Compléments de bases de données

COURS 2 :

PL/SQL

Décembre 5, 2019

ROMDHANI Senda

Lyon 1

* Cours extraits principalement des cours de Guillaume Cabanac, Fouad Dahak, Yacine Bouhabel, Amrouche Karima, Richard Grin

SOMMAIRe

- Introduction
- Éléments de base
- Curseurs
- Exceptions
- Procédures et fonction
- Déclencheurs (Triggers)

Petit rappel:

- PL/SQL est un langage qui intègre SQL et permet de programmer d'une manière procédurale (un langage procédurale contient simplement une série d'étapes à réaliser).
- Blocs anonymes: DECLARE, BEGIN, END
- On utilise des variables pour stocker les résultats des requêtes.
- Pour l'affichage des résultats il faut initialiser d'abord le buffer de sortie.
- Les boucles while, for..
- La directive SELECT … INTO permet de récupérer une seule ligne à la fois.

SOMMAIRe

Pourquoi les curseurs

En PL/SQL, on a parfois besoin de récupérer plusieurs lignes correspondant à une contrainte particulière. Cependant, la directive SELECT … INTO permet de récupérer une seule ligne à la fois.

Besoin d'un outil particulier permettant la manipulation des requêtes correspondant à plusieurs lignes au même temps : Les curseurs.

3. Curseurs

Définition des curseurs

- Un curseur est un pointeur vers un résultat d'une requête.
- Il existe deux types de curseurs :

- **Implicite :** un curseur peut être implicite (pas déclaré par l'utilisateur). Il est créé et géré en interne par le serveur afin de traiter les instructions SQL.
- **Explicite :** déclaré explicitement par le programmeur et qui permet de manipuler l'ensemble des résultats d'une requête.

Attributs des curseurs

Tous les curseurs ont des attributs que le programmeur peut utiliser :

- **%ROWCOUNT :** nombre de lignes traitées par le curseurs.
- **%FOUND :** vrai si au moins une ligne a été traitée par la requête.
- **%NOTFOUND :** vrai si aucune ligne n'a été traitée par la requête ou le dernier FETCH.
- **%ISOPEN** : vrai si le curseur est ouvert (utile seulement pour les curseurs explicites).

3.1. Curseurs implicites

Les curseurs implicites sont tous nommés SQL.

Exemple 1:

```
DECLARE
   nb_lignes NUMBER(10);
BEGIN
   DELETE FROM emp WHERE eno= 'E10';
   nb_lignes := SQL%ROWCOUNT;
 DBMS_OUTPUT.PUT_LINE('nb lignes supprimées est:'|| nb_lignes);
END;
```

ENO	ENAME	TITLE	CITY
E1	KRUNAL	MANAGER	Toronto
E2	KAUSHIK	PROGRAMMER	Toronto
E3	RAMESH	SUPPORT STAFF	Toronto
E4	JAGDISH	SUPPORT STAFF	Toronto
E5	DINESH	MANAGER	LONDON
E6	DIPU	SUPPORT STAFF	LONDON
E7	RAM	PROGRAMMER	LONDON
E8	SHYAM	PROGRAMMER	LONDON

➔ **Résultat de l'exécution** :

```
nombre de lignes supprimées est de:0
```

Il n y a aucun employé avec ENO=10. Donc aucune ligne supprimée.

3.1. Curseurs implicites

Exemple 2:

```
BEGIN
  FOR x IN (SELECT * FROM emp WHERE city='Toronto')
    dbms_Output.Put_Line(x.ename||''||x.title||'...
REALLY be raised :D');
  END LOOP;
END;
```

ENO	ENAME	TITLE	CITY
E1	KRUNAL	MANAGER	Toronto
E2	KAUSHIK	PROGRAMMER	Toronto
E3	RAMESH	SUPPORT STAFF	Toronto
E4	JAGDISH	SUPPORT STAFF	Toronto
E5	DINESH	MANAGER	LONDON
E6	DIPU	SUPPORT STAFF	LONDON
E7	RAM	PROGRAMMER	LONDON
E8	SHYAM	PROGRAMMER	LONDON

➔ **Résultat de l'exécution** :

```
KRUNAL MANAGER... should REALLY be raised :D
KAUSHIK PROGRAMMER... should REALLY be raised :D
RAMESH SUPPORT STAFF... should REALLY be raised :D
JAGDISH SUPPORT STAFF... should REALLY be raised :D
```

La boucle ouvre le curseur et récupère un enregistrement à la fois pour chaque boucle. A la fin de la boucle, le curseur est fermé.

3.2. Curseurs explicites

Pour traiter les SELECT qui renvoient plusieurs lignes:

1. Déclaration: les curseurs doivent être déclarés explicitement dans la zone DECLARE tout en spécifiant la requête SQL dedans.
2. Un curseur **explicite** doit être utilisé dans le code avec les commandes:
   - **OPEN:** ouvrir le curseur afin de l'utiliser dans la zone BEGIN
   - **FETCH**: Avancement ligne par ligne pour parcourir les résultats du curseur.
   - **CLOSE:** Fermeture du curseur après utilisation.

3.2. Curseurs explicites

Syntaxe de déclaration

```
CURSOR nom_du_cuseur IS
       un énoncé SELECT;
```

Remarques

Ne pas inclure la clause INTO dans la déclaration du curseur.

Exemple de déclaration

```
DECLARE
   CURSOR c1 IS
      SELECT ref, nom, qte
      FROM Article
      WHERE qte<500;
BEGIN
   …
END;
```

3.2. Curseurs explicites

Syntaxe d'OPEN

```
OPEN nom_du_curseur;
```

Syntaxe de FETCH

```
FETCH nom_du_curseur
   INTO [variable1, [variable2,…];
```

Remarques

- **La commande FETCH recherche les informations de la ligne en cours et les met dans les variables : variable1, variable2…**
- **On peut utiliser les attributs des curseurs (e.g., %NOTFOUND) pour tester le résultat du FETCH**

Préfixer le nom d'un curseur par c_ pour éviter les confusions

3.2. Curseurs explicites

Syntaxe de CLOSE

```
CLOSE nom_du_curseur;
```

Remarques

- **Le curseur doit être fermé après la fin du traitement des lignes. Il peut être rouvert si nécessaire.**
- **On ne peut pas rechercher des informations dans un curseur si ce dernier est fermé.**

3.2. Curseurs explicites

Exemple avec des variables définies par %TYPE

```
DECLARE
   CURSOR c_emp IS SELECT id, nom FROM emp where dept=30;
    v_id emp.id%TYPE;
   v_nom emp.nom%TYPE;
BEGIN
   OPEN c_emp;
   LOOP
      FETCH c_emp into v_id, v_nom;
      EXIT WHEN c_emp%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(v_id || ' ' || v_nom);
   END LOOP;
   CLOSE c_emp;
END;
```

3.2. Curseurs explicites

Exemple avec des variables définies par %ROWTYPE

```
DECLARE
   CURSOR c_emp IS SELECT id, nom FROM emp where dept=30;
    v_emp c_emp%ROWTYPE;
BEGIN
   OPEN c_emp;
   LOOP
      FETCH c_emp into v_emp;
      EXIT WHEN c_emp%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(v_emp.v_id || ' ' || v_emp.v_nom);
   END LOOP;
   CLOSE c_emp;
END;
```

3.2. Curseurs explicites

Utilisation de la boucle FOR

- La boucle FOR simplifie la programmation car elle évite d'utiliser explicitement les instructions OPEN, FETCH, CLOSE.
- En plus, elle déclare implicitement une variable de type ROW associé au curseur (ci-dessous, le type de 'variable' est curs%ROWTYPE).

Syntaxe de la boucle FOR

```
FOR variable IN curs LOOP
   …
END LOOP;
```

3.2. Curseurs explicites

Exemple avec une boucle FOR

```
DECLARE
   CURSOR c_client IS
      SELECT nom, adresse FROM clients;
BEGIN
   FOR v_client in c_client LOOP
      DBMS_OUTPUT.PUT_LINE('Nom : '||UPPER(c_client.nom) || ' Ville :' ||
      c_client.addresse);
   END LOOP;
END;
```

3.2. Curseurs explicites

Les curseurs paramétrés

- Un curseur paramétré peut servir plusieurs fois avec des valeurs de paramètres différentes.
- On doit fermer le curseur entre chaque utilisation de paramètres différents (sauf si on utilise la boucle FOR, car elle ferme le curseur automatiquement).

3.2. Curseurs explicites

Exemple avec un curseur paramétré

```
DECLARE
   CURSOR c_emp (p_dept NUMBER) IS
      SELECT dept, nom FROM emp where dept = p_dept;
BEGIN
   FOR v_emp in c_emp(10) LOOP
      DBMS_OUTPUT.PUT_LINE('Nom : '||UPPER(emp.nom));
   END LOOP;
   FOR v_emp in c_emp(20) LOOP
      DBMS_OUTPUT.PUT_LINE('Nom : '||UPPER(emp.nom));
   END LOOP;
END;
```

Les curseurs

En conclusion: déclaration

- Curseur nom_curseur votre_sélection_de_sélection
- Curseur nom_curseur (paramètre TYPE)
  votre_select_statement_using_param
- FOR x in (your_select_statement) LOOP ...