Fondements des bases de données

Programmation en PL/SQL Oracle – les curseurs

Équipe pédagogique BD

`http://liris.cnrs.fr/~mplantev/doku/doku.php?id=lif10_2016a`

Version du 13 octobre 2016

Curseurs

Les curseurs

Toutes les requêtes SQL sont associées à un *curseur* :

- Un curseur est un pointeur vers un résultat d'une requête
- Le curseur peut être implicite (pas déclaré par l'utilisateur) ou explicite.
- Les curseurs explicites permettent de manipuler l'ensemble des résultats d'une requête.

Les curseurs *implicites* sont tous nommés `SQL`

```
DECLARE
  v_nb_lignes NUMBER;
BEGIN
  DELETE FROM emp WHERE dep = 10;
  v_nb_lignes := SQL%ROWCOUNT;
  dbms_output.put_line('v_nb_lignes : ' || v_nb_lignes);
  -- ...
END;
```

Attributs des curseurs

Tous les curseurs ont des attributs que l'utilisateur peut utiliser :

`%ROWCOUNT` Nombre de lignes traitées par le curseur.

`%FOUND` Vrai si au moins une ligne a été traitée par la requête ou le dernier fetch.

`%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)

Les curseurs explicites

Pour traiter les `SELECT` qui renvoient plusieurs lignes

- Les curseurs doivent être déclarés explicitement. A la déclaration, on explicite la requête `SELECT` dont le résultat sera parcouru par le curseur.
- Le code doit les utiliser avec les commandes
  - `OPEN moncurseur`, pour ouvrir le curseur ;
  - `FETCH moncurseur`, pour avancer le curseur à la ligne suivante ;
  - `CLOSE moncurseur`, pour refermer le curseur

Utilisation

- On utilise souvent les curseurs dans une boucle `FOR` qui permet une utilisation *implicite* des instructions `OPEN`, `FETCH` et `CLOSE`.
- Généralement, on sort de la boucle quand l'attribut `NOTFOUND` est vrai.

Les curseurs explicites

Exemple de boucle LOOP pour les curseurs

```
DECLARE
   v_salaire EMP.sal%TYPE;
   v_total   EMP.sal%TYPE := 0;
   CURSOR c_salaires IS
         SELECT sal
         FROM emp;
BEGIN
   OPEN c_salaires;

   LOOP
     FETCH c_salaires INTO v_salaire;
     EXIT WHEN c_salaires%NOTFOUND;
     IF v_salaire IS NOT NULL THEN
       v_total := v_total + v_salaire;
       dbms_output.put_line('salaire : ' || v_salaire);
     END IF;
   END LOOP;

   CLOSE c_salaires;
   dbms_output.put_line('total : ' || v_total);
END;
```

☛ Ne pas oublier de fermer le curseur avec CLOSE

Les curseurs explicites

Type associé à un curseur avec %ROWTYPE

```
DECLARE
  CURSOR c_emp IS
        SELECT sal, dep
        FROM emp;
  v_emp       c_emp%ROWTYPE;
  v_total     v_emp.sal%type := 0;
BEGIN
  OPEN c_emp;

  LOOP
    FETCH c_emp INTO v_emp;
    EXIT WHEN c_emp%NOTFOUND;
    IF v_emp.sal IS NOT NULL THEN
      v_total := v_total + v_emp.sal;
      dbms_output.put_line('salaire : ' || v_emp.sal);
    END IF;
  END LOOP;

  CLOSE c_emp;
  dbms_output.put_line('total : ' || v_total);
END;
```

Boucle FOR pour un curseur

- Elle *simplifie* la programmation car elle évite d'utiliser explicitement les instruction `OPEN`, `FETCH` et `CLOSE`.
- En plus elle déclare *implicitement* une variable de type `ROW` associée au curseur.

Exemple

```
DECLARE
  CURSOR c_nom_emp IS
    SELECT nom, adresse
    FROM emp;
BEGIN
  FOR v_emp IN c_nom_emp LOOP
    dbms_output.put_line(
      'Employe : ' || UPPER(v_emp.nom) ||
      ' Ville : ' || v_emp.adresse);
  END LOOP;
END;
```

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

Curseurs 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 si on utilise pas la boucle `FOR` dédiée.

Exemple

```
DECLARE
  CURSOR c(p_dep EMP.dep%TYPE) IS
    SELECT dep, nom
    FROM emp
    WHERE dep = p_dep;
BEGIN
  FOR v_employe IN c(1) LOOP
          dbms_output.put_line('Dep 1 : ' || v_employe.nom);
        END LOOP;
        FOR v_employe IN c(2) LOOP
          dbms_output.put_line('Dep 2 : ' ||v_employe.nom);
        END LOOP;
END;
```

Fin du cours.