Université Mohammed V - Rabat

Ecole Mohammadia d'Ingénieurs

Filière Réseaux et Télécom

Bases de Données

Programmation PL/SQL

Mr N.EL FADDOULI

2022-2023

Plan

- Introduction
- Bloc PL/SQL
- Déclaration des variable
- Structure de contrôle
- Curseurs
- Les exceptions
- Les fonctions et procédures
- Les packages
- Les triggers

Les curseurs

- Définition d'un curseur
- Déclaration d'un curseur
- Utilisation d'un curseur

Curseur: définition

- Oracle crée des **zones de travail pour exécuter les ordres SQL**, stocker leurs résultats et les utiliser via un pointeur: **Curseur**
- Deux types de curseurs:
  - **Implicite**: créés automatiquement par Oracle, par exemple, lorsque la clause **INTO** accompagne le **SELECT**
  - **Explicite**: crées par le programmeur pour pouvoir traiter le résultat d'une requête **retournant plus d'un tuple**.
- L'utilisation d'un curseur explicite nécessite les étapes suivantes
  1. **Déclaration** du curseur : Section DECLARE
  2. **Ouverture** du curseur : Section BEGIN
  3. **Traitement** des lignes : Section BEGIN
  4. **Fermeture** du curseur : Section BEGIN ou EXCEPTION

Curseur: déclaration

- Déclaration du curseur:

```
DECLARE
        CURSOR  nom_curseur IS Requête_Select ;
        .......
```

Exemple: **employe** (…, **nomemp**, **sal**, **ville**, …)

```
DECLARE
    CURSOR emp_rabat  IS
    SELECT nomemp, sal FROM employe  WHERE upper(ville)= 'RABAT' ;
      …..
```


Curseur: Ouverture

- L'ouverture du curseur amorce l'**analyse** de la clause SELECT et son **exécution**.
- La réponse est déterminée et rangée dans un espace temporaire.
- L'ouverture du curseur se fait dans la section BEGIN du bloc

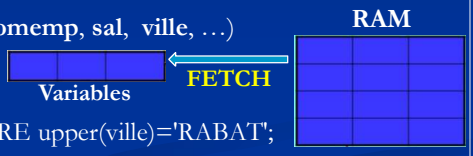
```
    OPEN  nom_curseur ;
```

Exemple:

```
DECLARE
CURSOR emp_rabat  IS
SELECT nomemp, sal FROM employe  WHERE upper(ville)= 'RABAT' ;
        ……..
BEGIN
        OPEN emp_rabat ;
        ……..
END ;
```

Curseur: Accès Aux Données

- L'accès aux données se fait par l'instruction: **FETCH INTO**
  FETCH ***nom_curseur*** **INTO** ***var1 , var2 , …***
- Le **FETCH** permet de récupérer le tuple **courant** de l'ensemble des tuples associés au curseur et stocker les valeurs dans des variables puis **se positionner sur le tuple suivant**.
- Pour traiter **plusieurs** lignes, il faut donc une boucle.



Exemple 1: La table utilisée est : **employe** (…, **nomemp**, **sal**, **ville**, …)

```
DECLARE
CURSOR emp_rabat IS
 SELECT nomemp, sal FROM employe WHERE upper(ville)='RABAT';
 nom employe.nomemp%TYPE ;    salaire employe.sal %TYPE ;
BEGIN
OPEN emp_rabat ;
FETCH emp_rabat INTO nom, salaire ;      -- On récupère le premier enregistrement
 WHILE emp_rabat%found LOOP              -- S'il y a un enregistrement récupéré
      ……                                 -- Traitement de l'enregistrement récupéré
    FETCH emp_rabat INTO nom, salaire ; --On récupère l'enregistrement suivant
END LOOP;
```

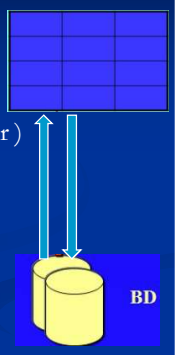

Curseur: Accès aux données

Exemple 2: **CREATE TABLE resultat (nom char(10), sal number (7,2))** ;

```
DECLARE
CURSOR emp_rabat IS
                SELECT nomemp, sal FROM employe WHERE upper(ville)= 'Rabat';
 nom employe.nomemp%TYPE ;    salaire employe.sal %TYPE ;
BEGIN
   OPEN emp_rabat ;
   FETCH emp_rabat INTO nom, salaire ;
   WHILE emp_rabat%found LOOP
        IF salaire >3000 THEN
                INSERT INTO resultat VALUES ( nom, salaire) ;
        END IF ;
        FETCH emp_rabat INTO nom, salaire ;
    END LOOP;
   CLOSE emp_rabat ;
END ;
```

Curseur: attributs

- Les attributs d'un curseur sont des indicateurs sur l'état d'un curseur.
- Quatre attributs permettent d'évaluer l'état du curseur.
  - **%NotFound** (vrai si le dernier FETCH n'a ramené aucun tuple).
  - **%Found** (inverse de %NotFound)
  - **%Rowcount** (compte le nombre de FETCH exécutés sur un curseur)
  - **%Isopen** (vrai si le curseur est ouvert)



Remarque :

Avec un curseur implicite crée par le SGBD Oracle, les mêmes attributs aux colonnes sont disponibles à condition de les préfixer par **SQL**. Ces attributs réfèrent au **dernier** curseur implicite utilisé par l'application

Exemple: Begin

```
        Update employe Set sal=sal+500 Where sal < 6000;
  dbms_output.put_line ("Nbre employés modifiés: ||SQL%ROWCOUNT) ;
            -- nombre de lignes affectées par une opération sur curseur implicite.
    End;
```


Utilisation simplifiée des curseurs

- L'utilisation **FOR LOOP** remplace **OPEN**, **FETCH** et **CLOSE**.
- Lorsque le curseur est invoqué, un **enregistrement** contenant le tuple courant est automatiquement créé avec les mêmes champs de données que ceux définis dans l'instruction **select**.

Exemple: CREATE TABLE resultat (nom varchar2(10), sal number (7,2))

```
DECLARE
  CURSOR employe_rabat IS
  SELECT nomempl, sal FROM employe WHERE upper(ville)= ' RABAT' ;
BEGIN
  FOR e IN employe_rabat LOOP
    IF e.sal >3000 THEN
        INSERT INTO resultat VALUES (e .nomempl, e .sal) ;
    END IF ;
  END LOOP;
END ;
```

L'attribut de curseur %ROWTYPE

- L'attribut **%ROWTYPE** permet de déclarer une variable de même type que l'enregistrement (la ligne) de la table

Syntaxe: ***Nom_de_variable nom_table*%ROWTYPE ;**

Exemple:

```
DECLARE
    e   empl%ROWTYPE;
```

- Avec un curseur:

  CURSOR ***nom_curseur*** **IS** ***Requete_SELECT***;

  ***nom_variable nom_curseur*%ROWTYPE** ;

- Les éléments de la structure sont identifiés par: ***nom_variable.nomcolonne***
- La structure est renseignée par le **FETCH** :

  FETCH ***nom_curseur*** **INTO** ***nom_variable***

L'attribut de curseur %ROWTYPE

Exemple :

```
DECLARE
CURSOR emp_rabat IS  SELECT  nomemp, sal FROM employe WHERE ville= 'Rabat';
 ligne    emp_rabat%ROWTYPE  ;
BEGIN
    OPEN   emp_rabat ;
    FETCH   emp_rabat  INTO  ligne;
    WHILE  emp_rabat%found  LOOP
          IF   ligne.sal >3000  THEN
                INSERT  INTO  resultat  VALUES ( ligne.nomemp, ligne.sal) ;
          END IF ;
          FETCH emp_rabat  INTO   ligne ;
     END  LOOP;
    CLOSE   emp_rabat ;
END ;
```

Utilisation d'une variable de type Record

- Un record permet de définir des types composites

Syntaxe :

```
TYPE  nom_record  IS   RECORD
          ( nom_ch1   type ,
            nom_ch2   type,
           ……..);
```

- Déclaration d'une variable de ce type: **nom_variable nom_record ;**

```
DECLARE
        TYPE  enreg_empl   IS  RECORD
            ( nom     employe.nomempl %TYPE,
              salaire   employe.sal %TYPE);
        e     enreg_empl ;
```


Curseur modifiable

- Un curseur qui comprend **plus** d'une table dans sa définition ne permet pas la modification des tables de BD.
- Seuls les curseurs définis sur **une seule table sans fonction d'agrégation** et de **regroupement** peuvent utilisés dans une MAJ: delete, update, insert avec le **CURRENT OF CURSOR**.

Curseur modifiable

```
Exemple: DECLARE
          CURSOR ce IS SELECT nomempl, sal, ville FROM employe FOR UPDATE ;
          nom      employe.nomempl %TYPE ;  salaire   employe.sal %TYPE ;
          ville    employe.ville %TYPE ;
        BEGIN
          OPEN ce ;
          FETCH ce INTO nom, salaire, ville ;
          WHILE ce%found LOOP
                IF ville IS NULL THEN
                    DELETE FROM employe WHERE CURRENT OF ce ;
                ELSE
                    UPDATE employe SET Ville =upper(Ville) WHERE CURRENT OF ce ;
                END IF ;
                FETCH ce INTO nom, salaire, ville ;
         END LOOP;
        CLOSE ce;
        END ;
```


Exercice (1/2)

- Ecrire un bloc PL/SQL permettant de:
  - Créer deux tables **Empl1** et **Empl2** qui contiennent les colonnes **empno**, **ename**, **sal** et **deptno** de la table **Emp.**
  - Utiliser un curseur pour sélectionner les colonnes **empno**, **ename**, **sal** et **deptno** de tous les employés de la table **Emp**.
  - Parcourir ce curseur afin d'insérer dans **Empl1** les employés gagnant plus que 1500 $ et dans **Empl2** les autres employés.
  - Afficher le nombre de tous les employés dans le curseur.

Exercice (2/2)

- Vider la table Empl1.
- Ecrire un bloc PL/SQL permettant de:
  - Lire deux réel A et B avec A ≤ B
  - Utiliser un curseur pour sélectionner les colonnes **empno**, **ename**, **sal** et **deptno** des employés de la table **Emp** qui ont un salaire entre A et B.
  - Parcourir les lignes de ce curseur afin de les insérer dans **Empl1.**
  - Afficher le nombre de tous les employés dans le curseur.