1. PL/SQL est un langage qui intègre SQL et permet de programmer de manière procédurale. Il est spécifique à Oracle. Pour SQL Server il existe par exemple un équivallent : TRANSAC SQL
2. Un bloc PL/SQL avec des exceptions a donc la structure suivante :

DECLARE -- mes déclarations de variables...

BEGIN -- début de la section executable

-- mes ordres SQL et PLSQL

EXCEPTION

-- mon traitement des exceptions

END; -- la fin du bloc executable

1. procédure PL/SQL / Une procédure est simplement un programme PL/SQL nommé, compilé et stocké dans la base.
   1. Pour créer la procédure :

-- hello.txt

-- cree une procedure stockee PL/SQL qui affiche Hello!

create or replace procedure hello

IS

BEGIN

dbms\_output.put\_line('Hello!');

END;

--

-- Pour executer la procedure:

-- SQL> SET SERVEROUTPUT ON !!!!!!!!!!!!!!!!!!!!!!

-- SQL> EXECUTE hello

* 1. pour executer la procedure :

execute hello;

1. fonctions PL/SQL : Une fonction est une procédure retournant une valeur.
   1. Création de la function PL/SQL :

CREATE OR REPLACE FUNCTION solde (no INTEGER)

RETURN REAL IS le\_solde REAL;

BEGIN

SELECT solde INTO le\_solde FROM clients

WHERE no\_cli = no;

RETURN le\_solde;

END;

* 1. Utilisation de la function dans une requete :

SELECT solde(1000) from dual ;

* 1. l'appel 'bête' de l'exécution d'une fonction comme si c'était une procédure (ouh la la !) ne fonctionne évidemment pas! mais on est souvent tenté de le faire quand même.

1. package PL/SQL : Un package est un module de programmes incluant procédures et / ou fonctions fonctionnellement dépendantes. Un package est composé de 2 parties :
   1. la spécification (introduite par ‘CREATE PACKAGE’) liste les entêtes de procédures et fonctions contenues dans le package,
   2. le corps du package (introduit par ‘CREATE PACKAGE BODY’) qui contient le code effectif des procédures et fonctions déclarées précédemment.
   3. Exemple :

CREATE PACKAGE clients AS -- spécifications du package

PROCEDURE insere\_client (no INTEGER, nom VARCHAR2, ...);

PROCEDURE supprime\_client (no INTEGER);

...

END;

CREATE PACKAGE BODY clients AS -- le corps du package

PROCEDURE insere\_client (no INTEGER, nom VARCHAR2, ...) IS

BEGIN

...

INSERT INTO clients VALUES (no, nom, ...);

END;

PROCEDURE supprime\_client (no INTEGER) IS

BEGIN

DELETE FROM clients WHERE no\_cli = no;

END;

...

END; -- du package

1. Trigger : Un trigger est un morceau de code PL/SQL, stocké dans la base, déclenché lors de l’occurrence d’un événement particulier. Exemple :

-- trigger déclenché lors d’une insertion

-- ou d’une modification de la table client

SQL> CREATE OR REPLACE TRIGGER aff\_discount

BEFORE INSERT OR UPDATE ON clients

FOR EACH ROW

WHEN (new.no\_cli > 0)

DECLARE

evol\_discount number;

BEGIN

evol\_discount := :new.discount - :old.discount;

DBMS\_OUTPUT.PUT\_LINE(' evolution : ' || evol\_discount);

END;

/

-- FOR EACH ROW signale qu’une modification de 4 lignes

-- par un seul UPDATE déclenche 4 fois le trigger.

-- Si on ne souhaite qu’un seul déclenchement ,

-- on omet simplement la clause FOR EACH ROW

1. Typage et déclaration de variables simples
   1. nomvar [constant] type [NOT NULL] [:= valeur | DEFAULT expression ]
   2. Exemple : X NUMEBR NOT NULL :=10.40;
2. Il existe plusieurs catégories de types de variables (une variable doit être toujours déclarée dans le bloc DECLARE) :
   1. simples (INTEGER, NUMBER, DATE, CHAR, BOOLEAN, VARCHAR, ...)
   2. composées (RECORD, TABLE, VARRAY , NESTED TABLE)
   3. référence
   4. pointeur de LOB (CLOB, BLOB, BFILES, NCLOB )
   5. type 'dynamique' %TYPE : qui définit une var du type de la colonne d'une table existante. Cette possibilité est fort utile. Elle fait référence aux caractéristiques d’une colonne de table (par exemple) et d’en récupérer automatiquement le type. Si le schéma de la base de données se trouve modifié (zone textuelle augmentée, zone numérique incorporant plus de décimales…), la variable en PL/SQL hérite d’emblée de ces modifications. Ce type a aussi le mérite de simplifier l’écriture de vos programmes. Par exemple, pour déclarer une variable v\_nom destinée à contenir le nom d’un employé de la table emp, on utilise le type %TYPE : v\_nom emp.ename%TYPE ;
   6. Type enregistrement : Pour déclarer un enregistrement, il faut avoir recours au type %ROWTYPE. Exemple : rec\_employe emp.%ROWTYPE ;
3. Affecter des valeurs aux variables PL/SQL : Il existe deux façons d’affecter des valeurs à des variables.
   1. La première utilise l’opérateur d’assignation, le signe « := ».
   2. La deuxième façon d’attribuer des valeurs à des variables consiste à effectuer un SELECT

De valeurs en provenance de la base de données. La syntaxe utilisée se présente comme suit :

SELECT colonne1, colonne2

INTO variable1, variable2

FROM table

[WHERE conditions] --le signe [] indique des clauses optionnelles

La clause INTO est obligatoire et l’ordre SELECT doit rapporter une seule ligne, sinon une erreur est générée. Si l’ordre SQL retourne plusieurs lignes, on utilisera obligatoirement un curseur.

1. Les curseurs : Oracle 10g utilise des espaces de travail pour exécuter les ordres SQL et en manipuler les données. Une instruction PL/SQL, le curseur, permet d’attribuer un nom à cet espace de travail et d’accéder aux données qu’il contient. Il existe deux sortes de curseurs : les curseurs implicites et les curseurs explicites. Le PL/SQL crée de manière implicite un curseur pour chaque ordre SQL, même pour ceux qui ne retournent qu’une ligne.
2. Les curseurs explicites : Pour les curseurs qui renvoient plus d’un enregistrement, vous pouvez déclarer explicitement un curseur, ce qui permet de traiter individuellement les lignes retournées.

-- recherche de tous les employés d’un département

DECLARE

CURSOR mon\_curseur IS

SELECT ename, sal, job FROM emp WHERE deptno = 10 ;

...

* 1. Les étapes de la vie d’un curseur : Les instructions de déclaration d’un curseur puis OPEN, FETCH et CLOSE permettent le contrôle du début à la fin de son existence. OPEN exécute l’ordre SQL associé au curseur, identifie le result set et positionne le curseur avant le premier enregistrement.L’instruction FETCH retourne l’enregistrement courant puis place le curseur surl’enregistrement suivant. Lorsque le dernier enregistrement est atteint, l’instruction CLOSE permet de fermer le curseur et libère les ressources allouées. Les étapes sont les suivantes :
     1. déclaration du curseur ;
     2. ouverture du curseur ;
     3. traitement des lignes du curseur ;
     4. fermeture du curseur.
  2. CURRENT-OF : permet d’updater les ligne renvoyer dans le curseur.

1. Les structures de contrôle
   1. IF-THEN-ELSE ;( mettre elsif au lieu du elseif)
   2. FOR <compteur> IN [REVERSE] <minimum> .. <maximum> LOOP
   3. ... ;

END LOOP;

* 1. LOOP

... ;

EXIT [WHEN ...];

END LOOP;

* 1. WHILE ... LOOP

... ;

END LOOP;

1. Pour executer : dbms\_output.put\_line , appeler : set serveroutput on;