Secteur Tertiaire Informatique
Filière étude - développement

Activité « Développer la persistance des données »

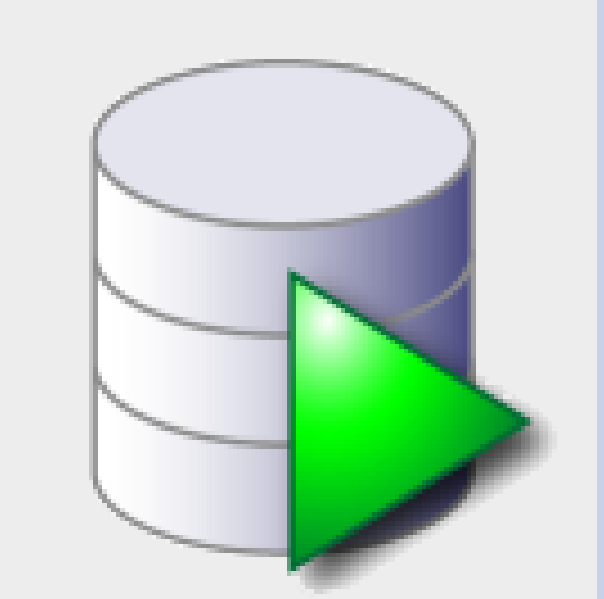
# PL/SQL – Mise en œuvre des procédures, des fonctions et des packages

Accueil

**Apprentissage**

Période en entreprise

Evaluation



Code barre

# SOMMAIRE

# I DEFINITION : PROCEDURE, FONCTION ET PACKAGE

Une **procédure** est une unité de traitement qui peut contenir des commandes SQL de manipulation de données (LMD), des instructions PL/SQL, des variables, des constantes, des curseurs et un gestionnaire d'erreurs.

Une **fonction** est une procédure qui retourne une valeur.

Les procédures et les fonctions sont créées comme des objets de la base appartenant à un utilisateur. Elles sont soumises donc à tous les mécanismes de sécurité et de confidentialité. Elles sont accessibles à travers les outils d'Oracle, tels que SQL*Plus, Sql Developer, etc. Il en résulte que plusieurs applications (outils) peuvent faire appel à la même procédure ou fonction.

Un **package** est une « encapsulation » de plusieurs procédures, fonctions, curseurs, variables au sein d'une seule unité nommée. Les packages offrent plusieurs avantages par rapport aux procédures et aux fonctions standard.

# II LES PROCEDURES ET LES FONCTIONS

La notion de procédure a été conçue dans l'esprit de grouper un ensemble de commandes SQL avec les instructions procédurales. Avec cette combinaison, l'utilisateur peut ainsi résoudre des problèmes complexes tout en ayant une flexibilité et une aisance dans son développement.

Les procédures et les fonctions sont utilisées pour augmenter considérablement la productivité. Elles servent également à gérer la sécurité et l'intégrité des données et à augmenter les performances.

Du point de vue de la sécurité, l'utilisateur peut autoriser l'accès à certaines tables seulement à travers les procédures. Les usagers bénéficiant du privilège d'accès aux tables à travers les procédures ne possèdent aucune autorisation d'accès à ces mêmes tables en dehors du cadre de ces procédures. Au niveau de l'intégrité, les procédures développées et testées assurent la même fonctionnalité indépendamment de la partie appelante. Autrement dit, la recompilation d'une procédure en cas de correction n'exige pas la recompilation de l'ensemble du code de l'application.

Les performances sont assurées par les facteurs suivants :

- Réduction du trafic sur le réseau (soumission d'un bloc PL/SQL au moteur au lieu d'une commande SQL).
- Compilation des procédures cataloguées (le moteur ne recompile pas les procédures au moment de l'exécution).
- Exécution immédiate de la procédure si elle est dans la SGA (réduction des accès disque).

- Partage de l'exécution d'une procédure par plusieurs utilisateurs (notion de mémoire partagée).

Pour créer un objet procédural, vous devez disposer du privilège système CREATE PROCEDURE pour votre schéma ou du privilège système CREATE ANY PROCEDURE pour la création dans un autre schéma
Pour autoriser un autre schéma à exécuter une procédure de votre schéma, vous devez lui octroyer le privilège EXECUTE

```
GRANT EXECUTE ON ma_procedure TO autre_schéma
```

## II.1 CREATION D'UNE PROCEDURE OU D'UNE FONCTION

La commande qui permet de créer une procédure est la suivante :

```
CREATE [OR REPLACE] PROCEDURE
    [schéma].nom_procédure [(liste d'arguments)]
    {IS | AS}
    bloc PL/SQL
```

Celle qui permet de créer une fonction est la suivante :

```
CREATE [OR REPLACE] FUNCTION
    [schéma].nom_fonction [(liste d'arguments)]
    RETURN type
    {IS | AS}
     bloc PL/SQL
```

L'option OR REPLACE permet de spécifier au système le remplacement de la procédure ou de la fonction si elle existe déjà dans la base.

L'utilisateur peut précéder le nom de la procédure (fonction) par celui d'un schéma s'il n'est pas dans cet environnement, à condition d'avoir les privilèges de création de procédures dans ce schéma.

<liste d'arguments> est composée des arguments d'entrée, de sortie et d'entrée/sortie, séparés par une virgule selon le format suivant :

```
liste d'arguments := nom d'argument [IN |OUT | IN OUT] type;…
```

Le mot clé IN indique que la variable est passée en entrée.

Le mot clé OUT indique que la variable est renseignée par la procédure puis renvoyée à l'appelant.

Le mot clé IN OUT est une combinaison des deux modes précédents. La variable est passée en entrée, renseignée par la procédure puis renvoyée à l'appelant (équivalant au passage de paramètres par référence dans les langages de programmation).

Le mot clé RETURN permet de spécifier le type de la donnée de retour de la fonction.

Enfin, le type de données est l'un des types reconnus par Oracle.

Le <bloc PL/SQL> doit commencer par le mot clé BEGIN et se terminer par END. Il peut être composé d'une partie déclarative, d'un corps de la procédure et d'un gestionnaire d'erreurs.

Le nom d'une procédure ne doit pas comporter les caractères : ( - ' " ) []

La création d'une procédure peut se faire à l'aide de l'outil SQL*Plus, mais nous conseillons fortement au lecteur d'utiliser l'outil Oracle SQL Developer pour la rédaction de ces procédures, la sauvegarde dans la base de données, ainsi que la compilation et l'exécution de celle-ci.

**La dernière ligne de chaque procédure doit être composée d'un seul caractère '/' pour spécifier au système l'exécution de sa création**.

Le fait d'avoir le source d'une procédure sur un fichier permet de corriger facilement la procédure avec l'option OR REPLACE.

Exemples : Créer une procédure qui permet de baisser le prix d'un article et une fonction qui affecte un numéro au client ou à l'article (en utilisant des séquences).

```
CREATE PROCEDURE baisse_prix ( Id IN NUMBER, Taux IN NUMBER)
IS
BEGIN
    UPDATE article SET article.prixunit = article.priunit*( 1 + Taux)
    WHERE article.idarticle = Id;
EXCEPTION
WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR
          (-20010,'Article Invalide : ' ||TO_CHAR(Id));
END;

CREATE FUNCTION numero (Nature IN CHAR)
RETURN NUMBER
IS
valeur NUMBER;
BEGIN
   IF Nature = 'C' OR Nature = 'c' THEN
      SELECT seqcl.nextval INTO valeur FROM dual;
   ELSIF Nature = 'A' OR Nature = 'a' THEN
      SELECT seqar.nextval INTO valeur FROM dual;
   ELSE valeur := -1 ;
   END IF;
   RETURN(valeur);
END;
```

En cas d'erreur de compilation, l'utilisateur doit la corriger sur le fichier et la soumettre ensuite au moteur avec l'option OR REPLACE.

## II.2 MODIFICATION D'UNE PROCEDURE OU D'UNE FONCTION

Il arrive parfois que l'application évolue, entraînant ainsi une modification dans le schéma de la base (suppression ou modification de tables). Pour permettre aux procédures (fonctions) existantes de prendre en compte ces modifications, il faut les recompiler avec la commande suivante :

```
ALTER {FUNCTION | PROCEDURE} [schéma.]nom COMPILE;
```

Cette commande recompile uniquement les procédures cataloguées standard. Il faut avoir les privilèges nécessaires pour réaliser cette recompilation.

Exemples : Recompiler la procédure et la fonction créées précédemment.

```
ALTER PROCEDURE baisse_prix COMPILE;
ALTER FUNCTION numero COMPILE;
```

## II.3 SUPPRESSION D'UNE PROCEDURE OU D'UNE FONCTION

Comme tout objet manipulé par Oracle, les procédures et les fonctions peuvent être supprimées si le besoin est ressenti. Cette suppression est assurée par la commande suivante :

```
DROP {FUNCTION | PROCEDURE} [schéma.]nom;
```

## II.4 EXECUTION D'UNE PROCEDURE

Sous SQL*Plus :

```
EXECUTE <nom procédure> [( argument1, argument2, ...) ]
```

Dans un bloc PL/SQL :

```
BEGIN
…
<nom procédure> [( argument1, argument2, ...) ] ;
…
END;
```

## II.5 EXECUTION D'UNE FONCTION

Sous SQL*Plus :

```
SELECT  <nom fonction> FROM DUAL;
```

Dans un bloc PL/SQL :

```
BEGIN
…
x := <nom procédure> [( argument1, argument2, ...) ] ;
```

```
…
END;
```

### II.6 VISUALISATION DES ERREURS DE COMPILATION

```
SHOW ERRORS
```

Vous pouvez aussi utiliser la vue USER_ERRORS du dictionnaire.

```
SELECT * FROM user_errors;
```

# III LES PACKAGES

La notion de package permet d'encapsuler des procédures, des fonctions, des curseurs et des variables comme une unité dans la base de données. Elle apporte un certain nombre d'avantages par rapport aux procédures et aux fonctions cataloguées.
En effet, les packages offrent un meilleur moyen de structuration et d'organisation du processus de développement. Le mécanisme de gestion de privilèges devient plus facile par rapport aux procédures (fonctions) cataloguées. En effet, l'attribution de privilèges d'utilisation des composantes d'un package se fait par une seule commande.

Les packages offrent un meilleur mécanisme de gestion de la sécurité. L'utilisateur peut spécifier au cours de la création d'un package des composantes publiques et des composantes privées. La séparation des déclarations des composantes d'un package de leur corps permet une meilleure flexibilité au développeur pour établir rapidement une maquette.

Enfin les performances peuvent être améliorées en utilisant les packages plutôt que les procédures cataloguées. Le moteur charge en mémoire le package entier quand une de ses procédures est appelée. Une fois le package en mémoire, le moteur n'a plus besoin d'effectuer des lectures (I/O disque) pour exécuter les procédures de ce même package.

### III.1 CREATION D'UN PACKAGE

La création d'un package se fait en deux étapes :

- Création des spécifications du package
- Création du corps du package

Les spécifications d'un package consistent à déclarer les procédures, les fonctions, les constantes, les variables et les exceptions qui peuvent être accessibles par le public. En d'autres termes, il s'agit de la déclaration des objets de type PUBLIC du package.

```
CREATE [OR REPLACE] PACKAGE [schéma.]nom_package
    {IS | AS} spécification PL/SQL
```

spécification PL/SQL ::=

déclaration de variable |
déclaration d'enregistrement |
déclaration de curseur |
déclaration d'exception |
déclaration de table PL/SQL |
déclaration de fonction |
déclaration de procédure.|

Le corps d'un package définit les procédures (fonctions), les curseurs et les exceptions qui sont déclarés dans les spécifications de la procédure. Il peut également définir d'autres objets de même type non déclarés dans les spécifications. Ces objets sont alors privés et ne peuvent en aucun cas être accédés en dehors du corps du package. La commande qui permet de créer le corps d'une procédure est la suivante :

```
CREATE [OR REPLACE] PACKAGE BODY [schéma.]nom_package
  {IS | AS} corps PL/SQL
```

Où

corps PL/SQL : :=
déclaration de variable |
déclaration d'enregistrement |
corps de curseur |
déclaration d'exception |
déclaration de table PL/SQL |
corps de fonction |
corps de procédure...

Les noms de la spécification et du corps du package doivent être les mêmes.

Les objets déclarés dans les spécifications du package sont rendus publics et peuvent être accédés à l'intérieur et à l'extérieur du package. Ainsi, les variables, les curseurs et les exceptions publics peuvent être utilisés par des procédures et des commandes qui n'appartiennent pas au package, à condition d'avoir le privilège EXECUTE sur ce package. Les objets privés sont déclarés dans le corps du package et n'auront comme étendue que le corps package. Les variables d'une procédure ne sont accessibles que par la procédure elle-même.

Il est important de signaler l'étendue (durée de vie) des variables, constantes et curseurs entre les procédures cataloguées et les packages. Les variables appartenant à une procédure standard ou du package ont une durée de vie limitée à l'exécution de la procédure. Une fois la procédure terminée, les variables et les constantes sont perdues.

Par contre, les mêmes objets déclarés au niveau des spécifications d'un package ou de son corps persistent le long de la session et après le premier appel à ce package. Quand une session commence, les variables et les curseurs sont initialisés à la valeur nulle, à moins qu'une initialisation ait été explicitement effectuée sur ces objets. Pour réaliser cette initialisation explicite, un package peut contenir dans son corps du code

exécuté seulement lors du premier appel à ce package. Ce code constitue un bloc séparé par les mots clés BEGIN et END.

Il est possible de nommer plusieurs procédures de la même façon. Cette possibilité permet de définir une procédure à plusieurs points d'entrée (nombre d'arguments et type de données différents).

### III.2 MODIFICATION D'UN PACKAGE

La modification d'un package concerne sa version compilée. En d'autres termes, il est important de recompiler le package afin que le noyau tienne compte de l'évolution de la base et que l'on puisse modifier ainsi sa méthode d'accès et son plan d'exécution.

Cette recompilation se fait par la commande suivante :

```
ALTER PACKAGE [schéma.]package
  COMPILE [PACKAGE | BODY];
```

En pratique, il est recommandé de sauvegarder les sources des packages dans des fichiers pour les reprendre en vue d'une modification de leur contenu. En cas de modification des sources, l'utilisateur doit recréer le package avec l'option REPLACE pour remplacer l'existant.

Exemple : Recompiler le package ges_vendeur.

```
ALTER PACKAGE ges_vendeur COMPILE PACKAGE;
```

### III.3 SUPPRESSION D'UN PACKAGE

La suppression d'un package se fait par la commande DROP comme suit :

```
DROP PACKAGE [schéma.]package;
```

**Etablissement référent**
*Direction de l'ingénierie Neuilly*

**Equipe de conception**
*Groupe d'étude de la filière étude - développement*

***Remerciements :***



Date de mise à jour 31/07/2014
afpa © Date de dépôt légal juillet 14

**afpa / Direction de l'Ingénierie13 place du Générale de Gaulle / 93108 Montreuil Cedex**
**association nationale pour la formation professionnelle des adultes**
**Ministère des Affaires sociales du Travail et de la Solidarité**