

PL /SQL RESUME

1. PL/SQL

Les éléments d'un bloc PL/SQL :

```
DECLARE
<section des déclarations>
BEGIN
<section des exécutions>
EXCEPTION
<section des exceptions>
END ;
```

2. Variables (et constantes)

· **Types de données :**

o **scalaires :** NUMBER, BINARY_INTEGER, INTEGER, BOOLEAN, DATE, CHAR, VARCHAR

o **composites :** RECORD, TABLE

· **Déclaration**

o **bornée :** pour déclarer une variable scalaire *Nom_variable type_donnees* [NOT NULL] [:= | DEFAULT valeur_initiale]

o **basée :** utilisation des attributs d'autres objets de la base de données

Nom_variable basée Nom_variable_origine%TYPE

Nom_variable basée Nom_table.Nom_colonne%TYPE

NB : %ROWTYPE donne à la variable la même structure (nom + type) que celle appliquée dans une ligne de la table.

· **Affectation de valeur :** (! même type de données des 2 cotés !) *plsql_variable := plsql_expression*

· **Sélectionner des valeurs de base de données et les affecter à des variables :**

SELECT les_attributs INTO les_variables FROM...

· **Demander à l'utilisateur d'entrer une chaîne de caractères (« variable de substitution) :**

&Nom_Attribut

· **Afficher:**

DBMS_OUTPUT.PUT_LINE (les_expressions);

DBMS_OUTPUT.PUT (les_expressions);

DBMS_OUTPUT.NEW_LINE;

NB : faire SET SERVEROUTPUT ON; pour permettre l'affichage.

3. Types de données

A. Variable de type table :

· **Définir un type de TABLE :**

TYPE nom_type IS TABLE OF {type_donnée | variable%TYPE | table.colonne%TYPE} [NOT NULL] INDEX BY BINARY_INTEGER;

· **Déclarer les tables PL/SQL de ce type :**

Nom_table_PLSQL Nom_type;

· **Référencer les lignes d'une table PL/SQL :**

Nom_table_PLSQL (valeur_clé_primaire);

B. Variable de type record:

Les records PL/SQL sont composés de champs chacun à un nom unique et ils peuvent appartenir à différents types de données. Le type record permet de recueillir des informations sur les attributs d'un objet.

· **Définir un type de RECORD :**

TYPE nom_type_record IS RECORD

(nom_champ_1 {type_donnée | variable%TYPE | table.colonne%TYPE} [NOT NULL]),

(nom_champ_1 {type_donnée | variable%TYPE | table.colonne%TYPE} [NOT NULL]),
...);

· **Déclarer les enregistrements de ce type :**

Nom_enregistrement Nom_type_record;

· **Référencer individuellement les champs d'un enregistrement :**

Nom_enregistrement .nom_champ;

4. Conditions et Répétitions

· IF-THEN-END IF :

```
IF <condition> THEN
<séquence d'instructions>
[ELSE IF <condition> THEN
<séquence d'instructions>]
[ELSE
<séquence d'instructions>]
END IF;
```

· CASE-END CASE :

```
CASE
    WHEN <condition1> THEN
        <séquence d'instructions1>
    WHEN <condition2> THEN
        <séquence d'instructions2>
    [ELSE <séquence d'instructions N>]
END CASE;
```

· LOOP :

```
LOOP
    <séquence d'instructions>
    EXIT[WHEN<condition>];
END LOOP;
<séquence d'instructions>
```

· FOR :

```
FOR <indice> IN[REVERSE] <enter_min>..<enter_max> LOOP
    <séquence d'instructions>
```

```
END LOOP;
```

· WHILE :

```
WHILE<condition> LOOP
    <séquence d'instructions>
END LOOP;
```

5. Gestion des transactions

· **COMMIT** : rend permanent tous les changements effectués durant la transaction courante.
COMMIT;

· **SAVEPOINT** : Place une étiquette dans le traitement d'une transaction.

```
SAVEPOINT <nom_point_sauvegarde>
```

· **ROLLBACK** : Annule toutes les actions appliquées depuis le début de la transaction ou le point de sauvegarde.

```
ROLLBACK[TO [SAVEPOINT] <nom_point_sauvegarde>]
```

6. Curseurs

Une instruction SELECT ... INTO renvoie exactement une ligne. Pour avoir plusieurs lignes, il faut utiliser les curseurs.

Un curseur permet de nommer le domaine privé et d'accéder aux informations stockées.

· **Curseurs implicites** : établis automatiquement → INSERT, UPDATE,DELETE, SELECT...INTO

· **Curseurs explicites** : doivent être déclarés

o **Déclaration du curseur** :

```
CURSOR <nom_curseur>
```

```
IS <instruction SELECT>
```

o **Ouverture du curseur** :

```
OPEN <nom_curseur>
```

o **Recherche des données provenant du curseur** :

→ l'instruction FETCH place le contenu de la ligne courante dans les variables locales.

```
FETCH <nom_curseur> INTO <les_variables>
```

o **Fermeture du curseur** :

```
CLOSE <nom_curseur>
```

o **Boucle FOR du curseur** :

```
FOR <nom_enregistrement>
```

```
IN <nom_curseur> LOOP
```

```
-- Traiter une ligne
```

```
END LOOP;
```

· **Attributs de curseur** :

o %NOTFOUND → détermine quand il ne reste plus de lignes à chercher

o %FOUND → contraire logique du précédent

o %ROWCOUNT → agir quand un nombre spécifié de lignes a été recherché

o %ISOPEN → retourne l'état du curseur

o Utilisation avec :

curseur implicite : SQL%ATTRIBUT

curseur explicite : <nom_curseur>% ATTRIBUT