PL / SQL
Les sous-programmes: procédures et fonctions

Ines BAKLOUTI

ines.baklouti@esprit.tn

Ecole Supérieure Privée d'Ingénierie et de Technologies

Plan

Introduction

- Les fonctions et procédures sont des **sous-programmes** considérées comme des blocs PL/SQL **nommés**.
- Les fonctions et procédures **stockées** sont des sous-programmes PL/SQL **autonomes** qui sont compilés et stockées dans le dictionnaire de données à l'aide d'une clause CREATE OR REPLACE FUNCTION / PROCEDURE.
- Une fonction ou procédure **non autonome** peut être déclarée dans un bloc PL/SQL, un sous-programme ou un package sans utiliser la clause CREATE OR REPLACE.

Plan

Les procédures et fonctions stockées (autonomes)

- Une procédure (ou fonction) autonome ou stockée est un sous-programme PL/SQL qui est conservé dans une base de données ORACLE et appelé par un utilisateur, directement ou indirectement.
- AVANTAGES
  - Efficacité : minimiser l'appel des requêtes SQL côté client en les remplaçant par des appels de procédures côté serveur intégrant plusieurs instructions SQL
  - Réutilisabilité : une procédure stockée peut être utilisée dans diverses situations (SQL,déclencheur,application)
  - Portabilité : une procédure stockée est indépendante de la version système d'exploitation ou des compilateurs
  - Maintenabilité : en appelant la même procédure à partir de plusieurs outils (SQL*PLUS, application, autre procédure stockée) on réduit le coût de maintenance de cette procédure centralisée.

Les procédures stockées

Syntaxe

CREATE [OR REPLACE] PROCEDURE nom_procédure
[(argument1 [mode passage1] type1. . . ,[argumentN [mode passageN] typeN])]
IS
–Section déclarative optionelle et sans utiliser le mot clé DECLARE
[déclaration_variables_locales]
BEGIN
–Section exécutable obligatoire
[section exception]
END [nom_procédure];

- Pour une procédure, il y a trois modes de passage de paramètre:
  - IN : en entrée (par défaut)
  - OUT : en sortie
  - IN OUT : en entrée et sortie

Les procédures stockées

Exemple

```
CREATE OR REPLACE PROCEDURE add_dept (dept_id IN
departments.department_id%TYPE, dept_name IN
departments.department_name%TYPE, nbre OUT number)
IS
BEGIN
    insert into departments(department_id,department_name)
values(dept_id,dept_name);
    commit;
    select count(*) into nbre from departments;
    dbms_output.put_line('Le nombre de département est : '||nbre);
END;
```

Les procédures stockées

Appel de procédures stockées

Exemple

```
DECLARE
nb number;
BEGIN
add_dept(300,'IT',nb);
end;
```

Les fonctions stockées

Syntaxe

```
CREATE [OR REPLACE] FUNCTION nom_fonction
[(argument1 [mode passage1] type1. . . ,[argumentN [mode passageN] typeN])]
RETURN type_retour IS
–Section déclarative optionelle et sans utiliser le mot clé DECLARE
[déclaration_variables_locales]
BEGIN
–Section exécutable obligatoire
[section exception]
END [nom_fonction];
```

- Tous les paramètres d'une fonction sont en mode IN (dans ce cas, on n'est pas obligé d'écrire le mode).

Les fonctions stockées

Exemple 1

```
CREATE OR REPLACE FUNCTION fn_check_sal (empno employees.employee_id%TYPE)
RETURN Boolean IS
    dept_id employees.department_id%TYPE;
    sal employees.salary%TYPE;
    avg_sal employees.salary%TYPE;
BEGIN
   SELECT salary,department_id INTO sal,dept_id FROM employees WHERE employee_id=empno;
    SELECT avg(salary) INTO avg_sal FROM employees WHERE department_id=dept_id;
IF sal > avg_sal THEN
    RETURN TRUE;
ELSE
    RETURN FALSE;
END IF;
END;
```

Les fonctions stockées

Exemple 2

```
CREATE OR REPLACE FUNCTION fn_dept_name (deptno number)
RETURN varchar IS
    dname varchar(30);
BEGIN
    select department_name into dname from departments where
department_id=deptno;
RETURN dname;
END;
```

Les fonctions stockées

Appel de fonctions stockées

Exemple 1

```
BEGIN
IF (fn_check_sal(124)) THEN
    DBMS_OUTPUT.PUT_LINE('Salary > average');
ELSE
    DBMS_OUTPUT.PUT_LINE('Salary < average');
END IF;
END;
```

Exemple 2

```
SELECT department_id No_dept, fn_dept_name(department_id)
Nom_dept,last_name Nom, first_name Prenom FROM employees order by
department_id;
```

Remarques

- Lors de la création d'un objet quelconque, tel qu'une table, une procédure ou une fonction, les entrées correspondantes sont créées dans la table user_objects. vous pouvez examiner le contenu de la table user_objects en exécutant la commande suivante:
  - SELECT object_name,object_type FROM user_objects;
- Le code source d'une procédure ou fonction stockées est enregistré dans la table user_source. Vous pouvez examiner le code source de la procédure en exécutant la commande suivante:
  - SELECT * FROM user_source WHERE name='FN_DEPT_NAME';
- Utilisez la commande DESCRIBE afin d'examiner les arguments et le type de données renvoyé par une fonction ou procédure.
  - Exemple : DESCRIBE fn_check_sal;

Plan

Les procédures non stockées

Syntaxe

PROCEDURE nom_procédure
[(argument1 [mode passage1] type1. . . ,[argumentN [mode passageN] typeN])]
IS
–Section déclarative optionelle et sans utiliser le mot clé DECLARE
[déclaration_variables_locales]
BEGIN
–Section exécutable obligatoire
[section exception]
END [nom_procédure];

Les procédures non stockées

Exemple

```
DECLARE
    a number:=10;
    b number:=30;
    s number;
    p number;
    PROCEDURE myproc(a in number, b in number, surf out number, peri out number)
    IS
    BEGIN
        surf:=a*b;
        peri:=(a+b)*2;
    END myproc;
BEGIN
    myproc(a,b,s,p);
    dbms_output.put_line('La surface: '||s);
    dbms_output.put_line('Le périmètre: '||p);
END;
```

Les fonctions non stockées

Syntaxe

FUNCTION nom_fonction

[(argument1 [mode passage1] type1. . . ,[argumentN [mode passageN] typeN])]

RETURN type_retour IS

–Section déclarative optionelle et sans utiliser le mot clé DECLARE

[déclaration_variables_locales]

BEGIN

–Section exécutable obligatoire

[section exception]

END [nom_fonction];

Les fonctions non stockées

Exemple

```
DECLARE
    a number:=10;
    b number:=30;
    c number;
    FUNCTION myfunc(a number, b number)
    RETURN number IS
        surf number;
    BEGIN
        surf:=a*b;
        return surf;
        END;
BEGIN
    c:=myfunc(a,b);
    dbms_output.put_line('La surface: '||c);
END;
```