RESUME LANGAGE PL/SQL

Università di Corsica Pasquale Paoli

Structure d'un bloc PL-SQL anonyme

```
[DECLARE
      --déclarations de variables, constantes
      -- exceptions et curseurs]
BEGIN [nombloc]
      --instructions
[EXCEPTIONS
      -- Traitement des exceptions]
END;  --ou END nombloc
/
```

2

Affectation et Structures de contrôle

- Affectation
  - variable:= valeur
- IF *condition* THEN ... ELSIF *condition* THEN ... ELSE ... END IF ;
- WHILE *condition* LOOP ... END LOOP ;
- FOR compteur IN min .. max
  - LOOP ... END LOOP ;
- LOOP ... EXIT WHEN *condition* END LOOP ;

3

Déclaration de variables

- DECLARE
  - variable1 type1;
  - variable2 type2 :=valeur;

•types SQL
•types structurés (tableaux record)
•référence à un type du dictionnaire de données

```
DECLARE
      NOM         CHAR(15);
      SALAIRE     NUMBER(7,2);
      EMBAUCHE    DATE ;
      RÉPONSE     BOOLEAN;
```

Remarque :PL/SQL n'est pas sensible à la casse

4

Types référençant le dictionnaire de données

- Type d'une colonne existante
  - nom_var **TABLE.COLONNE%TYPE**
- type record correspondant à une ligne d'une table existante
  - nom_var2 **TABLE%ROWTYPE**
- Type d'une variable existante
  - nom_var 3 **nom_var1%TYPE**

type record (colonnes de la table)

```
NOM      EMP.ENAME%TYPE;
ENREG    EMP%ROWTYPE;
COMMI    NUMBER(7,2);
SALAIRE  COMMI%TYPE;
```

5

Record (enregistrements)

```
DECLARE
   TYPE  TYPETUDIANT
      IS  RECORD (
         nom         ETUDIANT.NOM%TYPE,
         moyenne     ETUDIANT.MOYENNE%TYPE ) ;
   LIGNETU  TYPETUDIANT;
BEGIN
   LIGNETU.nom:='Toto';
   LIGNETU.moyenne:=18;
END
```

déclaration d'un type RECORD

déclaration d'une variable de type record

6

Tableaux

```
DECLARE
  TYPE TYPETABNOTE
     IS TABLE OF NUMBER(4,2)
     INDEX BY BINARY_INTEGER;
  TABNOTES TYPETABNOTE;

  i NUMBER;

BEGIN
   i:=1;

  TABNOTES(i) := 12.5;
END
```

déclaration d'un type tableau

type de numérotation

déclaration d'une variable de type tableau

7

Afficher des résultats avec le package DBMS_OUT

- Messages enregistrés dans un buffer côté serveur et affichés sur le poste client
- DBMS_OUTPUT.PUT_LINE(chaîne caractères);
  - Écriture avec saut de ligne
- DBMS_OUTPUT.PUT(chaîne caractères);
  - Écriture sans saut de ligne
- DBMS_OUTPUT.NEW_LINE;
  - Écriture d'un saut de ligne

Pour activer DBMS_OUT: SET SERVEROUT ON

8

SELECT INTO

Pour récupérer les données d'une sélection mono-ligne

- Clause permettant de récupérer le résultat d'une requête dans une (ou des) variables

```
SELECT col1, col2, . .  INTO var1, var2, . . .
FROM  table
WHERE .....
```

ATTENTION!
Le SFW ne doit retourner qu'une seule ligne

9

Ouverture d'un curseur

- BEGIN
  OPEN nom_curseur;
- L'ouverture déclenche:
  - Allocation mémoire pour les lignes du curseur
  - Analyse syntaxique et sémantique du SFW
  - Positionnement de verrous éventuels

10

Parcours d'un curseur

- Instruction **FETCH**
  - Parcourt une à une les lignes du curseur
  - Située en général dans une boucle
- la valeur de chaque colonne doit être stockée dans une **variable réceptrice**
- FETCH permet d'avancer le pointeur du curseur sur la ligne suivante et de récupérer cette ligne

  FETCH nom_curseur **INTO** liste_variables;

A tout moment le curseur pointe sur une ligne spécifique (la ligne courante)

11

Attributs d'un curseur

Indicateurs permettant de connaitre l'état d'un curseur.

- nom_curseur%FOUND
  - TRUE: le curseur pointe sur une ligne
  - FALSE: le parcours est terminé
- nom_curseur%NOTFOUND
  - TRUE: le parcours est terminé
- nom_curseur%ISOPEN
  - TRUE: le curseur est ouvert
- nom_curseur%ROWCOUNT
  - Nombre total de lignes du curseur

le dernier FETCH n'a pas ramené de ligne

```
BOUCLES DE PARCOURS
WHILE  SKIPAJA%FOUND
LOOP....END LOOP;
        OU
LOOP ...
 EXIT WHEN SKIPAJA%NOTFOUND
END LOOP;
```

12

Boucle simplifiée de parcours d'un curseur

```
FOR   varLigne IN  nom_curseur
LOOP
...
END LOOP ;
```

Variable RECORD de type ligne du curseur

- Pas de déclaration de variables réceptrices
- Pas d'ouverture
- Pas de Fetch
- Pas de close

13

Structure d'une procédure stockée

```
CREATE PROCEDURE nomp (p1 type1,
   p2 type 2,…) IS
        --déclarations de variables, constantes
        -- exceptions et curseurs
BEGIN [nomp]
        --instructions
[EXCEPTIONS
        -- Traitement des exceptions]
END;  --ou END nomp
```

14

Structure d'une fonction stockée

```
CREATE FUNCTION nomf (p1 type1,
   p2 type 2,…)
   RETURN type IS
        --déclarations de variables, constantes
        -- exceptions et curseurs
BEGIN [nomf]
        --instructions
        RETURN …;
END;  --ou END nomf
```

15

Définition d'un trigger

BEFORE, AFTER

INSERT, UPDATE, DELETE

```
CREATE TRIGGER nom moment evenement
ON nom_table FOR EACH ROW
        instructions;
```

- Oracle: Dans sqldeveloper, onglet spécifique associé à une table
- MySQL: Dans php myAdmin, fenêtre spécifique

16

Manipulation de la ligne courante

- Le trigger peut manipuler les valeurs des colonnes de la ligne sur laquelle il a été déclenché.
- :OLD.*col* désigne la valeur de la colonne col avant la mise à jour (OLD.col en MySql)
- :NEW.*col* désigne la nouvelle valeur de la colonne col après la mise à jour (NEW.col en MySql)

Pour pouvoir modifier la valeur d'une colonne, le trigger doit être exécuté avant la mise à jour (BEFORE)

17