Cours 2 : Les curseurs PL/SQL



Auteur : Riadh ZAAFRANI

2ème année SI
Février 2010

Présentation des curseurs

- L'une des plus importantes caractéristiques de **PL/SQL** est la possibilité de manipuler les données **ligne par ligne**.
- **SQL** est en effet un langage de type **tout ou rien**. Il est impossible de tester ou de modifier de manière sélective une ligne particulière dans un ensemble de lignes ramenées par un ordre **SELECT**.

Présentation des curseurs

- Lorsqu'on exécute un ordre **SQL** à partir de **PL/SQL**, **Oracle** alloue une zone de travail privée pour cet ordre.
- Les **curseurs PL/SQL** sont un mécanisme permettant de **nommer** cette zone de travail et de **manipuler** les données qu'elle contient.
- Un **curseur PL/SQL** permet de récupérer et de traiter les données de la base dans un programme PL/SQL, **ligne par ligne**.

Types des curseurs

- Il existe **deux sortes** de **curseurs** :
  - **Les curseurs explicites :** Ils sont créés et gérés par l'**utilisateur** pour traiter un ordre **SELECT** qui ramène plusieurs lignes. Le traitement du résultat pourra se faire ligne par ligne.
  - **Les curseurs implicites :** Ils sont générés **automatiquement** par le noyau pour toute requête SQL, même pour ceux qui ne retournent qu'une ligne.
- Les **curseurs implicites** ont les inconvénients suivants :
  - Ils sont moins performants que les curseurs explicites
  - Ils sont plus sujets aux erreurs de données
  - Ils laissent moins de contrôle au programmeur

Les curseurs explicites

- Pour les requêtes qui renvoient plus d'un enregistrement (même celles qui renvoient un seul enregistrement), vous pouvez déclarer explicitement un curseur, ce qui permet de traiter individuellement les lignes retournées.
- Un programme **PL/SQL** ouvre un curseur, traite les enregistrements retournés par l'ordre SQL, puis ferme le curseur.
- Le curseur permet **d'isoler l'enregistrement courant** d'un jeu de résultats.

Les étapes de la vie d'un curseur

- Les **étapes** d'utilisation d'un curseur explicite, pour traiter un ordre SELECT, sont les suivantes:
  - **Déclaration** du curseur
  - **Ouverture** du curseur
  - **Traitement des lignes**
  - **Fermeture** du curseur

La déclaration du curseur

- Tout **curseur explicite** utilisé dans un bloc **PL/SQL** doit obligatoirement être déclaré dans la section **DECLARE** du bloc, en précisant son nom et l'ordre SQL associé.
- La syntaxe de déclaration d'un curseur explicite est :
- **CURSOR** NOM_CURSEUR **[(**NOM_ARGUMENT TYPE :=VALEUR_DEFAUT **[,…])]** **IS** REQUETE**;**

La déclaration du curseur

- La requête peut contenir tous les **ordres SQL d'interrogation de données**, y compris les opérateurs ensemblistes UNION, INTERSECT ou MINUS.
- Les types d'arguments sont les suivants : **CHAR**, **NUMBER**, **DATE**, **BOOLEAN**; leur longueur n'est pas spécifiée.
- Le passage des valeurs des paramètres s'effectue à l'ouverture du curseur.

Les curseurs explicites : Exemples

- **SQL>** declare
  2 CURSOR c_employé IS SELECT nom,prénom, salaire, commission
  3 FROM employés
  4 ORDER BY nom;
- Nous avons créé dans cet exemple, un curseur **c_employé** qui contient les colonnes nom, prénom, salaire, commission pour l'ensemble des enregistrements de la table employés.

Les curseurs explicites : Exemples

- **SQL>**declare
  2 CURSOR c_produit (
  3 a_no_Fournisseur Produit.no_Fournisseur%TYPE:=1,
  4 a_code_catégorie Produit.code_catégorie%TYPE:=1)
  5 IS SELECT nomproduit, prix_unitaire
  6 FROM produit
  7 WHERE no_Fournisseur = a_no_Fournisseur
  8 AND code_catégorie = a_code_catégorie;
- Cet exemple expose la création d'un curseur **c_produit** contenant les colonnes **nomproduit**, **prix_unitaire** pour l'ensemble des produits du fournisseur et de la catégorie donnés, par l'intermédiaire des arguments **a_no_Fournisseur** et **a_code_catégorie**.

Les curseurs explicites : Exemples

- **Remarque :** Les expressions de calcul ou les fonctions SQL, qui se trouvent dans la clause **SELECT** de la requête du curseur, doivent comporter un **alias** pour pouvoir être référencées :
- **SQL>** declare
  2 CURSOR c_somme_sal
  3 IS SELECT fonction, sum(salaire) total_salaire
  4 FROM employés
  5 GROUP BY fonction;

Ouverture d'un curseur

- Dès que vous ouvrez le curseur, l'exécution de l'ordre SQL est lancée. Cette phase d'ouverture s'effectue dans la section **BEGIN** du bloc :
- **OPEN** NOM_CURSEUR **[(**VALEUR_ARGUMENT **[,…])];**
- **SQL>** declare
  2 CURSOR c_employé IS SELECT nom,prénom, salaire, commission
  3 FROM employés
  4 ORDER BY nom;
  5 Begin
  6 **open c_employé;**

Ouverture d'un curseur

- Les arguments spécifiés lors de la déclaration du curseur sont définis lors de l'ouverture du curseur. Chaque argument est affecté à une seule valeur selon deux modèles :
- **Association par position :** Dans ce cas, chaque argument est remplacé par la valeur occupant la même position dans la liste.
- **SQL>** declare
  2 CURSOR c_produit (
  3 a_no_Fournisseur Produit.no_Fournisseur%TYPE:=1,
  4 a_code_catégorie Produit.code_catégorie%TYPE:=1)
  5 IS SELECT nomproduit,prix_unitaire FROM produit
  6 WHERE no_Fournisseur = a_no_Fournisseur
  7 AND code_catégorie = a_code_catégorie;
  8 begin
  9 **open c_produit(2);**

Ouverture d'un curseur

- Dans l'exemple précédent, l'ouverture comporte un seul argument **v_no_Fournisseur** affecté à 2.
- L'argument **v_code_catégorie** ne figurant pas dans la déclaration, il est affecté avec sa valeur par défaut.
- Les arguments doivent être renseignées obligatoirement s'il n'y a pas de valeur par défaut déclarée.
- Les arguments associés par position sont affectés dans l'ordre de leur déclaration dans le curseur. Vous ne pouvez pas affecter le deuxième sans renseigner le premier.

Ouverture d'un curseur

- **Association par nom :** Dans ce cas, chaque argument peut être indiqué dans un ordre quelconque en faisant apparaître la correspondance de façon explicite sous la forme :
- **OPEN** NOM_CURSEUR **[(**NOM_ARGUMENT **=>** VALEUR_ARGUMENT**[,**…**])];**
- **SQL>** declare
  2 CURSOR c_produit (
  3 a_no_Fournisseur Produit.no_Fournisseur%TYPE:=1,
  4 a_code_catégorie Produit.code_catégorie%TYPE:=1)
  5 IS SELECT nomproduit,prix_unitaire FROM produit
  6 WHERE no_Fournisseur = a_no_Fournisseur
  7 AND code_catégorie = a_code_catégorie;
  8 begin
  9 **open c_produit(a_code_catégorie => 2);**

Traitement des lignes d'un curseur

- L'ordre **OPEN** a forcé l'exécution de l'ordre SQL associé au curseur.
- Il faut maintenant récupérer les lignes de l'ordre SELECT et les traiter une par une, en stockant la valeur de chaque colonne de l'ordre SQL dans une variable réceptrice.
- La commande **FETCH** permet de récupérer un enregistrement et transfère les valeurs projetés par l'ordre SELECT dans un enregistrement ou dans une liste de variables :
- **FETCH** NOM_CURSEUR **{**NOM_ENREGISTREMENT **|** NOM_VARIABLE**[,**…**])];**
- Pour récupérer l'ensemble des enregistrements, Il faut donc prévoir une boucle.

Statut d'un curseur

- Pour chaque exécution d'un ordre de manipulation du curseur, le noyau renvoie une information appelée **statut**, qui indique si l'ordre a été exécuté avec succès ou non. Cette information est disponible dans le programme par l'intermédiaire de **quatre attributs** rattachés à chaque curseur.
- **%FOUND** : C'est un attribut de type booléen. Le curseur explicite est **VRAI** si l'ordre FETCH ramène au moins une ligne.
  Le curseur implicite est **VRAI** si les instructions INSERT, UPDATE et DELETE traitent au moins une ligne.
- **%NOTFOUND** : C'est un attribut de type booléen. Le curseur explicite est **VRAI** si l'ordre FETCH ne ramène pas de lignes. Le curseur implicite est VRAI si les instructions INSERT, UPDATE et DELETE ne traitent aucune ligne.

Statut d'un curseur

- **%ISOPEN** : C'est un attribut de type booléen ; il est **VRAI** si le curseur est ouvert. Le curseur implicite est toujours **FAUX** car Oracle referme toujours les curseurs qu'il ouvre après chaque utilisation.
- **%ROWCOUNT** : Cet attribut est de type numérique. Le curseur implicite indique le **nombre de lignes traitées** par les ordres INSERT, UPDATE et DELETE. Le curseur explicite est **incrémenté** à chaque ordre FETCH ; il traduit donc la **n^{ième} ligne traitée**.
- La syntaxe de consultation d'un attribut est :
- **NOM_CURSEUR%ATTRIBUT**.

Statut d'un curseur

```
SQL> declare
  2  CURSOR c_produit ( a_no_Fournisseur
Produit.no_Fournisseur%TYPE:=1,
  3   a_code_catégorie Produit.code_catégorie%TYPE)
  4  IS SELECT nomproduit,prix_unitaire FROM produit
  5     WHERE no_Fournisseur = a_no_Fournisseur
  6      AND code_catégorie = a_code_catégorie;
  7   v_produit c_produit%ROWTYPE;
  8  begin
  9     open c_produit(a_code_catégorie  => 1);
 10     if c_produit%ISOPEN then
 11        dbms_output.put_line('La  valeur ROWCOUNT
:'|| c_produit%ROWCOUNT);
```

Statut d'un curseur

```
 12        loop
 13           fetch  c_produit  into  v_produit;
 14           exit  when  c_produit%NOTFOUND;
 15           dbms_output.put_line('Le produit : '''|| v_produit.nomproduit||
 16                        ''' est au prix '||v_produit.PRIX_UNITAIRE);
 17           dbms_output.put_line('La  valeur  ROWCOUNT  :  '||
c_produit%ROWCOUNT);
 18         end loop;
 19       end  if;
 20   close  c_produit;
 21  end;
 22  /
La  valeur  ROWCOUNT  :  0
Le produit : 'Chai' est au prix  90
La  valeur  ROWCOUNT  :  1
Le produit : 'Chang' est au prix  95
La  valeur  ROWCOUNT  :  2
```

Statut d'un curseur

Dans le cas d'un curseur implicite, un attribut est associé à un curseur par la notation **SQL%ATTRIBUT**. La valeur de l'attribut est relative au dernier ordre SQL exécuté avant son utilisation.

```
SQL>begin
  2    Update employés SET commission = 500
  3    Where numemployé = 10;
  4    if SQL%FOUND  then
  5    dbms_output.put_line('La valeur ROWCOUNT :'||
SQL%ROWCOUNT);
  6    end if;
  7  end;
  8  /
La valeur ROWCOUNT :1
```

Les boucles et les curseurs

Dans la mesure où l'utilisation principale d'un curseur est le parcours d'un ensemble de lignes ramenés par l'exécution SELECT associé, il peut être intéressant d'utiliser une syntaxe plus simple pour l'ouverture du curseur et le parcours de la boucle.

Oracle propose une variante de la boucle **FOR** qui déclare **implicitement** la variable de parcours, ouvre le curseur, réalise les **FETCH** successifs et ferme le curseur :

```
FOR NOM_ENREGISTREMENT IN NOM_CURSEUR LOOP
    INSTRUCTIONS
END LOOP
```

Les boucles et les curseurs

```
SQL> declare
  2   CURSOR c_produit ( a_no_Fournisseur
Produit.no_Fournisseur%TYPE:=1,
  3      a_code_catégorie  Produit.code_catégorie%TYPE)
  4   IS SELECT nomproduit,prix_unitaire FROM produit
  5      WHERE no_Fournisseur = a_no_Fournisseur
  6       AND code_catégorie = a_code_catégorie;
  7  begin
  8   for v_produit in c_produit(a_code_catégorie  => 1) loop
  9     dbms_output.put_line('Le produit : '''|| v_produit.nomproduit||
 10                        ''' est au prix '||v_produit.PRIX_UNITAIRE);
 11   end loop;
 12  end;
 13  /
Le produit : 'Chai' est au prix 90
Le produit : 'Chang' est au prix 95
```

Les boucles et les curseurs

Remarquez dans l'exemple précédent que la variable **v_produit** est définie automatiquement comme une variable de type enregistrement en lecture seule.

Il est également possible de ne pas déclarer le curseur dans la section **DECLARE**, mais de spécifier celui_ci directement dans l'instruction **FOR**.

Les boucles et les curseurs

```
SQL> begin
  2    for v_produit in (SELECT nomproduit,prix_unitaire
                         FROM  produit
  3                      WHERE no_Fournisseur =  1
  4                      AND code_catégorie  =1) loop
  5    dbms_output.put_line('Le produit : '''||
                                    v_produit.nomproduit
  6                 || ''' est au prix '||
v_produit.PRIX_UNITAIRE);
  7    end loop;
  8  end;
  9  /
Le produit : 'Chai' est au prix 90
Le produit : 'Chang' est au prix 95
```

Les boucles et les curseurs

PL/SQL permet aussi, pour la constitution de la requête, d'utiliser les variables déclarées dans le bloc ou toute autre variable accessible.

```
SQL> declare
  2    v_no_Fournisseur Produit.no_Fournisseur%TYPE;
  3    v_code_catégorie Produit.code_catégorie%TYPE;
  4    CURSOR c_produit
  5    IS SELECT nomproduit,prix_unitaire FROM produit
  6       WHERE no_Fournisseur = v_no_Fournisseur
  7        AND code_catégorie = v_code_catégorie;
```

Les boucles et les curseurs

```
  8  begin
  9     v_no_Fournisseur := 1;
 10     v_code_catégorie := 1;
 11     for v_produit in c_produit loop
 12        dbms_output.put_line('Le produit : '''||
v_produit.nomproduit||
 13                              ''' est au prix
'||v_produit.PRIX_UNITAIRE);
 14     end loop;
 15  end;
 16  /
Le produit : 'Chai' est au prix 90
Le produit : 'Chang' est au prix 95
```

Les boucles et les curseurs

Attention : Prenez garde aux noms des variables lorsque vous mélangez les **variables PL/SQL** et les **colonnes de la base** dans les requêtes à l'intérieur d'un bloc PL/SQL.

Si votre variable a le même nom que la colonne, Oracle utilise toujours la colonne.

Il n'y a pas d'erreur à la compilation, mais vous n'obtenez pas le résultat escompté.

Les boucles et les curseurs

```
SQL> declare
  2    numemployé employés.numemployé%TYPE:=1;
  3    CURSOR c_employé
  4    IS SELECT numemployé,nom FROM employés
  5       WHERE  numemployé = numemployé;
  6  begin
  7    for v_employé in c_employé loop
  8       dbms_output.put_line('numéro employé : '||
v_employé.numemployé||' a pour nom : '||v_employé.nom);
  9    end loop;
 10  end;
 11  /
numéro employé : 1 a pour nom : Zaafrani
numéro employé : 2 a pour nom : Chater
numéro employé : 3 a pour nom : Gharbi
numéro employé : 10 a pour nom : Ben Brahim
```

Les curseurs FOR UPDATE

Jusqu'à présent, tous les exemples des curseurs étaient en lecture seule. Aucune modification des données retournées n'a été effectuée.

Lorsqu'on lance un curseur avec un ordre SELECT sur la base pour récupérer des enregistrements, **aucun verrou** n'est mis sur les lignes sélectionnées.

Il y a toutefois des situations où l'on souhaite verrouiller un ensemble de lignes avant même de les avoir modifiés par programme.

Pour ce type de verrou, Oracle offre la clause **FOR UPDATE** dans la déclaration du curseur.

Les curseurs FOR UPDATE

- Lorsqu'on exécute un ordre **SELECT … FOR UPDATE**, les enregistrements ramenés sont **verrouillés** pendant toute la durée du travail.
- Personne ne peut modifier ces enregistrements avant qu'un ordre **ROLLBACK** ou **COMMIT** n'ait été exécuté.
- Lorsqu'un de ces ordres est exécuté, les verrous de lignes sont relâchés.
- **Attention :** Il est impossible d'exécuter un ordre **FETCH** sur un curseur **FOR UPDATE** après avoir fait **COMMIT** ou **ROLLBACK**. La position dans le curseur est perdue.

Les curseurs FOR UPDATE

- La syntaxe de déclaration d'un curseur en mise à jour est :
- **CURSOR NOM_CURSEUR**
  [(NOM_ARGUMENT TYPE :=VALEUR_DEFAUT
  [,…])] IS REQUETE
  FOR UPDATE [OF NOM_COLONNE [,…])]
  [{NOWAIT | WAIT NB_SECONDES}];
- **NOM_COLONNE :** Une ou plusieurs colonnes sur laquelle porte la clause FOR UPDATE.
- **NOWAIT :** Demande à Oracle de verrouiller les enregistrements immédiatement. Si les enregistrements sont déjà verrouillés, alors l'ouverture du curseur provoque une erreur.
- **WAIT :** Si les enregistrements sont déjà verrouillés, alors le programme attend NB_secondes pour le déverrouillage, sinon l'ouverture du curseur provoque une erreur.

Les curseurs FOR UPDATE

- **SQL>** declare
  2 CURSOR c_commande IS
  3 SELECT *
  4 FROM commandes
  5 WHERE TRUNC(DateEnvoi,'Month') = TRUNC(SYSDATE,'Month')
  6 FOR UPDATE OF DATECOMMANDE, PORT;
- La liste des colonnes spécifiée après le mot dé **OF** de la clause **FOR UPDATE** n'implique pas de ne modifier que les colonnes listées.
- Las verrous sont posées sur des lignes complètes; la liste OF est seulement un moyen de documenter plus clairement ce qu'on a l'intention de changer.

WHERE CURRENT OF

- L'instruction **WHERE CURRENT OF** permet d'accéder directement à la ligne ramenée par un l'ordre FETCH d'un curseur déclaré **FOR UPDATE**, afin de la traiter par une opération de mise à jour (**UPDATE** ou **DELETE**).
- Pour spécifier que l'on veuille traiter la ligne courante ramenée par l'ordre FETCH, on doit ajouter la clause : **WHERE CURRENT OF NOM_CURSEUR;** à l'opération de mise à jour (**UPDATE** ou **DELETE**).

WHERE CURRENT OF

- **SQL>** declare
  2 CURSOR c_employé IS SELECT nom,prénom, fonction, salaire, commission FROM employés
  3 FOR UPDATE OF SALAIRE, COMMISSION;
  4 v_employé c_employé%ROWTYPE;

WHERE CURRENT OF

```
5  begin
6    for v_employé in c_employé loop
7      if v_employé.fonction = 'Chef des ventes' and
8        v_employé.Salaire + v_employé.commission <
2000 then
9        Update employés SET salaire = salaire * 1.1
10       WHERE CURRENT OF c_employé;
11      dbms_output.put_line('Modification de salaire de
l''employé : '|| v_employé.nom||  ' ' ||v_employé.prénom);
12       end if;
13    end loop;
14    COMMIT;
15 end;
16 /
```

- **Modification de salaire de l'employé : Ben Brahim Nadra**