Applied Faculty

Database 3

الإجرائيات و التوابع

Procedures and Functions

مقدمة

- الإجرائيات و التوابع هي كتل برمجية مخزنة بشكل دائم في قاعدة البيانات
  - مكتوبة بلغة إجرائية لـ SQL
    - PL/SQL في Oracle
    - T-SQL في SQL Server
  - أو لغة برمجة خارجية مثل Java أو C++
  - يمكن استدعاء إجرائية مخزنة من قبل:
    - البرامج التطبيقية
    - الإجرائيات الأخرى
    - القوادح
  - الإجرائية لا تعيد قيمة
  - التابع يعيد قيمة معينة من نمط بيانات محدد

تعريف إجرائية

```
CREATE [OR REPLACE]
PROCEDURE <procedure_name> [(<list of parameters>)]
IS
   <variable declarations>
BEGIN
   <executable statements>
[EXCEPTION
   <exception handling>]
END [<procedure name>];
/
```

تعريف تابع

```
CREATE [OR REPLACE]
FUNCTION <function_name> [(<list of parameters>)]
RETURN <data_type> IS
   <variable declarations>
BEGIN
   <executable statements>
   RETURN <variable>;
[EXCEPTION
   <exception handling>]
END [<function_name>];
/
```

تعريف الوسطاء

```
<parameter_name> [IN|OUT|IN OUT] <data_type> [{ := | DEFAULT} <expression>]
```

- **IN وسيط دخل فقط**
  - يمكن استخدامه فقط بواسطة مصدر خارجي
  - لا يمكن تعديله ضمن الإجرائية
- **OUT وسيط خرج فقط**
  - لا يمكن أن يتلقى قيمة من مصدر خارجي
  - يمكن أن يتلقى قيمة ضمن الإجرائية فقط
- **IN OUT وسيط دخل و خرج معاً**
  - يمكن أن يتلقى قيمة من مصدر خارجي
  - يمكن للإجرائية أن تعدل قيمة الوسيط
  - يمكن للإجرائية الطالبة أن تستخدم القيمة المعدلة

مثال - إجرائية لإدراج سطر في جدول

```
CREATE OR REPLACE PROCEDURE
   add_employee(id NUMBER, name VARCHAR2, salary NUMBER)
IS
BEGIN
  INSERT INTO emp(empno, ename, sal)
  VALUES (id, name, salary);
END add_employee;
/
```

```
EXECUTE add_employee(8300,'Joe Chung',1750);
```

مثال – إجرائية لرفع رواتب موظفين قسم معين بنسبة معينة

```
CREATE PROCEDURE
     raise_salary(dno NUMBER, percentage NUMBER DEFAULT 0.5) IS
  CURSOR emp_cur (dept_no NUMBER) IS
    SELECT sal FROM emp WHERE deptno = dept_no
  FOR UPDATE OF SAL;
  empsal NUMBER(8);
BEGIN
  OPEN emp_cur(dno); -- Here dno is assigned to dept_no
  LOOP
    FETCH emp_cur INTO empsal;
    EXIT WHEN emp_cur%NOTFOUND;
    UPDATE EMP SET SAL = empsal * ((100 + percentage)/100)
    WHERE CURRENT OF emp_cur;
  END LOOP;
  CLOSE emp_cur;
  COMMIT;
END raise_salary;
/
```

إدخال القيم باستخدام وسيط من النوع IN

```
CREATE OR REPLACE PROCEDURE
     list_dept_employees(department IN INTEGER DEFAULT 10) IS
BEGIN
  FOR cur IN (SELECT * FROM emp
              WHERE deptno = department) LOOP
    DBMS_OUTPUT.PUT_LINE(cur.ename);
  END LOOP;
END list_dept_employees;
/
```

اسم الوسيط مع نمط بيانات غير مقيد

تم استخدام الوسيط في عبارة WHERE

```
SQL> set serveroutput on
SQL> exec list_dept_employees(10);
CLARK
KING
MILLER

PL/SQL procedure successfully completed.
```

القيمة الحالية للوسيط

إخراج القيم باستخدام وسيط من النوع OUT

إجرائية اسمية لإيجاد الموظف الأكبر عمراً في القسم

```
CREATE OR REPLACE PROCEDURE find_oldest_employee(
            department IN INTEGER,
            ename OUT emp.ename%TYPE,
            sal   OUT emp.sal%TYPE)
IS
  CURSOR b IS SELECT ename, sal FROM emp
              WHERE birth_date=(SELECT MIN(birth_date)
                                FROM emp
                                WHERE deptno = department);
BEGIN
  OPEN b;
  FETCH b INTO ename, sal;
  CLOSE b;
END find_oldest_employee;
/
```

إخراج القيم باستخدام وسيط من النوع OUT

إجرائية غير اسمية تستدعي الإجرائية السابقة

```
DECLARE
  CURSOR cur IS SELECT * FROM dept;
  cur_var cur%ROWTYPE;
  ename emp.ename%TYPE;
  sal   emp.sal%TYPE;
BEGIN
  OPEN cur;
  FETCH cur INTO cur_var;
  WHILE cur%FOUND LOOP
    find_oldest_employee(cur_var.deptno, ename, sal);
    DBMS_OUTPUT.PUT_LINE(cur_var.deptno||' '||ename||' '||sal);
    FETCH cur INTO cur_var;
  END LOOP;
  CLOSE cur;
END;
/
```

تعريف إجرائية ضمن إجرائية أخرى

```
DECLARE
  CURSOR cur IS SELECT * FROM dept;
  cur_var cur%ROWTYPE;
  v_ename emp.ename%TYPE;
  v_sal   emp.sal%TYPE;
  PROCEDURE find_oldest_employee(
            department IN INTEGER,
            ename OUT emp.ename%TYPE,
            sal   OUT emp.sal%TYPE) IS
    CURSOR b IS SELECT ename, sal FROM emp
                WHERE birth_date=(SELECT MIN(birth_date) FROM emp
                                  WHERE deptno = department);
  BEGIN
    OPEN b; FETCH b INTO ename, sal; CLOSE b;
  END find_oldest_employee;
BEGIN
  OPEN cur;
  FETCH cur INTO cur_var;
  WHILE cur%FOUND LOOP
    find_oldest_employee(cur_var.deptno, v_ename, v_sal);
    DBMS_OUTPUT.PUT_LINE(cur_var.deptno||' '||v_ename||' '||v_sal);
    FETCH cur INTO cur_var;
  END LOOP;
  CLOSE cur;
END;
/
```

التوابع

```
CREATE OR REPLACE FUNCTION goodday
RETURN VARCHAR2
IS
BEGIN
  RETURN 'Have a Good Day';
END;
/
```

```
DECLARE
  a VARCHAR2(30);
BEGIN
  a := goodday;
  DBMS_OUTPUT.PUT_LINE(a);
END;
/
```

استدعاء تابع ضمن عبارة SELECT

```
CREATE OR REPLACE FUNCTION age(bdate IN DATE)
RETURN NUMBER IS
  age NUMBER;
BEGIN
  age := TRUNC(MONTHS_BETWEEN(CURRENT_DATE, bdate)/12, 0);
  RETURN age;
END;
/
```

```
SELECT ename, age(birth_date)
  FROM emp
 WHERE deptno = 10;
```

استدعاء تابع ضمن إجرائية

```
CREATE OR REPLACE FUNCTION highest_seniority(dno IN INTEGER)
RETURN VARCHAR2 IS
  CURSOR b IS SELECT * FROM emp
              WHERE hiredate = (SELECT MIN(hiredate)
                                FROM emp
                                WHERE deptno = dno);
  b_var b%ROWTYPE;
BEGIN
  OPEN b;
  FETCH b INTO b_var;
  IF b%NOTFOUND THEN
    RETURN 'No employee in the department!';
  ELSE
    RETURN b_var.first_name||' '||b_var.last_name;
  END IF;
END;
/
```

استدعاء تابع ضمن إجرائية

```
DECLARE
  name VARCHAR2(50);
BEGIN
  FOR c IN (SELECT * FROM dept) LOOP
    name := highest_seniority(c.deptno);
    DBMS_OUTPUT.PUT_LINE (c.dname||' '||name);
  END LOOP;
END;
/
```