



www.dirasats.com

هذا الغلاف لا يعبر عن حقوق الملكية او فحوى الكتاب, فهو مجرد واجهة للموقع المحمل منه



شكرا لك على ثقتك بنا وعلى اختيار موقعنا

www.dirasats.com



من اجل تواصل معنا المرجو زيارة الموقع ستجد جميع المعلومات

www.dirasats.com

TP 5 : Concepts avancées des Curseurs Explicits

Objectifs :

- Déclaration et utilisation de curseurs explicites paramétrés
- Utilisation d'un curseur FOR UPDATE

1.

```
DECLARE
    v_current_deptno      dept.deptno%TYPE;
    v_emp                  VARCHAR2(50);
    CURSOR dept_cursor IS
        SELECT deptno
        FROM dept
        ORDER BY deptno;
    CURSOR emp_cursor(v_deptno NUMBER) IS
        SELECT ename || ' - Department ' || TO_CHAR(deptno)
        FROM emp
        WHERE deptno = v_deptno;
BEGIN
    OPEN dept_cursor;
    LOOP
        FETCH dept_cursor INTO v_current_deptno;
        EXIT WHEN dept_cursor%NOTFOUND;
        IF emp_cursor%ISOPEN THEN
            CLOSE emp_cursor;
        END IF;
        OPEN emp_cursor (v_current_deptno);
        LOOP
            FETCH emp_cursor INTO v_emp;
            EXIT WHEN emp_cursor%NOTFOUND;
            INSERT INTO messages (results)
            VALUES (v_emp);
        END LOOP;
        CLOSE emp_cursor;
    END LOOP;
    CLOSE dept_cursor;
    COMMIT;
END;
/
SQL> START p7q1.sql
SQL> SELECT *
      2 FROM messages;
```

2.

```
SQL> ALTER TABLE emp
      2 ADD stars VARCHAR2(100);
```

3.

```
SET VERIFY OFF
ACCEPT p_empno PROMPT ' Entrez le numéro d''employé : '
DECLARE
    v_empno emp.empno%TYPE := &p_empno;
    v_asterisk emp.stars%TYPE := NULL;
    v_sal emp.sal%TYPE;
BEGIN
    SELECT NVL(ROUND(sal/100), 0)
    INTO v_sal
    FROM emp
    WHERE empno = v_empno;
    FOR i IN 1..v_sal LOOP
        v_asterisk := v_asterisk || '*';
    END LOOP;
    UPDATE emp
    SET stars = v_asterisk
    WHERE empno = v_empno;
    COMMIT;
END;
/
SET VERIFY ON
SQL> START p4q5.sql
SQL> SELECT empno, sal, stars
      2 FROM emp
      3 WHERE empno IN (7934, 8000);
```

4.

```
SET VERIFY OFF
ACCEPT p_empno PROMPT 'Entrez le numéro d''employé : '
DECLARE
    v_empno emp.empno%TYPE := &p_empno;
    v_asterisk emp.stars%TYPE := NULL;
    CURSOR emp_cursor IS
        SELECT empno, NVL(ROUND(sal/100), 0) sal
        FROM emp
        WHERE empno = v_empno
        FOR UPDATE;
BEGIN
    FOR emp_record IN emp_cursor LOOP
        FOR i IN 1..emp_record.sal LOOP
            v_asterisk := v_asterisk || '*';
        END LOOP;
        UPDATE emp
        SET stars = v_asterisk
        WHERE CURRENT OF emp_cursor;
        v_asterisk := NULL;
    END LOOP;
```

```
COMMIT;
END;
/
SET VERIFY ON
SQL> START p7q2.sql
SQL> SELECT  empno, sal, stars
      2  FROM    emp
      3  WHERE   empno IN (7844, 7900, 8000);
```

TP 6 : Traitements des Exceptions

Objectifs :

- Traiter des exceptions nommées
- Créer et faire référence à des exceptions définies par l'utilisateur

1.

```
SET VERIFY OFF
ACCEPT p_sal PROMPT 'Entrez le montant du salaire : '
DECLARE
    v_ename emp.ename%TYPE;
    v_sal emp.sal%TYPE := &p_sal;
BEGIN
    SELECT      ename
    INTO    v_ename
    FROM      emp
    WHERE     sal = v_sal;
    INSERT INTO messages (results)
    VALUES (v_ename || ' - ' || TO_CHAR(v_sal));
EXCEPTION
    WHEN no_data_found THEN
        INSERT INTO messages (results)
        VALUES ('Aucun employé avec un salaire de' ||
                TO_CHAR(v_sal));
    WHEN too_many_rows THEN
        INSERT INTO messages (results)
        VALUES ('Plus d''un employé avec un salaire de ' ||
                TO_CHAR(v_sal));
    WHEN others THEN
        INSERT INTO messages (results)
        VALUES ('Une autre erreur est survenue.');
```

END;

/

```
SET VERIFY ON
SQL> START p8q1.sql
SQL> START p8q1.sql
SQL> START p8q1.sql
```

2.

```
SET VERIFY OFF
VARIABLE g_message VARCHAR2(40)
ACCEPT p_deptno PROMPT ' Entrer le numéro de département
: '
ACCEPT p_loc PROMPT ' Entrer la localité du département
: '
DECLARE
    e_invalid_dept EXCEPTION;
```



```

DECLARE
    v_sal          emp.sal%TYPE := &p_sal;
    v_low_sal      emp.sal%TYPE := v_sal - 100;
    v_high_sal     emp.sal%TYPE := v_sal + 100;
    v_no_emp       NUMBER(7);
    e_no_emp_returned EXCEPTION;
    e_more_than_one_emp EXCEPTION;
BEGIN
    SELECT      count(ename)
    INTO        v_no_emp
    FROM        emp
    WHERE sal between v_low_sal and v_high_sal;
    IF v_no_emp = 0 THEN
        RAISE e_no_emp_returned;
    ELSIF v_no_emp > 0 THEN
        RAISE e_more_than_one_emp;
    END IF;
EXCEPTION
    WHEN e_no_emp_returned THEN
        :g_message := 'Il n''y a pas d''employé avec un
salaire entre '||TO_CHAR(v_low_sal) || ' et '||
                        TO_CHAR(v_high_sal);
    WHEN e_more_than_one_emp THEN
        :g_message := 'Il y a '|| TO_CHAR(v_no_emp) ||
                        ' employé(s) avec un salaire entre '||
                        TO_CHAR(v_low_sal) || ' et '||
                        TO_CHAR(v_high_sal);

END;
/
SET VERIFY ON
PRINT g_message
SQL> START p8q3.sql

```

TP 7 : Création de procédures

Objectifs :

- **Décrire les différentes utilisations des procédures**
- **Créer des procédures côté client et côté serveur**
- **Créer des procédures avec des paramètres**
- **Déclarer des sous-programmes**
- **Appeler une procédure**
- **Supprimer une procédure**

1.a

```
CREATE OR REPLACE PROCEDURE add_prod
(v_prodid IN product.prodid%TYPE,
 v_descrip IN product.descrip%TYPE)
IS
BEGIN
    INSERT INTO product (prodid, descrip)
    VALUES      (v_prodid, v_descrip);
    COMMIT;
END add_prod;
```

1.b

```
SQL> START p3_1.sql
Procedure created.
SQL> EXECUTE add_prod (9999, 'SP TENNIS BALLS')
PL/SQL procedure successfully completed.
```

1.c

```
EXECUTE add_prod (100860, 'SP ADULT TENNIS RACKET')
begin add_prod(100860, 'SP ADULT TENNIS RACKET'); end;
*
ERROR at line 1
ORA-00001: unique constraint(SCOTT.PRODUCT_PRIMARY_KEY)
violated
ORA-06512: at "SCOTT.ADD_PROD", line 6
ORA-06512: at line 1
C'est la contrainte sur la clé primaire de la colonne
PRODID.
```


2.a

```
CREATE OR REPLACE PROCEDURE upd_prod
  (v_prodid IN product.prodid%TYPE,
   v_descrip IN product.descrip%TYPE)
IS
BEGIN
  UPDATE product
  SET   descrip = v_descrip
  WHERE prodid = v_prodid;
  IF SQL%NOTFOUND THEN
    RAISE_APPLICATION_ERROR(-20202,'No products updated.');
```

```
END IF;
```

```
END upd_prod;
```

2.b

```
SQL> START p3_2.sql
Procedure created.
```

```
SQL> EXECUTE upd_prod (9999, 'SP TENNIS NETS')
PL/SQL procedure successfully completed.
```

3.a

```
CREATE OR REPLACE PROCEDURE del_prod
  (v_prodid IN product.prodid%TYPE)
IS
BEGIN
  DELETE FROM product
  WHERE prodid = v_prodid;
  IF SQL%NOTFOUND THEN
    RAISE_APPLICATION_ERROR(-20203,'No products
deleted.');
```

```
END IF;
```

```
END DEL_PROD;
```

3.b

```
SQL> START p3_3.sql
Procedure created.
```

```
SQL> EXECUTE del_prod (9999)
PL/SQL successfully completed.
```

4.a

```
CREATE OR REPLACE PROCEDURE query_emp
  (v_empno IN emp.empno%TYPE,
   v_sal OUT emp.sal%TYPE,
   v_job OUT emp.job%TYPE)
IS
BEGIN
  SELECT sal, job
  INTO v_sal, v_job
  FROM emp
  WHERE empno = v_empno;
END query_emp;
```

4.b

```
SQL> START p3_4.sql
      Procedure created.

SQL> VARIABLE g_sal NUMBER
SQL> VARIABLE g_job VARCHAR2(15)
SQL> EXECUTE query_emp (7839, :g_sal, :g_job)
PL/SQL procedure successfully completed.

SQL> PRINT g_sal
      G_SAL
-----
      5000

SQL> PRINT g_job
      G_JOB
-----
PRESIDENT
```

4.c

```
SQL> execute query_emp (9898, :g_sal, :g_job)
      begin query_emp (9898, :g_sal, :g_job); end;
      *
ERROR at line 1:
ORA-01403: no data found
ORA-06512: at "SCOTT.QUERY_EMP", line 7
ORA-06512: at line 1
```

Il n'y a pas d'employés dans la table EMP avec le numéro EMPNO à 9898.

L'ordre SELECT ne ramène pas de données de la base, ce qui implique l'erreur PL/SQL NO_DATA_FOUND.