**PL/SQL**

Présentation :

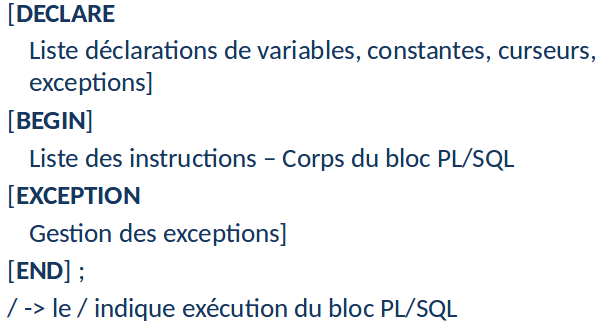
PL/SQL : programming language with SQL. Langage de programmation procédurale. Langage propriétaire Oracle mais qui ressemble beaucoup à de l’ADA. Spécifiquement adapté à la manipulation de bases de données : types requêtes, curseurs, traitement des exceptions. Permet de passer d’un monde ensembliste à un monde enregistrement par enregistrement

Côté serveur : Offre la possibilité de définir des objets persistants : procédures, fonctions, triggers

Côté client : permet d'écrire des blocs PL/SQL anonymes. Utilisable pour le développement d’interfaces graphiques, de masques de saisie (SQLForms), …

Eléments de syntaxe

Comme SQL, la casse n’est pas importante. Les identificateurs peuvent comporter des lettres, des chiffres, les caractères #, $, \_

Commentaires : 

- - sur une ligne

/\*

sur plusieurs lignes

\*/

Un bloc PL/SQL

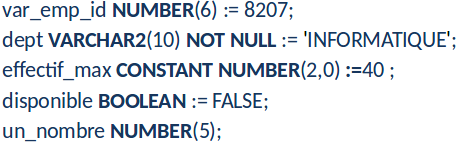
Variables et constantes :

Les variables peuvent être de types suivant :

* scalaire, recevant une valeur de type SQL (CHAR, NUMBER, VARCHAR, …) ou de type PL/SQL (sous type prédéfini:INTEGER ou défini par l’utilisateur)
* composé (RECORD, collection, types objets)
* Référence (REF) ou LOB (pour les données de grandes tailles)

Les contraintes NOT NULL doivent être suivies d’une clause d’initialisation :

identificateur [CONSTANT] typeDeDonnée [NOT NULL] [(:= | DEFAULT) expression];

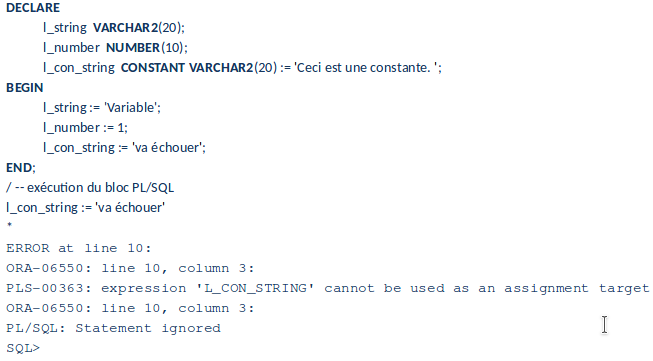


Exemple de variables et constances

Les déclarations multiples ne sont pas autorisés : nom, prenom VARCHAR(10); –Interdit

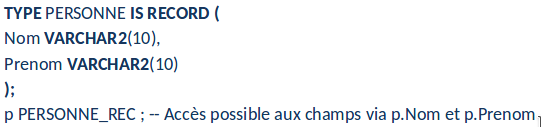
Les affectations des variables dans le bloc BEGIN … END suivent la syntaxe classique :=

un\_nombre := 6



Les types composés

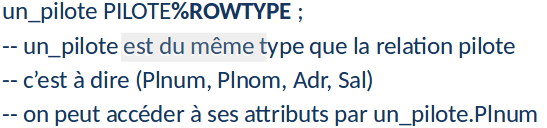
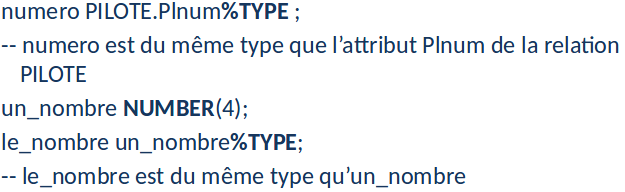
Record : semblable à une structure C. Tous les types sont de type SQL. Une variable de type record peut ressembler à une ligne dans une relation.

Type collections : TABLE, VARRAY (relationnel-objet)

Type objets : relationnel-objet

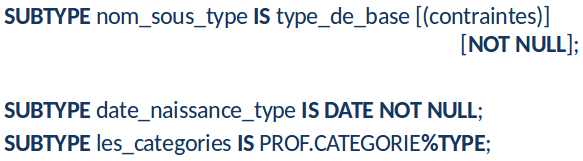
Les types implicites

| attribut%TYPE →signifie “ du même type” | attribut%ROWTYPE → “du même type d’enregistrement” |
| --- | --- |

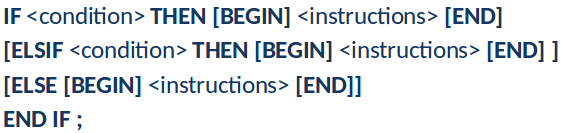


Notion de sous type

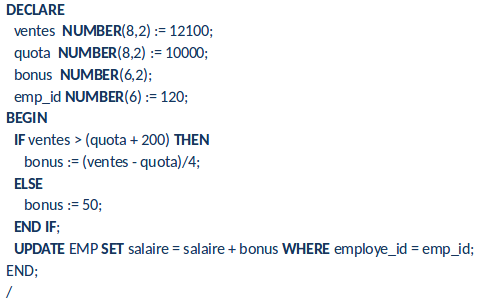
Chaque type prédéfini possède ses caractéristiques (domaine, opérateurs). Un sous type permet de restreindre les caractéristiques. Il en existe des prédéfinis : INTEGER, CHARACTER, POSITIVE. Il est possible de créer ses propres sous types.



Expressions et opérateurs

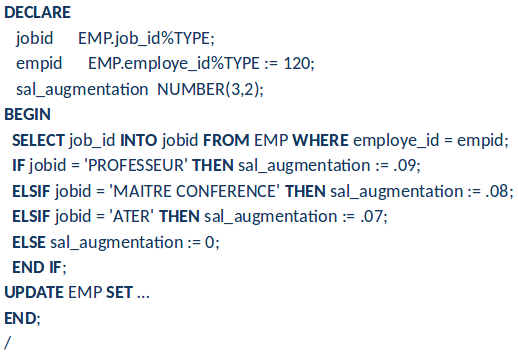
Les opérateurs de SQL sont valides en PL/SQL. Une opérande est une variable, une constante, un littéral, ou un appel à une fonction. Opérateur classiques : \*\*, +, -, \*, /,<, >, =, <=, >=, <>, !=, IS NULL, LIKE, BETWEEN, IN, NOT, AND, OR. Opérateurs de concaténation de chaînes : +, -, ||

Instructions conditionnelles

Les instructions conditionnelles ont une syntaxe classique (comparable à celle d’ADA) 

Exemple :

Ce bloc augmentera le salaire de l’employé 120 d’un bonus en fonction du nombre de ventes effectuées. La base de données est mise à jour.



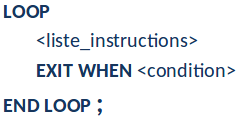
Exemple :

Ce bloc augmentera le salaire de l’employé 120 d’un bonus en fonction de sa catégorie (jobid).

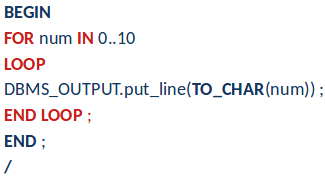
Les itérations

Les instructions d’itération sont tout à fait classiques :

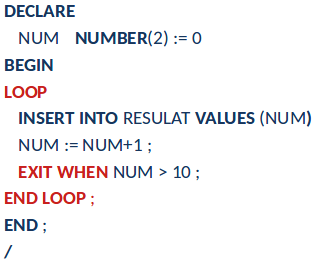
| Boucle FOR | Boucle WHILE |
| --- | --- |

Il existe aussi la possibilité de sortir avec une clause EXIT WHEN 

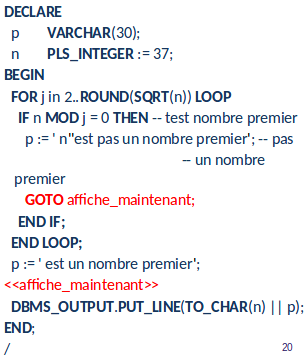
Boucle LOOP :



Exemple : affichage des 10 premiers nombres à l’écran:



Exemple : insertion dans la relation RESULTAT des 10 premières valeurs de 0 à 9

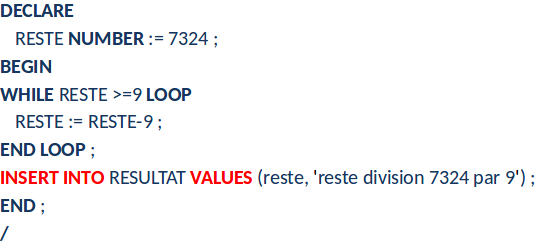


Les Branchements

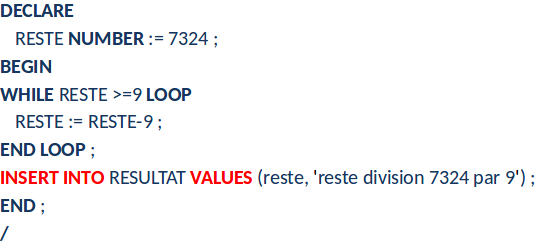
Le EXIT WHEN condition

Le GOTO : GOTO <étiquette>; où <étiquette> est spécifié dans le bloc sous la forme <<étiquette>>

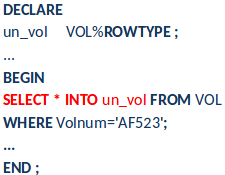
Exploitation des requêtes SQL

Les instructions SELECT, INSERT, DELETE, UPDATE peuvent être utilisées dans un bloc. Elles peuvent utiliser des variables du programme mais attention, les types doivent être compatibles et il faut utiliser des noms de variables différents. Il est possible d’affecter le retour d’une requête qui contient une seule valeur dans une variable avec SELECT …INTO. Pour les requêtes qui retournent plusieurs tuples il faut utiliser les curseurs. 

Exemple :sauvegarde dans la relation RESULTAT le contenu de la division de 7324 par 9



Exemple : récupération du nombre de vols stockés dans la base

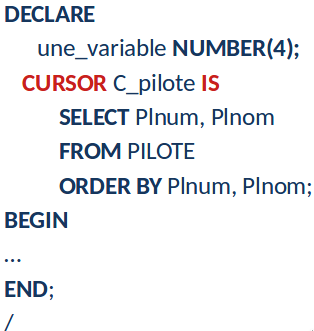


Exemple : récupération d’un tuple de la relation VOL de la base

Les curseurs

Rôle : établir la transition entre l’univers BD et celui des langages procéduraux classiques. pouvoir manipuler un à un tous les tuples retournés par une requête.

Un curseur est défini dans la partie déclarative d’un bloc PL/SQL par une requête d’interrogation en SQL (sa structure correspond aux attributs du SELECT), en suivant la syntaxe suivante: 



Exemple:

Gestion des curseurs

Dans le corps du bloc entre BEGIN et END

OPEN<nom\_curseur>; → exécute la requête de définition du curseur et alloue la place mémoire nécessaire; le curseur peut alors être perçu comme une suite d’enregistrements

CLOSE<nom\_curseur> → désactive le curseur et libère la place mémoire; le curseur est alors percu comme un ensemble indéfini.

FETCH<nom\_curseur> INTO <liste\_variables> ; → ramène le prochain enregistrement du curseur et renseigne les différentes variables réceptrices.

Attributs des curseurs

Il s’agit de propriétés booléennes prédéfinies des curseurs

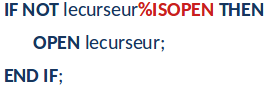
<nom\_curseurs>%NOTFOUND est à vrai si l’ordre FETCH ne retourne aucun enregistrement

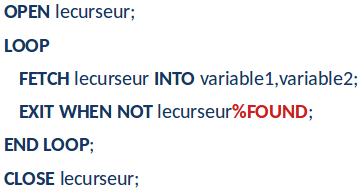
<nom\_curseur>%FOUND est à vrai si l’ordre FETCH retourne un enregistrement

<nom\_curseur>%ISOPEN est à vrai si le curseur est ouvert

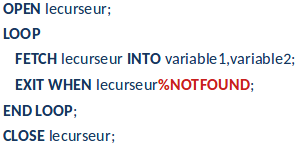
<nom\_curseur>%ROWCOUNT retourne le nombre de tuples qui ont été accédés via le curseur (0 avant le 1er fetch, puis 1, puis 2 …)

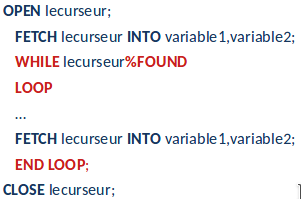
Exemple :



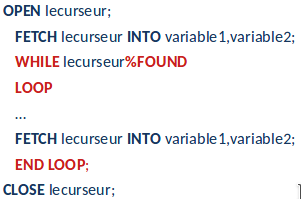
Exemple: %ISOPEN 

Exemple %FOUND

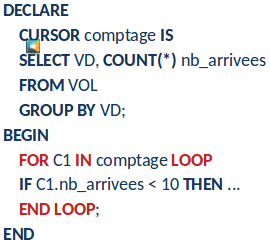


Exemple %NOTFOUND

Exemple dans un while :



Si l’ordre SELECT de définition du curseur comporte un calcul (horizontal ou vertical), il faut attribuer un alias au calcul pour pouvoir le manipuler ultérieurement. Dans l’exemple, le nombre de vol desservant chaque ville peut alors être manipulé par comptage.nb\_arrivees



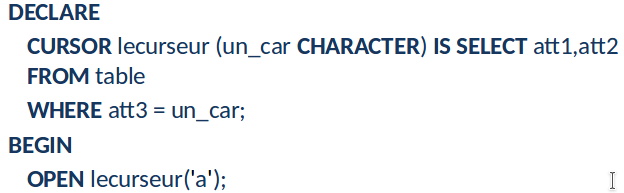
Il est possible de laisser le système

gérer les curseurs sans utiliser OPEN,

FETCH, CLOSE ni de déclaration de variable



Exemple :



Il est possible de passer un paramètre à un curseur

Les exceptions

Les exceptions permettent de contrôler des erreurs d’exécution; affichage de l’erreur ou traitement de l’erreur. Il existe 2 types d’exceptions :

* Exceptions définies par l’utilisateur dans la partie déclarative du bloc, elles sont déclenchées dans le corps du bloc, si une condition est remplie, par : 
* Exceptions prédéfinies, gérées par ORACLE, correspondant à des erreurs internes.

Les exceptions prédéfinies

Quelques exemples :

* NO\_DATA\_FOUND : déclenchée si une requête ne rend aucun résultat;
* ZERO\_DIVIDE : déclenchée s’il y a tentative de division par 0
* DUP\_VAL\_ON\_INDEX : déclenchée lors d’une tentative d’insertion d’une valeur dupliquée pour un attribut sur lequel est défini un index primaire.;
* INVALID\_NUMBER : déclenchée si une incompatibilité pour un type numérique est détectée.
* INVALID\_CURSOR déclenchée par exemple dans le cas d’accès à un curseur non ouvert.

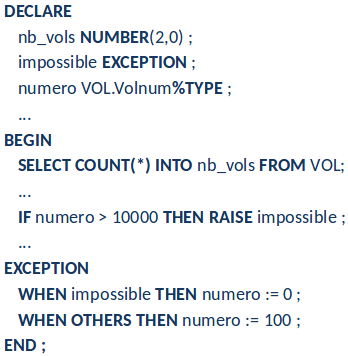
Les exceptions

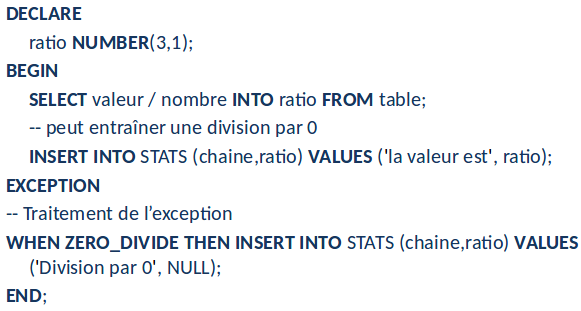
Le traitement des exceptions se fait dans la partie EXCEPTION du bloc PL/SQL par :

WHEN <nom\_exception> THEN [BEGIN] <liste\_instructions> [END] ;

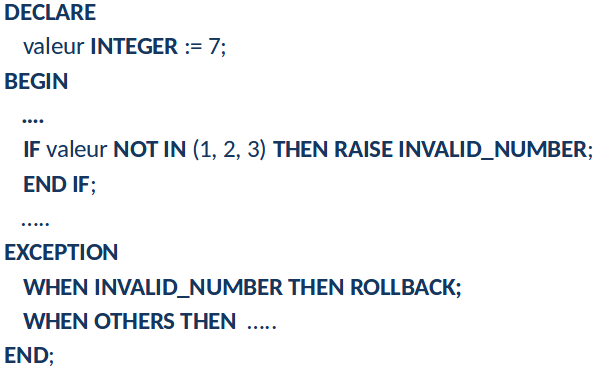
ou

WHEN OTHERS THEN <liste\_instructions>





Exemple : Exemple :



Exemple :

Les modules stockés

Un module stocké est un programme rangé dans la base de données qui peut être ainsi ré-utilisable et partageable (autorisation). Ces programmes peuvent être appelés à tout moment par un client et seront exécutés sur le serveur. Il est possible de définir des procédures ou des fonctions en PL/SQL.

| Les procédures | Les fonctions |
| --- | --- |
|  |  |

Entrées-sorties

Pour pouvoir afficher du texte à l’écran, utilisation du package DBMS\_OUTPUT → au niveau du prompt SQLPlus, exécuter l’instruction suivante : SET SERVEUROUTPUT ON

L’instruction pour afficher du texte ou le contenu d’une variable est : DBMS\_OUTPUT.PUT\_LINE(‘texte’ | | variable) ; ou | | est le caractère permettant la concaténation de chaînes.

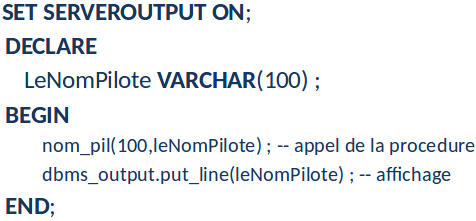
D’autres fonctions disponibles pour l’affichage:

dbms\_output.enable (autorise l’affichage)

dbms\_output.disable (interdit l’affichage)

dbms\_output.put\_line (affiche la chaîne et passe à la ligne)

dbms\_output.new\_line (passe à la ligne)



Dans le code: