Introduction au PL/SQL

Objectifs

- Maîtriser la syntaxe de PL/SQL
- Écrire des blocs et des packages

Syntaxe de PL/SQL

Objectifs

Maîtriser

- Structure d'un programme PL/SQL
- Les instructions en PL/SQL
- Les curseurs
- Les attributs
- Les exceptions

Syntaxe de PL/SQL

- Définition de PL/SQL
  - Langage procédural ➔Structures de programmation standard :
    - L'affectation.
    - Déclaration de variables et de constantes.
    - Définition de types abstraits;
    - Utilisation des structures de contrôle :
      - Structure conditionnelle.
      - Structure itérative.
    - Déclaration de procédure et de fonction.
  - Extension de SQL
    - Intègre directement les instructions de LMD.

Syntaxe de PL/SQL

- Structure d'un programme PL/SQL

  Un programme PL/SQL est composé de blocs imbriqués, chacun bloc est composé de sections :

  – Section de déclaration (optionnelle)
  – Section d'exécution (obligatoire)
  – Section d'exception (optionnelle)

Syntaxe de PL/SQL

- Sections d'un programme PL/SQL

Section déclaration
Section exécution
Section exception

Note. Un bloc PL/SQL peut être anonyme ou nommé telle que les procédures et les fonctions

Syntaxe de PL/SQL

Exemple

On suppose construite la table suivante :

Test(numéro, date_entrée)

```
Declare                                              S.D.
        max_record    Constant  number:=100;
        i    number :=1;
Begin                                                S.E.
    For  i  in 1 .. max_record    loop
         Insert   into   test
         Values(i , current_date);
    End loop;
End;
```

Bloc PL/SQL

Syntaxe de PL/SQL

Description

– Le bloc est anonyme (ni procédure, ni fonction).

– Il comporte une section de déclaration et une section d'exécution:

- La section de déclaration (declare)
  - Déclaration d'une constante de type number,
  - Déclaration d'une variable de type number.
- La section d'exécution (begin et end;)
  - Instruction itérative: for
  - Instruction sql : insert.

Syntaxe de PL/SQL

- **Section de déclaration**

Déclaration des variables et des constantes qui seront Utilisées dans la section d'exécution.

- La déclaration d'une variable :
  - nom,
  - type,
  - et peut avoir une valeur initiale,
  - et une terminaison par un point virgule.

La syntaxe de déclaration d'une variable

Nom_variable type [:=valeur_initiale* | *Default valeur_initiale];

Note. Par défaut, Les variables PL/SQL sont initialisées à NULL.

Syntaxe de PL/SQL

- **Section de déclaration (suite)**
  - La déclaration de constante :

    -nom,

    -type,

    -et une valeur initiale.

  La syntaxe de déclaration d'une constante :

  Nom_constante Constant type_constante := valeur_initiale ;

Syntaxe de PL/SQL

Exemples

- Date_naissance Date ;
- Salaire number (8,2) :=8000,00 ;
- Max_nombre Constant number :=100 ;

Syntaxe de PL/SQL

- Les types de données

  -Types SQL :Number, Varchar2, varchar, char, date

  -le type colonne : nom_table.champ%type

  -le type ligne (tuples): nom_table%rowtype

  -Type défini par l'utilisateur

  -Sous-type

Syntaxe de PL/SQL

- Variables de type colonne

  Syntaxe

  Nom_variable nom_table.colonne%type ;

  Exemple

  vsalaire employe.salaire%type ;

  Remarque

  Le type de la variable dépend de celui de la colonne de la table.

Syntaxe de PL/SQL

- Variables de type ligne

  Syntaxe

  Nom_variable nom_table%Rowtype ;

  Exemple1

  ligne_etudiant Etudiant%rowtype ;

  ligne_etudiant contient tous les champs de la table étudiant :

  - ligne_etudiant.nom :='Ali' ;
  - ligne_etudiant.année :=1 ;

Syntaxe de PL/SQL

- Variables de type ligne(suite)

  Exemple2

  vemp emp%rowtype

  ➔ vemp contient tous les champs de la table emp :

  vemp.salaire :=9000 ,00 ;

  vemp.nom='Ali';

  Remarque

  Une variable de type ligne dépend de la structure de la table de base.

Syntaxe de PL/SQL

- Commentaires
  - Commentaire ligne

    -- Ceci est un commentaire ligne
  - Commentaire plusieurs lignes

    /* Ceci est un commentaire

    sur

    (+) lignes */

Syntaxe de PL/SQL

- Section d'exécution
  - Section obligatoire délimitée par begin et end
  - Comporte l'ensemble des instructions à exécuter

  Les instructions incluent :
    - Instruction d'affectation.
    - Instruction NULL.
    - Instruction LMD.
    - Structures de contrôles.
    - Bloc PL/SQL.

Syntaxe de PL/SQL

- **Instructions d'affectation**

  Syntaxe

  nom_variable | *nom_constante* := *expression ;*

  Buts

  - Affecter la valeur d'une expression à une variable (section d'exécution)
  - Initialiser une variable (section de déclaration)
  - Affecter la valeur d'une expression à une constante (section de déclaration)

  Exemples :

  i :=i+1 ;

  max_nbre constant number :=100;

Syntaxe de PL/SQL

- **Instruction NULL**

  Syntaxe NULL **;**

  But Indiquer qu'aucune action ne doit être exécutée.

  Exemple

```
If  i<10  then
      i :=i+1 ;
Else
       NULL;
End if;
```

Syntaxe de PL/SQL

- Structures de contrôle

  Les structures conditionnelles

  - if
  - case

  Les structures itératives

  - Loop…end loop
  - While…end loop
  - for…end loop

Syntaxe de PL/SQL

- La structure if

  Syntaxe

```
If condition_1 then liste_instructions_1;
   [elsif condition_2 then liste_instructions_2]
        …
   [elsif condition_n then liste_instruction_n]
  [else      liste_instructions ]
end If;
```

Syntaxe de PL/SQL

- La structure if (exemple)

```
If ancienneté > 10 then
        Prime :=salaire*1.1;
elsif ancienneté > 5 then
        Prime :=salaire*0.5;
else
        Prime :=salaire*0.1;
end If;
```

Syntaxe de PL/SQL

- La structure case (choix multiple)

  Syntaxe 1

  ```
  Case expr
  when val1 then liste_instr1
  when val2 then liste_instr2
  …
  [else liste_instr]
  end case;
  ```

Syntaxe de PL/SQL

- La structure case(exemple)

```
Case a
when 1 then dbms_output.put_line('a=1');
when 1+2 then
dbms_output.put_line('a=3');
else dbms_output.put_line('a='||a);
end case;
```

Syntaxe de PL/SQL

- La structure case (choix multiple)

  Syntaxe 2

```
Case
when condition1 then liste_instr1
when condition2  then liste_instr2
…
[else liste_instr]
end case;
```

Syntaxe de PL/SQL

- La structure case(exemple)

```
Case
when a<=10 then
dbms_output.put_line('nombre petit');
when a > 10 then
dbms_output.put_line('nombre grand');
else dbms_output.put_line('a est inconnue');
end case;
```

Syntaxe de PL/SQL

- La structure loop..end loop

  Syntaxe

  ```
  Loop
          Liste_instrs
  End loop;
  ```

  But

  Exécuter plusieurs fois la liste d'instrs.

  Remarque

  Ces instructions seront exécutées de manière infinie.

  ➔Pour sortir de cette boucle on fait appel à l'instruction **EXIT**.

Syntaxe de PL/SQL

- L'instruction Exit

  Syntaxe

  Exit [when condition] ;

  condition à satisfaire pour sortir de la boucle encadrante

  But

  Quitter la boucle qui l'encadre avec ou sans condition.

  Exemple

```
Loop
    i:=i+1;
    Exit when i > 100;
End loop;
```

Syntaxe de PL/SQL

- La structure while… loop

  Syntaxe

  while *condition* ***loop***

  liste_instructions;

  end loop;

Syntaxe de PL/SQL

- La structure while… loop (exemple)

```
begin
i:=1 ;
While i<10 loop
    Insert into exemplaires(num_livre,num_invt)
     Values(1,i);
     i:=i+1;
end loop;
End;
```

Syntaxe de PL/SQL

- Instruction For…Loop

Variable déclarée

Syntaxe

For compteur in [reverse] Binf..Bsup loop

liste_instructions

end loop;

Syntaxe de PL/SQL

- Instruction For…Loop (exemple)

```
Declare
        i  int;
begin
        For i in 3..10 loop
            Insert into inventaire(num_livre,num_invt)
           Values(1,i);
        End loop;
end;
```

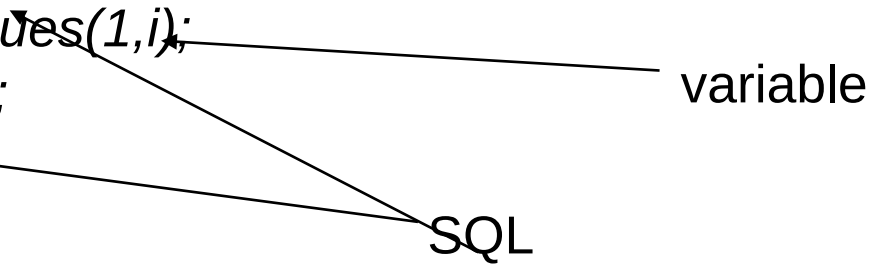
Syntaxe de PL/SQL

- **Les instructions SQL (LMD)**

  PL/SQL intègre les instructions SQL. L'exemple suivant donne une idée sur l'utilisation des instructions SQL dans un programme PL/SQL.

```
Declare
   i int ;
Begin
      For i in 3..10 loop
            Insert into inventaire(num_livre,num_invt)
            Values(1,i);
      End loop;
      commit;

end;
```



Syntaxe de PL/SQL

- L'instruction Select

  Syntaxe

  Select liste_colonnes_exprs
  into liste_variables | variable
  from liste_tables …
  avec
  liste_colonnes_exprs = liste_variables.

  But : Affectation des valeurs de liste_colonnes_exprs dans les variables de liste_variables ou dans les champs de la variable

Syntaxe de PL/SQL

- L'instruction Select(exemple 1)

```
declare
    salaire_max number;
begin
    Select max(salaire)
    Into  salaire_max
    From  emp
    Where   nserv='1';
    Dbms_output.put_line('salaire maximal est' || salaire_max);
End;
```

Syntaxe de PL/SQL

- L'instruction Select(exemple 2)

```
declare
    vemp emp%rowtype;
begin
    Select *
    Into  vemp
    From  emp
    Where   nemp='1';
    Dbms_output.put_line('sal :'|| vemp.salaire || '  '...);
End;
```

Syntaxe de PL/SQL

- L'instruction Select(exemple 3)

```
declare
    vsalaire number;
    vnom varchar2(20);
begin
    Select nom, salaire
    Into  vnom,vsalaire
    From  emp
    Where   nemp='1';
    Dbms_output.put_line('nom' || vnom || ' ' || 'salaire:' || vsalaire);
End;
```

Syntaxe de PL/SQL

- Procédure

  Syntaxe de déclaration

  Procedure *nom_procedure [(* liste_paramètres *)]* ***is|as***
  [liste_variables_constantes]
  begin
  liste_instructions ;
  end *[nom_procedure] ;*

  Chaque élément de liste_paramètres a la forme suivante :

  nom_paramètre [IN | OUT | IN OUT] type_données [{:= |default } valeur_initiale]

  Le IN | OUT | IN OUT s'appelle mode du paramètre.

Syntaxe de PL/SQL

- Procédure (suite)

  Le mode du paramètre

  –Le mode IN : le paramètre transmit une valeur à la procédure.

  ➔ le paramètre effectif peut être une constante, une expression ou une variable initialisée.

  –Le mode OUT : le paramètre reçoit une valeur de la procédure.

  ➔ paramètre effectif doit être une variable.

  –Le mode IN OUT : le paramètre transmet une valeur à la procédure et en reçoit une valeur.

  ➔Dans ce cas le paramètre effectif peut être une variable initialisée.

  Remarque

  Le mode in est le mode par défaut.

  Lors de la définition de la procédure on peut spécifier avec le mode IN une valeur par défaut ce qui n'est pas le cas avec OUT et IN OUT.

Syntaxe de PL/SQL

- Procédure (suite)
  - –Liste_variables_constantes :

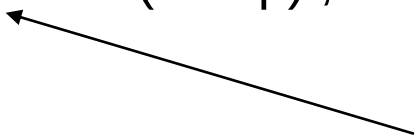
    définit les variables et les constantes locales de la procédure.
  - –Liste_instructions:

    Liste des instructions de la procédure

Syntaxe de PL/SQL

- Procédure (exemple 1)

```
Declare
    nemp   integer ;
Procedure  augment_salaire(num_emp  in int)  is
Begin
    Update emp
    Set  salaire = salaire *1.1
    Where  nemp=num_emp;
   commit;
End augment_salaire;
Begin  -- bloc anonyme
    nemp:=10;
   augment_salaire(nemp) ;
End ;
```



Appel à la procédure

Syntaxe de PL/SQL

- Procédure (exemple 2)

```
Declare
     nemp  integer ;
       vsal number;
Procedure  extraire_salaire(num_emp  in int :=1, sal out  number)
is
Begin
     select salaire into sal from emp
       Where  nemp=num_emp;
End extraire_salaire;
Begin  -- bloc anonyme
     nemp:=10;
    extraire_salaire(nemp, vsal);
  -- extraire_salaire(num_emp =>nemp ,  sal =>vsal);
    dbms_output.put_line(vsal);
End ;
```


Appel à la procédure

Syntaxe de PL/SQL

- Procédure (exemple 3)

```
Declare
--      nemp  integer ;
        vsal number;
Procedure  extraire_salaire(num_emp  in int :=1, sal out  number)
is
Begin
     select salaire into sal from emp
       Where  nemp=num_emp;
End augment_salaire;
Begin  -- bloc anonyme
--      nemp:=10;
    extraire_salaire(sal => vsal);
        dbms_output.put_line(vasl);
End ;
```

Appel à la procédure

Syntaxe de PL/SQL

- Fonction

  Syntaxe de déclaration

Function *nom_fonction [(liste_paramètres)]* ***return*** *type* ***is***

[liste_variables_constantes]

begin

liste_instructions ;

return valeur_retour ;

end [nom_fonction] ;

Syntaxe de PL/SQL

- Fonction (exemple)

Construire une fonction qui calcule la somme des salaires d'un service donné en argument.

```
Declare
        num_serv   integer;
Function  service_salaire(num_service in  integer) return number is
        Total_salaire  number ;
Begin
        select sum(salaire) into  Total_salaire
                From  emp
                Where  nserv=num_service;
         Return  Total_salaire;
End  service_salaire;
Begin --bloc anonyme
    num_serv :=10;
    dbms_output.put_line(service_salaire(num_serv));
End;
```

Appel à la fonction

Syntaxe de PL/SQL

- Bloc anonyme

  Syntaxe

  Declare

  Liste_variables_constantes_procedures_fonctions ;

  Begin

  Liste_instructions ;

  End ;

Syntaxe de PL/SQL

- Bloc anonyme (exemple)

Construire un bloc qui augmente de 10%, le salaire des employés de numéro 1 à 10. La valeur maximale sera déclarée comme une constante.

```
Declare
   max  constant   int :=10 ;
procedure  modif_salaire  is
begin
   update emp
   set salaire =salaire*1.1
   where nemp  between 1 and max ;
   commit;
end;
begin
          modif_salaire;
end;
```

Syntaxe de PL/SQL

- Procédure et fonction stockées

  Une procédure stockée est une procédure PL/SQL créée par la commande create et stockée dans la BD.

Syntaxe de création

```
Create [or replace] procedure  nomProcedure(…)
 is|as
 déclarations
 begin
 …
 end;
  /
```

Pour supprimer une procédure :

```
Drop procedure nomProcedure;
```

Pour compiler une procédure

```
Alter procedure compile;
```

Syntaxe de PL/SQL

- Procédure et fonction stockées(suite)

  Une fonction stockée est une fonction PL/SQL créée par la commande create et stockée dans la BD.

Syntaxe

create [or replace] function nomFonction(…) return type

is|as

…

Pour supprimer une fonction :

Drop function nomFunction;

Pour compiler une fonction

Alter function nomfonction;

Syntaxe de PL/SQL

- Procédure stockée(exemple)

  Calcul de la somme de salaire d'un service

```
Create procedure somSal(pserv number)
Is
    sal number;
Begin
    Select sum(salaire) into sal from emp
    Where nserv=pserv;
    Dbms_output.put_line('somme des salaires'||sal);
End;
```

Syntaxe de PL/SQL

- Fonction stockée(exemple)

```
Create Function somSal(pserv number) return number
is
    sal number;
Begin
    Select sum(salaire) into sal from emp
    Where nserv=pserv;
    return sal;
End;
```

Syntaxe de PL/SQL

- Les curseurs

  Définition

  –Un curseur est une structure qui décrit un ensemble de lignes issues d'une requête select

  –Un curseur est caractérisé par un nom

  –Un curseur permet la manipulation individuelle des lignes.

  Types de curseurs

  –Curseur implicite ==> si la requête select retourne au plus une ligne(into)

  –Curseur explicite ==> si la requête select retourne plusieurs lignes

  Les curseurs explicites doivent être déclarés

  Syntaxe

  Curso*r* *nom_curseur* ***is***

  Requête_séléction ;

Syntaxe de PL/SQL

- Curseur (exemple)

```
Cursor  C   is
Select num_etudiant  ,  prenom  ,  nom
From   etudiant
Where  annee='1' ;
```

Syntaxe de PL/SQL

- Manipulation d'un curseur
  - Ouverture d'un curseur

    Open *nomCurseur ;*

    Cette instruction réalise les tâches suivantes :

    1. Exécution de la requête associée au curseur.
    2. Récupération des lignes.
    3. Mettre (le pointeur) le curseur avant la première ligne.

  - Récupération des lignes du curseur

    Fetch *nom_curseur* ***into*** *liste_variables* |*variable ;*

    Cette instruction fait aussi passer le curseur(pointeur) à la ligne suivante.

  - Fermeture d'un curseur

    Close *nom_curseur ;*

Syntaxe de PL/SQL

- Manipulation d'un curseur(exemple)

```
Declare
  Cursor   c  is  --définition
    Select  salaire, nemp
    from   emp
    where  nserv=1;

  v_salaire    emp.salaire%type;
  v_nemp       emp.nemp%type ;
```

```
begin
 open  c ; --ouverture
 loop   --parcours de lignes
  fetch  c  into  v_salaire , v_nemp;
      exit  when  c%notfound;
      if  v_salaire< 5000  then
         update      emp
          set  salaire=salaire*1.1
          where nemp=v_nemp;
      end if;
end loop;
close c; --fermeture du curseur
end;
```

Syntaxe de PL/SQL

- Cursor for..loop

Ce type de curseur résume les instructions

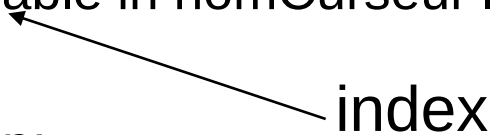
i. Open,
ii. fetch
iii. et close

Par

For variable in nomCurseur loop

…

End loop;



Syntaxe de PL/SQL

- Curseur for..loop (exemple)

```
Declare
   Cursor   c  is  --définition
      Select  salaire, nemp
      from   employe
      where  nserv=1;
begin
 for ligne in c loop
       if  ligne.salaire< 5000  then
            update      emp
            set  salaire=salaire*1.1
            where nemp=ligne.nemp;
       end if;
  end loop;
end;
```

Index non déclaré

Syntaxe de PL/SQL

- Curseur avec paramètres

  Un curseur peut avoir des paramètres

  Syntaxe

  Cursor *Nom_curseur (liste_paramètres)* ***is***

  Requêtes_ sélection ;

  Les paramètres effectifs sont transmis lors de l'ouverture du curseur :

  -Open nom_curseur (liste_paramètres) ;

  -For variable in nom_curseur (liste_paramètres) loop…

  Le mode des paramètres est in.

Syntaxe de PL/SQL

- Curseur avec paramètres (exemple)

```
Declare
   Cursor   c (pnserv int)  is  --définition
      Select  salaire, nemp
      from   employe
      where  nserv=pnserv;
begin
 for ligne in c(10) loop
       if  ligne.salaire< 5000  then
            update      emp
            set  salaire=salaire*1.1
            where nemp=ligne.nemp;
       end if;
  end loop;
end;
```

Paramètre effectif

Syntaxe de PL/SQL

- Curseur avec paramètres(exemple)

```
Declare
  Cursor   c ( pnserv int default 1) is
    Select  salaire, nemp
    from   employe
    where  nserv=pnserv;

 v_salaire    emp.salaire%type;
 v_nemp       emp.nemp%type ;
vnserv int;
```

```
Begin
vnserv:=10;
 open  c(vnserv) ; --ouverture
 loop   --parcours de lignes
   fetch  c  into  v_salaire , v_nemp;
       exit  when  c%notfound;
       if  v_salaire< 5000  then
          update   emp
           set  salaire=salaire*1.1
           where nemp=v_nemp;
       end if;
end loop;
close c; --fermeture du curseur
end;
```

Syntaxe de PL/SQL

- Curseur automatique

  -Curseur sans nom

  -La sélection associée est définie à l'exécution

  -La variable de parcours n'est pas déclarée.

  Syntaxe de création

  For ligne in (sélection) loop

  Liste_instrs

  End loop;

Requête seléction

Variable non déclarée

Syntaxe de PL/SQL

- Curseur variable

  Curseur qu'on peut ouvrir avec des requêtes sélections différentes.

Par exemple

- Open c for select * from emp;

  close c;

- Open c for select * from service;…

Syntaxe de PL/SQL

- Curseur variable (suite)

=>

- Définition d'un type de référence de curseurs

Syntaxe

type nomType is Ref cursor;

Exemple

type Tcv is ref cursor;

Syntaxe de PL/SQL

- Curseur variable (suite)

  -Déclaration d'une variable curseur

  Variable_curseur type_curseur

  -Exemple

  c Tcv;

  c accepte différentes requêtes SQL.

  -Ouverture d'un curseur variable

  Open variable_curseur for sélection;

  Note. Fetch et close

Syntaxe de PL/SQL

- Curseur variable (exemple)

  En utilisant (séparément) le même curseur:

  -Afficher le nom et le prénom des employés

  -Afficher le nombre d'employés et la moyenne des salaires

Syntaxe de PL/SQL

- Curseur variable (exemple)

```
Declare
        type tcv is ref cursor;
        c tcv;
        vnom emp.nom%type;
        vprénom emp.prénom%type;
        vmoy_sal number ;
        vcompt_emp number;
Begin
                                -->
```

Syntaxe de PL/SQL

- Curseur variable (exemple suite)

```
begin
   open c for select nom, prénom from emp;
   dbms_output.put_line('nom                    prénom');
   loop
         fetch c into vnom, vprénom;
         exit when c%notfound;
         dbms_output.put_line(vnom || ' ' ||  vprénom);
   end loop;
   close c;                                  -->
```

Syntaxe de PL/SQL

- Curseur variable (exemple fin)

```
open c for select count(nemp), avg(salaire) from
emp;
fetch c into vcompt_emp, vmoy_sal;
 dbms_output.put_line ('nbre emp  :'|| vcompt_emp);
 dbms_output.put_line('moyen salaire :'|| vmoy_sal );
 close c;

end; -- fin du bloc
```

Syntaxe de PL/SQL

- Curseur et MAJ des données

  Un curseur s'utilise pour MAJ des données

➔ Utilisation de

  - for update pour verrouiller les données à MAJ.
  - Current of pour référencer la ligne en cours dans le curseur

Syntaxe

cursor nomCursor [(liste_paramètres)] is req_sélection

for update [of liste_colonnes] [no wait| wait nbre_secondes]

Syntaxe de PL/SQL

- Curseur et MAJ des données(exemple 1)

```
declare
   cursor c is select nemp, nom, salaire, com from emp for
   update;
begin
    for enr in c loop
          if enr.com is null then
           update emp
           set com=salaire*0.1
           where current of c;
          end if;
     end loop;
end;
```

Syntaxe de PL/SQL

- Curseur et MAJ des données(exemple 2)

```
declare
   cursor c is select nemp, nom, salaire, com from emp for
   update;
begin
    for enr in c loop
          if enr.com is null then
           update emp
           set com=salaire*0.1
           where current of c;
          end if;
     end loop;
end;
```

Syntaxe de PL/SQL

- Package

Groupement de ressources : type, variable, procédure, fonction, curseur…

Définition d'un package :Spécification et implémentation

- Spécification

```
Create package nom_package is
        Liste de déclarations;
        Liste signatures de procédures_fonctions;
End;
- Exemple
Create package p is
        Curseur c is select * from emp;
        Procedure afficher(nemp number);
End;
```

Syntaxe de PL/SQL

- Package
- Implémentation

```
Create package body nom_package is
        Implémentation  procédures_fonctions;
End;
- Exemple
Create package body p is
        Procedure afficher(nemp number) is
        Begin
        …
        End;
End;
```

Syntaxe de PL/SQL

- Package

  - Accès aux éléments d'un package

  nom_package.ressource[(…)]

Syntaxe de PL/SQL

- Les attributs

Les objets PL/SQL possèdent des attributs qui renseignent sur leurs types ou leurs états :

- Type : le type de l'objet, i int; j i%type;
- Rowtype : la structure de l'objet table ou curseur, Enr emp%rowtype
- Found : propriété de curseur
  - -NULL à l'ouverture du curseur
  - -TRUE : au premier déplacement jusqu'au dernier déplacement.
  - -FALSE : si le pointeur du curseur est après le dernier élément du curseur
- Notfound vs Found
- Isopen : propriété de curseur
  - -TRUE : curseur ouvert
  - -FALSE : curseur fermé
- Rowcount : propriété de curseur, contient le nombre de lignes parcourues (fetchées).

Syntaxe de PL/SQL

- Les attributs (exemple 1)

Mise à jour des salaires des employés d'un service donné comme argument d'un curseur.

```
Declare
    Cursor  C (pnserv  int)  is  Select  *  from emp  Where  nserv=pnserv
    for update of salaire;
    nserv  em.nserv%type :=9;
    ligne_emp  emp%rowtype;
begin
   open  C(nserv);
   loop
    fetch  C  into ligne_emp;
      if C%found  then
         update  emp
         set  salaire=salaire*1.1
         where nemp=ligne_emp.nemp;
   Else
      exit;
   end if;
end loop;
commit;
close  C;
end;
```

Syntaxe de PL/SQL

- Les attributs (exemple 2)

Mise à jour des salaires des employés d'un service donné comme argument d'un curseur.

```
Declare
    Cursor  C (pnserv  int)  is  Select  *  from emp  Where  nserv=pnserv
    for update of salaire;
    ligne_emp  emp%rowtype;
Begin
  If  c%isopen then
  close C;
  open  C(1);
  loop
    fetch  C  into ligne_emp;
      if C%notfound  then
          Exit;
     end if
  update  emp
  Set salaire=salaire*1.1
   where current of c;
  end loop;
  commit;
  close  C;
  end;
```

Syntaxe de PL/SQL

- Les attributs (exemple 3)

Afficher le numéro, le nom et le rang(dans la req) des 20 premiers employés.

```
Declare
   Cursor  C  is       Select  *  from emp ;
      ligne_emp   emp%rowtype;
begin
  open  C;
  loop
      fetch  C  into  ligne_emp;
       Exit  when c%notfound;
      if  C%rowcount <=20  then
           DBMS.output.put_line(ligne_emp.nom ||'  '||ligne_emp.nempl ||'
           '||c%¨rowcount);
      Else exit;          End if;   End loop;
 Close  C;
End;
```

Syntaxe de PL/SQL

- Les enregistrements (record)

  -Définition d'un type enregistrement (déclaration ou package)

```
Type nomTypeEnregistrement is record (
      chmp1 type [not null] [:=expr],…
       );
```

  -Déclaration de variable

```
nomVariable nomTypeEnregistrement;
```

  - Manipulation

```
nomVariable.chmp1                    --individuelle
nomVariable1:= nomVariable2          --globale
```

Syntaxe de PL/SQL

- Les sous-types

Définition

-Un sous-type décrit un ensemble de valeurs d'un type dit type de base.

-Un sous-type définit une contrainte sur son type de base.

Définition d'un sous-type

Subtype nomSousType is tyepBase[(contrainte)] [not null];

Exemple

```
Subtype tsalaire is number(8,2) not null;      vsalaire tsalaire;
Subtype chaine is varchar2(100);               adresse chaine;
```

Syntaxe de PL/SQL

- Les packages

  Définition

  Objet de base de données qui regroupe des procédures, des fonctions, curseurs, type…

  La définition d'un package inclut

  -Définition de son interface(spécification) : éléments accessibles en dehors du package,

  -Définition du corps : implémentation des éléments de l'interface.

Syntaxe de PL/SQL

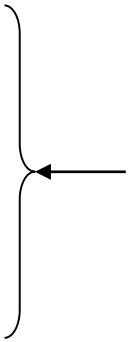
- Les packages : Définition de l'interface(spécification)

Syntaxe

Create [or replace] package nomPackage is|as
Définition de types_variables_constantes;
Définitions des curseurs;
Signatures des procédures_fonctions;
End [nomPackage];

Exemple

Create or replace package p_emp as
Type temp is record(emp.nemp%type…);
Vemp emp%rowtype;
Cursor c (pnserv int);
Procedure augSalaire(pnemp in int);
Function maxSal(pnserv int) return temp;
End p_emp;



Éléments accessible en dehors du package

Syntaxe de PL/SQL

- Les packages : Définition du corps (spécification)

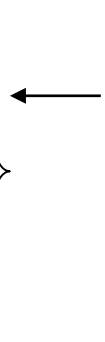
Syntaxe

```
Create [or replace] package body nomPackage is | as
    Définition des types;
    Définitions des curseurs;
    Définition des procédures_fonctions;
End [nomPackage];
```

Exemple

```
Create or replace package body p_emp as
    vemp temp;
    tauxIgr number;

    Procedure augSalaire(pnemp in int) is
    …
    End augSalaire;
    Function maxSal(pnserv int) return temp is
    …
    End p_emp;
```



Éléments accessibles en dehors du package

Syntaxe de PL/SQL

- Les exceptions

  -Une exception est une erreur qui survient lors de l'exécution d'un programme.

  -La gestion des exceptions sert à éviter les sorties inattendues du programme.

  -Les exceptions sont deux types

  Prédéfinies : VALUE_ERROR, INVALID_NUMBER…

  Définies : nomException Exception.

Syntaxe de PL/SQL

- Les exceptions (exemple)

```
Declare
    Vemp emp%rowtype;
Begin
    Select * into vemp from emp where nemp=10;
    dbms_output.put_Line(vemp.nemo || vemp.nom…);
Exceptions
    when no_data_found then
    dbms_output.put_Line(‘ emp inexistant’);
    when others then
    dbms_output.put_Line(‘Erreur inconnue’);
End ;
```

Syntaxe de PL/SQL

- Les exceptions(code et message d'exceptions)
  - La fonction SQLCODE retourne le code de l'exception
  - SQLERRM retourne le message associée à l'exception.
  - Ce message s'obtient par SQLERRM (SQLCODE)
  - Les exceptions prédéfinies ont sqlcode négatif sauf…
  - Les exceptions définies ont sqlcode=1 et

  Sqlerrm =user_defined exception…ora-000 normal…

Syntaxe de PL/SQL

- Les exceptions définies

  Pour déclarer une exception(section de déclaration):

  nomException Exception;

  Pour utiliser(soulever) une exception:

  Raise nomException;

Les exceptions prédéfinies sont automatiquement soulevées.

Syntaxe de PL/SQL

- Les exceptions (dénommer une exception)

Pour dénommer une exception

- Déclarer une exception

  Exc_emp exception;

- Utiliser le mot pragma Exception_init

  pragma Exception_init(Exc_emp, code-erreur);

Remarque

Les exceptions définies par pragma sont soulevées automatiquement.

Syntaxe de PL/SQL

- Les exceptions (exemple 1)

```
declare
   vemp emp%rowtype;
begin
   select * into vemp from emp where nemp=1;
   --affichage des informations
exception
   when no_data_found then
   dbms_output.put_line('Aucune ligne sélectionnée!');
   when others then
   dbms_output.put_line('Erreur inconnue!');
 end;
```

Syntaxe de PL/SQL

- Les exceptions (exemple 2)

```
declare
   vemp emp%rowtype;
begin
   select * into vemp from emp where nemp='a';
   --affichage des informations
exception
   when no_data_found then
   dbms_output.put_line('Aucune ligne sélectionnée!');
   when others then
   dbms_output.put_line('Erreur inconnue!');
 end;
```

Syntaxe de PL/SQL

- Les exceptions (exemple 3)

```
declare
        vnom varchar2(10);
begin
        select nom into vnom from emp where nemp=1;
        …
exception
        when no_data_found then
                dbms_output.put_line('Aucune ligne sélectionnée');
        when value_error then
                dbms_output.put_line('Erreur de valeurs!');
        when others then
                dbms_output.put_line('Erreur inconnue!');
end;
/
```

Syntaxe de PL/SQL

- Les exceptions (exemple 3)

```
declare
      exc1 exception;
 begin
 for enr in (select * from emp) loop
       if enr.com is null then
                  raise exc1;
      end if;
 end loop;
 exception
       when exc1 then
                  dbms_output.put_line('une commission non affectée');
       when others then
                  dbms_output.put_line('Erreur inconnue!');
 end;
/
```

Syntaxe de PL/SQL

- Les exceptions (exemple 4)

```
declare
    exc1 exception;
    pragma exception_init(exc1,-1722);
    exc2 exception;
    pragma exception_init(exc2,100);
vemp emp%rowtype;
begin
    select * into vemp from emp where nemp=100 | 'a'
exception
    when exc1 then
        dbms_output.put_line('Nombre non valide');
    when exc2 then
        dbms_output.put_line('Aucune ligne sélectionnée');
    when others then
        dbms_output.put_line('Erreur inconnue');
end;
```

Syntaxe de PL/SQL

- Les exceptions (exemple4)

```
declare
        exc1 exception;
        pragma exception_init(exc1,-1722);
        exc2 exception;
        pragma exception_init(exc2,100);
        exc3 exception;
        pragma exception_init(exc3,-1422);
        vemp emp%rowtype;
```

Syntaxe de PL/SQL

- Les exceptions (exemple4)

```
begin
 select * into vemp from emp where nserv=2;
      …
 exception
       when exc1 then
       dbms_output.put_line('Nombre non valide');
       when exc2 then
       dbms_output.put_line('Aucune ligne sélectionnée');
       when exc3 then
       dbms_output.put_line('plus d''une ligne!');
       when others then
       dbms_output.put_line('Erreur inconnue');
 end;
```