PL/ SQL

- Introduction:
  - PL/SQL (Procedural Language SQL) est un langage procedural structuré en BLOCS.
  - Extension du SQL: des requêtes SQL intégrées avec les structures de contrôle habituelles (alternatives, répétitives)
  - Un programme PL/SQL est composé de fonctions, de procédures, de triggers..
  - C'est un langage propriétaire d'ORACLE.
  - Peut s'exécuter comme bloc anonyme, procédure ou une fonction

PL/ SQL

- Les blocs PL/SQL: Un programme PL/SQL est constitué de trois blocs ou de trois sections.
  - La section déclarative (optionnelle)
  - La section de contrôle ou d'exécution (obligatoire)
  - La section de gestion des exception (optionnelle)
- La section déclarative: dans cette section, on déclare toutes les variables nécessaires à l'exécution du programme PL/SQL. Cette section commence en général par le mot réservé : DECLARE
- La section d'exécution: cette section contient des énoncés SQL ou PL/SQL. Elle débute par le mot réservé BEGIN et se termine par le mot réservé END.
- La section de gestion des exceptions: cette section commence par le mot réservé EXCEPTION. Si un erreur est générée lors de l'exécution d'un programme , celle-ci est envoyée au BLOC EXECPTION, ce qui donne la possibilité de la traiter et de ne pas mettre fin brutalement à l'exécution du Programme.

PL/ SQL

- Exemple de déclarations:

  DECLARE

  Numero NUMBER(4);

  nom VARCHAR2(30);

  Salaire NUMBER(8,2);

  Date_naissance DATE;

  // assignation de valeur

  Augmentatation NUMBER (4) := 5;

  Groupe VARCHAR2(10) := ‘groupe1’;

  // utilisation du %TYPE. Permet de déclarer des variable de même type que des variables déjà déclarée.

  Numero_Client yacoubsa.clients.numclient%type

  permet de déclarer le numéro du client du même type que le numclient de la table clients de l’usager yacoubsa.

PL/ SQL

- Suite Exemple de déclarations:

  Salaire_MIN NUMBER(7,2);

  Salaire_MAX SalaireMin%TYPE;

  Acteur VARCHAR2(30);

  Realisateur Acteur%TYPE :='Spielberg';

  // on peut donner le type ROWTYPE lorsqu'il s'agit de déclarer un enregistrement (une ligne).

  Enregistrement etudiant%ROWTYPE

Type de données:

- Les types SQL (number, varchar2(n), date, char(n), ..) http://download.oracle.com/docs/cd/E11882_01/appdev.112/e17126.pdf (page 110)
- Le type Boolean
- Le type variable %TYPE
- Le type rangée %ROWTYPE
- Les types CURSOR. (dynamique ou non)
- Les types RECORD.

Les instructions de contrôl
Instruction IF –END IF

- **IF - THEN**

  IF condition THEN
  Séquence_instructions;
  END IF;

- **IF - THEN - ELSE**

  IF condition THEN
  Séquence_instructions1;
  ELSE
  Séquence_instructions2;
  END IF;

Instruction IF –END IF

IF- THEN - ELSIF

```
IF condition THEN
        Séquence_instructions1;
ELSIF condition THEN
        Séquence_instructions2;
ELSIF condition THEN
        Séquence_instructions3;
ELSIF condition THEN
        Séquence_instructions4;
ELSE
        Séquence_instructions5;
END IF;
```

Exemple

```
Declare choix number;
begin
  IF choix =1 THEN
     delete from commander where numarticle = 100 ;
          ELSIF choix =2 THEN
     delete from commander where numarticle = 110;
 ELSE
      delete from commander where numarticle = 130;
  END IF;
END;
```

IF – END IF

- Dans une alternative, les mot réservés IF, THEN et END IF sont obligatoires. Les autres (ELSIF et ELSE)sont optionnels

Les procédures –fonctions packages et triggers

Les procédures stockées:

Définition:

Une procédure est un code PL/SQL défini par l'utilisateur et stocké dans la base de données. Ce qui permet d'éliminer la redondance de code.

Avantages:

- Le code SQL est précompilé.
- Exécution plus rapide (puisque stockée dans le serveur)
- Moins de code redondant

Les procédures stockées

- Syntaxe générale:

CREATE OR REPLACE PROCEDURE schema.NOMProcedure

(param1 [IN|OUT|IN OUT] typeparam1,

param2 [IN|OUT|IN OUT] Typeparam2, …) AS|IS

Déclarations des variables locales (sans DECLARE)

BEGIN

Bloc PL/SQL

END;

Syntaxe (suite)

- CREATE indique que l'on veut créer une procédure.
- OR REPLACE (facultative) permet d'écraser une procédure portant le même nom
- Param1, param2 sont les paramètres de la procédure
- IN :indique que le paramètre transmis par le programme appelant n'est pas modifiable par la procédure. (entrée) par défaut les paramètres sont IN
- OUT: indique que les paramètres sont modifiables par la procédure (sortie)
- IN OUT: combinaison de IN et OUT

Exemple

```
CREATE OR REPLACE PROCEDURE Augmentation
 (
 PNCODECOUR IN  RESULTATS.CODECOURS%Type,
 PNNOTE  IN NUMBER
 )
 AS
 BEGIN
 -- augmentation de la note
 Update RESULTATS Set NOTE = NOTE + PNNOTEIN
Where CODECOURS = PNCODECOUR  ;
COMMIT;
END;
```

EXEMPLE

```
CREATE OR REPLACE
PROCEDURE INSERECODETYPE
(
  CODE IN TYPELIVRE.CODETYPE%TYPE,
DESCR IN TYPELIVRE.DESCRIPTION%TYPE)
AS

BEGIN
INSERT INTO TYPELIVRE VALUES (CODE,DESCR);
END ;
```

EXEMPLE

```
CREATE OR REPLACE PROCEDURE afficher
(
pnum in armes.numitem%type,
sortie out armes.efficacite%type
) AS

BEGIN
select efficacite into sortie from armes where numitem = pnum;

END afficher;
```

Execution

- Pour exécuter une procédure dans SQL Developper , il suffit de taper l'instruction suivante:

EXECUTE nomProcedure(val1, val2,..)

EXECUTE augmentation('KED',10);

- Pour détruire une procédure, utiliser DROP PROCEDURE nomProcedure
- Lorsqu'une procédure a besoin de déclarer ses variables, pas besoins de DECLARE. Les variables sont déclarées entre IS |AS et BEGIN
- Pour exécuter une procédure avec un paramètre en OUT, il faudra déclarer la variable en OUT, utiliser un bind sur la variable de sortie et print pour voir le résultat:

```
variable res varchar2(30);
execute  afficher(2,:res);
print res;
```

Les fonctions

- Les fonctions sont considérées comme des procédures qui retournent des valeurs.
- Synthaxe.

```
CREATE OR REPLACE FUNCTION  NomFonction (param1 [IN|OUT|IN OUT ]
    typeparam1,
param2 [IN|OUT|IN OUT] Typeparam2, …)
RETURN TypeVariable  AS |IS
Declaration des variables locales (sans DECLARE)
BEGIN
   Bloc PL/SQL;
RETURN variable;
END ;
```

Exemple1

```
CREATE OR REPLACE FUNCTION afficher

(
 CODER IN TYPELIVRE.CODETYPE%TYPE
 )
  RETURN  VARCHAR2 AS

DESCR  varchar2(60);
BEGIN
  SELECT description  into DESCR  FROM TYPELIVRE where codetype = CODER;
  return  DESCR ;


END ;
```

cette fonction affiche la description (titre) d'un livre selon le code du livre

Exécution et suppression

- Execution : (dans SQL developper)

SELECT nomFonction (liste des valeurs) from DUAL (ou SYS.DUAL);

select afficher(21) from dual;

select total from dual;

- Détruire une fonction:

DROP FUNCTION nomFonction;

Le type CURSOR

- Un curseur est une zone de mémoire utilisée par Oracle pour récupérer les résultats de requêtes SQL.
- Il peut être explicite, il est donc associé à une seule requête
  - Exemple: CURSOR Resultat IS select nom, preneom from etudiant where nom like %POIT%;
- Il peut être dynamique, n'est pas associé à une seule requête SQL. Dans ce cas, lors de sa déclaration, il faudra utiliser le mot réservé REF.
  - Exemple: TYPE enregistrementEtudiant is REF CURSOR (la variable enregistrementEtudiant est de type CURSOR dynamique).
- C'est ce type de curseur que nous allons utiliser.

Le type CURSOR

- Un curseur dynamique est un curseur qui n'est pas associé à une seule requête SQL. Dans ce cas, lors de sa déclaration, il faudra utiliser le mot réservé REF.

- Un REF CURSOR est vu comme un pointeur sur une zone mémoire dans le serveur (contenant le résultat d'une requête).
- Un REF CURSOR n'est pas Updatable.
- Un REF CURSOR est forward-only.
- Il a un grand avantage lorsque votre procédure ou fonction PL/SQL est appelée par un langage de haut niveau comme C# ou JAVA.

Exemple

TYPE enregistrementEtudiant is REF CURSOR;

Pour affecter le résultat d'une requête SELECT à une variable de type REF CURSOR, il faut utiliser la syntaxe:

OPEN nomDuCurseur FOR SELECT …..

Le curseur sys_refcursor (par défaut) est un curseur dynamique.

Les Packages

- Un package est un objets de la base de données qui encapsule d'autres objets (procédures, fonctions ..)
- Un package a essentiellement deux parties:
  - la partie déclaration: dans cette partie sont déclarées les variables globales, les curseurs, les procédures et les fonctions
  - la partie corps du programme: dans cette partie, sont définies les procédures et les fonctions et les curseurs.

Les Packages

Avantages:

- Modularité : le fait de regrouper logiquement les éléments PL/SQL liés rend la compréhension plus facile des différents éléments du package.
- Facilité de développement : il est possible de définir uniquement la partie spécification du package. Le corps du package sera défini que pour l'exécution de l'application
- Meilleure performance : le package est présent en mémoire dès l'appel d'un élément qui le compose (fonction ou procédure) ce qui rend l'accès aux éléments du package beaucoup plus rapide que l'appel à des procédures ou à des fonctions indépendantes.

Les Packages

- Au fur et à mesure que vous modifiez votre package, enregistrez, en principe la compilation se fait en même temps
- Vous allez définir et compiler les procédures une à une. Il est très important de vérifier les procédures et fonctions une à la fois.
- La compilation de votre Body Package se fait de la même façon que la compilation de votre Package.
- Dans SQL Developer, vous pouvez tester certaines de vos procédures et fonctions en les exécutant tout en précédant leurs noms par le nom du package.
- Exemples:
  - SELECT gestionetudiants.afficher(420)FROM dual;
  - EXECUTE gestionetudiants.suppression(18);

Exemple 1/2

```
CREATE OR REPLACE
PACKAGE GESTIONEMPLOYES AS

TYPE EMPENR IS REF CURSOR;

PROCEDURE INSERTION (NUM IN NUMBER , NOM IN
    VARCHAR2,PRENOM IN  VARCHAR2, SALAIRE IN NUMBER );

PROCEDURE SUPPRESSION (NUM IN NUMBER);
PROCEDURE SELECTION (REQUETE OUT EMPENR );
FUNCTION TOTAL RETURN NUMBER;

END ;
```

Exemple 2/2

```
CREATE OR REPLACE  PACKAGE  BODY GESTIONEMPLOYES  AS
  PROCEDURE INSERTION (NUM IN NUMBER , NOM IN  VARCHAR2,PRENOM IN  VARCHAR2, SALAIRE IN NUMBER ) AS
    BEGIN
    INSERT INTO EMPLOYES (numemp, nomemp, prenomemp, salaireemp ) VALUES  (NUM,NOM,PRENOM ,SALAIRE);
    END;
      PROCEDURE SUPPRESSION (NUM IN NUMBER) AS
   BEGIN
   DELETE FROM EMPLOYES WHERE NUMEMP = NUM;
   END;
      FUNCTION  TOTAL RETURN NUMBER AS
    TOTALEMP NUMBER;
    BEGIN
      SELECT COUNT(*)  INTO TOTALEMP FROM EMPLOYES;
    RETURN TOTALEMP;
    END;
PROCEDURE SELECTION(REQUETE OUT EMPENR )AS
    BEGIN
    OPEN REQUETE FOR SELECT * FROM EMPLOYES ;
    END;
END GESTIONEMPLOYES;
```

Cursor : statique (explicite)

- Un Curseur statique ou explicite est obtenu par l'exécution d'une commande SQL.
- Pour son utilisation, il faut quatre étapes:
  - Déclaration du curseur
  - Ouverture du curseur OPEN
  - Lecture du curseur FETCH
  - Fermeture du curseur CLOSE
- Exemple:

Cursor : statique (explicite)

```
SET SERVEROUTPUT ON;
DECLARE
Nom1 USER1.ETUDIANTS.NOM%TYPE;Prenom1 varchar2(20);
CURSOR curseur1 IS SELECT nom, Prenom from etudiants;
BEGIN
OPEN curseur1;
LOOP
FETCH curseur1 INTO Nom1, Prenom1;
EXIT WHEN curseur1%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('le nom est '|| nom1 || 'le prenom est '||
    prenom1);
END LOOP;
CLOSE Curseur1;
END;
```

Cursor : statique (explicite)

Explications:

Les variables nom1 et prenom1 sont des variables locales destinées à recevoir le contenu du curseur.

La commande FETCH permet de lire ligne par ligne le contenu du curseur. À chaque fois que cette commande est appelée, le curseur avance au prochain enregistrement dans l'ensemble actif.

Curseur1%notfound : retourne vrai si le dernier FETCH échoue et ne retourne aucun enregistrement. %notfound est un attribut du curseur explicite.

Cursor : statique (explicite)

Autres attributs du curseur explicite:

%rowcount: retourne le nombre d'enregistrements trouvés

Exemple:

```
LOOP
    FETCH emp_curseur INTO
            emp_nom, dept_num
IF emp_cursor%rowcount>10
THEN EXIT;
END IF;
END LOOP;
```

Cursor : statique (explicite)

Autres attributs du curseur explicite:

%isopen: retourne vrai si le curseur est ouvert:

Exemple:

```
IF emp_curseur%isopen THEN
  FETCH ...
ELSE
    OPEN emp_curseur
END IF;
```