5- EXPLICIT CURSORS

Contenido

1. Introducció

Fins ara, només hem utilitzat SELECTS que retornen una sola fila. Per treballar amb **SELECTS que retornen més d'una fila** necessitem **cursors explícits**.

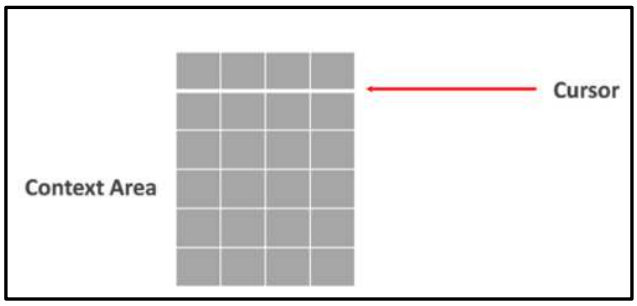
2. Context area i cursors

Context area

Zona de memòria en que Oracle emmagatzema les **dades** processades per cada sentència SQL.

Cada *context area*, i , per tant, cada sentència SQL que s'executa, té un cursor associat.

Es pot veure un cursor com un punter a la *context area*.



Tipus de cursors: implícit i explícit

CURSORS IMPLÍCITS

Definits automàticament per Oracle per a les sentències DML (INSERT, UPDATE, DELETE i MERGE) i SELECTs que retornen **una única fila**. Quan s'executa una d'aquestes sentències, es crea una cursor anomenat **SQL**.

Attribute	Description
SQL%FOUND	Boolean attribute that evaluates to TRUE if the most recent SQL statement returned at least one row.
SQL%NOTFOUND	Boolean attribute that evaluates to TRUE if the most recent SQL statement did not return even one row.
SQL%ROWCOUNT	An integer value that represents the number of rows affected by the most recent SQL statement.

CURSORS EXPLÍCITS

Declarats pel programador per SELECTS que retornen **més d'una fila**. Fan referència a la *context area* del SELECT i permeten accedir a les seves dades, recorrent les files del resultat d'una en una.

3. Treballar amb un cursor explícit

3.1. Exemple complet

Mostrar l'id i el nom de tots els departaments.

```
DECLARE
    -- Declara el cursor, associant-lo a un SELECT
    CURSOR cur_depts IS
        SELECT DEPARTMENT_ID, DEPARTMENT_NAME
        FROM DEPARTMENTS;
    v_department_id   departments.department_id%TYPE;
    v_department_name departments.department_name%TYPE;
BEGIN
    -- Obre el cursor -> Executa la consulta associada
    OPEN cur_depts;

    -- Obté les dades de la primera fila del resultat
    FETCH cur_depts INTO v_department_id, v_department_name;

    -- Bucle que recórrer totes les files del resultat
    WHILE cur_depts%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(v_department_id || ' ' || v_department_name);
        -- Obté les dades d'una fila
        FETCH cur_depts INTO v_department_id, v_department_name;
    END LOOP;

    -- Tanca el cursor
    CLOSE cur_depts;
END;
```

El nombre de files que retorna un SELECT s'anomena **ACTIVE SET** i s'emmagatzema a la *context area*.

3.2. Declaració del cursor: CURSOR ... IS SELECT

A l'apartat **DECLARE** del bloc de codi es dona **nom** al cursor i es vincula amb una **consulta**.

Un curso es pot associar qualsevol SELECT, incloent joins, SUBCONSULTES, ...

SINTAXI

```
CURSOR nom_cursor IS
   sentència_SELECT
```

EXEMPLE

```
CURSOR cur_depts IS
    SELECT DEPARTMENT_ID, DEPARTMENT_NAME
    FROM DEPARTMENTS;
```

Es poden utilitzar **variables PL/SQL** en el SELECT. Si s'utilitzen variables **han de ser declarades abans** del cursor.

3.3. Obrir el cursor: OPEN

Quan s'obre el cursor s'**executa la consulta associada** i les dades del resultat s'emmagatzemen a la *context area*.

El cursor apunta a la primera fila del resultat.

SINTAXI

```
OPEN nom_cursor;
```

EXEMPLE

```
OPEN cur_depts;
```

3.4. Recórrer el resultat: FETCH..INTO, %NOTFOUND, %FOUND

FETCH obté les dades de la fila del resultat a la que apunta el cursor i les guarda a les variables de la llista. El cursor passa a apuntar a la fila següent.

SINTAXI

```
FETCH nom_cursor INTO llista_variables
```

EXEMPLE

```
FETCH cur_depts INTO v_department_id, v_department_name;
```

BUCLE QUE RECÓRRER EL RESULTAT

```
-- Bucle que recórrer totes les files del resultat
WHILE cur_depts%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(v_department_id || ' ' || v_department_name);
    -- Obté les dades De la fila següent
    FETCH cur_depts INTO v_department_id, v_department_name;
END LOOP;
```

3.5. Tancar el cursor: CLOSE

Allibera la memòria de la *context area*.

SINTAXI

```
CLOSE nom_cursor;
```

EXEMPLE

```
CLOSE cur_depts;
```

- ✓ Es pot tornar a obrir el cursor amb un nou OPEN.
- ✓ Només es pot tornar a obrir un cursor si està tancat.
- ✓ Si s'intenta obtenir les dades d'un cursor tancat, es genera una excepció **INVALID_CURSOR**.

4. Atributs dels cursors explícits

nom_cursor%FOUND
Retorna cert si el darrer **FETCH** ha recuperat algun valor; sinó retorna retorna fals. ✓ Si el cursor **no** estava **obert** retorna **error** ✓ Si el cursor estava obert però **no** s'havia executat encara cap **FETCH**, retorna **NULL**. Utilitzat com a condició en els bucles que recorren el cursor.
nom_cursor %NOTFOUND
Fa el contrari que %FOUND.
nom_cursor %ROWCOUNT
Retorna el nombre de files recuperades fins al moment pel cursor (nombre de FETCH realitzats satisfactòriament).
nom_cursor %ISOPEN
Retorna veritable si el cursor està obert.

RECORDAR: No es poden utilitzar directament les propietats d'un cursor explícit en un sentència SQL. Només en una sentència PL/SQL.

S'ha de guardar el valor de la propietat en una variable i utilitzar la variable en la sentència SQL.

5. Records: %ROWTYPE

PROBLEMA AMB EL NOMBRE DE COLUMNES DEL RESULTAT

A l'exemple anterior, el SELECT del cursor només obté dos camps, *department_id* i *department_name*. Què passaria si el SELECT retornés totes les propietats d'EMPLOYEES ?

La taula EMPLOYEES té **12 columnes** →necessitem declarar **12 variables** , una per cada camp.

A més, el nom i tipus dels camps d'una taula poden canviar → s'hauria d'adaptar el codi PL/SQL als canvis que puguin tenir les taules.

```
DECLARE
  v_emp_id         employees.employee_id%TYPE;
  v_first_name     employees.first_name%TYPE;
  v_last_name      employees.last_name%TYPE;
  v_email          employees.email%TYPE;
  v_phone_number   employees.phone_number%TYPE;
  ... FIVE MORE SCALAR VARIABLES REQUIRED TO MATCH THE TABLE
  v_department_id employees.department_id%TYPE;
  CURSOR cur_emps IS
    SELECT * FROM employees
     WHERE department_id = 30;
BEGIN
  OPEN cur_emps;
  LOOP
    FETCH cur_emps INTO v_emp_id, v_first_name, EIGHT MORE HERE ...
        v_department_id;
  ...
```

SOLUCIÓ: **RECORDS**

Què és un registre ?

Un registre és un tipus de dades **compost**, format per un conjunt de camps, cada un amb el seu nom i tipus de dades.

Per accedir al valor d'un camp: `nom_registre.nom_camp`

Associar un registre amb un cursor: %ROWTYPE

Declara un registre amb el mateixos camps i del mateix tipus que les columnes del SELECT associat al cursor.

```
-- Declara el cursor
CURSOR cur_emps IS
    SELECT * FROM EMPLOYEES
       WHERE DEPARTMENT_ID = 30;

-- Declara el registre
v_emp_record cur_emps%ROWTYPE
```

v_emp_record.employee_id	v_emp_record.last_name	v_emp_record.salary
100	King	24000

EXEMPLE

```
DECLARE
    -- Declara el cursor
    CURSOR cur_emps IS
        SELECT * FROM EMPLOYEES
         WHERE DEPARTMENT_ID = 50;
    -- Declara un registre assicuat al cursor
    v_emp_record cur_emps%ROWTYPE;
BEGIN
    -- Obre el cursor
    OPEN cur_emps;

    -- Obté la primera fila
    FETCH cur_emps INTO v_emp_record;

    -- Bucle que recórrer el cursor
    WHILE cur_emps%FOUND LOOP
        -- Mostra les dades de la fila
        DBMS_OUTPUT.PUT_LINE(v_emp_record.EMPLOYEE_ID || ' ' ||
                             v_emp_record.LAST_NAME);
        -- Obté les dades de la fila següent
        FETCH cur_emps INTO v_emp_record;
    END LOOP;
    CLOSE cur_emps;
END;
```

```
124 Mourgos
141 Rajs
142 Davies
143 Matos
144 Vargas

Statement processed.


0.02 seconds
```

6. Recórrer les dades d'un cursor amb un bucle FOR..IN

SINTAXI

```
FOR nom_registre IN cursor LOOP
    sentències;
END LOOP;
```

Simplifica MOLT treballar amb cursors:

- ✓ Obre el cursor automàticament → no fa falta open
- ✓ Per a cada iteració del bucle fa el FETCH d'una fila al registre
- ✓ Acaba el bucle quan s'acaben les files
- ✓ Tanca el cursor automàticament → no fa falta close
- ✓ No fa falta declarar el record. El declara implícitament el FOR com a cursor%ROWTYPE, però només és vàlid dintre del FOR.

EXEMPLE

```
DECLARE
    CURSOR cur_emps IS
        SELECT * FROM EMPLOYEES
           WHERE DEPARTMENT_ID = 50;
BEGIN
    FOR v_emp_record IN cur_emps LOOP
            DBMS_OUTPUT.PUT_LINE(v_emp_record.EMPLOYEE_ID || ' ' ||
                                 v_emp_record.LAST_NAME);
    END LOOP;
END;
```

```
124 Mourgos
141 Rajs
142 Davies
143 Matos
144 Vargas

Statement processed.


0.05 seconds
```

Màxima simplificació: no declarar el cursor

Es pot simplificar encara més sense declarar el cursor, posant directament al FOR la sentència SELECT en la que es basa el CURSOR.

Tècnicament, el SELECT dintre del for és una subconsulta, per tant, ha d'anar entre parèntesis:

```
BEGIN
    FOR v_emp_record IN (SELECT * FROM EMPLOYEES
                         WHERE DEPARTMENT_ID = 50)
    LOOP
        DBMS_OUTPUT.PUT_LINE(v_emp_record.EMPLOYEE_ID || ' ' ||
                             v_emp_record.LAST_NAME);
    END LOOP;
END;
```

L'inconvenient és què, al no estar el cursor declarat explícitament, no es pot accedir a les seves propietats.

EXERCICIS 1, 2, 3 i 4

7. Cursors amb paràmetres

- ✓ Un **PARÀMETRE** és una *variable* utilitzada en la declaració d'un cursor.
- ✓ Utilitzats per enviar informació al cursor.
- ✓ Quan el cursor s'obre, es dona valor als paràmetres: `OPEN nom_cursor(paràmetres);`
- ✓ En el cas del bucle FOR, com l'OPEN va implícit al bucle, se li dona valor al paràmetre a l'IN:

```
FOR v_country_record IN cur_country(2) LOOP
```

- ✓ Els cursor només admeten **paràmetres d'entrada** i només són vàlids dintre del cursor.
- ✓ Un cursor no retorna cap valor.
- ✓ Cada cop que obrim el cursor el paràmetre pot tenir un valor diferent, de forma que el cursor pot retornar diferents resultats en funció del valor del paràmetre.

EXEMPLE

Crear un cursor que mostri l'id i el nom dels països d'una determinada regió.

```
DECLARE
    CURSOR cur_country(p_region_id NUMBER) IS
        SELECT COUNTRY_ID, COUNTRY_NAME
        FROM COUNTRIES
        WHERE REGION_ID = p_region_id;
    v_country_record cur_country%ROWTYPE;
BEGIN
    FOR v_country_record IN cur_country(2) LOOP
        DBMS_OUTPUT.PUT_LINE(v_country_record.country_id || ' ' ||
                             v_country_record.country_name);
    END LOOP;
END;
```

DECLARAR DEL CURSOR

```
CURSOR cur_country(p_region_id NUMBER) IS
    SELECT COUNTRY_ID, COUNTRY_NAME
    FROM COUNTRIES
    WHERE REGION_ID = p_region_id;
```

S'afegeix darrera del nom del cursor la llista de paràmetres entre parèntesi.

- ✓ Els paràmetres poden ser de qualsevol tipus escalar. No fa falta indicar la mida.
- ✓ Els paràmetres s'utilitzen a la clàusula **WHERE** del SELECT del cursor.

OBRIR EL CURSOR

```
OPEN cur_country(2);
```

- ✓ Es dona valor a cada paràmetre del cursor.
- ✓ Es pot obrir el cursor vàries vegades, donant un valor diferent al paràmetre cada cop, obtenint, per tant, valors diferents.

8. *Multiple* cursor

En aplicacions reals, serà necessari utilitzar més d'un cursor en el mateix bloc PL/SQL. Molts cops els **cursors** estan **relacionats**: les dades que s'obtenen amb un cursor són paràmetres duna altre cursor.

EXERCICIS 5, 6 i 7