PL SQL

Séance 4

Pr. M’barek ELHALOUI

Révision/ Questions

- Structures de contrôle conditionnelles :
  - If….then….elseif….else…endif ;
- Structures de contrôle répétitives :
  - While …Loop…end Loop ;
  - Loop….end Loop ;
  - For…In…Loop…End Loop ;
- Structure de contrôle de choix :
  - Case…when…then…else…end case ;

- Questions ?

Exercices (Correction)

- Ecrie un programme PL SQL qui permet de réaliser une rotation des éléments d'un un tableau vers la droite.
- Ecrie un programme PL SQL qui permet le classement d'un tableau de 10 chiffres entiers du plus petit au plus grand.

Corrigé de l'exercice 1

```
-- Exercice Permutation circulaire à droite d'une table
DECLARE
TYPE TableType IS TABLE OF NUMBER INDEX BY BINARY_INTEGER ;
tab TableType;
i  NUMBER ; j NUMBER ;
BEGIN
    for i in 1..10 loop
    tab(i) := i ;
    end loop;

    j:= tab(10);

    for i in reverse 2..10 loop
    tab(i) := tab(i-1) ;
    end loop;

    tab (1):= j;

    for i in 1..10 loop
    dbms_output.put_line(tab(i));
    end loop;

END ;
```

Corrigé de l'exercice 2

```
 --Exercice Trie d'un tableau d'entiers
DECLARE
TYPE TableType IS TABLE OF NUMBER (2) ;
tab  TableType := TableType(6,8,1,9,15,21,3,35,13,11);
i NUMBER ;
j NUMBER :=0 ;
temp NUMBER ;
BEGIN
    i:=1 ;
    while i <= 10 loop
        j:=i+1;
            while j <= 10 loop
                if tab(i) > tab(j)
                then
                temp:= tab(i) ;
                tab (i) := tab (j);
                tab (j) := temp;
                End if;
            j:=j+1;
            end loop;
    i:=i+1;
    end loop;

    for i in 1..10 loop
    dbms_output.put_line(tab(i));
    end loop;
END ;
```

Les Curseurs

- Un **curseur** est une **zone mémoire de taille fixe** qui sert à **exécuter les instructions SQL** par le moteur SQL et à **stocker** les informations en cours de traitement ;
- Sert comme pointeur de résultats d'une requête ;
- Permet de sélectionner plusieurs lignes et les parcourir tuple par tuple ;
- Deux types de curseurs :
  - Implicite
  - Explicites

Les Curseurs Implicites

- Un curseur implicite est géré automatiquement par le noyau oracle pour les requêtes SQL (Select, Update, Insert, Delete, ...)
- permet d'obtenir des informations sur la requête réalisée, grâce aux attributs SQL%attribut. Il applique l'attribut sur la dernière instruction SQL exécutée.
- Les curseurs implicites sont tous nommés SQL :
    - SQL%rowcount : nombre de lignes affectées par la dernière instruction.
    - SQL%found : TRUE si la dernière instruction affecte au moins 1 ligne.
    - SQL%notfound : TRUE si la dernière instruction n'affecte aucune ligne.
- **Exemple :**

```
Delete from Employe where ...
if SQL%rowcount > 10 then
-- on a supprimé plus de 10 lignes
...
end if ;
```

Les Curseurs Explicites

- Un curseur explicite est gérés par l'utilisateur pour traiter un ordre **Select** qui renvoient plusieurs lignes;
- Déclaré dans la Section **DECLARE** par la syntaxe **CURSOR** *nom_curseur* **IS** *requête* **;**

- Un curseur peut être paramétré : **CURSOR** nom_curseur (*A1 type1, A2 type2, ...*) **IS** requête **;**
- Les paramètres (*A1, A2,..)* d'un curseur paramétré sont passés lors de l'ouverture du curseur (***OPEN*****)** ;
- L'exécution de la requête se fait lors de l'ouverture du curseur ;
- Souvent utilisé dans boucle pour le parcours et traitement ligne par ligne.

Les Curseurs Explicites

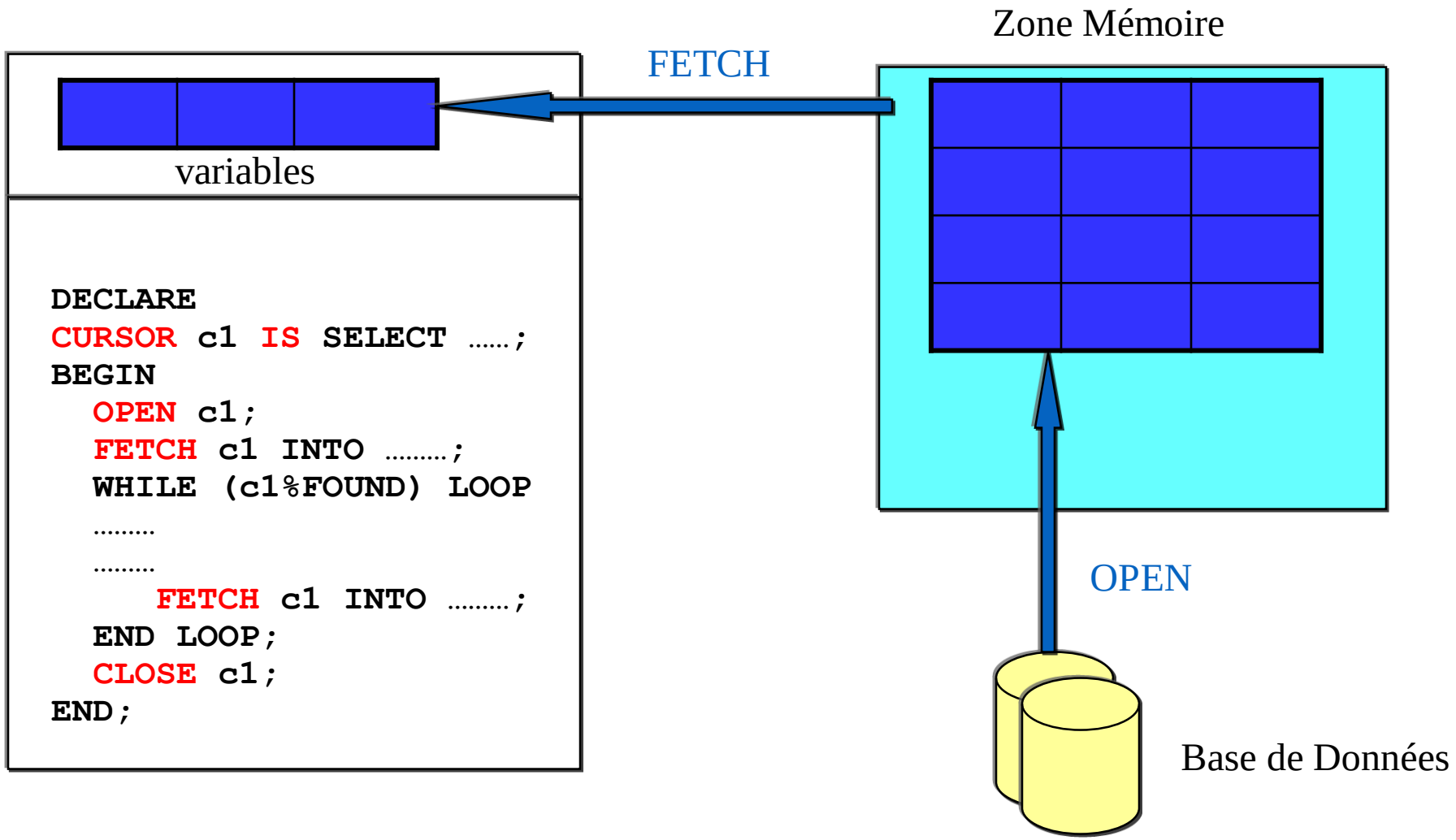
Quatre étrapes :

- Déclaration du curseur : **DECLARE**
  - Requête SQL (Ordre Select *)* sans exécution
- Ouverture du curseur : **OPEN**
  - Se fait section ***Begin*** du Bloc.
  - Allocation d'un espace mémoire par la requête exécutée
  - Eventuel verrouillage préventif (*curseur pour update..*)
- Traitement des lignes : **FETCH**
  - Chaque FETCH ramène une seule ligne à la fois
  - Une boucle pour traitement de plusieurs lignes
  - Les attributs *(%ISOPEN,…)* indiquent l'état du curseur.
- Fermeture du curseur : **CLOSE**
  - Libération de l'espace mémoire

Les Curseurs : Attributs

Attribut	Description
%ISOPEN	Retourne ***True*** si le curseur est ouvert
%FOUND	Retourne ***True*** si l'exécution est correcte de l'ordre SQL
%NOTFOUND	Retourne ***True*** si l'exécution est incorrecte de l'ordre SQL
%ROWCOUNT	Retourne le ***Nombre de lignes traitées*** par l'ordre SQL, évolue à chaque ligne traitée par un FETCH (zéro au départ)

Les curseurs



Les Curseurs : Exemple

```
DECLARE
CURSOR c_Emp IS
        SELECT * FROM Employe WHERE dept = '15' and Salary < 3000 ;
Nombre Integer :=0 ;
Somme Employe.Salary%Type :=0 ;
Rec_Emp Employe%ROWTYPE ; -- Enregistrement pour charger les lignes du curseur

BEGIN
    open  c_Emp ;
    loop
        fetch  c_Emp  into Rec_Emp ;
        exit when  c_Emp%notfound ;
        if  Rec_Emp.Salary is not null then
        Nombre := Nombre + 1;
        Somme := Somme + Rec_Emp.Salary ;
        end if;
    end loop;
    close c_Emp; -- A ne pas oublier
    DBMS_OUTPUT.put_line('Nombre de salariés ayant moins de 3000 DH :'|| Nombre);
    DBMS_OUTPUT.put_line('Total des salaires moins de 3000 DH :'|| Somme);
END;
```

Les Curseurs : Exemple avec while

```
DECLARE
CURSOR c_Emp IS
        SELECT * FROM Employe WHERE Salary < 3000 ;
code  Employe.Id_Employe%Type  ;
nom Employe.First_name%Type  ;
salaire Employe.Salary%Type  ;

BEGIN
    open  c_Emp ;
        FETCH c_Emp INTO code, nom, salaire;
        WHILE c_Emp%FOUND
        LOOP
        dbms_output.putline('Matricule : ' || Code ||', Nom : ' || nom ||', Salaire : '|| salaire);
        FETCH c_Emp INTO code, nom, salaire;
        end loop;
    close c_Emp; -- A ne pas oublier
END;
```

Les Curseurs : Pour Update

- Pour mise à jour (*Update*) …ou suppression (*Delete*)
- Affectent seulement le tuple courant de *FETCH*
- Variable Rec_Emp déclarée implicitement de type CURSOR
- Exemple : Augmentation de 3% des salaires moins de 3000 DH du département num 15

```
DECLARE
CURSOR c_Emp IS
        SELECT * FROM Employe WHERE dept = '15' and Salary < 3000 FOR UPDATE OF Salary;
BEGIN
    --Pas d'ouverture de curseur
    --Pas de Fetch

        For Rec_Emp In c_Emp Loop
        update Employe ;
        SET salary = Rec_Emp.salary * 1.03 WHERE current of c_Emp ;
        end loop;
        Commit;
    --Pas de cloture du curseur
END;
```

Exercices

- Créer un curseur paramétré qui permet de récupérer les employés d'un département donné passé en paramètre.
- Afficher le Matricule; Nom, Prénom, Salaire des 5 employés qui ont les salaires les plus élevés.

Gestion des Exceptions

- Toute erreur (SQL ou applicative) entraîne automatiquement un débranchement vers le paragraphe EXCEPTION ;
- Une exception peut être prédéfinie par le système (*exemple : division par 0*) ou définie par l'utilisateur ;
- Le traitement d'une exception se fait par la règle WHEN
- Déclaration d'une exception :

```
nomerreur EXCEPTION;
PRAGMA EXCEPTION_INIT(nomerreur,n°erreur);
```

```
DECLARE
        --déclarations
BEGIN
        --exécutions
EXCEPTION
    WHEN exception1 THEN
    .....
    WHEN exception2 THEN
    .....
    WHEN OTHERS THEN
    .....
END ;
/
```

Gestion des Exceptions

- Exemple d'exceptions internes

Code d'erreur SQLCODE	Erreur
+1403	NO_DATA_FOUND
-1	DUP_VAL_ON_INDEX
-6502	VALUE_ERROR
-1001	INVALID CURSOR
-1722	INVALID NUMBER
-6501	PROGRAM ERROR
-1017	LOGIN DENIED
-1422	TOO_MANY_ROWS
-1476	ZERO_DIVIDE

Gestion des Exceptions

- Exemple :

```
DECLARE
   salaire numeric(8,2);
   salaire_trop_bas EXCEPTION;
BEGIN
   select sal into salaire from emp
     where matr = 50;
   if salaire < 300 then
     raise salaire_trop_bas;
   end if;
   -- suite du bloc
EXCEPTION
   WHEN salaire_trop_bas THEN . . .;
   WHEN OTHERS THEN
     dbms_output.put_line(SQLERRM);
END;
```