés manuellement par le programmeur pour parcourir plusieurs lignes retournées par une requête SQL.

➢ Curseurs Implicite

Gérés automatiquement par le noyau dans les cas suivants :

- Une instruction SELECT exécutée sous SQL Developer
- Une instruction SELECT donnant une seule ligne de résultat sous PL/SQL
- Les instructions UPDATE, INSERT et DELETE

Les Curseurs

➢ Curseurs Explicite

Obligatoires pour un SELECT susceptible de produire **plusieurs lignes résultat**

Déclaration manuelle : Ils doivent être définis dans la section DECLARE d'un bloc PL/SQL.**Étapes contrôlées :**

Le programmeur doit :

- **Déclarer** le curseur(DECLARE)
- **Ouvrir** le curseur pour exécuter la requête(OPEN)
- **Récupérer** les lignes avec (FETCH).
- **Fermer** le curseur une fois terminé(CLOSE)
- **Utilisation flexible :** Idéal pour parcourir des résultats ligne par ligne ou effectuer un traitement personnalisé.

Les Curseurs

➢ Déclaration

- Déclaration d'un curseur: consiste à nommer un curseur et à lui associer une requête
- La déclaration se fait dans la section DECLARE d'un bloc PL/SQL CURSOR nom-curseur IS requête;

Syntaxe

```
DECLARE
CURSOR <nom_du_curseur> IS SELECT ... FROM ...
BEGIN ...
END;
```

Les Curseurs

➢ Ouvertures

- Consiste à nommer un curseur et à lui associer une requête (Declare cursor is….)
- L'instruction open réalise les tâches suivantes (open …) :
  - Exécution de la requête associée au curseur.
  - Récupération des lignes.
  - Mettre (le pointeur) le curseur avant la première ligne.
- **Syntaxe**

```
DECLARE CURSOR <nom_du_curseur> IS SELECT ... FROM ...
BEGIN ...
OPEN <nom_du_curseur>
END;
```

Les Curseurs

➢ Défiliement des données

Récupération des lignes du curseur

Cette instruction fait aussi passer le curseur à la ligne suivante.

```
Fetch      nom_curseur      into      liste_variables |variable ;
```

- Autant d'instructions FETCH que de lignes résultats :
  - *FETCH nom-curseur INTO liste-variables;*
  - *ou FETCH nom-curseur INTO nom-enregistrement;*

➢ Fermeture :

Libère les ressources allouées au curseur.

```
CLOSE mon_curseur;
```

Exemples

Considérons la tables employés:

id_employe	nom	salaire	departement
1	Malak	5000	Finance
2	Anouar	4500	IT
3	Sarah	4800	Finance
4	Yassine	5200	HR
5	Amine	5100	Finance

```
DECLARE
   CURSOR C1 IS
      SELECT nom, salaire
      FROM employes
      WHERE departement = 'Finance';

   v_nom VARCHAR2(50);
   v_salaire NUMBER;
BEGIN
   OPEN C1;

   LOOP
      FETCH C1 INTO v_nom, v_salaire;
      EXIT WHEN C1%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Nom : ' || v_nom || ', Salaire : ' || v_salaire);
   END LOOP;

   CLOSE C1;
END;
/
```

Le curseur affiché va sélectionner les employés appartenant au département "Finance" et affichera leur **nom** et **salaire** ligne par ligne dans le format :

```
Nom : [nom], Salaire : [salaire]
```

```
Nom : Malak, Salaire : 5000
Nom : Sarah, Salaire : 4800
Nom : Amine, Salaire : 5100
```

Les Curseurs

- **Deuxième forme : Utilisation d'un RECORD personnalisé pour regrouper les colonnes.**

```
DECLARE
   CURSOR C1 IS
      SELECT nom, salaire    FROM employes  WHERE departement = 'Finance';
   TYPE EmployeRecord IS RECORD (   v_nom VARCHAR2(50), v_salaire NUMBER );
   r_employe EmployeRecord;
BEGIN
   OPEN C1;
   LOOP
      FETCH C1 INTO r_employe;
      EXIT WHEN C1%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Nom : ' || r_employe.v_nom || ', Salaire : ' ||
r_employe.v_salaire);
   END LOOP;

   CLOSE C1;
END;
/
```

Les Curseurs

- **3ème forme Utilisation de %ROWTYPE, qui correspond automatiquement à la structure des colonnes du curseur**

```
DECLARE
   CURSOR C1 IS
      SELECT nom, salaire    FROM employes   WHERE departement = 'Finance';
   r_employe C1%ROWTYPE;
--r_employe une var composite dont la struct est définie par c1
BEGIN
   OPEN C1;
   LOOP
      FETCH C1 INTO r_employe;
      EXIT WHEN C1%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Nom : ' || r_employe.nom || ', Salaire : ' ||
r_employe.salaire);
   END LOOP;
   CLOSE C1;
END; /
```

Quatrième forme Boucle intégrée simplifiée

```
DECLARE
   CURSOR C1 IS
      SELECT nom, salaire  FROM employes
      WHERE departement = 'Finance';
BEGIN
   FOR r_employe IN C1 LOOP
      DBMS_OUTPUT.PUT_LINE('Nom : ' || r_employe.nom || ', Salaire
: ' || r_employe.salaire);
   END LOOP;
END;/
```

Les Curseurs

➢ Statut Curseur

Statut	Description
%FOUND	TRUE si une ligne a été récupérée avec succès.
%NOTFOUND	TRUE si aucune ligne n'a été récupérée.
%ISOPEN	TRUE si le curseur est ouvert.
%ROWCOUNT	Retourne le nombre de lignes récupérées jusqu'à présent (commence à 0).

Les Curseurs

```
DECLARE
   CURSOR C1 IS
      SELECT nom, salaire
      FROM employes
      WHERE departement = 'Finance';

   v_nom VARCHAR2(50);
   v_salaire NUMBER;
```

Les Curseurs

```
BEGIN
   OPEN C1;

   LOOP
      FETCH C1 INTO v_nom, v_salaire;

      -- Vérifications des attributs
      IF C1%FOUND THEN
         DBMS_OUTPUT.PUT_LINE('Nom : ' || v_nom || ', Salaire : ' || v_salaire);
         DBMS_OUTPUT.PUT_LINE('Nombre de lignes récupérées : ' || C1%ROWCOUNT);
      END IF;

      EXIT WHEN C1%NOTFOUND; -- Arrête la boucle si aucune ligne n'est trouvée
   END LOOP;

   DBMS_OUTPUT.PUT_LINE('Le curseur est toujours ouvert ? ' || C1%ISOPEN);

   CLOSE C1;
   DBMS_OUTPUT.PUT_LINE('Le curseur est maintenant fermé ? ' || NOT C1%ISOPEN);
END;
/
```

Les Curseurs

➢ Curseurs Automatiques

Les **curseurs automatiques** sont des **curseurs implicites** gérés automatiquement par Oracle. Ils sont créés et utilisés sans noms et **sans intervention du programmeur** pour exécuter et gérer les résultats d'une commande SQL. Ces curseurs sont particulièrement utiles pour les opérations **INSERT**, **UPDATE**, **DELETE**, ou un **SELECT INTO** (qui récupère une seule ligne).

```
For ligne in (sélection) loop
 -- La variable de parcours n'est pas déclarée.
   Liste_des_instructions;
End loop;
```

Les Curseurs

➢ Curseurs Variables

Les **curseurs variables**, également appelés **Ref Cursors**, sont des curseurs **dynamiques** qui permettent de manipuler et exécuter des requêtes SQL **à la volée (**cela signifie qu'il peut être **associé à une requête SQL différente au moment de l'exécution** donc dynamique).

Contrairement aux curseurs classiques, ils ne sont pas associés à une requête fixe, ce qui les rend flexibles pour exécuter des requêtes variables au moment de l'exécution.

```
TYPE ref_cursor_type IS REF CURSOR;
```

- La requête est assignée au curseur au **moment de l'ouverture**, et tu peux choisir une requête différente à chaque fois.
- La requête est dynamique, et le curseur peut s'adapter aux besoins à chaque exécution.

Les Curseurs

➢ **Exemple** **Requête fixe (curseur classique)**

```
DECLARE
  CURSOR mon_curseur IS
    SELECT nom FROM employes WHERE departement = 'Finance'; -- Requête fixe
BEGIN
  OPEN mon_curseur; -- Toujours la même requête
END;
/
```

Le curseur est **toujours lié** à departement = 'Finance'.Si tu veux exécuter une autre condition, tu dois déclarer un **nouveau curseur**.

Requête dynamique (curseur variable)

```
DECLARE
   TYPE ref_cursorIS REF CURSOR;
   my_cursor ref_cursor;
BEGIN
   -- 1ère requête
   OPEN my_cursor FOR
      SELECT nom FROM employes WHERE departement = 'Finance';

   -- 2ème requête
   OPEN my_cursor FOR
      SELECT nom FROM employes WHERE salaire > 5000;

   -- Chaque ouverture utilise une requête différente
   CLOSE my_cursor ;
END;
/
```

Les Curseurs

➢ Curseurs Avec pramètres

Un **curseur avec paramètres** est un curseur qui accepte des **valeurs externes (paramètres)** lors de son ouverture. Cela permet de rendre le curseur plus **dynamique** et **réutilisable**, car il peut exécuter des requêtes SQL similaires avec des valeurs différentes.

```
CURSOR nom_curseur (param1 type, param2 type) IS requete de selection;
```

Les paramètres effectifs sont transmis lors de l'ouverture du curseur :

```
Open  nom_curseur (liste_paramètres) ;
For variable in nom_curseur (liste_paramètres) loop…
```

Les Curseurs

- Curseurs Avec pramètres

```
Declare
   Cursor    c (pnserv int)  is
   Select  salaire, nemp
       from    employe
       where   nserv=pnserv;
begin
 for ligne in c(10) loop
        if  ligne.salaire< 5000  then
             update       emp
             set  salaire=salaire*1.1
             where nemp=ligne.nemp;
        end if;
  end loop;
end;
```

Les Curseurs

➢ Curseurs Avec pramètres

```
Declare
   Cursor    c ( pnserv int default
1) is
      Select  salaire, nemp
      from    employe
      where   nserv=pnserv;

   v_salaire     emp.salaire%type;
   v_nemp        emp.nemp%type ;
  vnserv int;
Begin
vnserv:=10;
open  c(vnserv) ; --ouverture
--Cela signifie que la requête
récupère les employés du service
10.
```

```
loop   --parcours de lignes
   fetch  c  into  v_salaire ,
v_nemp;
       exit  when  c%notfound;
     if  v_salaire< 5000  then
           update   emp
            set
salaire=salaire*1.1
            where nemp=v_nemp;
       end if;
end loop;
close c; --fermeture du curseur
end;
```

➢ Mise à Jour des lignes d'un curseurs

Pour mettre à jour directement les lignes d'un curseur, on utilise la clause WHERE CURRENT OF. Cette clause identifie la ligne courante du curseur et permet de la mettre à jour ou de la supprimer sans avoir à spécifier d'identifiant dans la condition.

Possibilité d'utiliser la clause FOR UPDATE dans la déclaration du curseur. Cela permet d'utiliser la clause CURRENT OF nom-curseur

- For update pour verrouiller les données à MAJ.
- Current of pour référencer la ligne en cours dans le curseur

Les Curseurs

- Mises à jours des lignes de curseurs

```
Declare
Cursor c is select comission from salaries for update of
comission;
begin
      for enr in c loop
       if enr.comission is null then
        update salaries
              set comission=salaire*0.1
              where current of c;
             end if;
       end loop;
end;
```

```
DECLARE
   CURSOR emp_curseur IS
      SELECT nemp, salaire
      FROM employes
      WHERE departement = 'Finance'
      FOR UPDATE; -- Verrouille les lignes pour permettre leur mise à jour

BEGIN
   OPEN emp_curseur;

   LOOP
      FETCH emp_curseur INTO v_nemp, v_salaire;
      EXIT WHEN emp_curseur%NOTFOUND;

      IF v_salaire < 5000 THEN
         UPDATE employes – on peut faire delete aussi
         SET salaire = salaire * 1.1
         WHERE CURRENT OF emp_curseur; -- Met à jour la ligne courante
      END IF;
   END LOOP;

   CLOSE emp_curseur;
END;
/
```