2. Cursors

NF3. Excepcions, cursors i transaccions

UF3 - Llenguatges SQL: DCL i extensió procedimental

Cursors

2.1. Cursores

Un **cursor** es un identificador y no una variable, por tanto, sólo hace referencia a una consulta y no se puede asignar valores ni usarse en expresiones.

Hay dos tipos de cursores:

- *Cursores implícitos*, que son los cursores declarados para todas las sentencias DML y PL/SQL SELECT. En este caso una subconsulta devuelve una sola fila; en caso contrario daría error.

- *Cursores explícitos*, que son los cursores declarados y nombrados por el programados. En este caso una consulta devuelve varias filas.

2.2. Cursores implícitos

Los cursores implícitos son los más sencillos de usar y hemos venido usándolos hasta ahora, puesto que están definidos en sentencia DML y PL/SQL.

Atributos	Descripción
SQL%FOUND	Devuelve cierto si el último INSERT, UPDATE, DELETE o SELECT … INTO …, han afecta a una o más filas.
SQL%NOTFOUND	Devuelve cierto si el último INSERT, UPDATE, DELETE o SELECT … INTO …, ha fallado, es decir, no han afectado a ninguna fila.
SQL%ROWCOUNT	Devuelve el número de filas afectadas por el último INSERT, UPDATE, DELETE o SELECT … INTO …
SQL%ISOPEN	Devuelve siempre falso, puesto que Oracle cierra automáticamente el cursor después de cada orden SQL. Pero en caso de error, y quedar abierto algún cursor devolvería cierto.

2.3. Cursores explícitos

Recordemos que estos tipos de cursores se usan para trabajan con consultas que devuelven más de una fila. Estos cursores cuentan con cuatro operaciones básicas para manipularlos y son:

1. **Operación para declarar el cursor**. La sintaxis para declarar un cursor es la siguiente:

```
CURSOR <nombre_cursor> IS SELECT <sentencia_SELECT>;
```

2. **Operación para abrir el cursor**. Cuando se ejecuta la operación de abrir el cursor, se ejecuta automáticamente la sentencia SELECT asociada y el resultado se almacena en las estructuras internas de memoria. La sintaxis para abrir el cursor es la siguiente:

```
OPEN <nombre_cursor>;
```

3. **Operación para recoger la información.** La sintaxis para captar la información, es la siguiente:

```
FETCH <nombre_cursor> INTO {variable|<lista_variables>};
```

donde,

nombre_cursor ⇒ nombre que recibe el cursor.

variable ⇒ se usará para recoger la información de todas las columnas consultadas. Esta variable se ha de declarar de la siguiente manera:

```
<variable> <nombre_cursor>%TYPE
```

lista_variables ⇒ se usa para recoger información de la columna correspondiente a la sentencia SELECT y ha de ser del mismo tipo que las columnas.

Se ha de tener en cuenta que cada FETCH recupera una fila y el cursor avanza automáticamente a la fila siguiente.

4. **Operación de cierre del cursor**. La sintaxis para cerrar la información, es la siguiente:

```
CLOSE <nombre_cursor>;
```

2.3.1. Ejemplo de cursor explícitos

Visualizar todos los empleados que hay en la base de datos.

```
SET SERVEROUTPUT ON
SET VERIFY OFF
SET ECHO OFF
-- Declaración del cursor y variables donde almacenar
-- los datos que contendrá el cursor
DECLARE
 CURSOR c1 IS
  SELECT employee_id, first_name
  FROM employees;
 var_empno employees.employee_id%TYPE;
 var_ename employees.first_name%TYPE;
```

```
BEGIN
 DBMS_OUTPUT.PUT_LINE('Los empleados que hay en nuestra base de
datos son:');
 OPEN c1;  -- se abre el cursor
 DBMS_OUTPUT.PUT_LINE('CODIGO          NOMBRE');
 LOOP  -- bucle para tratar el cursor
  -- se carga en las variables PL/SQL lo que contiene la fila
  -- actual del cursor
  FETCH c1 INTO var_empno,var_ename;
  -- se comprueba si el cursor está vacío. Si está vacío sale del bucle.
  -- Si no está vacío, continúa en el bucle, imprime los datos y vuelve
  -- al FETCH
  EXIT WHEN c1%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE(var_empno||' - '||var_ename);
 END LOOP;
 CLOSE c1;  -- cierra el cursor
END;
/
SET VERIFY ON
SET ECHO ON
```

Ejecución del ejemplo de cursor:

```
Los empleados que hay en nuestra base de datos son:
CODIGO - NOMBRE
100 - Steven
101 - Neena
102 - Lex
103 - Alexander
104 - Bruce
107 - Diana
124 - Kevin
141 - Trenna
142 - Curtis
143 - Randall
144 - Peter
149 - Eleni
174 - Ellen
176 - Jonathan
178 - Kimberely
200 - Jennifer
201 - Michael
202 - Pat
205 - Shelley
206 - William
```

2.3.2. Atributos de cursores explícitos (I)

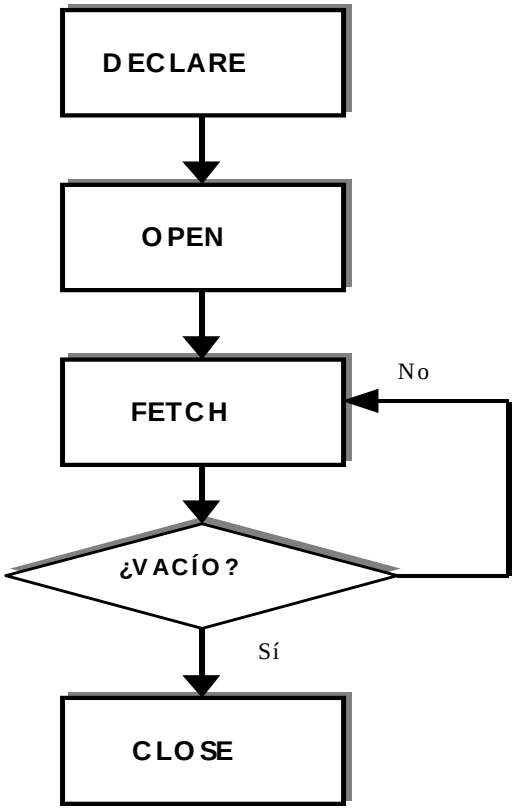
Atributos	Descripción
<nombrecursor>%FOUND	Esta cláusula se suele usar en un bucle como condición y devuelve cierto si el último FETCH ha recuperado algún valor; si no es así, devuelve falso. Si el cursor no estaba abierto devuelve un error y si el curso está abierto, pero no se ha ejecutado ningún FETCH devuelve NULL.
<nombrecursor>%NOTFOUND	Devuelve falso si el último FETCH ha recuperado algún valor y cierto en caso contrario. Es el caso contrario al %FOUND y se usa como condición para salir del bucle.
<nombrecursor>%ROWCOUNT	Devuelve el número de filas recuperadas por el cursor.
<nombrecursor>%ISOPEN	Devuelve verdadero si el cursor está abierto.

Atributos de cursores explícitos (II)

		%FOUND	%ISOPEN	%NOTFOUND	%ROWCOUNT
OPEN	Antes	Invalid_cursor	Falso	Invalid_cursor	Invalid_cursor
	Después	NULL	Verdadero	NULL	0
Primer FETCH	Antes	NULL	Verdadero	Falso	0
	Después	Verdadero	Verdadero	Falso	1
Siguientes FETCH	Antes	Verdadero	Verdadero	Falso	1
	Después	Verdadero	Verdadero	Falso	…
Último FETCH	Antes	Verdadero	Verdadero	Falso	N
	Después	Falso	Verdadero	Verdadero	N
CLOSE	Antes	Falso	Verdadero	Verdadero	N
	Después	Invalid_cursor	Falso	Invalid_cursor	Invalid_cursor

2.4. La estructura cursor FOR ... LOOP

1) Declarar el cursor.
2) Declarar una variable que recogerá los datos del cursor.
3) Abrir el cursor.
5) Recuperar las filas extraídas de la consulta con el FETCH, introduciendo los datos en la variable, procesándolos y comprobando si hay datos recuperados o no.
5) Cerrar el cursor.



Aunque no es difícil realizar estos cinco pasos para trabajar con un cursor, PL/SQL cuenta con la estructura cursor `FOR ... LOOP`, que simplifica los cinco pasos en:

1) Declarar el cursor.
2) Procesar el cursor con la nueva estructura: abre, recorre y cierra el cursor.

La sintaxis de la estructura es la siguiente:

```
FOR <nombre_var> IN <nombre_cursor> LOOP
    <instrucciones>
END LOOP;
```

2.4.1. Ejemplo de cursor (usando la estructura FOR)

Visualizar todos los empleados ordenados por el nombre ascendentemente, que hay en la base de datos.

```
SET SERVEROUTPUT ON
SET VERIFY OFF
SET ECHO OFF
-- Declaración del cursor y variables donde almacenar
-- los datos que contendrá el cursor
DECLARE
 CURSOR c1 IS
  SELECT employee_id, first_name
  FROM employees;
  ORDER BY first_name;
```

```
BEGIN
 DBMS_OUTPUT.PUT_LINE('Los empleados que hay en nuestra base de
datos son:');
 DBMS_OUTPUT.PUT_LINE('CODIGO          NOMBRE');
 -- Estructura FOR, que abre el cursor, lo recorre y lo cierra.
 -- La variable var no hace falta declararla
 FOR var IN c1 LOOP
  DBMS_OUTPUT.PUT_LINE(var.employee_id ||' - ' ||var.first_name);
 END LOOP;
END;
/
SET VERIFY ON
SET ECHO ON
```

Ejecución del ejemplo de cursor:

```
Los empleados que hay en nuestra base de datos son:
CODIGO -   NOMBRE
103 - Alexander
104 - Bruce
142 - Curtis
107 - Diana
149 - Eleni
174 - Ellen
200 - Jennifer
176 - Jonathan
124 - Kevin
178 - Kimberely
102 - Lex
201 - Michael
101 - Neena
202 - Pat
144 - Peter
143 - Randall
205 - Shelley
100 - Steven
141 - Trenna
206 - William
```

2.5. Cursores para actualizar las filas

Hasta ahora hemos estado trabajando con cursores para seleccionar datos, pero también se puede usar dichos cursores para actualizar filas. Para ello, tenemos la cláusula `FOR UPDATE`, que tiene la siguiente sintaxis:

```
CURSOR <nombre_cursor> <declaración_del_cursor> FOR UPDATE
```

donde,

- *`nombre_cursor`* ⇒ nombre que recibe el cursor.
- *`declaración_del_cursor`* ⇒ sentencia SQL que se va a asociar al cursor.
- *`FOR UPDATE`* ⇒ indica que las filas seleccionadas por el cursor van a ser actualizadas o borradas.

Algunas consideraciones a tener en cuenta sobre la cláusula FOR UPDATE:

- Todas las filas seleccionadas serán bloqueadas cuando se abre el cursor y se desbloquearan al terminar las actualizaciones.

- Una vez que se declara un cursor FOR UPDATE, se ha de incluir el parámetro CURRENT OF en la cláusula WHERE. Además, se ha de especificar si se desea actualizar (UPDATE) o borrar (DELETE) filas. La sintaxis es la siguiente:

```
[UPDATE/DELETE ] … WHERE CURRENT OF <nombre_cursor>
```

En algunas ocasiones usar la cláusula `FOR UPDATE` puede plantear algunos problemas, ya que:

- Se bloquean todas las filas de la `SELECT` y no sólo la que se está actualizando.
- Si se ejecuta un `COMMIT`, después ya no se puede ejecutar un `FETCH`.

Para evitar estos problemas se puede usar la cláusula **`ROWID`** en lugar de `FOR UPDATE`, que indica la fila que se va a actualizar. La sintaxis es la siguiente:

```
CURSOR <nombre_cursor> IS SELECT <campos> ROWID FROM <tabla>;
```

Después de ejecutar el `FETCH` se guardará el número de la fila en una variable, de esta manera se podrá usar dicho número después en la cláusula `WHERE` de la actualización o el borrado. La sintaxis de la cláusula `ROWID` en el `WHERE` es la siguiente:

```
[UPDATE/DELETE]… WHERE
    ROWID = variable_que_almacena_número(rowid)
```

donde,

`ROWID` ⇒ devuelve el número de fila donde se encuentra el registro en la tabla.

2.6. Cursores con parámetros

```
CURSOR <nombre_cursor> [(parámetro1, parámetro2, …)]
   IS SELECT <sentecia_SELECT>;
```

donde,

nombre_cursor ⇒ nombre que recibe el cursor.

parámetro1, parámetro2,… ⇒ son los parámetros formales[1] y tiene la siguiente sintaxis:

```
nombre_variable [IN] tipo_dato [{:= | DEFAULT} valor]
```

sentencia_SELECT ⇒ es la sentencia donde intervendrá los parámetros.

[1] Estos parámetros son de entrada y son locales al cursor, por tanto, sólo pueden ser referenciados dentro de la consulta.

Para abrir un cursor con parámetros, contamos con la cláusula OPEN y con la siguiente sintaxis:

```
OPEN <nombre_cursor> [(parámetro1, parámetro2,…)];
```

2.7. Ejemplos de cursor

Ejemplo1: Script que muestre los empleados que hay en la tabla correspondiente (códigos y nombre de los empleados), ordenados por el nombre.

Se incorpora el tratamiento de excepciones, para controlar cuando no hay más empleados en la tabla. La excepción será programada por el usuario.

```
Los empleados que hay en nuestra base de datos son:
CODIGO  -  NOMBRE
103 - Alexander
104 - Bruce
142 - Curtis
107 - Diana
149 - Eleni
174 - Ellen
200 - Jennifer
176 - Jonathan
124 - Kevin
178 - Kimberely
102 - Lex
201 - Michael
101 - Neena
202 - Pat
144 - Peter
143 - Randall
205 - Shelley
100 - Steven
141 - Trenna
206 - William
NO HAY MAS EMPLEADOS
```

```
SET SERVEROUTPUT ON
SET VERIFY OFF
SET ECHO OFF
-- Declaración del cursor y variables donde almacenar
-- los datos que contendrá el cursor.
-- Declaración de la excepción no_filas
DECLARE
 CURSOR c1 IS
  SELECT employee_id, first_name
  FROM employees
  ORDER BY ename;
 var_empno employees.employee_id%TYPE;
 var_ename employees.first_name%TYPE;
 no_filas EXCEPTION;
```

```
BEGIN
 DBMS_OUTPUT.PUT_LINE('Los empleados que hay en nuestra
base de datos son:');
 OPEN c1;
 DBMS_OUTPUT.PUT_LINE('CODIGO - NOMBRE');
 LOOP
  FETCH c1 INTO var_empno,var_ename;
  IF c1%NOTFOUND THEN
    RAISE no_filas;
    EXIT;
  END IF;
  DBMS_OUTPUT.PUT_LINE(var_empno||' - '||var_ename);
 END LOOP;
 CLOSE c1;
```

```
-- Tratamiento de las excepciones
EXCEPTION
 WHEN no_filas THEN
  DBMS_OUTPUT.PUT_LINE ('NO HAY MAS EMPLEADOS');
 WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE ('ERROR!!!(sin definir)');
END;
/
SET VERIFY ON
SET ECHO ON
```

Ejemplo2: Script que muestre los empleados que hay en la tabla correspondiente (código, nombre, salario y oficio), ordenados por el nombre ascendentemente. Usando la estructura `OPEN – FETCH – CLOSE.`

```
Los empleados que hay en nuestra base de datos son:
CODIGO - NOMBRE - SALARIO - OFICIO
----------------------------------
103 - Alexander - 9000 - IT_PROG
104 - Bruce - 6000 - IT_PROG
142 - Curtis - 3100 - ST_CLERK
107 - Diana - 4200 - IT_PROG
149 - Eleni - 10500 - SA_MAN
174 - Ellen - 11000 - SA_REP
200 - Jennifer - 4400 - AD_ASST
176 - Jonathan - 8600 - SA_REP
124 - Kevin - 5800 - ST_MAN
178 - Kimberely - 7000 - SA_REP
102 - Lex - 17000 - AD_VP
201 - Michael - 13000 - MK_MAN
101 - Neena - 17000 - AD_VP
202 - Pat - 6000 - MK_REP
144 - Peter - 2500 - ST_CLERK
143 - Randall - 2600 - ST_CLERK
205 - Shelley - 12000 - AC_MGR
100 - Steven - 24000 - AD_PRES
141 - Trenna - 3500 - ST_CLERK
206 - William - 8300 - AC_ACCOUNT
```

```
SET SERVEROUTPUT ON
SET VERIFY OFF
SET ECHO OFF
-- Declaración del cursor y variables donde almacenar
-- los datos del cursor
DECLARE
 CURSOR c1 IS
  SELECT employee_id, first_name, salary, job_id
  FROM employees
  ORDER BY first_name;
 var_empno employees.employee_id%TYPE;
 var_ename employees.first_name%TYPE;
 var_salario employees.salarry%TYPE;
 var_oficio employees.job_id%TYPE;
```

```
BEGIN
 DBMS_OUTPUT.PUT_LINE('Los empleados que hay en nuestra base
de datos son:');
 OPEN c1;
  DBMS_OUTPUT.PUT_LINE('CODIGO - NOMBRE - SALARIO - OFICIO');
  DBMS_OUTPUT.PUT_LINE('--------------------------------');
 LOOP
  FETCH c1 INTO var_empno, var_ename, var_salario, var_oficio;
  EXIT WHEN c1%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE(var_empno||' - '||var_ename||' -
'||var_salario||' - '||var_oficio);
 END LOOP;
 CLOSE c1;
END;
/
SET VERIFY ON
SET ECHO ON
```

Ejemplo3: Script que muestre los empleados que hay en la tabla correspondiente (código, nombre, salario y oficio), ordenado por el nombre ascendentemente. Usando una variable del tipo:

```
var_empleado c1%ROWTYPE.
```

```
Los empleados que hay en nuestra base de datos son:
CODIGO - NOMBRE - SALARIO - OFICIO
----------------------------------
103 - Alexander - 9000 - IT_PROG
104 - Bruce - 6000 - IT_PROG
142 - Curtis - 3100 - ST_CLERK
107 - Diana - 4200 - IT_PROG
149 - Eleni - 10500 - SA_MAN
174 - Ellen - 11000 - SA_REP
200 - Jennifer - 4400 - AD_ASST
176 - Jonathan - 8600 - SA_REP
124 - Kevin - 5800 - ST_MAN
178 - Kimberely - 7000 - SA_REP
102 - Lex - 17000 - AD_VP
201 - Michael - 13000 - MK_MAN
101 - Neena - 17000 - AD_VP
202 - Pat - 6000 - MK_REP
144 - Peter - 2500 - ST_CLERK
143 - Randall - 2600 - ST_CLERK
205 - Shelley - 12000 - AC_MGR
100 - Steven - 24000 - AD_PRES
141 - Trenna - 3500 - ST_CLERK
206 - William - 8300 - AC_ACCOUNT
```

```
SET SERVEROUTPUT ON
SET VERIFY OFF
SET ECHO OFF
-- Declaración del cursor y la variable donde almacenar
-- los datos del cursor
DECLARE
 CURSOR c1 IS
  SELECT employee_id, first_name, salary, job_id
  FROM employees
  ORDER BY first_name;
 var_empleado c1%ROWTYPE;
```

```
BEGIN
 DBMS_OUTPUT.PUT_LINE('Los empleados que hay en nuestra base
de datos son:');
 OPEN c1;
  DBMS_OUTPUT.PUT_LINE('CODIGO - NOMBRE - SALARIO - OFICIO');
  DBMS_OUTPUT.PUT_LINE('--------------------------------');
 LOOP
  FETCH c1 INTO var_empleado;
  EXIT WHEN c1%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE(var_empleado.employee_id||' -
'||var_empleado.first_name||' - '||var_empleado.salary||' -
'||var_empleado.job_id);
 END LOOP;
 CLOSE c1;
END;
/
SET VERIFY ON
SET ECHO ON
```

Ejemplo4: Script que muestre los empleados que hay en la tabla correspondiente (código, nombre, salario y oficio), ordenados por el salario descendentemente. Usando la estructura FOR.

```
Los empleados que hay en nuestra base de datos son:
CODIGO - NOMBRE - SALARIO - OFICIO
------------------------------
100 - Steven - 24000 - AD_PRES
101 - Neena - 17000 - AD_VP
102 - Lex - 17000 - AD_VP
201 - Michael - 13000 - MK_MAN
205 - Shelley - 12000 - AC_MGR
174 - Ellen - 11000 - SA_REP
149 - Eleni - 10500 - SA_MAN
103 - Alexander - 9000 - IT_PROG
176 - Jonathan - 8600 - SA_REP
206 - William - 8300 - AC_ACCOUNT
178 - Kimberely - 7000 - SA_REP
104 - Bruce - 6000 - IT_PROG
202 - Pat - 6000 - MK_REP
124 - Kevin - 5800 - ST_MAN
200 - Jennifer - 4400 - AD_ASST
107 - Diana - 4200 - IT_PROG
141 - Trenna - 3500 - ST_CLERK
142 - Curtis - 3100 - ST_CLERK
143 - Randall - 2600 - ST_CLERK
144 - Peter - 2500 - ST_CLERK
```

```
SET SERVEROUTPUT ON
SET VERIFY OFF
SET ECHO OFF
DECLARE
 CURSOR c1 IS
 SELECT employee_id, first_name, salary, job_id
  FROM employees
  ORDER BY salary DESC;
BEGIN
 DBMS_OUTPUT.PUT_LINE('Los empleados que hay en nuestra base
de datos son:');
 DBMS_OUTPUT.PUT_LINE('CODIGO - NOMBRE - SALARIO - OFICIO');
 DBMS_OUTPUT.PUT_LINE('-----------------------------');
 FOR var IN c1 LOOP
  DBMS_OUTPUT.PUT_LINE(var.employee_id||' -
'||var.first_name||' - '||var.salary||' - '||var.job_id);
 END LOOP;
END;
/
SET VERIFY ON
SET ECHO ON
```

Ejemplo5: Modifica el salario de los empleados que tienen la comisión NULA, la modificación es sumarle al salario la comisión.

EMPLOYEE_ID	SALARY	COMMISSION_PCT
100	24000	0,2
149	10500	0,2
174	11000	0,3
176	8600	0,2
178	7000	0,15

EMPLOYEE_ID	SALARY	COMMISSION_PCT
100	24000,2	0,2
149	10500,2	0,2
174	11000,3	0,3
176	8600,2	0,2
178	7000,15	0,15

```
SET SERVEROUTPUT ON
SET VERIFY OFF
SET ECHO OFF
-- Declaración del cursor en modo actualización
-- y la variable donde almacenar los datos del cursor
DECLARE
 var_salario employees.salary%TYPE;
 CURSOR c1 IS
  SELECT employee_id, salary, commission_pct
  FROM employees
  WHERE commission_pct IS NOT NULL
  FOR UPDATE;
 var_empleado c1%ROWTYPE;
```

```
BEGIN
 OPEN c1;  -- abrir el cursor
 FETCH c1 INTO var_empleado; -- leer el cursor
 -- si está lleno el cursor
 WHILE c1%FOUND LOOP
  var_salario:= var_empleado.salary + var_empleado.commission_pct;
  -- actualización del campo de la tabla física de la base de
  -- datos, teniendo en cuenta que sólo se actualizará los
  -- registros que están cargados en el cursor (comisión no nula)
  UPDATE employees SET salary = var_salario
   WHERE CURRENT OF c1;
  FETCH c1 INTO var_empleado;
 END LOOP;
 CLOSE c1;
```

```
EXCEPTION
 WHEN NO_DATA_FOUND THEN
  DBMS_OUTPUT.PUT_LINE ('ERROR: no hay datos en la tabla');
 WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE ('ERROR!!!!!!!');
END;
/
SET VERIFY ON
SET ECHO ON
```

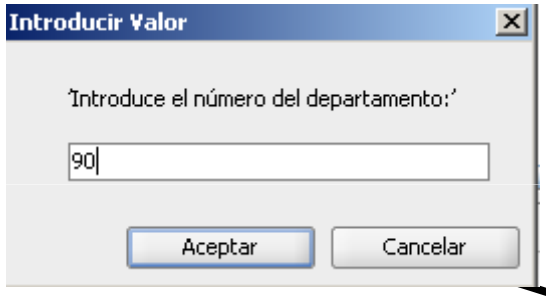
Ejemplo6: Script que modifique el salario de los empleados que tienen la comisión NULA, la modificación es sumarle al salario la comisión (igual que el ejemplo anterior). Usando la cláusula ROWID.

```
SET SERVEROUTPUT ON
SET VERIFY OFF
SET ECHO OFF
-- Declaración del cursor en modo actualización (incluyendo el ROWID)
y la variable donde almacenar los datos del cursor
DECLARE
 var_salario employees.salary%TYPE;
 CURSOR c1 IS
  SELECT employee_id, salary, commission_pct, ROWID
  FROM employees
  WHERE commission_pct IS NOT NULL
  FOR UPDATE;
 var_empleado c1%ROWTYPE;
```

```
BEGIN
 OPEN c1;
 FETCH c1 INTO var_empleado;
 WHILE c1%FOUND LOOP
   UPDATE employees
   SET salary = var_empleado.salary + var_empleado.commission_pct
   WHERE ROWID = var_empleado.ROWID;
   FETCH c1 INTO var_empleado;
 END LOOP;
 CLOSE c1;
EXCEPTION
 WHEN NO_DATA_FOUND THEN
  DBMS_OUTPUT.PUT_LINE ('ERROR: no hay datos en la tabla');
 WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE ('ERROR!!!!!!!');
END;
/
SET VERIFY ON
SET ECHO ON
```

Ejemplo7: Script que muestre todos los empleados que pertenecen a un departamento en concreto (código, nombre del empleado y número del departamento al que pertenece).

El usuario introducirá por teclado el código del departamento y se le pasará como parámetro al cursor.




```
bloque anónimo terminado
El empleado es 100 se llama Steven y trabaja en el departamento 90
El empleado es 101 se llama Neena y trabaja en el departamento 90
El empleado es 102 se llama Lex y trabaja en el departamento 90
```

```
SET SERVEROUTPUT ON
SET VERIFY OFF
SET ECHO OFF
ACCEPT var_numemp PROMPT 'Introduce el número del empleado: '
ACCEPT var_numdep PROMPT 'Introduce el número del departamento:'
-- Declaración del cursor incluyendo el parámetro que se le pasará
-- y la variable donde almacenar los datos del cursor
DECLARE
 CURSOR c1
  (var_deptno employees.department_id%TYPE)
  IS
   SELECT employee_id, first_name, department_id
   FROM employees
   WHERE department_id = var_deptno;
 var_empleado c1%ROWTYPE;
```

```
BEGIN
 OPEN c1(&var_numdep); -- abrir cursor con parámetro
 LOOP
  FETCH c1 INTO var_empleado;
  DBMS_OUTPUT.PUT_LINE ('El empleado es '||
var_empleado.employee_id ||' se llama '||
var_empleado.first_name ||' y trabaja en el departamento '||
var_empleado.department_id);
  EXIT WHEN c1%NOTFOUND;
 END LOOP;
 CLOSE c1;
EXCEPTION
 WHEN NO_DATA_FOUND THEN
  DBMS_OUTPUT.PUT_LINE ('ERROR: no hay datos en la tabla');
 WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE ('ERROR!!!!!!!');
END;
/
SET VERIFY ON
SET ECHO ON
```

Preguntes!!!!!