Les Packages

Utilisation

Les packages permettent d'encapsuler des éléments de type PL/SQL dans une unité nommée dans la base de données. Un ou plusieurs objets des catégories suivantes peuvent donc être regroupés quand ils ont un lien entre eux : des Procédures, Fonctions, Variables, Constantes, Curseurs, Exceptions.
Les packages sont compilés lors de la création et sont stockés comme des objets de schéma normaux.
Ils apportent un certain nombre d'avantages par rapport aux procédures et aux fonctions isolées.

Avantages

Structuration de développement

Les packages offrent un meilleur moyen de structuration et d'organisation du processus de développement. Ils permettent une modularisation des applications : chaque module forme une unité et dispose d'interfaces clairement définies.
Le mécanisme de gestion de privilèges devient plus facile par rapport aux procédures et fonctions cataloguées. En effet, l'attribution de privilèges d'utilisation des composantes d'un package se fait par une seule commande portant sur l'ensemble du package.

Développement de composants

Le package offre un meilleur mécanisme de gestion de la sécurité. La séparation des déclarations des composants d'un package de leur corps permet une meilleure flexibilité au développeur.
Il est possible de définir des composants publics, c'est-à-dire visibles hors du package, et privés, c'est-à-dire visibles uniquement par les autres composants du package.
La notion de package favorise la réutilisation de code et permet la « surcharge » des procédures et fonctions (overloading), ce que nous verrons plus tard.

Gestion des performances

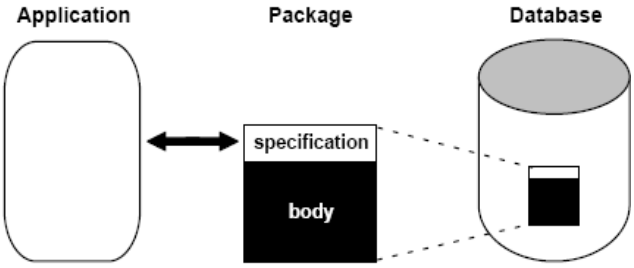
Les performances peuvent être améliorées en utilisant les packages plutôt que les procédures et fonctions cataloguées. Le moteur charge en mémoire le package entier quand une de ses procédures ou fonction est appelée pour toute la durée de la session. Une fois le package en mémoire, le moteur n'a plus besoin d'effectuer de lectures (I/O disque) pour exécuter les autres procédures ou fonctions de ce même package.

Constituants d'un package

Un package est constitué de deux parties distinctes :

- Les spécifications du package (`PACKAGE`)
- Le corps du package (`PACKAGE BODY`)

Les applications qui utiliseront un package le feront à travers les spécifications qui peuvent être vues comme l'interface du package.



Spécifications du package

Les spécifications d'un package contiennent les déclarations des procédures, des fonctions, des constantes, des variables, des types et des exceptions publiques. En d'autres termes, il s'agit de la déclaration des objets qui pourront être utilisés par des objets externes au package.
La création des spécifications d'un package se fait à l'aide de la commande suivante :

```
CREATE [OR REPLACE] PACKAGE package_name
       [AUTHID {CURRENT_USER | DEFINER}] --s'exécute sous le nom du créateur
   {IS | AS}                              -- ou de l'utilisateur connecté
   [PRAGMA SERIALLY_REUSABLE;]  -- le package ne reste pas en mémoire
   [collection_type_definition ...]
   [record_type_definition ...]
   [subtype_definition ...]
   [collection_declaration ...]
   [constant_declaration ...]
   [exception_declaration ...]
   [object_declaration ...]
   [record_declaration ...]     -- Attention :
   [variable_declaration ...]   -- les variables et les curseurs « publics »
   [cursor_spec ...]            -- sont valables pour toute la session (sont
   [function_spec ...]          -- utilisés par tous les sous-pgm)
   [procedure_spec ...]
   [call_spec ...]              -- appel à du java ou du C
   [PRAGMA RESTRICT_REFERENCES(assertions)...]    -- ne lit et/ou n'écrit pas
END [package_name];             -- dans la base de données et/ou les packages
```

Corps du package

Le corps d'un package implémente les éléments qui sont déclarés dans les spécifications du package. 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'un package est la suivante :

```
CREATE [OR REPLACE] PACKAGE BODY package_name {IS | AS}
   [PRAGMA SERIALLY_REUSABLE;]
   [collection_type_definition ...]
   [record_type_definition ...]
   [subtype_definition ...]
   [collection_declaration ...]
   [constant_declaration ...]
   [exception_declaration ...]
   [object_declaration ...]
   [record_declaration ...]
   [variable_declaration ...]
   [cursor_body ...]
   [function_spec ...]
   [procedure_spec ...]
   [call_spec ...]
BEGIN
   sequence_of_statements;
END [package_name];
```

L'identificateur de la spécification et celui du corps du package doivent être les mêmes, ce qui permet au système de les associer.

Droits d'accès et modules PL/SQL stockés

Pour pouvoir exécuter un module PL/SQL stocké il faut en être le propriétaire, ou disposer du droit EXECUTE sur cet objet (GRANT EXECUTE ON…TO…). Attention : Les droits ne peuvent être gérés à travers les rôles et doivent être donnés directement à l'utilisateur.

Si l'on dispose des droits d'exécution, il y a deux cas de figures définis lors de la compilation de la procédure, pour les droits de manipulation des objets utilisés par les modules PL/SQL :

- Si la clause AUTHID est définie avec DEFINER (valeur par défaut), le module est exécuté avec les droits de son propriétaire. Dans ce cas, les droits sont contrôlés lors de la compilation.
- Par contre, si elle est définie avec CURRENT USER, c'est avec les droits de l'utilisateur courant que sera exécuté le module.

La surcharge ou Overloading

PL/SQL permet à deux sous-programmes d'un même package de porter le même nom. Cette option est intéressante quand les sous-programmes doivent accepter un nombre de paramètres variables ou encore des types de données différents.
Par exemple, le package suivant comporte deux procédures portant le même nom mais acceptant des paramètres de types différents et traitant ceux-ci de façon appropriée :

```
CREATE PACKAGE pkg_employe IS
  FUNCTION NbEmployésDuDépartement (i_dep_no  IN NUMBER) RETURN INTEGER;
  FUNCTION NbEmployésDuDépartement (i_dep_nom IN VARCHAR2) RETURN INTEGER;
END pkg_employe;
/
CREATE PACKAGE BODY pkg_employe IS
  FUNCTION NbEmployésDuDépartement (i_dep_no IN NUMBER) RETURN INTEGER IS
     v_nb INTEGER;
  BEGIN
     SELECT COUNT(*) INTO v_nb FROM exe_employe WHERE emp_dep_no = i_dep_no;
     RETURN v_nb;
  END NbEmployésDuDépartement;

  FUNCTION NbEmployésDuDépartement (i_dep_nom IN VARCHAR2) RETURN INTEGER IS
     v_nb INTEGER;
  BEGIN
     SELECT COUNT(*) INTO v_nb
        FROM exe_employe JOIN exe_dept on dep_no = emp_dep_no
        WHERE UPPER(dep_nom) = UPPER(i_dep_nom);
     RETURN v_nb;
  END NbEmployésDuDépartement;
END pkg_employe;
```

Pour utiliser des objets d'un package, il faut préfixer leur nom du nom du package :

```
BEGIN
   dbms_output.put_line( pkg_employe.NbEmployésDuDépartement(3) );
   dbms_output.put_line( pkg_employe.NbEmployésDuDépartement('RH') );
END;
```

Visibilités des objets

Les objets déclarés dans les spécifications du package sont 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 par le corps du package et n'auront comme étendue que le corps du 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 après le premier appel à ce package. On va donc trouver trois sortes de variables dans un package :

- Les variables publiques : elles sont déclarées dans les spécifications du package. Elles sont visibles par tous les objets du package, ainsi que ceux externes au package, tout en respectant la gestion des droits classiques (GRANT)
- Les variables privées, globales : elles sont déclarées uniquement dans le corps du package. Elles sont visibles par tous les objets du package, mais uniquement par ceux-ci
- Les variables locales : elles sont déclarées dans une procédure ou une fonction et ne sont visibles que de celle-ci.

Initialisation de variables

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, un package peut contenir dans son corps un bloc d'initialisation exécuté seulement lors du premier appel à ce package. Ce code constitue un bloc séparé par les mots clés BEGIN et END à la fin du package body.

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 [schema.]package COMPILE [ PACKAGE | SPECIFICATION | BODY ];
```

Recompilation d'un package

Pour les modifications de l'implémentation des objets, on utilisera la commande CREATE OR REPLACE PACKAGE/PACKAGE BODY, qui recréée un objet package. Il n'est pas nécessaire de sauvegarder les sources des package dans des fichiers pour les reprendre en vue d'une modification de leur contenu, car la source est stockée dans la base de données.

Exemple : Recompiler le package pkg_employe :

```
ALTER PACKAGE pkg_employe COMPILE PACKAGE;
```

Suppression d'un package

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

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

Vues du dictionnaire

Certaines vues du dictionnaire référençant les procédures et les packages sont mentionnées ci-dessous :

- USER_SOURCE -- code PL/SQL stocké dans le dictionnaire
- USER_OBJECT_SIZE -- taille occupée par les objets de schéma
- USER_ERRORS -- erreurs de compilation

Ainsi, par exemple, il est possible de consulter le dictionnaire pour connaître l'espace en mémoire qu'occupent les packages :

```
SELECT * FROM user_object_size WHERE TYPE LIKE 'PACKAGE%';
```

	NAME	TYPE	SOURCE_SIZE	PARSED_SIZE	CODE_SIZE	ERROR_SIZE
1	PKG_EMPLOYE	PACKAGE	178	385	223	0
2	PKG_EMPLOYE	PACKAGE BODY	548	0	673	0

Dépendances d'objets

Un package qui contient des erreurs de programmation existe dans la base de données mais n'est pas compilé. Il a un statut INVALID. De même, lors de son exécution, s'il essaie d'atteindre un objet inexistant (supprimé), il sera invalidé par le noyau. Il est donc important de connaître les dépendances entre objets de programmation, afin de les recompiler, de les modifier ou de les détruire.

Une analyse d'impact simple peut être faite en traquant les dépendances directes sur l'objet analysé en interrogeant la vue du dictionnaire USER_DEPENDENCIES.

```
SELECT * FROM user_dependencies WHERE NAME LIKE 'PKG%';
```

	NAME	TYPE	REFEREN	REFERENCED_NAME	REFERENCED_TYPE	DEPENDENC
1	PKG_EMPLOYE	PACKAGE	SYS	STANDARD	PACKAGE	HARD
2	PKG_EMPLOYE	PACKAGE BODY	SYS	STANDARD	PACKAGE	HARD
3	PKG_EMPLOYE	PACKAGE BODY	SYSTEM	PKG_EMPLOYE	PACKAGE	HARD
4	PKG_EMPLOYE	PACKAGE BODY	SYSTEM	EXE_EMPLOYE	TABLE	HARD
5	PKG_EMPLOYE	PACKAGE BODY	SYSTEM	EXE_DEPT	TABLE	HARD

Packages fournis

La base de données dispose de package fournis par Oracle. Ces objets, qui appartiennent au schéma SYS, peuvent être utilisés par les développeurs. La liste exhaustive de ces objets est différente pour chaque version de la base de données, mais on peut signaler quelques classiques :

- DBMS_OUTPUT Informations de sorties des procédures stockées
- DBMS_DDL Procédures et fonctions d'accès au LDD
- DBMS_UTILITY Analyse d'objets d'un schéma
- DBMS_SESSION Modification de la session utilisateur
- DBMS_SHARED_POOL Accès au SQL partagé
- DBMS_TRANSACTION Contrôle logique des transactions
- DBMS_MAIL Lien avec Oracle*Mail
- DBMS_PIPE Envoi de messages du serveur (PIPE UNIX)
- DBMS_ALERT Signalisation d'événement sur la database
- DBMS_LOCK Appel par l'application de lock
- …

Ces packages, tout comme d'autres existant dans la base de données, peuvent être considérés comme des librairies enrichissant le langage PL/SQL.