SMI-S5

SGBD ORACLE : Procédures & Fonctions

Pr. Issam QAFFOU

Laboratoire Ingénierie des Systèmes d'Information
Département d'Informatique
FSSM-UCA

Introduction

- Une procédure stockée ou une fonction est un bloc stocké dans la base de données.
- Ce bloc peut être appelé par n'importe quel autre bloc PL/SQL que ce soit anonyme ou stocké aussi.
- On peut appeler les procédures et les fonctions par « sous programmes ».
- Une fonction ou une procédure stockée peut être appelée de n'importe où: SQL*PLUS, Oracle Forms & Report ou de l'intérieur d'un autre sous-programme.

Introduction

- Vous pouvez partitionner la partie logique de votre application en utilisant les sous-programmes.
- Vous mettez toute la logique demandée pour l'accès aux données dans la base de données.
- Vous mettez toutes les activités client dans la couche client.
- La différence entre une procédure et une fonction:
  - Une procédure se comporte comme une commande SQL ou PL/SQL (commit par exemple).
  - Une fonction est une partie d'une expression qui a une valeur (substr par exemple). Elle accepte des données et renvoie une valeur.

Création (Procédure stockée)

- Syntaxe:

```
CREATE [OR REPLACE] PROCEDURE <nom_procédure>
        [(argument [IN | OUT | IN OUT] type [DEFAULT valeur]
    . . .
    argument [IN | OUT | IN OUT] type)]
IS | AS
    <partie_déclaration>
BEGIN
    <corps_procédure>
[EXCEPTION
    <gestion d'exception>]
END;
```

- On utilise REPLACE pour remplacer une procédure qui existe déjà portant le même non que la notre.

Création (Procédure stockée)

Liste d'arguments:

- Ce sont les arguments ou paramètres de la procédure. Vous pouvez en déclarer autant que vous voulez.
- On déclare un argument suivi de son mode qui peut être:
  - IN: pour les paramètres entrés seulement. Ne sera pas retourné dans le résultat.
  - OUT: pour les paramètres de sortie seulement.
  - IN OUT: les deux; les paramètres peuvent être d'entrée et retournés en résultat.
- Vous suivez le mode par le type de l'argument (number, date, etc.).

Création (Procédure stockée)

- Paramètres actuels et formels:
  - Les paramètres passés (fournis) lors de l'appel d'une procédure s'appellent: paramètres actuels.
  - Les paramètres utilisés dans la définition de la procédure s'appellent: paramètres formels.
- Lors de l'exécution, les paramètres actuels sont copiés dans les paramètres formels pour être utilisés dans le corps de la procédure.
- Tout argument ayant le mode OUT ou IN OUT est alors copié du paramètre formel vers le paramètre actuel quand la procédure est terminée.
- **N.B**: Tout ce qui est déclaré au sein d'une procédure lui est strictement propre.

Création (Procédure stockée)

- Exemple: pour éviter d'appeler insert into autant de fois que de lignes qui existent dans une table pour la remplir, vous pouvez utiliser une procédure stockée comme suit:

```
CREATE OR REPLACE PROCEDURE debug
                  (     p_program_name IN VARCHAR2
                  ,     p_text         IN VARCHAR2)
IS
BEGIN
      INSERT INTO debug
      (     program_name
      ,     text
      ,     logged_at ) VALUES
      (     p_program_name
      ,     p_text
      ,     SYSDATE);
END;
```

Création (Procédure stockée)

- Exemple (suite): il suffit d'appeler et exécuter cette procédure là vous voulez par son nom et ses paramètres actuels.
  - **Syntaxe**:

    debug('Mon programme', 'Mettre du texte …');
  - **Exécution**:

    EXEC debug('Mon programme', 'Mettre du texte …');

- La commande SQL*PLUS EXEC ne fait qu'encadrer le code PL/SQL (dans ce cas l'appel de la procédure) avec un BEGIN et END pour la rendre un bloc anonyme.

Création (Fonctions)

- La création d'une fonction est quasiment identique à celle d'une procédure stockée, avec une simple différence c'est qu'une fonction doit avoir une valeur de retour.
- Donc on l'affecte à une variable, par exemple:

```
Nom_cours := GetNom(p_ccode);
```

- La fonction GetNom prend un seul paramètre (le code du cours) et retourne en résultat le nom du cours correspondant.
- Une fonction peut être utilisée n'importe où avec une expression: affectation, IF, traitement itératif, etc.

Création (Fonctions)

- Syntaxe: elle a la même syntaxe qu'une procédure mais on ajoute un return

```
CREATE [OR REPLACE] Function <nom_fonction>
        [(argument [IN | OUT | IN OUT] type [DEFAULT valeur]
    . . .
    argument [IN | OUT | IN OUT] type)]
    RETURN <type_valeur_retournée>
IS | AS
    <partie_déclaration>
BEGIN
    <corps_fonction>
[EXCEPTION
    <gestion d'exception>]
Return <valeur>;
END;
```

Création (Fonctions)

- Exemple: une fonction qui calcule le montant de frais non payé restant pour un étudiant.

```
create or replace function getMontantRestant(prollno number)
return number
is
    v_frais number(5);
    v_mntpaye number(5);
begin
-- obtenir le montant total payé par un étudiant
    select sum(total) into v_mntpaye
    from payement
    where rollno = prollno;
-- obtenir les frais des cours dans lesquels
--l'étudiant s'est adhéré
    select frais into v_frais
    from cours
    where ccode = ( select ccode from batches
    where bcode in
                ( select bcode from Etudiant
                where rollno = prollno)
    );
-- retourner la différence
    return v_frais - v_mntpaye;
end;
/
```

Création (Fonctions)

- Exemple (suite): cette fonction peut être appelée
  - dans un block pl/sql comme suit:

```
begin
        dbms_output.put_line(getMontantRestant(10));
end;
```

  - dans une requête SQL comme toute fonction standard:

```
select rollno, getMontantRestant(rollno)
from Etudiant;
```