PL SQL

Séance 1

Pr. M’barek ELHALOUI

Introduction : Pourquoi PL SQL ?

- **SQL :**
  - Un langage non procédural (ne contient pas d'instructions procédurales telles que les boucles, les conditions,…)
  - Ne permet pas de réaliser les traitements qui nécessitent parfois des calculs sur la base de données.
  - Besoin de créer des traitements complexes destinés à être stockés sur le serveur.
  - Nécessité d'un langage procédural pour lier plusieurs requêtes SQL avec des variables et avec la structures des langages habituels (Langage de 4ème génération).

Introduction : Pourquoi PL SQL ?

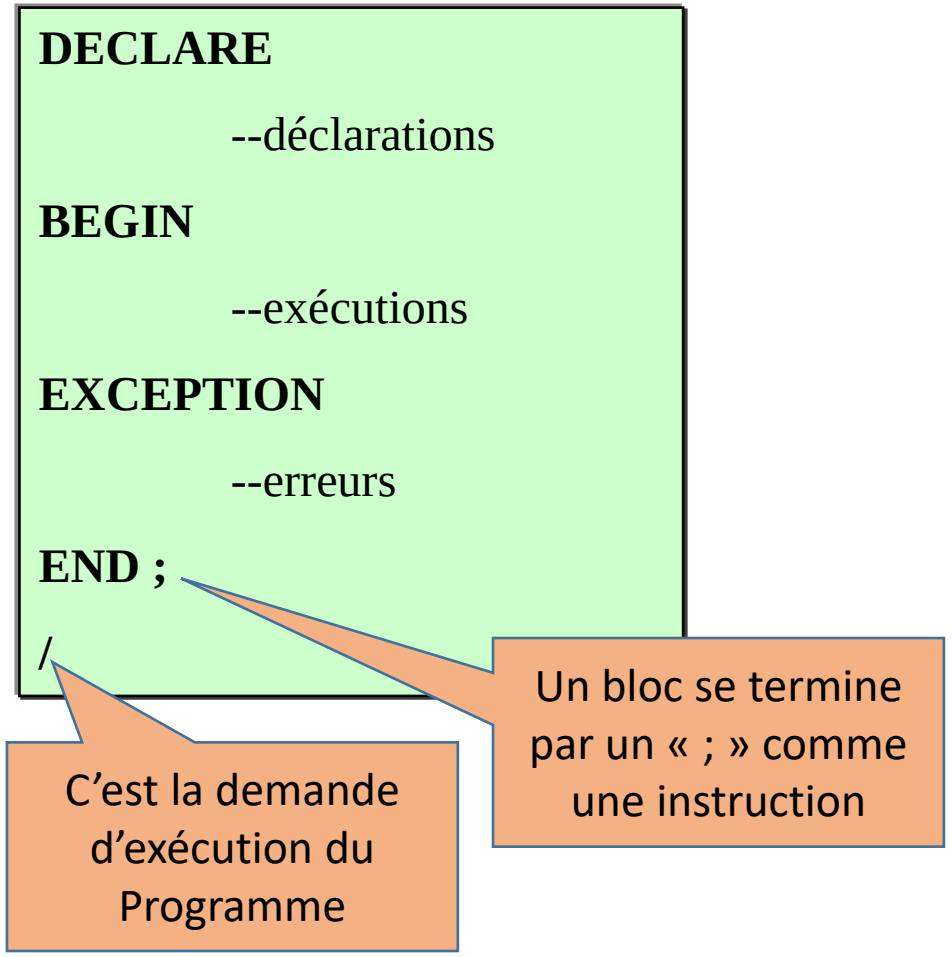
- **PL SQL :**
  - Procedural Language extensions to SQL (Extension de SQL)
  - Procédural : Surcouche procédurale à SQL, boucles, contrôles, affectations, exceptions, ….
  - Syntaxe proche de Pascal ou ADA
  - Chaque programme est un bloc (BEGIN – END)
  - Programmation adaptée pour :
    - Transactions
    - Une architecture Client - Serveur

Introduction : PL SQL

- **Caractéristiques :**
  - **Extension de SQL** : des requêtes SQL cohabitent avec les structures de contrôle habituelles de la programmation structurée (blocs, alternatives, boucles);
  - **Les variables :** permettent la mémorisation des résultats des requêtes et l'échange d'informations entre ces dernières et avec le reste du programme.
  - Définition de **sous-programmes**
  - Gestion des erreurs à l'exécution **(exceptions)**
- **Avantages :**
  - Prise en charge du langaue SQL
  - Permet de définir des fonctions, des procédures ou encore des déclencheurs (triggers).
  - Portabilité : toutes les BDs Oracle comportent un moteur d'exécution PL/SQL
  - Meilleure Performance

Structure d'un Programme PL SQL

- **Section déclaration** (optionnelle) :
  - Variables locales simples
  - Variables tableaux
  - curseurs
- **Section Begin du Bloc** (Obligatoire) :
  - Section des ordres exécutables
  - Ordres SQL
  - Instructions PL SQL
- **Section EXCEPTION** (optionnelle) :
  - Réception en cas d'erreur
  - Exceptions SQL ou utilisateur
- **Section « End ; » du Bloc** (Obligatoire)



Structure d'un Programme PL SQL

- PL/SQL n'est pas sensible à la casse,
- Les identificateurs (noms) comportent des lettres, chiffres, $, #, ..
- On peut avoir des sous-blocs (blocs imbriqués).
- Une instruction se termine par ";"
- Les opérateurs de SQL sont valides en PL/SQL .
- Les commentaires :
  - **-- ceci est un commenatire sur une ligne**
  - **/* ceci est un commentaire**
  - **sur plusieurs**
  - **lignes */**

```
DECLARE
        --déclarations
BEGIN
        --exécutions
    BEGIN
        --exécutions
    EXCEPTION
        --erreurs
    END ;
EXCEPTION
        --erreurs
END ;
/
```

Les types de Bloc

Anonyme

```
[DECLARE]
 --déclarations
BEGIN
 --exécutions
[EXCEPTION]
 --erreurs
END ;
```

Procédure

```
PROCEDURE name
IS
 BEGIN
  --exécutions
[EXCEPTION]
        --erreurs
 END ;
```

Fonction

```
FUNCTION name
RETURN datatype
IS
 BEGIN
   --exécutions
  RETURN value;
[EXCEPTION]
        --erreurs
 END ;
```

Structure d’un Programme PL SQL

Pour afficher le contenu d'une variable, les procédures :

- **DBMS OUTPUT.PUT()** et **DBMS OUTPUT.PUT LINE()** prennent en argument une valeur a afficher ou une variable dont la valeur est à afficher.
- Par défaut, les fonctions d'affichage sont désactivées.
- Il convient, à moins que vous ne vouliez rien voir s'afficher, de
- les activer avec la commande : **SET SERVEROUTPUT ON** .

Exemple : Programme « Bonjour »

- - - La sortie écran ne marche que si le serveur d'impression est ouvert. Faire **SET SERVEROUTPUT ON** sous SQL Plus pour basculer sur le mode sortie console.

```
SQL> SET SERVEROUTPUT ON

SQL> begin
```

- - - DBMS_OUTPUT.PUT_LINE permet d'écrire sur la console SQLPlus.

```
2         DBMS_OUTPUT.PUT_LINE('Bonjour');
3       end;
```

- **- - demande d'exécution du programme tapée**

```
4       /
```

-

Bonjour

Les Variables

- Les variables sont utilisées pour :
  - Stockage temporaire des données
  - Manipulation des valeurs stockées
  - Réutilisation
- Exigences d'identificateurs oracle :
  - Maximum 30 caractères
  - Commencent par une lettre
  - Peuvent contenir lettres, chiffres, _, $ et #
- Pas sensibles à la casse
- Portée dans le bloc et dans le bloc fils (règles des langages à blocs)
- Doivent être déclarées avant d'être utilisées

Les Variables

Principaux types :

- Variables scalaires →Types habituels correspondants aux types Oracle ou SQL :
  - Issus de SQL : **CHAR**, **NUMBER**, **DATE**, **VARCHAR2**
  - Types PL/SQL : **BOOLEAN**, **SMALLINT**, **BINARY_INTEGER**, **DECIMAL**, **FLOAT**, **INTEGER**, **REAL**, **ROWID**
- Types composés → adaptés à la récupération des colonnes et lignes des tables SQL : **%TYPE**, **%ROWTYPE**
  - Enregistrements (record)
  - Tables

Les Variables : Déclaration

nom variable [CONSTANT] type [[NOT NULL] [:= expression | DEFAULT expression];

- **nom variable :** représente le nom de la variable composé de lettres, chiffres, $, _ ou # Le nom de la variable ne peut pas excéder 30 caractères et commence par une lettre
- **CONSTANT :** indique que la valeur ne pourra pas être modifiée dans le code du bloc PL/SQL
- **Type :** représente le type de la variable (types usuels de ORACLE)
- **NOT NULL :** indique que la variable ne peut pas être NULL, et dans ce cas **expression** doit être indiqué.

NB :
- Si une variable est déclarée avec l'option CONSTANTE, elle doit être initialisée
- Si une variable est déclarée avec l'option NOT NULL, elle doit être initialisée
- Déclarations multiples interdites : ~~(i, j integer)~~

Les Variables : Déclaration

- Une variable se déclare de la sorte :

```
BECLARE
/* Forme de la déclaration : Nom type [:= initialisation]; */

   nom varchar2(50) := 'Alami';
   dateRec date; /* initialisation implicite avec null */


BEGIN … END;
```

Les Variables Simples

- Variables de type SQL :

```
nbr      NUMBER(2)  ;
nom      VARCHAR(30)  ;
minimum CONSTANT INTEGER := 5 ;
salaire NUMBER(8,2)  ;
debut    NUMBER NOT NULL ;
```

- Variables de type booléen (TRUE, FALSE, NULL)

```
fin      BOOLEAN ;
reponse BOOLEAN DEFAULT TRUE ;
ok       BOOLEAN := TRUE;
```

Variables faisant référence au dictionnaire de données

- Référence à une colonne (table, vue) : Variable de même type qu'un attribut (colonne)

```
vsalaire    employe.salaire%TYPE;
vnom        etudiant.nom%TYPE;
Vcomm       vsalaire%TYPE;
```

- Référence à une ligne (table, vue) : Variable de même type qu'un schéma (ligne)

```
vemploye    employe%ROWTYPE;
vetudiant   etudiant%ROWTYPE;
```

- Contenu d'une variable : variable.colonne

```
vemploye.adresse
```