

# PARTIE 1 - LE LANGAGE PROCEDURAL D'ORACLE : LE LANGAGE PL/SQL

## I – INTRODUCTION

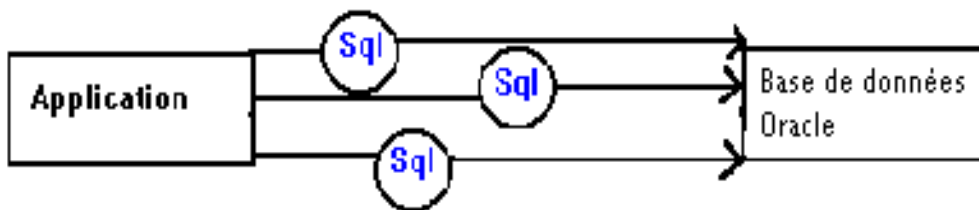
Le PL/SQL est le langage procédural d'ORACLE, c'est une extension du langage SQL qui est un langage ensembliste.

PL/SQL = Procédural Language / SQL

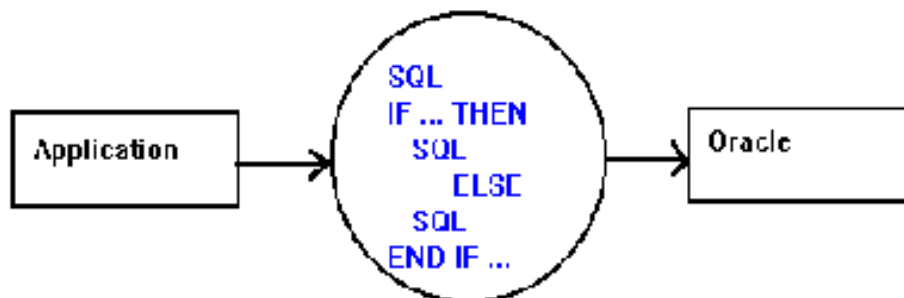
L'intérêt du PL/SQL est de pouvoir dans un même traitement allier la puissance des instructions SQL et la souplesse d'un langage procédural.

Le fonctionnement de PL/SQL est basé sur l'interprétation d'un "bloc" de commandes. Ce mode de fonctionnement permet d'obtenir des gains de transmission et des gains de performances :

Dans l'environnement SQL, les ordres du langage sont transmis et exécutés les uns à la suite des autres



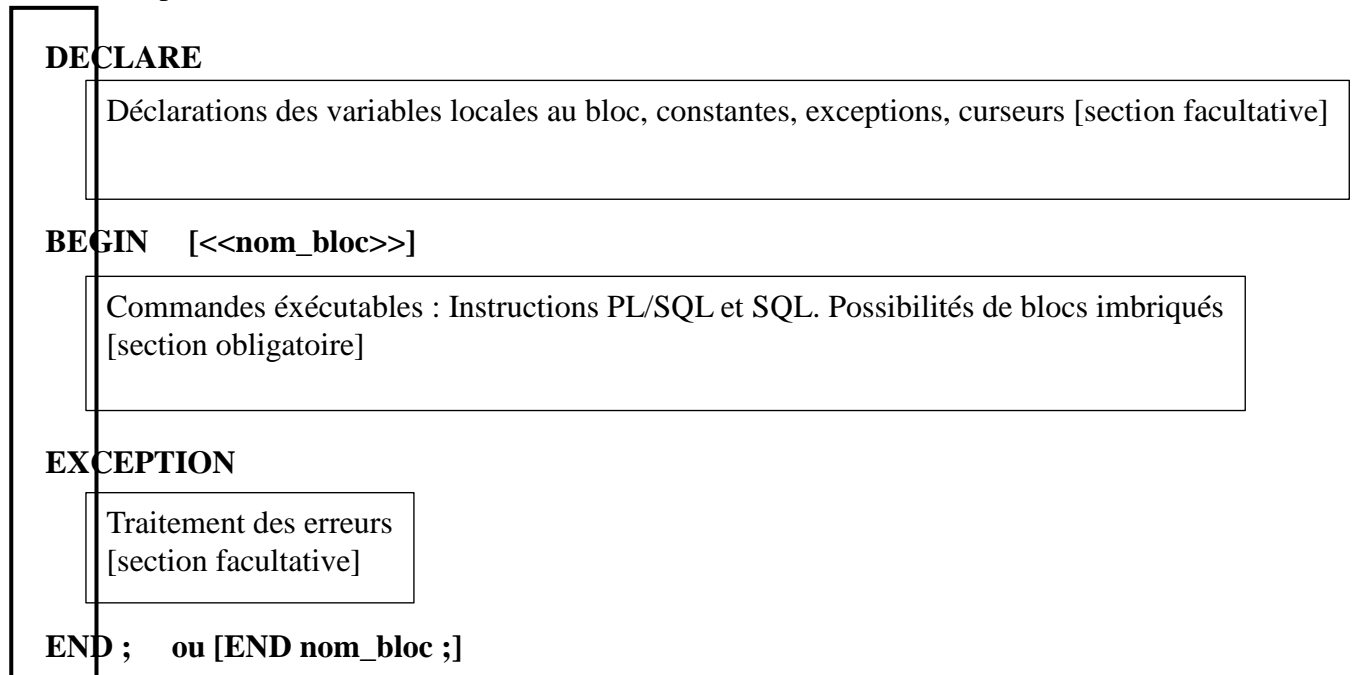
Dans l'environnement PL/SQL ; les ordres SQL et PL/SQL sont regroupés en BLOCs ; un bloc ne demande qu'un seul transfert et une seule exécution de l'ensemble des commandes contenues dans le bloc.



## II – LE BLOC PL/SQL

PL/Sql n'interprète pas une commande, mais un ensemble de commandes contenu dans un programme ou "bloc" PL/Sql.

Un bloc est composé de trois sections :



Chaque instruction de n'importe quelle section doit se terminer par un ';'.

Possibilité de placer des commentaires : -- commentaire sur une ligne

ou /\* commentaire sur

plusieurs lignes \*/

### **EXEMPLE sous SQL\*PLUS de Personnel Oracle 7**

#### **① Ecriture du programme PL/SQL sous le bloc notes nommé « PLSQL\_EX1.sql »**

```
DECLARE          -- Début du programme
    sal_emp number(7,2);      -- variable locale au bloc

BEGIN
    /* Sélectionner le salaire de l'employé saisi au préalable dans SQL*PLUS (num_emp) ,
       l'augmenter de 10% si ce salaire est inférieur à 1000 */
    SELECT sal into sal_emp FROM emp
       where empno = '&num_emp';

    If sal_emp < 1000 Then
        UPDATE emp    SET sal = sal * 1.1
           WHERE empno = '&num_emp';
    end if;

    /* commit;*/

END;
```

/ -- Ne pas oublier ce slash qui termine le fichier  
**Test de notre premier programme :**

### ① Sous SQL\*PLUS, visualisation de la table emp

SQL> select \* from emp;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	800		20
7499	ALLEN	SALESMAN	7698	20/02/81	1600	300	30

.....

### ② Sous SQLPLUS définir la variable num\_emp

SQL> define num\_emp=7369

ou

SQL> PROMPT " Numéro du salarié désiré ?" -- ou écrire ces 2 lignes directement

SQL> ACCEPT num\_emp -- dans le programme PL/SQL avant DECLARE

### ③ Appel du programme Plsql\_ex1.sql écrit précédemment et sauvegardé sous c:\orawin95\gautier:

SQL> start ..\gautier\PLSQL\_EX1 (start ou @)

ancien 8: where empno = '&num\_emp';

nouveau 8: where empno = '7369';

ancien 12: WHERE empno = '&num\_emp';

nouveau 12: WHERE empno = '7369';

Procédure PL/SQL terminée avec succès.

### ③ Vérification de la modification sur la table emp

SQL> select \* from emp;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	880		20
7499	ALLEN	SALESMAN	7698	20/02/81	1600	300	30

.....

### III – DECLARATION DES VARIABLES

La partie déclarative dans un bloc PL/Sql, peut comporter trois types de déclarations.

Elle est délimitée par les mots-clés **DECLARE**, qui spécifie le début et **BEGIN**, qui signifie la fin de la déclaration et le début de la partie des commandes.

Les types de déclarations possibles dans cette partie sont les suivants :

- déclaration des variables et des constantes,
- déclaration de curseurs,
- déclaration des exceptions.

Les types de variables utilisées en PL/Sql sont les suivantes :

- variables locales
  - ⇒ de type Oracle : reconnu par Oracle
  - ⇒ faisant référence au dictionnaire de données
- variables de l'environnement extérieur à PL/SQL
  - ⇒ champs d'écran de Forms
  - ⇒ variables hôtes définies en langage hôte dans Pro\*
  - ⇒ variables définies dans Sql\*Plus (préfixées de &)

La déclaration d'une variable consiste à allouer un espace pour stocker et modifier une valeur. Elle est typée et peut recevoir une valeur par défaut et/ou un statut NOT NULL.

#### 1. Variables ou constantes locales de type Oracle et PL/Sql

**Nom-de-variable [CONSTANT] type [[NOT NULL] := expression] ;**

**Type** Char(n), Number(n[,m]), date, boolean

**Expression** peut être une constante ou un calcul faisant éventuellement référence à une variable précédemment déclarée

*DECLARE*

```
Nom_du_client Char(30) ;  
X              number / + 1 ;      -- initialisation  
PI             constant    number(7,5) := 3.14159 ;  
Rayon          number := 1 ;  
Surface        number(15,5) := pi * Rayon **2 ;  
Reponse        boolean ;
```

## 2. Variables faisant référence au même type qu'une colonne d'une table ou même type qu'une autre variable

**Nom-de-variable nom\_table.nom-colonne %type;**

ou

**Nom-de-variable1 Nom-de-variable2%type ;**

*DECLARE*

*Emp\_Nom EMP.Empno%type ; -- Même type que la propriété Empno*

*X number(10,3) ;*

*Y X%type ; -- Du même type que x donc number(10,3)*

Ceci est intéressant pour des raisons de simplification d'écriture et d'évolution .

## IV– VALORISATION DES VARIABLES PL/SQL

Trois possibilités de valorisation de variables sont disponibles :

1. par l'opérateur d'affectation : ':=' ,
2. par la clause **Select ... Into ...** .
3. par le traitement d'un **curseur** dans la section Begin. (que nous aborderons par la suite)

### a) affectation de valeur

**Nom\_Variable := Expression ;**

**Expression peut-être :**

- une constante, une variable, un calcul

**Les opérateurs de calcul sont :**

- + ; - ; \* ; / ; \*\* ; ||

*BEGIN*

*X := 0 ;*

*Vnom := 'Monsieur' || Vnom ; -- concaténation*

*Y := (X+5) \* Y ;*

### b) La clause select ... into

La difficulté dans l'utilisation de la clause Select résulte du nombre de lignes ou d'occurrences retourné.

Si le Select retourne une et une seule ligne l'affectation s'effectue correctement.

Par contre,

Si le Select retourne 0 ligne : NO\_DATA\_FOUND ( test « nom\_variable IS NULL »)

Si le Select retourne plusieurs lignes : TOO\_MANY\_ROWS , une erreur PL/SQL est générée.

**SELECT {\*/Liste d'expression} INTO Liste de variables FROM ... ;**

```

DECLARE
  VRef CHAR(10) ;
  VPrix Articles.Prix%TYPE ;
  Clt Clients.%ROWTYPE

BEGIN
  SELECT RefArt, PrixArt INTO Vref, Vprix
    FROM Articles WHERE DesArt = 'Cadeau';
  SELECT * INTO Clt
    FROM Clients WHERE NoClt = 10 ;

END ;

```

## **V – STRUCTURES DE CONTRÔLES**

### **a) Structure alternative**

```

IF condition Then
    Instructions ;
[Else instructions ; ]

[ELSEIF condition Then
    instructions ;
    [Else commandes ;] ]
END IF;

```

Seules les clauses IF, THEN, END IF sont obligatoires.

La condition peut utiliser les variables définies ainsi que tous les opérateurs présents dans SQL  
 =, <, >, <=, >=, <>, IS NULL, IS NOT NULL, BETWEEN, LIKE, AND, OR, etc..

### **b) La boucle POUR**

```

FOR compteur IN exp_debut .. exp_fin
LOOP
    ...
    instructions ;
    ...
END LOOP ;

```

Règles :

- ◆ Déclaration implicite de la variable compteur
- ◆ exp\_debut, exp\_fin : sont des constantes, expressions ou variables
- ◆ compteur : est une variable de type entier, locale à la boucle. Elle s'incrémente de 1, après chaque traitement du contenu de la boucle, jusqu'à ce qu'il atteigne la valeur de droite

### **c) La boucle TANT QUE**

```

WHILE condition
LOOP    ...
    instructions ;

    ...
END LOOP;

```

La condition est une expression définie en combinant les opérateurs : <, >, =, !=, <=, >=; and, or, like, etc... Expression est une constante, une variable, le résultat d'une fonction.

## **VI – ECHANGES AVEC L'EXTERIEUR**

A priori il n'existe pas d'instruction d'affichage et de saisie dans le langage PL/SQL.

### **Sous SQL\*Plus :**

⇒ **on peut définir une variable ( réservation d'une zone mémoire), et l'afficher à la fin du programme PL/SQL**

```

SQL> variable x number
SQL> start ../gautier/plsql_ex2

```

Procédure PL/SQL terminée avec succès.

```

SQL> print x
      X
-----

```

5

Remarque : X est préfixée par : → variable hôte.

```

--- EXEMPLE 2 : PLSQL_EX2.sql
BEGIN

    SELECT COUNT(*) INTO :X FROM DEPT;
END;
/

```

⇒ **on peut saisir une valeur à rechercher dans SQLPLUS (ou sous le fichier avant le DECLARE)**

```

SQL >@ ../gautier/plsql_ex3
prompt "nom du département désiré" -- ou ces 2 lignes sous SQL*PLUS
accept dept_nom

```

```

-- Exemple PLSQL_EX3.sql
DECLARE
    res dept%rowtype; -- Même type qu'une ligne de la table
BEGIN
    SELECT * into res from dept
        where dname = '&dept_nom';
END;
/

```

Mais le résultat de la requête ne s'affiche pas

⇒ **on peut définir une constante dans SQL\*PLUS ou dans le fichier**

```
SQL>define dept_nom = 'SALES'
SQL>@ ../gautier/plsql_ex3           // sans les lignes prompt et accept
Mais rien ne s'affiche :
```

⇒ **La meilleure solution pour récupérer le résultat de l'exécution d'un programme PL/SQL consiste à créer une table résultat comportant les champs que l'on désire puis d'afficher à la fin du programme cette table.**

Sous NOTPAD

```
-- Exemple PLSQL_EX4.sql
Prompt "Quel est le département désiré"
accept numero
create table resultat(num number(2), nom char(14))
/

DECLARE
  numero dept.deptno%type;
  nomdept dept.dname%type;
BEGIN
  SELECT deptno, dname into numero, nomdept from dept
                                where deptno = '&numero';
  INSERT INTO resultat
                                values(numero, nomdept);
END;
/
select * from resultat
/
drop table resultat
/
```

Sous SQL\*PLUS

```
SQL> @ ../gautier/plsql_ex4
"Quel est le département désiré"
10
```

Table créée.

```
ancien 6:  where deptno = '&numero';
nouveau 6:  where deptno = '10';
```

Procédure PL/SQL terminée avec succès.

```
NUM NOM
-----
10 ACCOUNTING
Table supprimée.
```

**Remarque :** Dans un précompilateur Oracle, les variables sont désormais partagées avec le bloc PL/SQL. Ce bloc est inséré entre 2 délimiteurs EXEC SQL EXECUTE et END-EXEC.

## **VII – UTILISATION DU PACKAGE DBMS\_OUTPUT**



Sous Oracle 7 , le package DBMS\_OUTPUT permet d'afficher des messages à l'écran dans des programmes PL/SQL. Cela va faciliter le test et le « débogage » des programmes.

Pour cela il faut sous SQL\*PLUS autoriser l'utilisation de l'instruction d'affichage.

**SQL> SET ServerOUTPUT ON [SIZE 80000]**

(Optionnel : précise le nombre de caractères maximum à afficher)

Dans un programme PL/SQL on peut alors utiliser l'instruction :

**DBMS\_OUTPUT.PUT\_LINE('message .....') ;**

*-- Exemple PLSQL\_EX4b.sql*

*Prompt "Quel est le département désiré"*

*accept numero*

**DECLARE**

*numero dept.deptno%type;*

*nomdept dept.dname%type;*

**BEGIN**

*SELECT deptno, dname into numero, nomdept from dept  
where deptno = '&numero';*

***Dbms\_Output.put\_line('le département ' || TO\_CHAR(numero) || ' a pour nom ' || nomdept) ;***

**END;**

**/**

```
SQL> @ c:\asql\plsql_ex4b
"Quel est le département désiré"
20
ancien 6:  where deptno = '&numero';
nouveau 6:  where deptno = '20';
le département 20 a pour nom RESEARCH
```

Procédure PL/SQL terminée avec succès.

**SQL>**

Remarque : Pour ne plus autoriser l'affichage : set ServerOutput off

## VIII – EXERCICES D'APPLICATION

Nous utiliserons la base de données Employée

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	1064,8		20
7499	ALLEN	SALESMAN	7698	20/02/81	1600	300	30
7521	WARD	SALESMAN	7698	22/02/81	1250	500	30
7566	JONES	MANAGER	7839	02/04/81	2975		20
7654	MARTIN	SALESMAN	7698	28/09/81	1250	1400	30
7698	BLAKE	MANAGER	7839	01/05/81	2850		30
7782	CLARK	MANAGER	7839	09/06/81	2450		10
7788	SCOTT	ANALYST	7566	19/04/87	3000		20
7839	KING	PRESIDENT		17/11/81	5000		10
7844	TURNER	SALESMAN	7698	08/09/81	1500	0	30
7876	ADAMS	CLERK	7788	23/05/87	1100		20
7900	JAMES	CLERK	7698	03/12/81	950		30
7902	FORD	ANALYST	7566	03/12/81	3000		20
7934	MILLER	CLERK	7782	23/01/82	1300		10

14 ligne(s) sélectionnée(s).

### Exercice 1 : « Exo1\_plsql.sql »

Ecrire le programme PL/SQL

- L'utilisateur saisit un nom d'employé
- Si cet employé n'a pas de travail défini, vous devez afficher le nom de l'employé suivi de n'a pas de travail.
- Si cet employé est un vendeur (SALESMAN) vous devez lui attribuer 1000 francs de commission et afficher le nom de l'employé suivi de a 1000 frs de commission
- Si cet employé est dans la table mais n'est pas vendeur, vous devez lui affecter 0 de commission et afficher le nom de l'employé suivi de n'a pas de commission

Vérifier si les modifications ont bien été effectuées dans la table.

### Exercice 2 : « Exo2\_plsql.sql »

Ecrire un programme PL/SQL permettant d'afficher la factorielle de 9.

Rappel  $9! = 9 * 8 * 7 \dots * 1$

### Exercice 3 : « Exo3\_plsql.sql »

Vous devez rechercher en premier lieu le salaire de l'employé 7902. (déclaration d'une constante)

Tant que le salaire < 4000, vous devez continuer à chercher le salaire du chef de l'employé et ainsi de suite : Select .... where empno = chef ;

Dans notre exemple cela s'arrêtera à King

Vous devez ainsi afficher le nom et le salaire sur lequel le programme se termine.