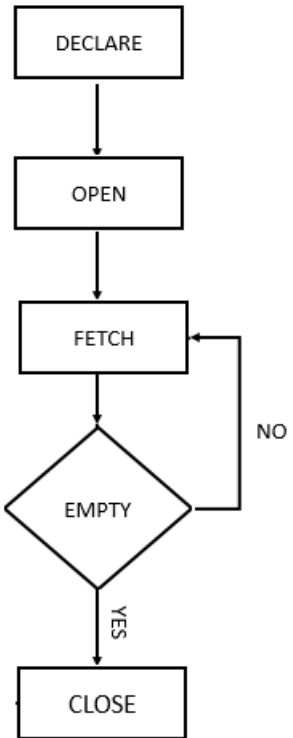


## Cursors

Definició	Format
<p>Un cursor és una punter que apunta al resultat d'una query. PL/SQL té 2 tipus de cursors, implícits i explícits</p> <p><b>Implicit cursors</b></p> <p>Quan Oracle- <i>PL/SQL</i> executa una sentència com : SELECT INTO, INSERT, UPDATE, and DELETE, automàticament crea un cursor implícit.</p> <p>Oracle gestiona internament el cicle d'execució d'un cursor però només permet obtenir informació de la query lligada al cursor i del seu estat.</p> <p>Els cursors implícits no són elegants quan la query retorna 0 o més d'una fila, ja que això provoca les excepcions: NO_DATA_FOUND o TOO_MANY_ROWS respectivament.</p> <p><b>Explicit cursors</b></p> <p>Un cursor explícit és una sentència SELECT declarada de manera explicita a la secció DECLARE d'un bloc de PL/SQL</p> <p>D'un cursor EXPLÍCIT es té el control de la seva execució, quan és fa: OPEN, FETCH, i CLOSE.</p>	<p>La següent imatge mostra el cicle d'execució d'un cursor:</p>  <pre> graph TD     DECLARE[DECLARE] --&gt; OPEN[OPEN]     OPEN --&gt; FETCH[FETCH]     FETCH --&gt; EMPTY{EMPTY}     EMPTY -- YES --&gt; CLOSE[CLOSE]     EMPTY -- NO --&gt; FETCH   </pre> <ol style="list-style-type: none"> <li>1. DECLARE CURSOR</li> <li>2. OPEN CURSOR</li> <li>3. FETCH CURSOR MENTRE HI HA FILES</li> <li>4. CLOSE CURSOR</li> </ol>

Cursors EXPLÍCITS	Exemples
<pre> DECLARE   CURSOR &lt;cursor_name&gt; IS &lt;SELECT statement^&gt;   &lt;cursor_variable declaration&gt; BEGIN   OPEN &lt;cursor_name&gt;;   FETCH &lt;cursor_name&gt; INTO &lt;cursor_variable&gt;;   .   .   CLOSE &lt;cursor_name&gt;; END; </pre> <p>Tal com es mostra al codi anterior, el cursor EXPLICIT s'ha de definir a la secció declaració d'un bloc de PL/SQL block, es crea específicament per a la select que s'ha definit per al programa.</p> <ul style="list-style-type: none"> <li>En la definició sintàctica que es mostra abans es pot veure que en la secció DECLARE es defineix el cursor i la variable que farem servir per manipular les dades obtingudes pel cursor.</li> <li>La variable associada al cursor s'acostuma a definir com del tipus del cursor: &lt;cursor_name&gt;%ROWTYPE.</li> </ul>	<pre> declare   cursor guru_det is select emp_name from emp;   lv_emp_name emp.emp_name%type; begin   open guru_det;   loop     fetch guru_det into lv_emp_name;     if guru_det%notfound then       exit;     end if;     dbms_output.put_line('Employee Fetched: '                            lv_emp_name);   end loop;   dbms_output.put_line('rows fetched is'                            guru_det%rowcount);   close guru_det; end; </pre> <p>En aquest exemple podem veure com, es declara, s'obre , es recuperen dades i es tanca un cursor.</p>

Cursors EXPLÍCITS	Exemples
<p><b>Declarar el cursor</b></p> <ul style="list-style-type: none"> <li>Declarar el cursos significa simplement crear un context d'execució <u>amb nom</u> per a la SELECT declarada en el cursor. El nom d'aquest context és el nom de cursor.</li> </ul> <p><b>Obrir, OPEN, el cursor</b></p> <ul style="list-style-type: none"> <li>Obrir el cursor implica donar instrucció al servidor de BD per a que assigni espai de memòria per al cursor. Així ja està punt per a recuperar dades perquè té un lloc on guardar-les.</li> </ul> <p><b>Llegir, FETCH, dades del cursor</b></p> <ul style="list-style-type: none"> <li>Quan es fa aquest procés les files « fetched » es guarden a l'espai de memòria reservat pel cursor. La recuperació de dades,files, és d'una en una.</li> <li>Cada sentència fetch recupera les dades d'un registre resultant de la select, els valors d'aquest registre es pot accedir fent servir la variable associada al cursor.</li> </ul> <p><b>Tancar, CLOSE, el cursor</b></p> <ul style="list-style-type: none"> <li>Una vegada s'han recuperat totes les dades, o bé abans si s'escau, s'ha de tancar el cursor per tal d'alliberar l'espai de memòria que s'ha reservat pel context del cursor.</li> </ul> <p>Si es declara un cursos dins un bloc anònim, un procediment o una funció, el cursor es tancarà automàticament quan el bloc de codi acabi la seva execució.</p> <p>NOTA: si es tanca o es fa fetch d'un cursor que no s'ha obert es produirà l'excepció INVALID_CURSOR exception.</p>	<pre> DECLARE CURSOR c_sales IS     SELECT * FROM sales ORDER BY total DESC; r_sales c_sales%ROWTYPE;  BEGIN     -- reset credit limit of all customers     UPDATE customers SET credit_limit = 0;      OPEN c_sales;      LOOP         FETCH c_sales INTO r_sales;         EXIT WHEN c_sales%NOTFOUND;          -- update credit for the current customer         UPDATE customers         SET credit_limit =             CASE WHEN 10000 &gt; r_sales.credit                 THEN r_sales.credit             ELSE 10000             END         WHERE customer_id = r_sales.customer_id;     END LOOP;     CLOSE c_sales; END; </pre> <p>En aquest exemple podem veure com, es fa servir un cursor per actualitzat dades.</p>

Cursors EXPLÍCITS	Exemples										
<p><b>Cursor Attributes</b></p> <p>Ambdós cursors: implícits i explícits, tenen atributs que es poden consultar. Aquest atributs donen informació de l'estat d'execució del cursor.</p> <p>A sota hi ha una taula on es mostren aquest atributs i per a que es poden fer servir.</p> <table border="1" data-bbox="159 528 1144 1018"> <thead> <tr> <th>Cursor Attribute</th><th>Description</th></tr> </thead> <tbody> <tr> <td>%FOUND</td><td>It returns the Boolean result 'TRUE' if the most recent fetch operation fetched a record successfully, else it will return FALSE.</td></tr> <tr> <td>%NOTFOUND</td><td>This works oppositely to %FOUND it will return 'TRUE' if the most recent fetch operation could not able to fetch any record.</td></tr> <tr> <td>%ISOPEN</td><td>It returns Boolean result 'TRUE' if the given cursor is already opened, else it returns 'FALSE'</td></tr> <tr> <td>%ROWCOUNT</td><td>It returns the numerical value. It gives the actual count of records that got affected by the DML activity.</td></tr> </tbody> </table> <p>Si un cursor no està obert quan consultem el valor dels atributs:</p> <ul style="list-style-type: none"> <li>• %FOUND, %NOTFOUND, % ROWCOUNT</li> </ul> <p>retornarà <b>INVALID_CURSOR</b>.</p>	Cursor Attribute	Description	%FOUND	It returns the Boolean result 'TRUE' if the most recent fetch operation fetched a record successfully, else it will return FALSE.	%NOTFOUND	This works oppositely to %FOUND it will return 'TRUE' if the most recent fetch operation could not able to fetch any record.	%ISOPEN	It returns Boolean result 'TRUE' if the given cursor is already opened, else it returns 'FALSE'	%ROWCOUNT	It returns the numerical value. It gives the actual count of records that got affected by the DML activity.	<pre> declare     cursor guru_det is select emp_name from emp;     lv_emp_name emp.emp_name%type; begin     open guru_det;     loop         fetch guru_det into lv_emp_name;         if guru_det%notfound then             exit;         end if;         dbms_output.put_line('Employee Fetched: '                                   lv_emp_name);     end loop;     dbms_output.put_line('rows fetched is'                              guru_det%rowcount);     close guru_det; end; </pre> <p>En aquest exemple podem veure com, es fan servir els atributs d'un cursor en un bucle <b>LOOP</b>.</p>
Cursor Attribute	Description										
%FOUND	It returns the Boolean result 'TRUE' if the most recent fetch operation fetched a record successfully, else it will return FALSE.										
%NOTFOUND	This works oppositely to %FOUND it will return 'TRUE' if the most recent fetch operation could not able to fetch any record.										
%ISOPEN	It returns Boolean result 'TRUE' if the given cursor is already opened, else it returns 'FALSE'										
%ROWCOUNT	It returns the numerical value. It gives the actual count of records that got affected by the DML activity.										

Cursors EXPLÍCITS	Exemples
	<pre>DECLARE     v_sal_TOTAL employees.salary%TYPE:=0;     cursor c1 IS SELECT employee_id, salary, commission_pct         FROM employees WHERE commission_pct IS NOT NULL;     var_empleado c1%ROWTYPE; BEGIN     OPEN c1; -- obrir el cursor     FETCH c1 INTO var_empleado; -- llegir 1era vegada     WHILE c1%FOUND LOOP -- mentre hi ha dades         v_sal_TOTAL:=v_sal_TOTAL+var_empleado.salary;         -- seguir llegint fins acabar         FETCH c1 INTO var_empleado;     END LOOP;     CLOSE c1; EXCEPTION     WHEN NO_DATA_FOUND THEN         DBMS_OUTPUT.PUT_LINE ('ERROR: no hi ha dades');     WHEN OTHERS THEN         DBMS_OUTPUT.PUT_LINE ('ERROR!!!!!!!'); END;</pre> <p>En aquest exemple podem veure com, es fan servir els atributs d'un cursor en un bucle <b>WHILE</b>.</p>

Cursors EXPLÍCITS	Exemples
<p><b>Cursor FOR LOOP</b></p> <p>En un bucle FOR LOOP s'executa el codi dins el loop una vegada per a cada enter (ex: <i>for i in 1..10</i>) dins el rang especificat. De manera semblant el cursor FOR LOOP executa el codi dins el loop una vegada per a cada fila resultant de la query associada al cursor.</p> <p>El cursor FOR LOOP crea implícitament una variable d'índex del tipus del cursor i obre el cursor.</p> <p>En cada passada del loop, iteració, el cursor FOR LOOP recupera una fila del resultat que es pot manipular mitjançant la variable d'índex que s'ha creat implícitament. Si al query associada al cursor no retorna cap resultat el cursor FOR LOOP tanca el cursor.</p> <p>El cursor també es tanca si dins el loop hi ha una sentència que passa el control fora del loop, ex: EXIT or GOTO, o bé si es produeix una excepció.</p> <div data-bbox="163 874 654 1109"> <pre>FOR record IN cursor_name LOOP     /* process records*/     ... END LOOP;</pre> </div> <div data-bbox="163 1141 654 1332"> <p><b>1) record</b></p> <p>The record is the name of the index that the cursor FOR LOOP statement declares implicitly as a %ROWTYPE record variable of the type of the cursor.</p> </div> <div data-bbox="665 874 1155 1085"> <p>The record variable is local to the cursor FOR LOOP statement. It means that you can only reference it inside the loop, not outside. After the cursor FOR LOOP statement execution ends, the record variable becomes undefined.</p> </div> <div data-bbox="665 1133 1155 1241"> <p><b>2) cursor_name</b></p> <p>The cursor_name is the name of an explicit cursor that is opened when the loop starts.</p> </div>	<pre>DECLARE     CURSOR c_product     IS         SELECT             product_name, list_price         FROM             products         ORDER BY             list_price DESC;  BEGIN     FOR r_product IN c_product     LOOP         dbms_output.put_line( r_product.product_name    ' :                                    r_product.list_price );     END LOOP; END;</pre> <p>En aquest exemple podem veure com es tracta un cursor dins un bucle FOR</p>

## Cursors EXPLÍCITS

## Exemples

```
BEGIN
FOR r_product IN (
    SELECT
        product_name, list_price
    FROM
        products
    ORDER BY list_price DESC
)
LOOP
    dbms_output.put_line( r_product.product_name ||
                          ': $' || r_product.list_price
    );
END LOOP;
END;
```

Aquest exemple fa el mateix que hem vista a l'exemple anterior però en comptes de declarar el cursor a la secció DECLARE, es posa la sentència del cursor dins la declaració de rang del bucle FOR

## Cursors IMPLÍCITS

Els cursors implícits es creen i destrueixen automàticament, ho fa el servidor Oracle quan s'executa una sentència SQL dins un bloc de PL/SQL.

El servidor ORACLE s'encarrega de fer open, fetch i close sense necessitat de programar-ho explícitament, ho fa per defecte quan s'executa una sentència.

### PL/SQL Single Row Implicit Cursors

- *This type of implicit cursors process at most one row by a SELECT INTO clause with one or more columns assigning them to individual variables or to a collective record type variable.*
- *If the SQL encounters more than one row while processing, the block fails with an ORA-01422: exact fetch returns more than requested number of rows error at run time and %ROWCOUNT yields 1, not the actual number of rows that satisfy the query.*
- *The value of the SQL%ROWCOUNT attribute refers to the most recently executed SQL statement from PL/SQL. To save an attribute value for later use, assign it to a local variable immediately.*

```
DECLARE
    v_id    employees.employee_id%&type;
    v_email employees.email%&type;
BEGIN
    select employee_id, email
    into v_id, v_email
    from employees
    where employee_id=120;
    -- nombre de files
    dbms_output.put_line('total rows' || sql%rowcount);
END;
```

En aquest codi d'exemple es pot veure una SELECT que recupera 1 columnes i assigna el seu valor a 2 variables. El nombre de files resultants d'aquesta query no potser més gran d'1 perquè sinó saltaria una excepció. Però podem consultar quantes files s'han recuperat fent servir l'atribut del cursor: ROWCOUNT , que retornarà 1.

Sí bé això no té gaire sentit fer-ho en una select sí que té més sentit en una sentència d'actualització com la que es mostra a sota:

```
BEGIN
    delete from employees
    where manager_id=120;
    -- nombre de files esborrades
    dbms_output.put_line('total rows' || sql%rowcount);
END;
```

Així podem saber si aquest DELETE ha esborrat unes quantes files o cap



## Cursors IMPLÍCITS

Oracle creates a memory area, known as the context area, for processing an SQL statement, which contains all the information needed for processing the statement; for example, the number of rows processed, etc.

A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set.

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

In PL/SQL, you can refer to the most recent implicit cursor as the SQL cursor, which always has attributes such as %FOUND, %ISOPEN, %NOTFOUND, and %ROWCOUNT.

Any SQL cursor attribute will be accessed as **sql%attribute\_name**

```
DECLARE
    total_rows number(2);
BEGIN
    UPDATE customers
    SET salary = salary + 500;
    IF sql%notfound THEN
        dbms_output.put_line('no customers selected');
    ELSIF sql%found THEN
        total_rows := sql%rowcount;
        dbms_output.put_line
            ( total_rows || ' customers selected ');
    END IF;
END;
```

Cursors amb paràmetres	Exemple
<p>Un cursor explícit pot acceptar paràmetres. Cada vegada que s'obre aquest tipus de cursor se li poden passar paràmetres diferents, això farà que cada vegada pugui retornar un conjunt resultat diferent.</p> <p>Per a declarar cursors amb paràmetres es fer servir la següent sintaxis:</p> <pre>CURSOR cursor_name (parameter_list) IS cursor_query;</pre> <p>A la SELECT associada al cursor es poden fer servir el paràmetres del cursor en qualsevol punt de la query on es pugui posar un valor constant, a la clàusula where per exemple .</p> <p>Els paràmetres del cursor només són accessibles per la query del cursor, no es poden referenciar fora de la query associada al cursor.</p> <p>Per obrir un cursor amb paràmetres la sintaxis és la següent:</p> <pre>OPEN cursor_name (value_list)</pre> <p>Un exemple de cursor amb paràmetres es mostra a la dreta.</p>	<pre>DECLARE r_product products%rowtype; CURSOR c_product (low_price NUMBER, high_price NUMBER) IS SELECT * FROM products WHERE list_price BETWEEN low_price AND high_price;  BEGIN  -- show mass products dbms_output.put_line('Mass products: '); OPEN c_product(50,100); LOOP     FETCH c_product INTO r_product;     EXIT WHEN c_product%notfound;     dbms_output.put_line(r_product.product_name    ': '                           r_product.list_price);  END LOOP; CLOSE c_product;  -- show luxury products dbms_output.put_line('Luxury products: '); OPEN c_product(800,1000); LOOP     FETCH c_product INTO r_product;     EXIT WHEN c_product%notfound;     dbms_output.put_line(r_product.product_name    ': '                           r_product.list_price);  END LOOP; CLOSE c_product;  END;</pre>

## Cursors amb paràmetres

A un cursor amb paràmetres se li poden associar valors de defecte com es mostra a sota:

```
CURSOR cursor_name (
    parameter_name datatype:=default_value,
    parameter_name datatype:=default_value,
    ...
) IS
    cursor_query;
```

Si obres el cursor sense indicar cap paràmetre, s'agafaran els valors de defecte indicats en la declaració del cursor.

## Exemple

```
DECLARE
    CURSOR c_revenue (in_year NUMBER :=2017 ,
                      in_customer_id NUMBER := 1)
    IS
        SELECT SUM(quantity * unit_price) revenue
        FROM order_items
        INNER JOIN orders USING (order_id)
        WHERE status = 'Shipped' AND
              EXTRACT( YEAR FROM order_date) = in_year
        GROUP BY customer_id
        HAVING customer_id = in_customer_id;

    r_revenue c_revenue%rowtype;
BEGIN
    OPEN c_revenue;
    LOOP
        FETCH c_revenue INTO r_revenue;
        EXIT WHEN c_revenue%notfound;
        -- show the revenue
        dbms_output.put_line(r_revenue.revenue);
    END LOOP;
    CLOSE c_revenue;
END;
```

En aquest exemple podem veure com es pot fer OPEN d'un cursor sense passar-li els paràmetres, llavors farà servir els valors de defecte.

Cursors FOR UPDATE	Exemple
<p>Per realitzar actualitzacions a partir d'un cursor cal afegir FOR UPDATE al final de la declaració del cursor:</p> <pre>CURSOR nom_cursor &lt;declaracions&gt; FOR UPDATE</pre> <p>Això indica que les files retornada pel cursor es volen actualitzar o esborrar. Per fer-ho s'inclourà la condició CURRENT OF nom_cursor a la clàusula WHERE. Així es pot esborrar o actualitzar la darrera fila recuperada pel cursor, el darrer FETCH. Sintaxi:</p> <pre>{UPDATE DELETE}... <b>WHERE CURRENT OF</b> nom_cursor.</pre> <p><b><i>LIMITACIONS DE CURSORS FOR UPDATE</i></b></p> <p>No es pot fer servir <i>cursor ...for update</i> si la consulta associada conté:</p> <ul style="list-style-type: none"> <li>• l'operador distinct,</li> <li>• funcions d'agrupació</li> <li>• la clàusula group by</li> <li>• operadors de conjunts: joins</li> </ul>	<p>Exemple:</p> <p>Apujar el sou a tots els empleats del departament passat com a paràmetre d'entrada. El percentatge d'increment també es passarà com a paràmetre</p> <pre>CREATE OR REPLACE PROCEDURE subir_salario     (num_dept NUMBER, incre NUMBER) IS     CURSOR cur IS         SELECT oficio, salario         FROM empleados         WHERE cod_dept=num_dept <b>FOR UPDATE;</b>     reg cur%ROWTYPE;     inc NUMBER (8); BEGIN     OPEN cur;     FETCH cur INTO reg;     WHILE cur%FOUND LOOP         inc :=(reg.salario/100)* incre;         UPDATE empleados SET salario=salario+inc             <b>WHERE CURRENT OF cur;</b>         FETCH cur INTO reg;     END LOOP; END subir_salario;</pre>

## Cursors FOR UPDATE

La sintaxis d'un cursor **SELECT FOR UPDATE** és la següent:

```
CURSOR cursor_name
IS
  cursor_query
  FOR UPDATE [OF column_list] [NOWAIT];
```

Els paràmetres són:

- cursor\_name

El nom del cursor.

- select\_statement

La select que donarà lloc a les files de resultat.

- column\_list

Opcional. La llista de columnes, d'entre les columnes de la select, que es volen actualitzar.

- NOWAIT

Opcional. El cursor no espera que els recursos estiguin lliures.

*The NOWAIT method is used for retrieval, so when the data is found to be locked by another session, it will quickly return the ORA-00054 error if the resources are busy*

## Exemple

**sense NOWAIT**

Un cursor FOR UPDATE que no ha especificat l'opció NOWAIT, es pot trobar en la següent situació:

- SESSIO 1, executa una select for update del treballador 7900. Per aquesta sessió el resultat es que recupera la fila demanada.

```
SQL> /*-----*
SQL> * Session 1: SELECT...FOR UPDATE *
SQL> *-----*/
SQL> select empno, ename, job
  2  from emp
  3  where empno = 7900 for update;

   EMPNO  ENAME          JOB
-----
   7900  JAMES          CLERK

SQL>
```

- SESSIO 2, executa una select for update del treballador 7900. Es queda penjada esperant que la SESSIO 1 alliberi el recurs, això no passarà fins que la SESSIO 1 faci commit o rollback.

```
SQL> /*-----*
SQL> * Session 2: SELECT...FOR UPDATE *
SQL> *-----*/
SQL> select empno, ename, job
  2  from emp
  3  where empno = 7900 for update;
```

## Cursors FOR UPDATE

Exemple de FOR UPDATE d'una columna en concret:

```
CURSOR c_emp
IS
  SELECT employee_id, manager_id
  FROM employees
  FOR UPDATE OF manager_id;
```

*The SELECT FOR UPDATE statement allows you to lock the records in the cursor result set. You are not required to make changes to the records in order to use this statement. The record locks are released when the next commit or rollback statement is issued.*

*The FOR UPDATE clause is generally used in cases where an online system needs to display a set of row data on a screen and they need to ensure that the data does not change before the end-user has an opportunity to update the data.*

*However, the FOR UPDATE clause in SQL will cause the SQL to "hang" if one of the requested rows is locked by another user. If you try to access the rows with the NOWAIT clause, you will get an error message, ORA-00054 Resource busy and acquire with NOWAIT specified.*

## Exemple

**amb NOWAIT**

Un cursor FOR UPDATE que ha especificat l'opció NOWAIT, es pot trobar en la següent situació:

- SESSIO 1, executa una select for update del treballador 7900. Per aquesta sessió el resultat es que recupera la fila demanada.

```
SQL> /*-----*
SQL> * Session 1: SELECT...FOR UPDATE *
SQL> *-----*/
SQL> select empno, ename, job
  2  from emp
  3  where empno = 7900 for update;

   EMPNO  ENAME          JOB
-----
   7900  JAMES          CLERK

SQL>
```

- SESSIO 2, executa una select for update del treballador 7900. Li retorna un error.

```
SQL> /*-----*
SQL> * Session 2: SELECT...FOR UPDATE NOWAIT *
SQL> *-----*/
SQL> select empno, ename, job
  2  from emp
  3  where empno = 7900 for update nowait;
select empno, ename, job
*
ERROR at line 1:
ORA-00054: resource busy and acquire with NOWAIT
specified or timeout expired

SQL>
```

Ha trobat el recurs ocupat i no s'ha esperat.

ROWID FOR UPDATE	Exemple
<p>Fer servir ROWID és la manera més ràpida d'accedir una fila de dades.</p> <p>Si estàs preparant un program per :</p> <ul style="list-style-type: none"> <li>• accedir a les dades d'una fila</li> <li>• fer algun processament</li> <li>• i com a conseqüència actualitzar la fila</li> <li>• tot en la mateixa transacció</li> </ul> <p>es pot millorar la «performance» fent servir la ROWID.</p> <pre>SELECT oficio, salario , ROWID FROM empleados WHERE cod_dept=num_dept</pre> <p><i>For each row in the database, the ROWID pseudocolumn returns the address of the row. Oracle Database rowid values contain information necessary to locate a row:</i></p>	<p>Exemple: procediment <i>apujar el sou</i> fent servir ROWID.</p> <pre>CREATE OR REPLACE PROCEDURE subir_salario (num_dept NUMBER, incre NUMBER) IS CURSOR cur IS SELECT oficio, salario , ROWID FROM empleados WHERE cod_dept=num_dept FOR UPDATE; reg cur%ROWTYPE; inc NUMBER (8); BEGIN OPEN cur; FETCH cur INTO reg; WHILE cur%FOUND LOOP inc :=(reg.salario/100)* incre; UPDATE empleados SET salario=salario+inc WHERE ROWID = reg.ROWID; FETCH cur INTO reg; END LOOP; END subir_salario;</pre>

## Webgrafia

Enllaços web	
<a href="#">...plsql/t_plsql_cursors.htm</a>	Implicit vs. Explicit Cursors
<a href="#">rowids-for-plsql-performance</a>	ROWIDs for PL/SQL Performance
<a href="#">.../plsql/plsql-transaction.php</a>	PL/SQL Transaction Commit, Rollback, Savepoint, Autocommit
<a href="#">...question/rollback-in-oracle-exception-060312/</a>	Rollback in Oracle Exception
<a href="#">When should I use 'for update nowait' in cursors?</a>	When should I use 'for update nowait' in cursors?
<a href="#">.../how-to-lock-a-row-in-oracle/</a>	How to Lock a Row: SELECT FOR UPDATE