SMI-S5

SGBD ORACLE : Curseurs

Pr. Issam QAFFOU

Laboratoire Ingénierie des Systèmes d'Information
Département d'Informatique
FSSM-UCA

C'est quoi un Curseur?

- Les instructions de type select … into … ne fonctionnent que sur des requêtes ne retournant qu'une unique ligne.
- Ce n'est pas le cas souvent dans toutes les applications Oracle. Quelle est la solution?
- Solution = Curseur
- **Un curseur est une variable dynamique** qui prend pour valeur le résultat d'une **requête multi-lignes.**
- **Un curseur est une zone mémoire** qui permet de traiter **individuellement chaque ligne renvoyée** par un SELECT.
- Le curseur permet d'accéder au champ où les commandes SQL sont exécutées et les informations sont stockées.

Deux types de Curseur

- En PL/SQL, il y a deux types de curseur:
  - Implicite: déclaré pour toute commande du langage de manipulation de données et pour les requêtes ne retournant qu'une seule ligne.
  - Explicite: utilisé en pl/sql au cas où on veut retourner plusieurs lignes. On l'utilise quand on ne veut pas traiter les exceptions: TOO_MANY_ROWS et NO_DATA_FOUND.
- Les curseurs explicites sont créés par l'utilisateur pour chercher et récupérer plus d'une ligne à partir de la base de données.

Quand l'expliciter?

- Le curseur explicite, stocke les enregistrements résultats d'un « select » et permet de les récupérer un par un pour pouvoir les traiter.
- La récupération se fait par « FETCH »
- «FETCH» récupère la ligne courante du curseur et copie les données dans des variables.
- Le traitement à effectuer est à préciser par l'utilisateur.

Les attributs d'un curseur

- Ils permettent d'avoir des informations sur un curseur.
- La syntaxe utilisée: <nom_curseur>%<nom_attribut>
- Ces attributs peuvent être utilisés pour des curseurs implicites comme pour des curseurs explicites mais les résultats sont différents.

Nom de l'attribut	Résultat retourné
NOTFOUND	Vrai, si aucune ligne n'est récupérée
FOUND	Vrai, si une ligne est récupérée
ROWCOUNT	Nbre de lignes stockées dans le curseur
ISOPEN	Vrai, si le curseur est déjà ouvert

Les attributs d'un curseur

- Les résultats sont différents pour le curseur implicite

Nom de l'attribut	Résultat retourné
NOTFOUND	Vrai, si aucune ligne n'est affectée par une opération DML
FOUND	Vrai, si toutes les lignes résultats sont affectées
ROWCOUNT	Nbre de lignes affectées par l'opération DML la plus récente

- ISOPEN est propre au curseur explicite.
- Puisque le curseur est implicite alors on accède à ses informations par: SQL%<nom_attribut>

Les attributs d'un curseur

- Exemple:

```
Begin
    update cours set
frais=frais*1.2
     where durée>25;
    /* si le Nbre de lignes
affectées est
     Supérieur à 5 */
    If SQL%ROWCOUNT > 5
         then Rollback;
         Else commit;
    End if;
End;
```

Création et utilisation

- Pour travailler avec un curseur explicite, le programmeur doit suivre les étapes suivantes:
  1. Déclaration
  2. Ouverture
  3. Récupération et traitement
  4. Fermeture
- Ces étapes sont à suivre par ordre mais ne sont pas inéluctables.
- On peut utiliser un curseur explicite en utilisant la boucle FOR qui permet de faire toutes ces étapes automatiquement.

Création et utilisation

Déclaration

- La déclaration du curseur se fait dans la zone de déclaration dans le bloc PL/SQL.
- La syntaxe générale est:

  Cursor <nom_curseur> [(paramètre[, paramètre, …])] is

  <requête_select>;

- Exemple:

curseur explicite sans paramètre

```
Declare
  Cursor cours_d is
     Select ccode,
nom
     From cours;
Begin
…
End;
```

Ouverture

- Pour pouvoir travailler sur le curseur déclaré, il faut l'ouvrir.
- Pour ouvrir un curseur on utilise "OPEN"
- "OPEN" exécute la commande select et met les lignes récupérées dans le curseur. Elle les prépare pour le traitement.
- Syntaxe générale: OPEN <nom_curseur> [(arg_1,…)];
- Arg_i sont les valeurs affectées aux paramètres du curseur s'il y en a.
- Si le curseur n'a pas de paramètres on écrit simplement: OPEN <nom_curseur>;

Ouverture

- Exemple

```
Declare
  Cursor cours_d is
      Select ccode,
nom
      From cours;
Begin
      OPEN cours_d;
      …
End;
```

Extraction d'une ligne

- Une fois le curseur est ouvert, le programmeur peut parcourir les lignes stockées dedans.
- Pour récupérer ligne par ligne on utilise "FETCH"
- "FETCH" prend les données de l'enregistrement courant et copie les valeurs de ses colonnes dans des variables citées après INTO.
- Syntaxe: FETCH <nom_curseur> INTO var1, var2,…
- Pour chaque colonne de l'enregistrement courant doit correspondre une variable.
- Les types des variables doiventt correspondre aux types des colonnes stockés dans le curseur.

Extraction d'une ligne

- Puisque "FETCH" récupère une seule ligne chaque fois, alors il faut utiliser une boucle pour pouvoir parcourir toutes les lignes.
- L'instruction "FETCH" est écrite entre loop et end loop.
- Pour que la boucle ne soit pas infinie, il faut lui indiquer de sortir après avoir parcouru toutes les lignes.
- La condition d'arrêt est exprimée par l'attribut "notfound" en écrivant: <nom_curseur>%notfound;

Extraction d'une ligne

- Exemple

```
declare
    cursor cours_d is
      select ccode, nom
      from cours;
    v_ccode cours.ccode%type;
    v_nom cours.nom%type;
begin
     open cours_d;
     loop
       fetch cours_d into v_ccode, v_nom;
       exit when cours_d%notfound;
      end loop;
end;
```

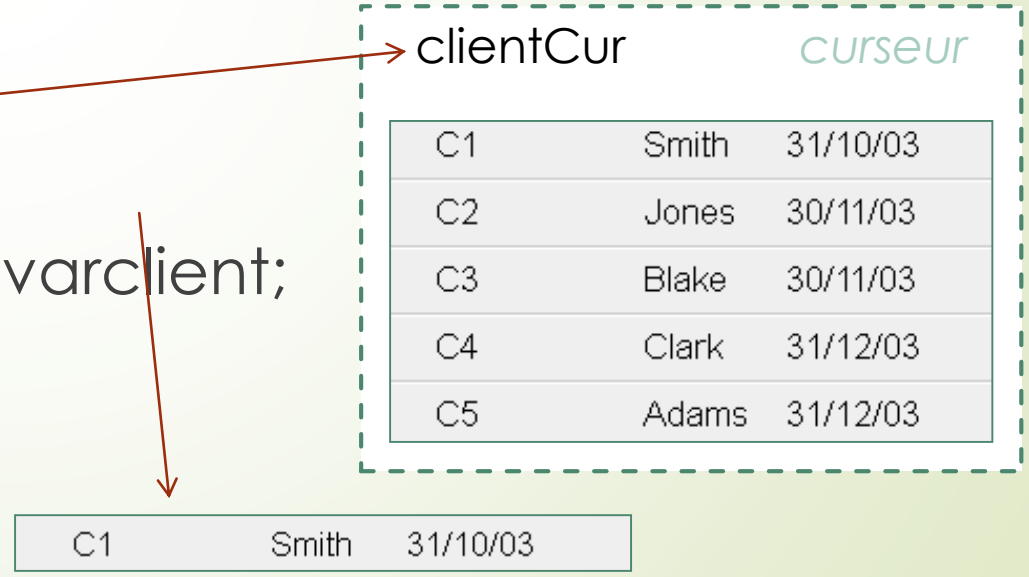
Fermer un curseur

- Pour libérer de la mémoire, il faut fermer le curseur après avoir terminé tous les traitements sur toutes les lignes stockées dans ce curseur.
- La fermeture se fait à l'aide de "Close"
- Syntaxe: close <nom_curseur>;
- Exemple

```
Declare
  Cursor cours_d is
…
Begin
      …
      …
      Close cours_d;
End;
```

Principe d'un curseur

```
DECLARE
    CURSOR clientCur IS SELECT * FROM Client;
    varclient Client%ROWTYPE; /*rowtype résume tous les
    types de Client, comme les structure en C */
BEGIN
    OPEN clientCur ;
    FETCH clientCur  INTO  varclient;
    CLOSE clientCur ;
END;
/
```



Boucle FOR

- Elle est utilisée pour simplifier le parcours.
- Les étapes de parcours ne sont pas nécessaires quand on utilise la boucle for.
- Syntaxe

```
For <rowtype_variable> IN
<nom_curseur>
Loop
    Instructions;
End loop;
```

Boucle FOR

- Exemple

```
DECLARE
cursor cours_c is
…
BEGIN
   open cours_c;
   loop
      fetch cours_c into
v_ccode,v_frais;
      exit when cours_c%notfound;
      . . .
   end loop;
   close cours_c;
END;
```

```
declare
   cursor cours_c is
      select ccode,frais
      from cours;
begin
/*Le curseur est ouvert et une
ligne est récupérée. La boucle
est terminée quand tous les
enregistrements sont récupérés */
   for rec in cours_c
   loop
      if rec.fee > 5000 then
          -- traitement à effectuer
      end if;
    end loop;
end;
```

Exemple

- sélectionner l'ensemble des employés dont le salaire ne dépasse pas 6000 et les augmenter de 500:

```
DECLARE
   CURSOR SalCur IS
        SELECT * FROM EMP WHERE SAL<6000.00;
   employe EMP%ROWTYPE;
BEGIN
   OPEN SalCur;
   LOOP
      FETCH SalCur INTO employe;
      EXIT WHEN SalCur%NOTFOUND;
      UPDATE EMP
      SET SAL=SAL+500.00
      WHERE EMPNO=employe.empno;
   END LOOP;
   CLOSE SalCur ;
END;
/
```

```
BEGIN
   FOR employe IN SalCur
   LOOP
      UPDATE EMP
      SET SAL=6500.00 WHERE
      EMPNO=employe.empno;
   END LOOP;
END;
```

Curseur paramétré

- Un curseur paramétré est un curseur dont la requête contient des variables dont les valeurs ne seront fixées qu'à l'ouverture.
- On précise la liste des noms et des type des paramètres entre parenthèses après le nom du curseur
- Syntaxe:

  CURSOR <nom_curseur> (param1, param2,...) IS <requête>;

- Exemple

```
CURSOR enfants ( numparent
NUMBER) IS
   SELECT *
   FROM PERSONNE
   WHERE pere = numparent
         OR mere = numparent ;
```

Exemple Récapitulatif

```
DECLARE
        CURSOR parent IS
                SELECT *
                FROM PERSONNE;
        p parent%rowtype;
        CURSOR enfants (numparent NUMBER) IS
                SELECT *
                FROM PERSONNE
                WHERE pere = numparent
                OR mere = numparent;
        e enfants%rowtype;
BEGIN
        FOR p IN parent LOOP
                DBMS_OUTPUT.PUT_LINE('Les enfants de '|| p.prenom ||
                        ' ' || p.nom || ' sont : ');
                FOR e IN enfants(p.numpers) LOOP
                        DBMS_OUTPUT.PUT_LINE(' * '|| e.prenom
                                || ' ' || e.nom     );
                END LOOP;
        END LOOP;
END;
/
```