http://docs.oracle.com/database/121/index.htm#

5

Procédures et fonctions stockées (sous-programmes PL/SQL)

Procédures et fonctions

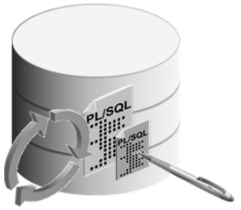
- Ce sont des blocs PL/SQL nommés
- Elles sont également appelées sous-programmes PL/SQL
- Elles présentent des structures de bloc semblables à celle des blocs anonymes
  - Section déclarative facultative (sans le mot-clé DECLARE)
  - Section exécutable obligatoire
  - Section facultative de traitement des exceptions

Différences entre blocs anonymes et sous-programmes

Blocs anonymes	Sous-programmes
Blocs PL/SQL non nommés	Blocs PL/SQL nommés
Compilés chaque fois	Compilés une seule fois
Non stockés dans la base de données	Stockés dans la base de données
Ne peuvent pas être appelés par d'autres applications	Peuvent être appelés par d'autres applications, car ils sont nommés
Ne renvoient pas de valeurs	Les fonctions doivent renvoyer des valeurs
Ne peuvent pas accepter de paramètres	Peuvent accepter des paramètres

5.1

Procédures stockées



Généralités

- Une procédure est un bloc PL/SQL nommé qui exécute une action
- Une procédure est stockée dans la base données (objet de schéma)
- Une procédure est un objet réutilisable

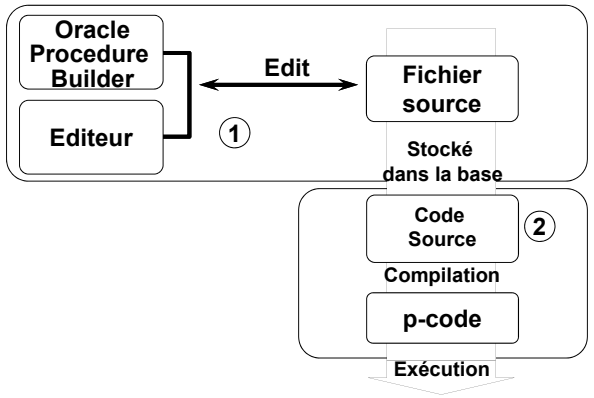
Structure d'une procédure PL/SQL

En-tête
IS
    Section déclarative
BEGIN
    Section exécution
[EXCEPTION
    Section Exception]
END NomProcédure;

Syntaxe

```
CREATE [OR REPLACE] PROCEDURE nom_procédure
 [(paramètre1 [mode1] type1,-- types sans spécification de taille
  paramètre2 [mode2] type2,
  . . .)]
IS
PL/SQL Block;     -- corps de la procédure
```


Développement d'une procédure



Développement d'une procédure

1. **Saisir le texte de l'ordre CREATE OR REPLACE PROCEDURE avec un éditeur et le sauver sous forme de fichier source (extension *.sql*)**

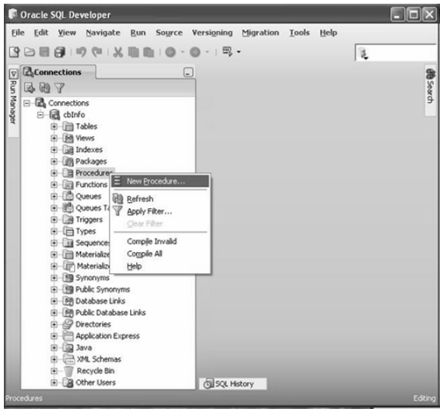
Sous SQL*Plus

2. **Compiler le code par START (ou STA) nom_procédure → p-code généré**

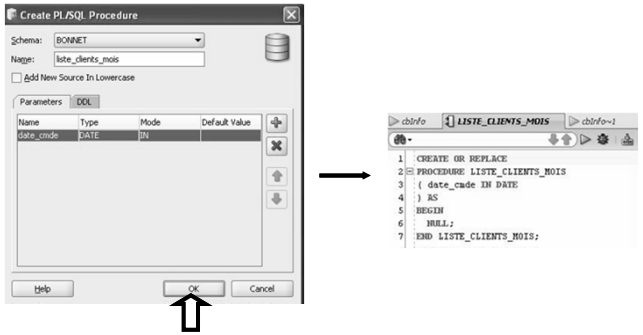
3. **Visualiser les erreurs par SHOW errors**

4. **Appeler la procédure pour l'exécuter EXECUTE (ou EXEC) nom_procédure([arg1, arg2, …, argn])**

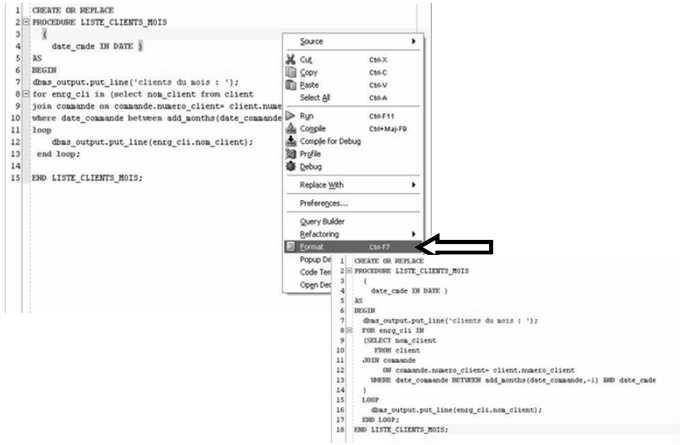
Sous SQL Developer



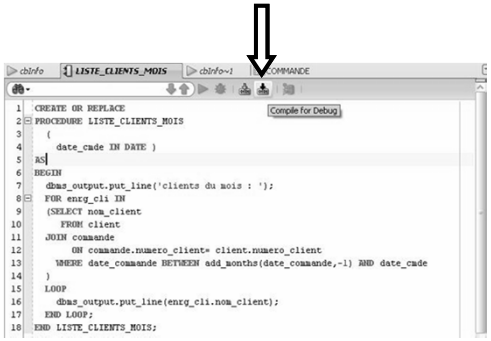
Création d'une procédure - suite



Création d'une procédure - suite

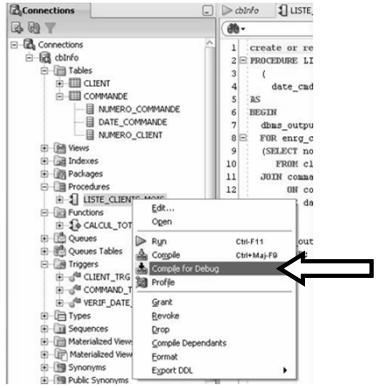


Compilation d'une procédure

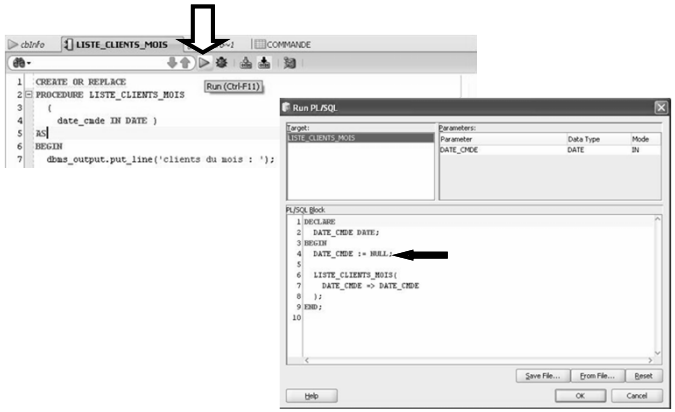


Compilation d'une procédure

OU



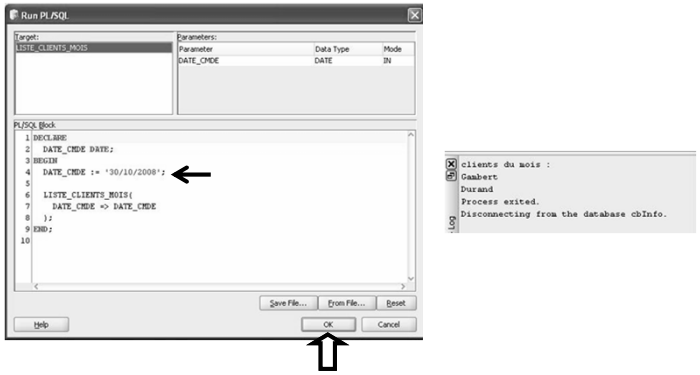
Exécution d'une procédure



Exécution d'une procédure

OU

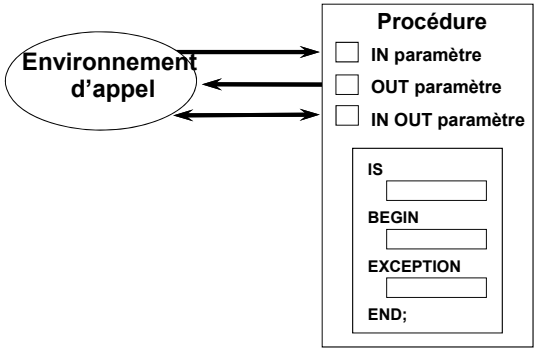
Exécution d'une procédure - suite



Appel des procédures

- **Pour exécuter une procédure, il faut en être propriétaire ou disposer du privilège objet EXECUTE**
- **Appel d'une procédure à partir :**
  - **de blocs anonymes**
  - **d'une autre procédure**
  - **de packages (voir chapitre 6)**

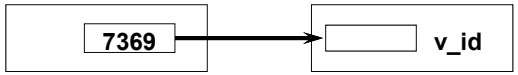
Paramètres d'une procédure



Mode d'un paramètre

IN	OUT	IN OUT
Par défaut	*Doit être spécifié*	*Doit être spécifié*
Valeur transmise à la procédure	Valeur transmise à l'environnement	Valeur transmise à la procédure et renvoyée à l'environnement
paramètre formel variable ou constante	variable non initialisée	variable initialisée
paramètre effectif variable, constante ou expression	variable	variable

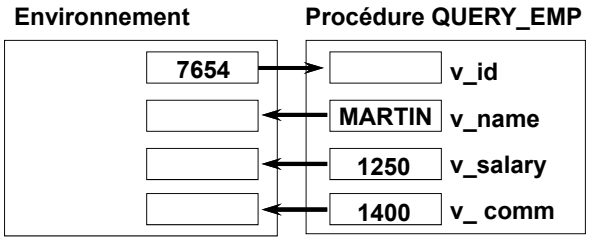
Mode IN: exemple de création et d'appel de la procédure raise_salary




```
SQL> CREATE OR REPLACE PROCEDURE raise_salary
  2  (v_id in emp.empno%TYPE)
  3  IS
  4  BEGIN
  5      UPDATE emp
  6      SET    sal = sal * 1.1
  7      WHERE  empno = v_id;
  8  END raise_salary;
  9  /
Procedure created

SQL> EXECUTE raise_salary (7369)
PL/SQL procedure successfully completed
```


Mode OUT: exemple




```
SQL> CREATE OR REPLACE PROCEDURE query_emp
  1  (v_id     IN   emp.empno%TYPE,
  2   v_name   OUT  emp.ename%TYPE,
  3   v_salary OUT  emp.sal%TYPE,
  4   v_comm   OUT  emp.comm%TYPE)
  5  IS
  6  BEGIN
  7    SELECT   ename, sal, comm
  8    INTO     v_name, v_salary, v_comm
  9    FROM     emp
 10    WHERE    empno = v_id;
 11  END query_emp;
 12  /
```


Exemple : compilation et appel de la procédure query_emp sous SQL*Plus

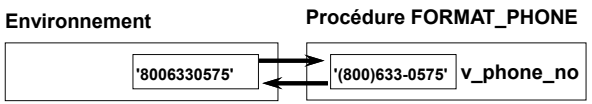
```
SQL> START query_emp.sql
Procedure created
```

```
SQL> VARIABLE g_name varchar2(15)
SQL> VARIABLE g_salary     number
SQL> VARIABLE g_comm       number
```

```
SQL> EXECUTE query_emp (7654, :g_name, :g_salary,
  2  :g_comm)
PL/SQL procedure successfully completed
```

```
SQL> PRINT g_name
G_NAME
---------------
MARTIN
```

Mode IN OUT: exemple




```
SQL> CREATE OR REPLACE PROCEDURE format_phone
  2  (v_phone_no IN OUT VARCHAR2)
  3  IS
  4  BEGIN
  5   v_phone_no := '(' || SUBSTR(v_phone_no,1,3) ||
  6                 ')' || SUBSTR(v_phone_no,4,3) ||
  7                 '-' || SUBSTR(v_phone_no,7);
  8  END format_phone;
  9 /
```


Exemple d'appel de format_phone sous SQL*Plus

```
SQL>VARIABLE g_phone_no varchar2(15)

SQL> BEGIN :g_phone_no := '8006330575'; END;
  2  /
PL/SQL procedure successfully completed

SQL> EXECUTE format_phone (:g_phone_no)
PL/SQL procedure successfully completed

SQL> PRINT g_phone_no
```

```
G_PHONE_NO
---------------
(800)633-0575
```


Appel d'une procédure dans un bloc PL/SQL

```
DECLARE
  v_id NUMBER := 7900;
BEGIN
  raise_salary(v_id);    --appel procédure par son nom
  COMMIT;
...

END;
```


Appel d'une procédure dans une autre procédure

```
SQL> CREATE OR REPLACE PROCEDURE process_emps
  2  IS
  3    CURSOR emp_cursor IS
  4    SELECT empno
  5    FROM   emp;
  6  BEGIN
  7    FOR emp_rec IN emp_cursor LOOP
  8       raise_salary(emp_rec.empno);  --appel procédure
                                          par son nom
  9    END LOOP;
 10    COMMIT;
 11  END process_emps;
 12 /
```

Transmission des valeurs de paramètres

- Par position
- Par Nom =>
- Combinaison

Transmission des valeurs de paramètres - exemple

```
SQL> CREATE OR REPLACE PROCEDURE add_dept
  1  (v_name  IN dept.dname%TYPE  DEFAULT 'unknown',
  2   v_loc   IN dept.loc%TYPE    DEFAULT 'unknown')
  3  IS
  4  BEGIN
  5     INSERT INTO dept
  6     VALUES (dept_deptno.NEXTVAL, v_name, v_loc);
  7     COMMIT;
  8  END add_dept;
  9  /
```

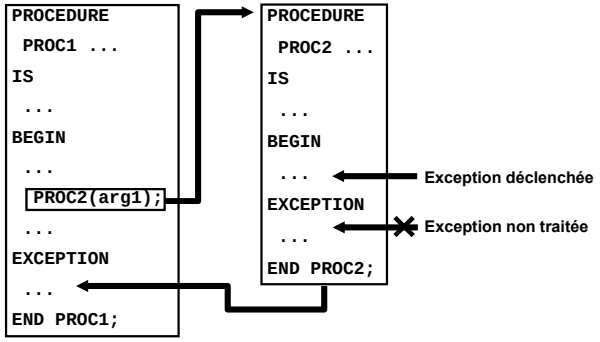

Transmission des valeurs de paramètres - exemple

```
SQL> begin
  2  add_dept;
  3  add_dept ( 'TRAINING', 'NEW YORK');
  4  add_dept ( v_loc => 'DALLAS', v_name =>
                'EDUCATION') ;
  5  add_dept ( v_loc => 'BOSTON') ;
  6  end;
  7  /
PL/SQL procedure successfully completed
```

```
SQL>     SELECT * FROM dept;

DEPTNO   DNAME              LOC
------   --------------     -------------
   ...   ...                ...
    41   unknown            unknown
    42   TRAINING           NEW YORK
    43   EDUCATION          DALLAS
    44   unknown            BOSTON
```


Procédures et Exceptions



Suppression d'une Procédure

Ordre SQL: DROP PROCEDURE

Syntaxe

```
DROP PROCEDURE nom_procédure;
```

Exemple

```
SQL> DROP PROCEDURE raise_salary;
Procedure dropped.
```


Exercices

5.2

Fonctions stockées

Généralités

- Une fonction est un bloc PL/SQL nommé qui exprime une action
- Une fonction renvoie une et une seule valeur
- Une fonction est stockée dans la base données (objet de schéma)
- Une fonction est un objet réutilisable
- Une fonction stockée peut être utilisée dans une expression ou comme valeur de paramètre pour un autre sous-programme

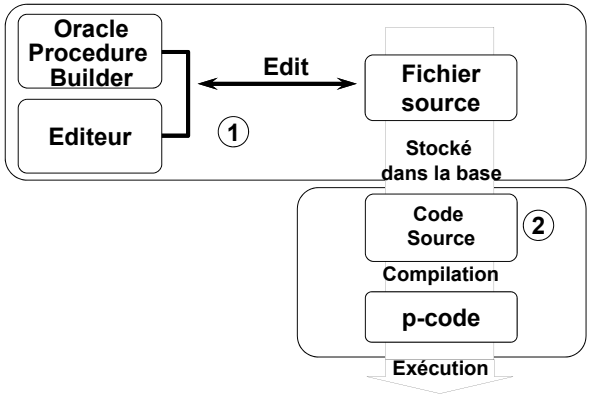
Structure d'une fonction PL/SQL

En-tête - - ordre RETURN
IS
Section déclarative
BEGIN
Section exécution - - ordre RETURN
[EXCEPTION
Section Exception]
END NomFonction;

Syntaxe pour créer une Fonction

```
CREATE [OR REPLACE] FUNCTION nom_fonction
 (paramètre1 [mode1] type1, -- types sans spécification de taille
  paramètre2 [mode2] type2,
  . . .
RETURN type      -- type de la valeur résultat
IS
PL/SQL Block;    -- corps de la fonction
```


Développement d'une fonction



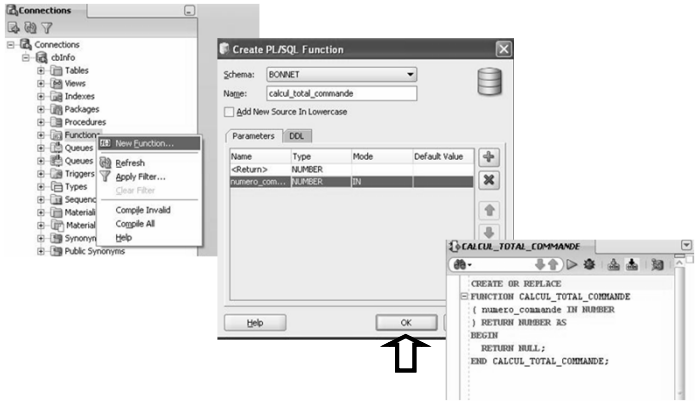
Développement d'une fonction

1. **Saisir le texte de l'ordre CREATE OR REPLACE FUNCTION avec un éditeur et le sauver sous forme de fichier source (extension *.sql*)**

Sous SQL*Plus

2. **compiler le code par START (ou STA) nom_fonction → p-code généré**
3. **Visualiser les erreurs par SHOW errors**
4. **Appeler la fonction pour l'exécuter EXECUTE (ou EXEC)**

Sous SQL Developer



Création d'une fonction sous SQL*Plus : Exemple

```
SQL> CREATE OR REPLACE FUNCTION get_sal
  2   (v_id  IN   emp.empno%TYPE)
  3  RETURN NUMBER
  4  IS
  5    v_salary     emp.sal%TYPE :=0;
  6  BEGIN
  7    SELECT sal
  8    INTO   v_salary
  9    FROM   emp
 10    WHERE  empno = v_id;
 11    RETURN (v_salary);
 12 END get_sal;
 13 /
```


Exécution d'une fonction

- Pour exécuter une fonction, il faut en être propriétaire ou disposer du privilège objet EXECUTE
- Utilisation dans une expression PL/SQL – résultat dans une variable locale

```
DECLARE
  sal employees.salary%type;
BEGIN
  sal := get_sal(100);
  DBMS_OUTPUT.PUT_LINE('The salary is: '|| sal);
END;
```


- Utilisation dans une expression PL/SQL – résultat dans une variable hôte

```
VARIABLE b_salary NUMBER
EXECUTE :b_salary := get_sal(100)
```

- Utilisation en paramètre d'un autre sous-programme

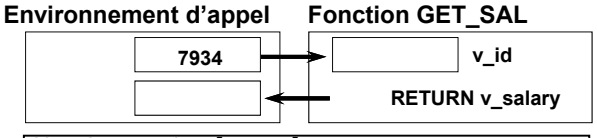
```
EXECUTE dbms_output.put_line(get_sal(100))
```

- Utilisation dans un ordre SQL (voir restrictions)

```
SELECT job_id, get_sal(employee_id)
FROM employees;
```

Exemple sous SQL*Plus

Création d'une variable de lien pour recevoir le résultat de la fonction et exécution de la fonction




```
SQL> START get_salary.sql
Procedure created

SQL> VARIABLE g_salary number

SQL> EXECUTE :g_salary := get_sal(7934)
PL/SQL procedure successfully completed

SQL> PRINT g_salary
          G_SALARY
------------------
              1300
```


Utilisation d'une fonction utilisateur dans un ordre SQL

- Dans la liste de projection d'un ordre `SELECT`
- Dans les expressions conditionnelles des clauses `WHERE` ou `HAVING`
- Dans les clauses `CONNECT BY`, `START WITH`, `ORDER BY`, et `GROUP BY`
- Dans la clause `VALUES` d'un ordre `INSERT`
- Dans la clause `SET` d'un ordre `UPDATE`

Partout où une fonction SQL prédéfinie peut être appelée

Utilisation d'une fonction utilisateur dans un ordre SQL : RESTRICTIONS

- La fonction ne doit accepter que des paramètres `IN`
- Les type des paramètres d'entrée sont limités aux types de données SQL (et non des types spécifiques PL/SQL)
- Le type de la valeur renvoyée peut être tout type de données SQL
- A l'appel, les paramètres doivent être spécifiés en utilisant la notation par position

AUTRES RESTRICTIONS

- Pour une fonction appelée à partir d'une instruction `SELECT`, les ordres `INSERT`, `UPDATE`, ou `DELETE` ne sont pas autorisés dans le corps de cette fonction
- Pour une fonction appelée à partir d'une instruction `UPDATE` ou `DELETE` sur une table `T`, les ordres `SELECT`, `INSERT`, `UPDATE`, ou `DELETE` sur cette table `T` ne sont pas autorisés dans le corps de cette fonction
- Pour une fonction appelée à partir une instruction SQL, les ordres `COMMIT` ou `ROLLBACK` ne sont pas autorisés
- Les procédures / fonctions appelées dans le corps de la fonction doivent respecter les restrictions précédentes

Suppression d'une Fonction

Par l'ordre DROP FUNCTION

Syntaxe

```
DROP FUNCTION nom_fonction;
```

Exemple

```
SQL> DROP FUNCTION get_salary;
Function dropped.
```


Vues du dictionnaire de données relatives aux procédures / fonctions

- **user_source**

```
desc user_source
Nom                            NULL     Type
------------------------------ -------- ----------------------
NAME                                    VARCHAR2(30)
TYPE                                    VARCHAR2(12)
LINE                                    NUMBER
TEXT                                    VARCHAR2(4000)
```

```
SELECT text FROM user_source where type
='FUNCTION' order by line;
```

```
  TEXT
1 FUNCTION check_sal (p_empno employees.employee_id%type) RETURN Boolean IS
2 FUNCTION get_sal
3 v_dept_id employees.department_id%TYPE;
4 (v_id  IN   emp.empno%TYPE)
5  v_sal     employees.salary%TYPE;
6    RETURN NUMBER
...
```


- **user_objects**

'PROCEDURE'
'FUNCTION'

```
desc user_objects
Nom                            NULL     Type
------------------------------ -------- ----------------
OBJECT_NAME                             VARCHAR2(128)
SUBOBJECT_NAME                          VARCHAR2(30)
OBJECT_ID                               NUMBER
DATA_OBJECT_ID                          NUMBER
OBJECT_TYPE                             VARCHAR2(19)
CREATED                                 DATE
LAST_DDL_TIME                           DATE
TIMESTAMP                               VARCHAR2(19)
STATUS                                  VARCHAR2(7)
TEMPORARY                               VARCHAR2(1)
GENERATED                               VARCHAR2(1)
SECONDARY                               VARCHAR2(1)
NAMESPACE                               NUMBER
EDITION_NAME                            VARCHAR2(30)
```

```
SELECT created, status FROM user_objects
where object_name='GET_SAL';
```

	CREATED	STATUS
1	07/04/14	VALID

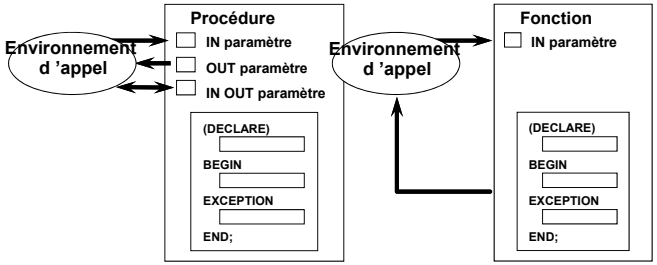
- **user_procedures**

```
desc user_procedures
Nom              NULL Type
---------------- ---- -------------
OBJECT_NAME           VARCHAR2(128)
PROCEDURE_NAME        VARCHAR2(30)
OBJECT_ID             NUMBER
SUBPROGRAM_ID         NUMBER
OVERLOAD              VARCHAR2(40)
OBJECT_TYPE           VARCHAR2(19)
AGGREGATE             VARCHAR2(3)
PIPELINED             VARCHAR2(3)
IMPLTYPEOWNER         VARCHAR2(30)
IMPLTYPENAME          VARCHAR2(30)
PARALLEL              VARCHAR2(3)
INTERFACE             VARCHAR2(3)
DETERMINISTIC         VARCHAR2(3)
AUTHID                VARCHAR2(12)
```

```
SELECT OBJECT_NAME FROM USER_PROCEDURES WHERE
OBJECT_TYPE IN ('PROCEDURE','FUNCTION');
```

- **user_errors : erreurs de syntaxe PL/SQL**

Procédure ou Fonction?



Procédure ou Fonction?

Procédure	*Fonction*
Exécutée comme une instruction PL/SQL	Utilisée dans une expression
Pas de type de retour	Type de RETOUR défini
Peut fournir plusieurs valeurs résultats	Une seule valeur résultat

Exercices