

Chapitre 4 : Gestion des curseurs





Les curseurs

- › Toutes les requêtes SQL sont associées à un curseur
- › Ce curseur représente la zone mémoire utilisée pour *analyser* et exécuter la requête
- › On distingue deux types de curseurs:
 - **Les curseurs implicites**
 - **Les curseurs explicites**



Les curseurs

- › Les requêtes SQL INSERT, UPDATE and DELETE sont gérés par des curseurs implicites
- › Pour la requête SQL SELECT:
 - Des **curseurs implicites** sont créés automatiquement pour les instructions SELECT INTO qui ramène exactement 1 ligne.
 - Des **curseurs explicites** sont **créés et gérés par l'utilisateur** pour traiter les ordres SELECT qui ramène 0, 1 ou plusieurs lignes.



Notion de curseur

- › Le tableau suivant indique, les commandes SQL susceptibles de jouer le rôle de curseur implicite et/ou explicite :

Opération	Curseur Implicite	Curseur Explicite
MAJ	Insert, Update, Delete	Ø
Interrogation	Select ... into	Select

Les curseurs implicites



Les curseurs implicites

- › Déclarés et gérés automatiquement par ORACLE lors de l'exécution de la requête.
- › Associés aux ordres INSERT, UPDATE et DELETE.
- › Associés à l'ordre SELECT s'il retourne exactement une seule ligne.
- › **Attention :** Si l'ordre SELECT ne retourne pas exactement une seule ligne, une exception sera générée.



Exemples de curseur implicite

Exemple 1 :

DECLARE

v_mat emp.matr%TYPE;

v_nom emp.nom%TYPE;

BEGIN

v_nom := 'Dupond';

v_mat := 600;

INSERT INTO emp (matr, nom)

VALUES (v_mat, v_nom);

END;

/



Exemples de curseur implicite

Exemple 2 :

DECLARE

v_name emp.nom%TYPE := 'Durand';

BEGIN

UPDATE emp

SET nom = v_name

WHERE matr = 600;

END;

/



Exemples de curseur implicite

Exemple 3 :

DECLARE

v_matr emp.matr%TYPE := 600;

BEGIN

DELETE FROM emp

WHERE matr = v_matr;

END;

/



Exemples de curseur implicite

Exemple 4 :

DECLARE

V_dname dept.dname%TYPE;

V_Loc dept.loc%TYPE;

BEGIN

SELECT dname, loc

INTO v_dname, v_loc  Obligatoire

FROM dept

WHERE deptno = 13;

...

END;



Attributs du curseur implicite

- › PL/SQL fournit des attributs permettant d'évaluer le résultat d'un curseur implicite:

Attribut	Description
SQL%ROWCOUNT	Entier : Nombre de lignes affectées par le <u>dernier</u> ordre SQL
SQL%FOUND	Booléen : TRUE si le <u>dernier</u> ordre SQL affecte au moins une ligne
SQL%NOTFOUND	Booléen : TRUE si le <u>dernier</u> ordre SQL n'affecte aucune ligne
SQL%ISOPEN	Booléen : Toujours FALSE pour les curseurs implicites

- › Ces attributs sont utilisables comme des fonctions dans les ordres PL/SQL MAIS PAS dans les commandes SQL.



Exemple

BEGIN

UPDATE EMP

SET sal = sal*1.1

WHERE deptno = 10;

IF (SQL%NOTFOUND) THEN

DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' ligne mise à jour');

-- SQL%ROWCOUNT ici égale à 0

ELSIF(SQL%FOUND) THEN

DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' lignes mise à jour');

END IF;

-- SQL%ISOPEN est toujours FALSE

END;

/



Curseur implicite : Exceptions du select

Condition	Nom Exception
Le SELECT identifie plus d'une ligne ▪ Erreur Oracle Server ORA-01422	TOO_MANY_ROWS
Le SELECT n'identifie aucune ligne ▪ Erreur Oracle Server ORA-01403	NO_DATA_FOUND

Les curseurs explicites



Les curseurs explicites

- › ***Les curseurs explicites*** sont déclarés et nommés par le programmeur
- › Ils sont utilisés pour traiter des ordres ***Select*** qui ramène ***0, 1 ou plusieurs lignes***
- › Les curseurs explicites sont indispensables pour traiter les requêtes select qui retournes 0 ou plusieurs lignes.
- › Ils sont manipulés avec des instructions spécifiques dans la partie exécutable des blocs PL/SQL.



Les curseurs explicites

- › Un curseur explicite doit être explicitement:
 - Déclaré dans la section DECLARE
 - Géré par le développeur dans la section exécutable :
 - › Ouverture du curseur
 - › Lecture et traitement des lignes
 - › Fermeture du curseur.



Curseur Explicite : Cycle de vie

Cycle de vie

- **Cursor** : Déclaration du nom du curseur et de la requête.
- **Open** : Ouverture du curseur et exécution de la requête.
- **Fetch** : Récupération des lignes, une par une.
- **Close** : Fermeture du curseur et libération des ressources.



Curseur Explicite : Cycle de vie

Cycle de vie

Section déclaration

- **Cursor** : Déclaration du nom du curseur et de la requête.

Section exécution

- **Open** : Ouverture du curseur et exécution de la requête.
- **Fetch** : Récupération des lignes, une par une.
- **Close** : Fermeture du curseur et libération des ressources.



Déclaration d'un curseur

› Se fait dans la section DECLARE

› Syntaxe :

CURSOR nom_curseur [(paramètre1,paramètre2,...)] **IS** *ordre_select*;

DECLARE

CURSOR c1 **IS**

SELECT nom, moyenne

FROM etudiant

WHERE dept = 'info';

BEGIN

...

END;



Ouverture d'un curseur

- › Se fait dans la section Exécution
- › Syntaxe : **OPEN** nom_curseur [(paramètre_effectif,...)];

```
DECLARE  
  
    CURSOR c1 IS  
    SELECT nom, moyenne  
    FROM etudiant  
    WHERE dept = 'info';  
  
BEGIN  
  
    ...;  
    open    c1;  
    ...;  
  
END;
```



Accès aux lignes d'un curseur

- › Se fait dans la section Exécution
- › Les lignes ramenées sont traitées une par une
- › la valeur de chaque colonne doit être stockée dans une variable réceptrice
- › Syntaxe : **FETCH** nom_curseur **INTO** variable1, variable2, ...;
- › **Fonctionnement :**
 - Ramène le contenu de la ligne courante
 - Assigne les données dans les variables de INTO
 - Déplace le pointeur vers la ligne suivante



Accès aux lignes d'un curseur

```
DECLARE
```

```
CURSOR c1 IS SELECT nom, moyenne FROM etudiant WHERE Dept='info';  
v_nom etudiant.nom%TYPE;  
v_moyenne etudiant.moyenne%TYPE;  
e1 NUMBER :=0;  
e2 NUMBER :=0;
```

```
BEGIN
```

```
OPEN c1;
```

```
LOOP
```

```
FETCH c1 INTO v_nom, v_moyenne;
```

```
EXIT WHEN c1%NOTFOUND; --Test de sortie
```

```
IF v_moyenne < 10 THEN e1:=e1+1;
```

```
ELSE e2:=e2+1;
```

```
END IF;
```

```
END LOOP;
```

```
....
```

```
END;
```



Fermeture d'un curseur

- › Se fait dans la section Exécution
- › Libération de l'espace mémoire alloué
- › Syntaxe : **CLOSE** nom_curseur;
- › **Remarque :**
 - Toute opération sur un curseur fermé (**Fetch**, **%attribut**) engendra l'exception prédéfinie **INVALID_CURSOR**
 - Après fermeture, un curseur peut être réouvert si nécessaire



Fermeture d'un curseur

DECLARE

CURSOR c1 **IS SELECT** nom, moyenne **FROM** etudiant **WHERE** Dept='info';
v_nom etudiant.nom%TYPE;

v_moyenne etudiant.moyenne%TYPE;

e1 NUMBER :=0;

e2 NUMBER :=0;

BEGIN

OPEN c1;

.....

CLOSE c1;

END;



Curseur Explicite : Exemple complet

```
DECLARE
    CURSOR c1 IS SELECT nom, moyenne FROM etudiant WHERE Dept='info';
    v_nom etudiant.nom%TYPE;
    v_moyenne etudiant.moyenne%TYPE;
    e1 NUMBER :=0;
    e2 NUMBER :=0;
BEGIN
    OPEN c1;
    LOOP
        FETCH c1 INTO v_nom, v_moyenne;
        EXIT WHEN c1%NOTFOUND; --Test de sortie
        IF v_moyenne < 10 THEN e1:=e1+1;
        ELSE e2:=e2+1;
        END IF;
    END LOOP;
    CLOSE c1;
    DBMS_OUTPUT.PUT_LINE(e2 || ' étudiants ont réussi');
    DBMS_OUTPUT.PUT_LINE(e1 || ' étudiants n ont pas réussi');
END;
```



Attributs d'un curseur explicite

› Syntaxe : nom_curseur%attribut

Attribut	Description
ROWCOUNT	Entier : Nombre de lignes parcourues jusqu'à présent.
FOUND	Booléen : TRUE si le <u>dernier</u> FETCH a ramené une ligne FALSE si pas de ligne ramenée.
NOTFOUND	Booléen : TRUE si le <u>dernier</u> FETCH n'a pas ramené une ligne FALSE s'il a ramené une ligne
ISOPEN	Booléen : TRUE si le curseur est ouvert FALSE si le curseur est fermé



Exemple - %FOUND

DECLARE

```
CURSOR c1 IS SELECT nom, moyenne FROM etudiant WHERE Dept='info';  
v_nom etudiant.nom%TYPE;  
v_moyenne etudiant.moyenne%TYPE;  
e1 NUMBER :=0;  
e2 NUMBER :=0;
```

BEGIN

```
OPEN c1;  
FETCH c1 INTO v_nom, v_moyenne;  
WHILE c1%FOUND  
LOOP  
    IF v_moyenne < 10 THEN e1:=e1+1;  
    ELSE e2:=e2+1;  
    END IF;  
    FETCH c1 INTO v_nom, v_moyenne;  
END LOOP;  
CLOSE c1; .....
```

END;



Exemple - %NOTFOUND

DECLARE

```
CURSOR c1 IS SELECT nom, moyenne FROM etudiant WHERE Dept='info';  
v_nom etudiant.nom%TYPE;  
v_moyenne etudiant.moyenne%TYPE;  
e1 NUMBER :=0;  
e2 NUMBER :=0;
```

BEGIN

```
OPEN c1;
```

```
LOOP
```

```
    FETCH c1 INTO v_nom, v_moyenne;
```

```
    EXIT WHEN c1%NOTFOUND;
```

```
    IF v_moyenne < 10 THEN e1:=e1+1;
```

```
    ELSE e2:=e2+1;
```

```
    END IF;
```

```
END LOOP;
```

```
CLOSE c1;
```

```
.....
```

```
END;
```



Exemple - %ISOPEN

DECLARE

```
CURSOR c1 IS SELECT nom, moyenne FROM etudiant WHERE Dept='info';  
v_nom etudiant.nom%TYPE;  
v_moyenne etudiant.moyenne%TYPE;  
e1 NUMBER :=0;  
e2 NUMBER :=0;
```

BEGIN

```
IF NOT(c1%ISOPEN) THEN OPEN c1; END IF;
```

LOOP

```
FETCH c1 INTO v_nom, v_moyenne;
```

```
EXIT WHEN c1%NOTFOUND;
```

```
IF v_moyenne < 10 THEN e1:=e1+1;
```

```
ELSE e2:=e2+1;
```

```
END IF;
```

```
END LOOP;
```

```
CLOSE c1; .....
```

END;



Exemple - %ROWCOUNT

DECLARE

CURSOR c1 IS SELECT nom, moyenne FROM etudiant WHERE Dept='info';
v_nom etudiant.nom%TYPE;

v_moyenne etudiant.moyenne%TYPE;

e1 NUMBER :=0;

e2 NUMBER :=0;

BEGIN

OPEN c1;

LOOP

FETCH c1 INTO v_nom, v_moyenne;

EXIT WHEN c1%NOTFOUND OR c1%ROWCOUNT > 5;

IF v_moyenne < 10 THEN e1:=e1+1;

ELSE e2:=e2+1;

END IF;

END LOOP;

CLOSE c1;

END;

Quand on ouvre le curseur, c'est la première ligne qui est pointée. Ensuite, à chaque FETCH, on passe à la ligne suivante. Lorsqu'on veut savoir combien de lignes on a déjà récupérées (le numéro de la ligne en cours), il faut utiliser l'attribut %ROWCOUNT



Curseur Explicite et Enregistrement

› SYNTAXE :

Nom_enregistrement nom_curseur%ROWTYPE;

DECLARE

CURSOR c1 **IS SELECT** nom, moyenne **FROM** etudiant **WHERE** Dept='info';

Rec1 c1%ROWTYPE; -- Rec1 de même schéma que c1

e1 **NUMBER** :=0;

e2 **NUMBER** :=0;

BEGIN

OPEN c1;

LOOP

FETCH c1 **INTO** Rec1;

EXIT WHEN c1%NOTFOUND;

IF Rec1.moyenne < 10 **THEN** e1:=e1+1;

ELSE e2:=e2+1;

END IF;

END LOOP;

CLOSE c1;

END;



Curseur Explicite : Les instructions utilisées

Instruction	Description
cursor nomCur is requête ;	Déclare le curseur.
open nomCur ;	Ouvre le curseur en chargeant les lignes en mémoire centrale. Aucune exception levée si la requête ne renvoie aucune ligne.
fetch nomCur into var1, ..., varN ;	Avance d'une ligne et affecte les valeurs de la ligne aux variables.
close nomCur ;	Ferme le curseur.
nomCur%isOpen	true si le curseur est ouvert, false sinon.
nomCur%notFound	true si le dernier fetch n'a pas renvoyé de ligne (fin de curseur), false sinon.
nomCur%found	true si le dernier fetch a renvoyé une ligne, false sinon.
nomCur%rowCount	Nombre de lignes parcourues jusqu'à présent.
nomCur%rowType	Définit un enregistrement de même structure que le curseur



Plus simple : Boucle dédiée aux curseurs explicites

- › PL/SQL offre une boucle FOR spécifique plus simple pour les curseurs explicites. Elle prend en charge toutes les opérations du curseur (OPEN, FETCH, EXIT et CLOSE)
- › Le record (nom_record) est déclaré implicitement

```
DECLARE
    CURSOR nom_curseur [(paramètre, ...)] IS ordre_select;
BEGIN
    FOR nom_record IN nom_curseur [(paramètre,...)]
    LOOP
        ...
        -- traitement de la ligne courante
    END LOOP;
END;
```



Plus simple : Boucle dédiée aux curseurs explicites

DECLARE

CURSOR c1 **IS SELECT** nom, moyenne **FROM** etudiant ;

-- PAS DE DECLARATION DE VARIABLE DE RECEPTION

e1 **NUMBER** :=0;

e2 **NUMBER** :=0;

BEGIN

-- PAS D'OUVERTURE DE CURSEUR et PAS DE FETCH

FOR rec_c1 **IN** c1 **LOOP**

IF rec_c1.moyenne < 10 **THEN** e1:=e1+1;

ELSE e2:=e2+1;

END IF;

END LOOP;

-- PAS DE CLOSE

DBMS_OUTPUT.PUT_LINE(e2 || ' etudiants ont réussi');

DBMS_OUTPUT.PUT_LINE(e1 || ' etudiants n ont pas réussi');

END;

rec_c1 est une variable de réception de type enregistrement de même structure que c1

Pas besoin de la déclarer



Plus simple : Boucle dédiée aux curseurs explicites

- › La boucle FOR dédié aux curseurs explicites peut être utilisé sans déclaration de curseur en utilisant un curseur anonyme.
- › Syntaxe :

```
FOR nom_record IN (Commande_SELECT)
LOOP
    ...
    -- traitement de la ligne courante
END LOOP;
```



Plus simple : Boucle dédiée aux curseurs explicites

DECLARE

-- PAS DE DECLARATION DU CURSEUR

-- PAS DE DECLARATION DE VARIABLE DE RECEPTION

e1 NUMBER :=0;

e2 NUMBER :=0;

BEGIN

-- PAS D'OUVERTURE DE CURSEUR et PAS DE FETCH

FOR rec_c1 IN (SELECT nom, moyenne FROM etudiant)

LOOP

IF rec_c1.moyenne < 10 THEN e1:=e1+1;

ELSE e2:=e2+1;

END IF;

END LOOP;

-- PAS DE CLOSE

DBMS_OUTPUT.PUT_LINE(e2 || ' etudiants ont réussi');

DBMS_OUTPUT.PUT_LINE(e1 || ' etudiants n ont pas réussi');

END;



Exercice

- › Ecrire un bloc PL/SQL qui sélectionne l'ensemble des employés dont le salaire ne dépasse pas 1000 DT et les augmente de 500 DT.



Solution

```
DECLARE
```

```
    CURSOR cur_sal IS SELECT * FROM EMP WHERE SAL < 1000;
```

```
BEGIN
```

```
    FOR rec_emp IN cur_sal
```

```
    LOOP
```

```
        UPDATE EMP
```

```
        SET SAL= SAL + 500 WHERE EMPNO = rec_emp.EMPNO;
```

```
    END LOOP;
```

```
END;
```



Exercice

› Soit la table employé

EMPLOYEE (matricule, nom, prénom, salaire, prime)

› Ecrire un bloc PL/SQL permettant de chercher les employés gagnant moins de 3000 DT, et suivant les cas on augmente leur prime.

- Si le prime < 500, on l'augmente par 20%
- Sinon on l'augmente par 10 %



Solution

```
DECLARE
    CURSOR inc_prime IS SELECT * FROM employe WHERE salaire < 3000;
BEGIN
    FOR rec_inc_prime IN inc_prime
    LOOP
        IF rec_inc_prime.prime < 500 THEN
            UPDATE employe
                SET prime = prime * 1.2
                WHERE matricule = rec_inc_prime.matricule;
        ELSE
            UPDATE employe
                SET prime = prime * 1.1
                WHERE matricule = rec_inc_prime.matricule;
        END IF;
    END LOOP;
END;
```

Les curseurs explicites paramétrés





Curseurs explicites 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 (***sauf si on utilise la boucle FOR dédiée qui ferme automatiquement le curseur***).



Curseurs explicites paramétrés : Déclaration

› Syntaxe :

› **Cursor nom_curseur (nom_paramètre type DEFAULT valeur_par_défaut)**

- nom_paramètre : représente le nom de la variable paramètre
- type : représente le type SQL de la variable paramètre (doit correspondre en type avec la colonne visée)
- valeur_par_défaut : représente la valeur par défaut du paramètre (doit correspondre en type avec celui du paramètre)



Exemple 1 :

```
DECLARE
```

```
    CURSOR c1(p_dept etudiant.dept%TYPE) IS SELECT nom, moyenne FROM etudiant
```

```
    WHERE dept = p_dept;
```

```
    Rec1 c1%ROWTYPE;
```

```
    e1 NUMBER :=0;
```

```
    e2 NUMBER :=0;
```

```
BEGIN
```

```
    OPEN c1('info');
```

```
    LOOP
```

```
        ...
```

```
    END LOOP;
```

```
    CLOSE c1;
```

```
    OPEN c1('physique');
```

```
    LOOP
```

```
        ...
```

```
    END LOOP;
```

```
    CLOSE c1;
```

```
END;
```



Exemple 2 :

```
DECLARE
    CURSOR c1(p_dept etudiant.dept%TYPE) IS SELECT nom, moyenne
    FROM etudiant where dept = p_dept;
    e1 NUMBER :=0;
    e2 NUMBER :=0;
BEGIN
    FOR rec_c1 IN c1('info') LOOP
        ...
    END LOOP;
    FOR rec_c1 IN c1('physique') LOOP
        ...
    END LOOP;
    ...
END;
```



Remarques

- › Le seul ordre qui offre le choix entre curseur implicite et explicite est le *SELECT* lorsqu'il ne ramène qu'une seule ligne.
- › Il n'existe pas de curseur explicite de mise à jour
- › Un curseur explicite ne peut être qu'un **SELECT** : utiliser systématiquement des curseurs explicites pour les ordres **SELECT**

Les curseurs explicites : La commande FOR UPDATE





Curseur Explicite : FOR UPDATE

- › Si l'on utilise un curseur pour faire des mises à jour, on peut 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 du bloc PL/SQL.
- › Cependant, si quelqu'un d'autre a déjà verrouillé les mêmes lignes, il faudra attendre que l'autre libère le verrou pour que les lignes puissent être mises à jour.



Curseur Explicite : FOR UPDATE

- › 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;
```



Curseur Explicite : WHERE CURRENT OF

- › 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.
- › Donc pas besoin d'une condition WHERE sur la valeur d'un champ.



Curseur Explicite : WHERE CURRENT OF

› 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;



Exercice

- › Ecrire un bloc PL/SQL qui sélectionne l'ensemble des employés dont le salaire ne dépasse pas 1000 DT et les augmente de 500 DT.



Solution

```
DECLARE
```

```
    CURSOR cur_sal IS SELECT * FROM EMP WHERE SAL < 1000 FOR UPDATE;
```

```
BEGIN
```

```
    FOR rec_emp IN cur_sal
```

```
    LOOP
```

```
        UPDATE EMP SET SAL= SAL + 500 WHERE CURRENT OF cur_sal;
```

```
    END LOOP;
```

```
END;
```



Exercice

› Soit la table employé

EMPLOYEE (matricule, nom, prénom, salaire, prime)

› Ecrire un bloc PL/SQL permettant de chercher les employés gagnant moins de 3000 DT, et suivant les cas on augmente leur prime.

- Si le prime < 500, on l'augmente par 20%
- Sinon on l'augmente par 10 %

Solution



```
DECLARE
```

```
    CURSOR inc_prime IS SELECT * FROM employe WHERE salaire < 3000
```

```
    FOR UPDATE OF prime;
```

```
    V_prime employe.prime%type
```

```
BEGIN
```

```
    FOR rec_inc_prime IN inc_prime LOOP
```

```
        IF rec_inc_prime.prime < 500 THEN
```

```
            V_prime := rec_inc_prime.prime * 1.2;
```

```
        ELSE
```

```
            V_prime := rec_inc_prime.prime * 1.1;
```

```
        END IF;
```

```
        UPDATE employe SET prime = V_prime WHERE CURRENT OF inc_prime;
```

```
    END LOOP;
```

```
END;
```