

TP PL/SQL

Dans ce TP, vous allez apprendre le langage PL/SQL proposé par Oracle. Voici un tableau récapitulatif des commandes PL/SQL principales.

Bloc PL/SQL	<pre> DECLARE -- Déclaration constantes/variables BEGIN -- Commandes/instructions EXCEPTION -- Traitement des erreurs à l'exé. END; / -- Exécution automatique lors du « start » </pre>
Déclaration de variable	<pre>Nom Variable TYPE_VARIABLE;</pre>
Affectation	<pre>Nom Variable:=valeur;</pre>
Tests	<pre> SELECT attribut INTO Nom_Variable FROM table; IF condition1 THEN -- Instructions ELSEIF condition2 THEN -- (Optionnel) -- Instructions ELSE -- (Optionnel) -- Instructions END IF; </pre>
Boucles	<pre> FOR compteur IN [REVERSE] min..max LOOP -- Instructions END LOOP; WHILE condition LOOP -- Instructions END LOOP; </pre>
Curseurs	<pre> -- Déclaration CURSOR Nom_Curseur IS Requête_SQL; -- Utilisation FOR tuple IN Nom_Curseur LOOP -- Instructions -- Ex. Nom_Variable:=tuple.attribut; END LOOP;-- NB : tuple est de type Nom Curseur%ROWTYPE </pre>
Exceptions	<pre> -- Déclarer Nom Exception EXCEPTION; -- Lever RAISE Nom Exception; -- Traiter WHEN Nom Exception THEN -- Instruction; </pre>

Partie 1

Le schéma de la BDD utilisée est composé de 2 tables EMP et DEPT.

1. Faire le nécessaire afin de les créer dans votre schéma avec les contraintes nécessaires, et d'insérer les données suivantes :

EMP

NUMEMP	NOM_COMPLET	PROFESSION	CHEF	SALAIRE	NUMDEP
1	BARTH Florent	CHEF DE PROJET	3	13000	2
2	XAVIER Richard	CHERCHEUR	3	21000	1
3	NICOLLE Chris	CHERCHEUR		25000	1
4	BLAKE John	DEVELOPPEUR	6	8000	2
5	DUPONT Jean	DEVELOPPEUR	3	9000	2
6	MARTIN Alexandre	COMPTABLE	3	10000	3
7	RAY Benjamin	COMPTABLE	3	10000	3
8	MILLER Pascal	DEVELOPPEUR	3	9000	2
9	FORD John	DIRECTEUR	3	30000	4

DEPT

NUMDEP	NOMDEP	NOMLOC
1	RECHERCHE	DIJON
2	DEVELOPPEMENT	NEW-YORK
3	FACTURATION	PARIS
4	DIRECTION	LONDRES

A) Initiation PL/SQL

Soit le programme PL/SQL suivant :

```
DECLARE
    --
    n NUMBER(2);
    --
    CURSOR employes IS SELECT numemp, nomemp, salaire FROM EMP;
    --
    newsal emp.salaire%TYPE;
    --
    empv EXCEPTION;
BEGIN
    --
    SELECT COUNT(*) FROM EMP;
    --
    IF n=0 ALORS
        --
        RAISE empv;
    END IF;
    --
    FOR employe IN employes LOOP
        --
        newsal=employe.salaire+50;
        --
        UPDATE EMP SET SALAIRE=newsal where NUMEMP = employe.numemp;
    END LOOP;
    Commit;

    --
EXCEPTION
    --
    WHEN empv THEN dbms_output.put_line('Message d''erreur !');
END;
```

2. Commenter le programme PL/SQL et corriger les erreurs s'il y en a.
 3. Remplacer la Chaîne "Message d'erreur" en fin de programme par un message plus approprié.
 4. Quelle variable dans ce programme peut être assimilée à un « RecordSet ».
 5. Exécuter ce programme sous SQL Developer, que fait ce programme ?
- Pour pouvoir afficher les messages sous SQL Developer (dbms_output.put_line), vous devez exécuter préalablement :

```
SQL> set serveroutput on
```

6. Corriger les erreurs

B) Vos premiers pas avec PL/SQL

7. Ecrire une **procédure** qui affiche « Hello Word ». Utiliser la syntaxe suivante :

```
CREATE OR REPLACE PROCEDURE nom_proc (nom_param [in] type_param,...) AS
-- Déclaration de variables.
BEGIN
-- Corps de la procédure
END;
```

Puis vous devez utiliser une commande PLSQL qui exécute la procédure.

```
DECLARE
BEGIN
    Nom_procedure;
END;
```

8. Ecrire une fonction PLSQL qui prend en paramètre un nom d'employé et renvoie son salaire. Vous devez utiliser la syntaxe suivante

```
CREATE OR REPLACE FUNCTION nom_fonction (nom_param in type_param)
RETURN Type_de_Retour IS
-- Déclaration de variables.
BEGIN
-- Corps de la fonction
RETURN Valeur_de_retour;
END;
```

Pour voir le résultat de votre fonction, vous pouvez utiliser la syntaxe suivante :

```
DECLARE
BEGIN
    dbms_output.PUT_LINE (Nom_de_Fonction (Parametres, ...) );
END;
```

Sur le même principe, modifier votre fonction pour qu'elle prenne en entrée un nom et qu'elle renvoie le message suivant :

« NUMERO = *Valeur*, POSTE = *Valeur*, SALAIRE = *Valeur* »

C) Approfondissement

9. Ecrire une procédure *affecter_emp_dept* qui prend en paramètre un nom d'employé et un nouveau département, et affecte au département cet employé.
10. Ecrire une fonction *exist_emp* qui renvoie VRAI si l'employé passé en paramètre se trouve dans la table EMP sinon elle renvoie FAUX.
11. Ecrire une fonction *app_emp_dept* qui affiche «cet employé appartient au département xxxxxxxxxx» si l'employé fait bien partie du département passé en paramètre sinon affiche «cet employé n'appartient pas au département xxxxxxxxxx»
12. Ecrire un programme *maj_emp* qui met à jour les salaires selon la nouvelle politique de la direction. Vous devez :
 - Réduire les salaires des employés du département *Recherche* de 200€
 - Augmenter les salaires du département *Développement* de 40€
 - Multiplier par 2 les salaires du département *Direction*
 - Mettre les salaires des employés du département *Facturation* à la moyenne des salaires des employés.

Partie 2

Sous PLSQL, vous pouvez, comme pour certains langages, regrouper vos procédures et fonctions dans des paquets (packages).

Un paquetage se décompose en deux parties : la spécification et le corps.
La spécification doit être compilée avant le corps. Pour la créer:

```
CREATE OR REPLACE package Nom_Package as

function Nom_Fonction1 RETURN Type_de_retour;
function Nom_Fonction2(parametre IN typeParametre)
RETURN Type_de_retour;
...
procedure Nom_Procedure1;
procedure Nom_Procedure2 (parametre IN typeParametre);
...
end Nom_Package;
/
```

Création du Corps :

```
CREATE OR REPLACE package body Nom_Package as

function Nom_Fonction1 RETURN Type_de_retour
DECLARE
BEGIN
...
END Nom_Fonction ;

...

END Nom_Package;
/
```

13. Supprimer les procédures et les fonctions déjà créées dans ce TP
14. Maintenant, recréer ces mêmes fonctions et les procédures mais regroupées dans un paquetage « **PACK_IQ** ».
15. Essayer d'appeler de nouveau une des fonctions ou procédures. Que se passe-t-il ? Quelle est la solution ?