N. ABDAT

Le langage PL/SQL

Nov 2021

Langage PL/SQL

(**P**rocedural **L**anguage / **S**tructured **Q**uery **L**anguage)

PL/SQL est un langage fournissant une interface procédurale au SGBD Oracle.

Caractéristiques de PL/SQL

- Extension de SQL : des requêtes SQL cohabitent avec les structures de contrôle habituelles de la programmation structurée (blocs, alternatives, boucles)
- Un programme est constitué de procédures et de fonctions
- Des variables permettent l'échange d'information entre les requêtes SQL et le reste du programme

Utilisation de PL/SQL

- PL/SQL peut être utilisé pour l'écriture des procédures et des triggers.
- Il sert également pour écrire des fonctions utilisateurs qui peuvent être utilisées dans les requêtes SQL (en plus des fonctions prédéfinies).
- Il est aussi utilisé dans des outils Oracle, *Forms* et *Report*….

BLOC PL/SQL :

Chaque bloc PL/SQL peut être constitué de 3 sections :

- une section facultative de déclaration et initialisation de types, variables et constantes
- une section obligatoire contenant les instructions d'exécution
- une section facultative de gestion des erreurs

BLOC PL/SQL :

[**DECLARE** … déclarations et initialisation]

BEGIN
… instructions exécutables

[**EXCEPTION** … interception des erreurs]

END;

Exemples de déclarations et initialisations :

DECLARE
mot **CHAR(5);** note **NUMBER (4,2) := 10;**
x NUMBER(4) **:=** 0;
v1 **table%ROWTYPE;** *(type du tuple d'une table*)
v2 **table.attribut%TYPE;** *(type d'un attribut d'une table*)

Le *IF* dans PL/SQL :

si-alors

```
IF condition THEN instructions; END IF;
```

si-alors-sinon

```
IF condition THEN instructions; ELSE instructions; END
IF;
```

Imbrications de conditions

```
IF condition1 THEN instructions;
                ELSIF condition2 THEN instructions;
                ELSIF ….. ;
                 ELSE instructions;
END IF;
```


Le *CASE* dans PL/SQL :

```
CASE variable
        WHEN expr1 THEN instructions1;
         WHEN expr2 THEN instructions2; …
        WHEN exprN THEN instructionsN;
         [ELSE instructionsN+1;]
END CASE;
```

```
CASE variable
        WHEN condition1 THEN instructions1;
        WHEN condition2 THEN instructions2; …
        WHEN conditionN THEN instructionsN;
        [ELSE instructionsN+1;]
END CASE;
```

Les boucles dans PL/SQL :

Boucle Pour :	**Boucle Tant que :**	**Boucle Répéter :**
FOR I IN 1 . . 10 LOOP instructions; END LOOP;	WHILE condition LOOP instructions; END LOOP;	LOOP instructions; EXIT WHEN condition; END LOOP ;

PRODEDURE ET FONCTIONS

Procédure :

CREATE [OR REPLACE] PROCEDURE nomProcédure
[(paramètre [**IN** | **OUT** | **IN OUT**] typeSQL [:= | DEFAULT] ....)]
IS Bloc PL/SQL ;

Fonction :

CREATE [**OR REPLACE**] **FUNCTION** nomFonction
[(paramètre [**IN** | **OUT** | **IN OUT**] typeSQL .....)] **RETURN** typeSQL
IS Bloc PL/SQL contenant Return;

EXTRAIRE UN TUPLE :

SELECT liste_attributs **INTO** liste_variables **FROM** nomTable **……** ;

Exemples:

```
SELECT * INTO v1
FROM Bateau WHERE nbat=103;
```

v1= (103, 'EL BAHDJA', 'BNA')

```
SELECT nombat INTO v2
FROM Bateau WHERE nbat=104;
```

v2= 'LA COLOMBE '

Remarque :
Une requête SELECT … INTO... doit renvoyer un seul enregistrement.
Une requête SELECT … INTO... qui renvoie plusieurs enregistrements, ou qui n'en renvoie aucun, génère une erreur PL/SQL .

EXTRAIRE PLUSIEURS TUPLES : LES CURSEURS (CURSOR) :

Pour traiter des requêtes renvoyant plusieurs enregistrements, on utilise des curseurs PL/SQL.

Définition

Le curseur est une zone de mémoire de taille fixe, utilisé par le moteur de la base Oracle pour analyser et interpréter les ordres SQL.

Il existe deux types de curseurs :

Le curseur implicite : Curseur SQL généré et géré par ORACLE pour chaque ordre SQL.

Le curseur explicite : Curseur SQL généré et géré par l'utilisateur pour traiter un ordre SELECT qui ramène plusieurs lignes.

Étape d'utilisation d'un curseur explicite :

Le curseur explicite permet de traiter individuellement chaque ligne renvoyée par un SELECT. Il est décrit dans la partie déclarative et est ouvert dans le code du programme, il s'évalue alors et va se charger en extrayant les données de la base.

Déclaration

Tout curseur explicite utilisé dans un bloc PL/SQL doit être déclaré dans la section DECLARE du bloc en donnant son nom et l'ordre SELECT associé.

Syntaxe **CURSOR** nom_curseur **IS** ordre_select **;**

Exemple : DECLARE CURSOR cr **IS SELECT** nbat, nombat **FROM** bateau**;**

Ouverture

Après avoir déclaré le curseur, on "ouvre" celui-ci pour faire exécuter l'ordre SELECT. L'ouverture déclenche :

- l'allocation mémoire du curseur,
- l'analyse syntaxique et sémantique de l'ordre SELECT

L'ouverture du curseur se fait dans la section BEGIN du bloc.

Syntaxe **OPEN** nom_curseur **;** **Exemple : OPEN** cr **;**

Traitement des lignes

Après l'exécution du SELECT, on peut parcourir tout le curseur en récupérant les lignes une par une dans une variable locale.

FETCH permet le positionnement sur la ligne suivante et chargement de l'enregistrement courant dans 1 ou plusieurs variables.

Syntaxe **FETCH** nomCurseur **INTO** listeVar | nomRECORD

Exemple : **FETCH** cr **INTO** b;
La variable b est déclarée comme suit : b cr**%ROWTYPE;**

Fermeture
Le curseur est fermé à la fin du traitement .
La fermeture du curseur se fait dans la section BEGIN du bloc.

Syntaxe **CLOSE** nom_curseur **;**

-Un curseur, durant son existence (de l'ouverture à la fermeture), contient en permanence l'adresse de la ligne courante.
-Un programme PL/SQL peut travailler avec plusieurs curseurs en même temps.

Boucle FOR (gestion semi-automatique)
-FOR de curseur évite les directives OPEN, FETCH et CLOSE.
-La boucle s'arrête d'elle-même à la fin de l'extraction de la dernière ligne du curseur.
-La variable de réception du curseur est automatiquement déclarée .
-L'accès aux valeurs des colonnes se fait également par la notation pointée.

Exemple

```
DECLARE   CURSOR  cr IS SELECT nbat, nombat FROM bateau;
BEGIN
  for b in cr loop
        dbms_output.put_line('Le bateau N° ' || b.nbat||' a pour nom '|| b.nombat);
  end loop;
END;
```