Tema 10

1.- Ejemplos de creación de procedimientos con cursores.

1) Desarrollar un procedimiento que visualice el apellido y la fecha de alta de todos los empleados ordenados por apellido.

```
CREATE OR REPLACE PROCEDURE ver_emple

AS

CURSOR c_emple IS

SELECT APELLIDO, FECHA_ALT

FROM EMPLE

ORDER BY APELLIDO;

v_apellido VARCHAR2(10);

v_fecha DATE;

BEGIN

OPEN c_emple;

FETCH c_emple into v_apellido, v_fecha;

WHILE c_emple%FOUND LOOP

DBMS_OUTPUT.PUT_LINE( v_apellido||' * '||v_fecha);

FETCH c_emple into v_apellido,v_fecha;

END LOOP;

CLOSE c_emple;

END ver_emple;
```

2) Codificar un procedimiento que muestre el nombre de cada departamento y el número de empleados que tiene.

```
CREATE OR REPLACE PROCEDURE ver_emple_depart

AS
```

```
CURSOR c_emple IS
SELECT dnombre, COUNT(emp_no)
FROM emple e, depart d
WHERE d.dept_no = e.dept_no(+)
GROUP BY dnombre;
v_dnombre depart.dnombre%TYPE;
v_num_emple BINARY_INTEGER;
BEGIN
OPEN c_emple;
FETCH c_emple into v_dnombre, v_num_emple;
WHILE c_emple%FOUND LOOP
DBMS_OUTPUT.PUT_LINE(v_dnombre||' * '||v_num_emple);
FETCH c_emple into v_dnombre,v_num_emple;
END LOOP;
CLOSE c_emple;
END ver_emple_depart;
```

3) Escribir un programa que visualice el apellido y el salario de los cinco empleados que tienen el salario más alto.

```
CREATE OR REPLACE PROCEDURE emp_5maxsal
AS
CURSOR c_emp IS
SELECT apellido, salario FROM emple
ORDER BY salario DESC;
vr_emp c_emp%ROWTYPE;
i NUMBER;
BEGIN
```

```
i:=1;
OPEN c_emp;
FETCH c_emp INTO vr_emp;
WHILE c_emp%FOUND AND i<=5 LOOP
DBMS_OUTPUT.PUT_LINE(vr_emp.apellido ||
' * '|| vr_emp.salario);
FETCH c_emp INTO vr_emp;
i:=I+1;
END LOOP;
CLOSE c_emp;
END emp_5maxsal;
```

OPEN...FETCH...

2 .- Ejemplos de como como recorrer un cursor.

4) Codificar un programa que visualice los dos empleados que ganan menos de cada oficio.

```
CREATE OR REPLACE PROCEDURE emp_2minsal
AS
CURSOR c_emp IS
SELECT apellido, oficio, salario FROM emple
ORDER BY oficio, salario;
vr_emp c_emp%ROWTYPE;
oficio_ant EMPLE.OFICIO%TYPE;
i NUMBER;
BEGIN
OPEN c_emp;
```

```
oficio_ant:='*';
FETCH c_emp INTO vr_emp;
WHILE c_emp%FOUND LOOP
IF oficio_ant <> vr_emp.oficio THEN
oficio_ant := vr_emp.oficio;
i := 1;
END IF;
IF i <= 2 THEN
DBMS_OUTPUT.PUT_LINE(vr_emp.oficio||' * '
||vr_emp.apellido||' * '
||vr_emp.salario);
END IF;
FETCH c_emp INTO vr_emp;
i:=I+1;
END LOOP;
CLOSE c_emp;
END emp_2minsal;
```

5) Desarrollar un procedimiento que permita insertar nuevos departamentos según las siguientes especificaciones:

Se pasará al procedimiento el nombre del departamento y la localidad.

El procedimiento insertará la fila nueva asignando como número de departamento la decena siguiente al número mayor de la tabla.

Se incluirá gestión de posibles errores.

```
CREATE OR REPLACE PROCEDURE insertar_depart(
nombre_dep VARCHAR2,
```

```
loc VARCHAR2)
AS
CURSOR c_dep IS SELECT dnombre
FROM depart WHERE dnombre = nombre_dep;
v_dummy DEPART.DNOMBRE%TYPE DEFAULT NULL;
v_ulti_num DEPART.DEPT_NO%TYPE;
nombre_duplicado EXCEPTION;
BEGIN
/* Comprobación de que el departamento no está duplicado */
OPEN c_dep;
FETCH c_dep INTO v_dummy;
CLOSE c_dep;
IF v_dummy IS NOT NULL THEN
RAISE nombre_duplicado;
END IF;

/* Captura del último número y cálculo del siguiente */
SELECT MAX(dept_no) INTO v_ulti_num FROM depart;

/* Inserción de la nueva fila */
INSERT INTO depart VALUES ((TRUNC(v_ulti_num, -1)+10)
, nombre_dep, loc);
EXCEPTION
WHEN nombre_duplicado THEN
DBMS_OUTPUT.PUT_LINE('Err. departamento duplicado');
RAISE;
WHEN OTHERS THEN
RAISE_APPLICATION_ERROR(-20005,
```

'Err. Operación cancelada');

END insertar_depart;

6) Codificar un procedimiento reciba como parámetros un numero de departamento, un importe y un porcentaje; y suba el salario a todos los empleados del departamento indicado en la llamada. La subida será el porcentaje o el importe indicado en la llamada (el que sea más beneficioso para el empleado en cada caso empleado).

```
CREATE OR REPLACE PROCEDURE subida_sal1(
num_depar emple.dept_no%TYPE,
importe NUMBER,
porcentaje NUMBER)
AS
CURSOR c_sal IS SELECT salario,ROWID
FROM emple WHERE dept_no = num_depar;
vr_sal c_sal%ROWTYPE;
v_imp_pct NUMBER(10);
BEGIN
OPEN c_sal;
FETCH c_sal INTO vr_sal;
WHILE c_sal%FOUND LOOP

/* Guardar en v_imp_pct el importe mayor */
v_imp_pct :=
GREATEST((vr_sal.salario/100)*porcentaje,
v_imp_pct);

/* Actualizar */
UPDATE EMPLE SET SALARIO=SALARIO + v_imp_pct
```

```
WHERE ROWID = vr_sal.rowid;


FETCH c_sal INTO vr_sal;

END LOOP;

CLOSE c_sal;

EXCEPTION

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE('Err. ninguna fila actualizada');

END subida_sal1;
```

7) Escribir un procedimiento que suba el sueldo de todos los empleados que ganen menos que el salario medio de su oficio. La subida será de el 50% de la diferencia entre el salario del empleado y la media de su oficio. Se deberá asegurar que la transacción no se quede a medias, y se gestionarán los posibles errores.

```
CREATE OR REPLACE PROCEDURE subida_50pct

AS

CURSOR c_ofi_sal IS

SELECT oficio, AVG(salario) salario FROM emple

GROUP BY oficio;

CURSOR c_emp_sal IS

SELECT oficio, salario FROM emple E1

WHERE salario <

(SELECT AVG(salario) FROM emple E2

WHERE E2.oficio = E1.oficio)

ORDER BY oficio, salario FOR UPDATE OF salario;


vr_ofi_sal c_ofi_sal%ROWTYPE;

vr_emp_sal c_emp_sal%ROWTYPE;

v_incremento emple.salario%TYPE;
```

```
BEGIN
COMMIT;
OPEN c_emp_sal;
FETCH c_emp_sal INTO vr_emp_sal;
OPEN c_ofi_sal;
FETCH c_ofi_sal INTO vr_ofi_sal;
WHILE c_ofi_sal%FOUND AND c_emp_sal%FOUND LOOP

/* calcular incremento */
v_incremento :=
(vr_ofi_sal.salario - vr_emp_sal.salario) / 2;

/* actualizar */
UPDATE emple SET salario = salario + v_incremento
WHERE CURRENT OF c_emp_sal;

/* siguiente empleado */
FETCH c_emp_sal INTO vr_emp_sal;

/* comprobar si es otro oficio */
IF c_ofi_sal%FOUND and
vr_ofi_sal.oficio <> vr_emp_sal.oficio THEN
FETCH c_ofi_sal INTO vr_ofi_sal;
END IF;
END LOOP;
CLOSE c_emp_sal;
CLOSE c_ofi_sal;
```

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK WORK;

RAISE;

END subida_50pct;

8) Diseñar una aplicación que simule un listado de liquidación de los empleados según las siguientes especificaciones:

- El listado tendrá el siguiente formato para cada empleado:

****

Liquidación del empleado:.....(1) Dpto:.....(2) Oficio:.....(3)

Salario : .....(4)

Trienios :.....(5)

Comp. Responsabil :.....(6)

Comisión :.....(7)

Total :.....(8)

****

- Donde:

1 ,2, 3 y 4 Corresponden al apellido, departamento, oficio y salario del empleado.

5 Es el importe en concepto de trienios. Cada trienio son tres años completos desde la fecha de alta hasta la de emisión y supone 5000 Ptas.

6 Es el complemento por responsabilidad. Será de 10000Ptas por cada empleado que se encuentre directamente a cargo del empleado en cuestión.

7 Es la comisión. Los valores nulos serán sustituidos por ceros.

8 Suma de todos los conceptos anteriores.

– El listado irá ordenado por Apellido.

```
CREATE OR REPLACE PROCEDURE liquidar
AS
CURSOR c_emp IS
SELECT apellido, emp_no, oficio, salario,
NVL(comision,0) comision, dept_no, fecha_alt
FROM emple
ORDER BY apellido;
vr_emp c_emp%ROWTYPE;
v_trien NUMBER(9) DEFAULT 0;
v_comp_r NUMBER(9);
v_total NUMBER(10);
BEGIN
FOR vr_emp in c_emp LOOP

/* Calcular trienios. Llama a la función trienios
creada en el ejercicio 11.8 */
v_trien := trienios(vr_emp.fecha_alt,SYSDATE)*5000;

/* Calcular complemento de responsabilidad. Se
encierra en un bloque pues levantará NO_DATA_FOUND*/
BEGIN
SELECT COUNT(*) INTO v_comp_r
FROM EMPLE WHERE DIR = vr_emp.emp_no;
v_comp_r := v_comp_r *10000;
EXCEPTION
WHEN NO_DATA_FOUND THEN
v_comp_r:=0;
```

```
END;

/* Calcular el total del empleado */
v_total := vr_emp.salario + vr_emp. comision +
v_trien + v_comp_r;

/* Visualizar datos del empleado */
DBMS_OUTPUT.PUT_LINE('************************************');
DBMS_OUTPUT.PUT_LINE(' Liquidacion de : '|| vr_emp.apellido
||' Dpto: ' || vr_emp.dept_no
|| ' Oficio: ' || vr_emp.oficio);
DBMS_OUTPUT.PUT_LINE(RPAD('Salario:',16)
||LPAD(TO_CHAR(vr_emp.salario,'9,999,999'),12));
DBMS_OUTPUT.PUT_LINE(RPAD('Trienios: ',16)
|| LPAD(TO_CHAR(v_trien,'9,999,999'),12));
DBMS_OUTPUT.PUT_LINE('Comp. Respons: '
||LPAD(TO_CHAR(v_comp_r,'9,999,999'),12));
DBMS_OUTPUT.PUT_LINE(RPAD('Comision: ' ,16)
||LPAD(TO_CHAR(vr_emp.comision,'9,999,999'),12));
DBMS_OUTPUT.PUT_LINE('------------------');
DBMS_OUTPUT.PUT_LINE(RPAD(' Total : ',16)
||LPAD(TO_CHAR(v_total,'9,999,999') ,12));
DBMS_OUTPUT.PUT_LINE('************************************');
END LOOP;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('No se ha encontrado ninguna fila');
END liquidar;
```

/* Nota: También se puede utilizar una cláusula SELECT más compleja:

CURSOR c_emp IS

SELECT APELLIDO, EMP_NO, OFICIO,

(EMP_CARGO * 10000) COM_RESPONSABILIDAD,

SALARIO, NVL(COMISION, 0) COMISION, DEPT_NO,

TRIENIOS(FECHA_ALT, SYSDATE) * 5000 TOT_TRIENIOS

FROM EMPLE,(SELECT DIR,COUNT(*) EMP_CARGO FROM EMPLE

GROUP BY DIR) DIREC

WHERE EMPLE.EMP_NO = DIREC.DIR(+)

ORDER BY APELLIDO;

de esta forma se simplifica el programa y se evita la utilización de variables de trabajo. */

9) Crear la tabla T_liquidacion con las columnas apellido, departamento, oficio, salario, trienios, comp_responsabilidad, comisión y total; y modificar la aplicación anterior para que en lugar de realizar el listado directamente en pantalla, guarde los datos en la tabla. Se controlarán todas las posibles incidencias que puedan ocurrir durante el proceso.

CREATE TABLE t_liquidacion (

APELLIDO VARCHAR2(10),

DEPARTAMENTO NUMBER(2),

OFICIO VARCHAR2(10),

SALARIO NUMBER(10),

TRIENIOS NUMBER(10),

COMP_RESPONSABILIDAD NUMBER(10),

COMISION NUMBER(10),

TOTAL NUMBER(10)

);

CREATE OR REPLACE PROCEDURE liquidar2

```
AS
CURSOR c_emp IS
SELECT apellido, emp_no, oficio, salario,
NVL(comision,0) comision, dept_no, fecha_alt
FROM emple
ORDER BY apellido;
vr_emp c_emp%ROWTYPE;
v_trien NUMBER(9) DEFAULT 0;
v_comp_r NUMBER(9);
v_total NUMBER(10);
BEGIN
COMMIT WORK;
FOR vr_emp in c_emp LOOP

/* Calcular trienios. Llama a la función trienios
creada en el ejercicio 11.8 */
v_trien := trienios(vr_emp.fecha_alt,SYSDATE)*5000;

/* Calcular complemento de responsabilidad. Se
encierra en un bloque pues levantará NO_DATA_FOUND*/
BEGIN
SELECT COUNT(*) INTO v_comp_r
FROM EMPLE WHERE DIR = vr_emp.emp_no;
v_comp_r := v_comp_r *10000;
EXCEPTION
WHEN NO_DATA_FOUND THEN
v_comp_r:=0;
END;
```

```
/* Calcular el total del empleado */
v_total := vr_emp.salario + vr_emp. comision +
v_trien + v_comp_r;

/* Insertar los datos en la tabla T_liquidacion */
INSERT INTO t_liquidacion
(APELLIDO, OFICIO, SALARIO, TRIENIOS,
COMP_RESPONSABILIDAD, COMISION, TOTAL)
VALUES
(vr_emp.apellido, vr_emp.oficio, vr_emp.salario,
v_trien, v_comp_r, vr_emp.comision, v_total);
END LOOP;
EXCEPTION
WHEN OTHERS THEN
ROLLBACK WORK;
END liquidar2;
```