

# **Stockage de traitements avec Oracle et PL/SQL**

## **Packages Fonctions et Procédures**

# PL/SQL :

## procédures et fonctions (1/2)

- Une procédure est une unité de traitement qui peut contenir des commandes sql de manipulation de données et des instructions PL/SQL
- Une fonction est une procédure qui retourne une valeur

# procédures et fonctions (2/2)

```
CREATE [or replace] FUNCTION nom_fonction (liste des  
paramètres)
```

```
RETURN type
```

```
is
```

```
    bloc/sql
```

```
CREATE [or replace] PROCEDURE nom_procédure (liste  
des paramètres)
```

```
is
```

```
    bloc/sql
```

# PL/SQL : les curseurs (1/8)

- Un curseur est une zone de travail de l'environnement utilisateur qui contient des informations permettant l'exécution d'un ordre de sélection SQL.
- Un curseur est utilisé pour traiter une requête SQL (SELECT) ramenant plus de une ligne.

# PL/SQL : les curseurs (2/8)

- Deux types de curseur
  - Curseur implicite
  - Curseur explicite
- Quatre étapes\_
  - Déclaration du curseur
  - Ouverture du curseur
  - Traitement des lignes
  - Fermeture du curseur

# PL/SQL : les curseurs (3/8)

- Déclaration du curseur :

```
CURSOR nom_curseur [(nom_par  
type_par,...)] IS
```

```
SELECT commande;
```

- Ouverture du curseur :

```
OPEN nom_cur [ (param [, param] ...)];
```

# PL/SQL : les curseurs (4/8)

- Traitements des lignes générées :  
**FETCH** nom\_cur **INTO** { nom\_var  
[,nom\_var] ... | nom\_enreg };
- Utilisation d'une structure répétitive de type  
**LOOP .... END LOOP**
- Fermeture du curseur en fin d'utilisation :  
**CLOSE** nom\_cur ;

# PL/SQL : les curseurs (5/8)

Declare

```
CURSOR cur_client IS select * from client ;  
    Nom_record client%type ;
```

Begin

```
OPEN cur_client ;
```

```
LOOP
```

```
    FETCH cur_client into nom_record ;  
    Traitements éventuels
```

```
END LOOP ;
```

```
CLOSE cur_client ;
```

End ;

# PL/SQL : les curseurs (6/8)

- Il existe une une forme condensée d'utilisation du curseur
- Deux étapes :
  - Déclaration du curseur
  - Utilisation de la syntaxe **FOR** nom\_variable **IN** nom\_curseur
- La syntaxe **FOR ... IN** ouvre, effectue le fetch et ferme le curseur automatiquement

# PL/SQL : les curseurs (7/8)

Declare

```
CURSOR cur_client IS select * from client ;
```

Begin

```
FOR curseur IN cur_client
```

```
LOOP
```

```
    Traitements éventuels
```

```
END LOOP ;
```

End ;

# PL/SQL : les curseurs(8/8)

- Quatre pseudo attributs permettent de connaître le statut du curseur à un instant donné :
  - **%FOUND** : vrai si exécution correcte SQL
  - **%NOTFOUND** : vrai si exécution incorrecte SQL
  - **%ISOPEN** : vrai si curseur déjà ouvert
  - **%ROWCOUNT** : donne le ligne traitée par l'ordre FETCH (évolue à chaque FETCH)

# PL/SQL : Les package 1/10

- Un package permet de regrouper des modules de traitement qui ont un lien logique entre eux sous une seule entité.
- Les modules de traitement peuvent être :
  - procédures
  - fonctions
  - exceptions
  - variables
  - curseurs
  - constantes

# PL/SQL : Les package 2/10

- Un package est composé de deux parties créées et compilées séparément :
  - Une partie **déclaration** (spécification) qui contient la déclaration des modules qui sont accessibles de l'extérieur ( **PUBLIC** )
  - Une partie **body** (corps) qui contient les définitions des modules déclarés dans la partie déclaration ainsi que les déclarations et les définitions des modules de type **PRIVE**.

# PL/SQL : Les package 3/10

- On choisira de créer des packages dans le but :
  - d'une meilleur organisation des programmes (regroupement des objets par thème comme par exemple un package qui regroupera tous les objets chargés de gérer les clients et dans un autre package tous les objets chargés de gérer les commandes ...)
  - d'une approche objet

# PL/SQL : Les package 4/10

- **Partie déclaration**

```
CREATE [OR REPLACE] PACKAGE
```

```
  nom_package
```

```
[IS|AS]
```

```
{[déclaration de procédure ;]
```

```
|[déclaration de fonctions ;]
```

```
|[déclaration de variable ;]
```

```
|...}
```

# PL/SQL : Les package 5/10

- **Partie body**

```
CREATE [OR REPLACE] PACKAGE BODY  
  nom_package
```

```
[IS|AS]
```

```
{[déclaration et corps de procédure ;]
```

```
|[déclaration et corps de fonctions ;]
```

```
|[déclaration de variable ;]
```

```
|...}
```

```
END nom_package
```

# PL/SQL : Les package 6/10

- **Exemple**

```
create or replace package exemple1
```

```
is
```

```
-- déclaration d'une variable globale au package--
```

```
vglobal NUMBER :=0 ;
```

```
procedure test(pnom in client.nom%type);
```

```
function moyenne return number;
```

```
end exemple1;
```

```
/
```

# PL/SQL : Les package 7/10

- **Exemple**

```
CREATE OR REPLACE PACKAGE BODY exemple1
AS
  PROCEDURE test(pnom in client.nom%type)
  IS
  BEGIN
  END test;
  FUNCTION moyenne RETURN NUMBER
  IS
  BEGIN
  RETURN(1);
  END moyenne;
END exemple1;
/
```

# PL/SQL : Les package 8/10

- -- initialise la variable vglobal à 500--
- **SQL>exec exemple1.vglobal:=500**
- -- déclaration de la variable moyenne de type number--
- **SQL>VARIABLE moyenne NUMBER**
- -- appel de la fonction moyenne--
- **EXECUTE :moyenne :=  
exemple1.moyenne**
- -- affichage du résultat de la fonction--
- **Print moyenne**

# PL/SQL : Les package 9/10

- **Pour modifier la partie spécification ou la partie corps d'un package, il suffit de modifier le texte source correspondant et d'exécuter l'un des ordres suivants :**

**REPLACE PACKAGE nom\_package**

**REPLACE PACKAGE BODY nom\_package**

# PL/SQL : Les package 10/10

- **Pour supprimer un package entier**  
**DROP PACKAGE nom\_package**
- **Pour supprimer la partie BODY d'un package**  
**DROP PACKAGE BODY nom\_package**