Tema 11

1.- Construir un disparador de base de datos que permita auditar las operaciones de inserción o borrado de datos que se realicen en la tabla emple según las siguientes especificaciones:

- En primer lugar se creará desde SQL*Plus la tabla auditaremple con la columna col1 VARCHAR2(200).

- Cuando se produzca cualquier manipulación se insertará una fila en dicha tabla que contendrá:

- Fecha y hora

- Número de empleado

- Apellido

- La operación de actualización INSERCIÓN o BORRADO

```
CREATE TABLE auditaremple (

col1 VARCHAR2(200)

);


CREATE OR REPLACE TRIGGER auditar_act_emp

BEFORE INSERT OR DELETE

ON EMPLE

FOR EACH ROW

BEGIN

IF DELETING THEN

INSERT INTO AUDITAREMPLE

VALUES(TO_CHAR(sysdate,'DD/MM/YY*HH24:MI*')

|| :OLD.EMP_NO|| '*' || :OLD.APELLIDO || '* BORRADO ');
```

```
ELSIF INSERTING THEN

INSERT INTO AUDITAREMPLE

VALUES(TO_CHAR(sysdate,'DD/MM/YY*HH24:MI*')

|| :NEW.EMP_NO || '*' || :NEW.APELLIDO||'* INSERCION ');

END IF;

END;
```

2.- Escribir un trigger de base de datos un que permita auditar las modificaciones en la tabla empleados insertado en la tabla auditaremple los siguientes datos:

- Fecha y hora

- Número de empleado

- Apellido

- La operación de actualización: MODIFICACIÓN.

- El valor anterior y el valor nuevo de cada columna modificada. (solo las columnas modificadas)

```
CREATE OR REPLACE TRIGGER audit_modif

BEFORE UPDATE ON EMPLE

FOR EACH ROW

DECLARE

v_cad_inser auditaremple.col1%TYPE;

BEGIN

v_cad_inser := TO_CHAR(sysdate,'DD/MM/YY*HH24:MI*') ||:OLD.EMP_NO ||'*
MODIFICACION *';


IF UPDATING ('EMP_NO') THEN

v_cad_inser := v_cad_inser
```

```
||:OLD.EMP_NO|| '*'|| :NEW.EMP_NO;
END IF;

IF UPDATING ('APELLIDO') THEN
v_cad_inser := v_cad_inser
||:OLD.APELLIDO|| '*'||:NEW.APELLIDO;
END IF;

IF UPDATING ('OFICIO') THEN
v_cad_inser := v_cad_inser
||:OLD.OFICIO|| '*'||:NEW.OFICIO;
END IF;

IF UPDATING ('DIR') THEN
v_cad_inser := v_cad_inser
||:OLD.DIR|| '*'||:NEW.DIR;
END IF;

IF UPDATING ('FECHA_ALT') THEN
v_cad_inser := v_cad_inser
||:OLD.FECHA_ALT||:NEW.FECHA_ALT;
END IF;

IF UPDATING ('SALARIO') THEN
v_cad_inser := v_cad_inser
||:OLD.SALARIO|| '*'||:NEW.SALARIO;
```

```
END IF;

IF UPDATING ('COMISION') THEN
v_cad_inser := v_cad_inser
||:OLD.COMISION|| '*'||:NEW.COMISION;
END IF;

IF UPDATING ('DEPT_NO') THEN
v_cad_inser := v_cad_inser
||:OLD.DEPT_NO|| '*'||:NEW.DEPT_NO;
END IF;

INSERT INTO AUDITAREMPLE VALUES(v_cad_inser);
END;
```

3.- Suponiendo que disponemos de la vista

```
CREATE VIEW DEPARTAM AS
SELECT DEPART.DEPT_NO, DNOMBRE, LOC, COUNT(EMP_NO) TOT_EMPLE
FROM EMPLE, DEPART
WHERE EMPLE.DEPT_NO (+) = DEPART.DEPT_NO
GROUP BY DEPART.DEPT_NO, DNOMBRE, LOC;
```

Construir un disparador que permita realizar operaciones de actualización en la tabla depart a partir de la vista dptos, de forma similar al ejemplo del trigger t_ges_emplead. Se contemplarán las siguientes operaciones:

- Insertar departamento.

- Borrar departamento.

- Modificar la localidad de un departamento.

```
CREATE OR REPLACE TRIGGER ges_depart

INSTEAD OF DELETE OR INSERT OR UPDATE

ON DEPARTAM

FOR EACH ROW

BEGIN

IF DELETING THEN

DELETE FROM depart WHERE dept_no = :old.dept_no;

ELSIF INSERTING THEN

INSERT INTO depart

VALUES(:new.dept_no, :new.dnombre, :new.loc);

ELSIF UPDATING('loc') THEN

UPDATE depart SET loc = :new.loc

WHERE dept_no = :old.dept_no;

ELSE

RAISE_APPLICATION_ERROR

(-20001,'Error en la actualización');

END IF;

END;
```

4.- Escribir un paquete completo para gestionar los departamentos. El paquete se llamará gest_depart y deberá incluir, al menos, los siguientes subprogramas:

- insertar_nuevo_depart: permite insertar un departamento nuevo. El procedimiento recibe el nombre y la localidad del nuevo departamento. Creará el nuevo departamento comprobando que el nombre no se duplique y le asignará como número de departamento la decena siguiente al último número de departamento utilizado.

- borrar_depart: permite borrar un departamento. El procedimiento recibirá dos números de departamento de los cuales el primero corresponde al departamento que queremos borrar y el segundo al departamento al que pasarán los empleados del departamento que se va eliminar. El procedimiento se encargará de realizar los cambios oportunos en los números de departamento de los empleados correspondientes.

- modificar_loc_depart: modifica la localidad del departamento. El procedimiento recibirá el número del departamento a modificar y la nueva localidad, y realizará el cambio solicitado.

- visualizar_datos_depart: visualizará los datos de un departamento cuyo número se pasará en la llamada. Además de los datos relativos al departamento, se visualizará el número de empleados que pertenecen actualmente al departamento.

- visualizar_datos_depart: versión sobrecargada del procedimiento anterior que, en lugar del número del departamento, recibirá el nombre del departamento. Realizará una llamada a la función buscar_depart_por_nombre que se indica en el apartado siguiente.

- buscar_depart_por_nombre: función local al paquete. Recibe el nombre de un departamento y devuelve el número del mismo.

/********** Cabecera o especificación del paquete **********/

CREATE OR REPLACE PACKAGE gest_depart AS

```
PROCEDURE insert_depart
(v_nom_dep VARCHAR2,
v_loc VARCHAR2);
PROCEDURE borrar_depar
(v_dep_borrar NUMBER,
v_dep_nue NUMBER);
PROCEDURE cambiar_localidad
(v_num_dep NUMBER,
v_loc VARCHAR2);
PROCEDURE visualizar_datos_depart
(v_num_dep NUMBER);
PROCEDURE visualizar_datos_depart
(v_nom_dep VARCHAR2);
END gest_depart;
/

/****************** Cuerpo del paquete **********************/
CREATE OR REPLACE PACKAGE BODY gest_depart AS
FUNCTION buscar_depart_por_nombre /* Función privada */
(v_nom_dep VARCHAR2)
RETURN NUMBER;
/************************************************************/

PROCEDURE insert_depart(
v_nom_dep VARCHAR2,
v_loc VARCHAR2)
```

```
AS
ultimo_dep DEPART.DEPT_NO%TYPE;
nombre_repetido EXCEPTION;
BEGIN

/*Comprobar dpt repetido(Puede levantar NO_DATA_FOUND)*/
DECLARE
nom_dep depart.DNOMBRE%TYPE;
nombre_repetido EXCEPTION;
BEGIN
SELECT dnombre INTO nom_dep FROM depart
WHERE dnombre = v_nom_dep;
RAISE insert_depart.nombre_repetido;
EXCEPTION
WHEN NO_DATA_FOUND THEN
NULL;
WHEN TOO_MANY_ROWS THEN
RAISE insert_Depart.nombre_repetido;
END; /* Fin del bloque de comprobación
de departamento repetido */

/* Calcular el número de departamento e insertar */
SELECT MAX(DEPT_NO) INTO ultimo_dep FROM DEPART;
INSERT INTO DEPART VALUES ((TRUNC(ultimo_dep, -1) +10),
v_nom_dep,v_loc);
EXCEPTION
```

```
WHEN nombre_repetido THEN
DBMS_OUTPUT.PUT_LINE
('Err. Nombre de departamento duplicado');
WHEN NO_DATA_FOUND THEN /* Si no había ningún departamento */
INSERT INTO DEPART VALUES (10,v_nom_dep,v_loc);
END insert_depart;
/*************************************************************/

PROCEDURE borrar_depar
(v_dep_borrar NUMBER,
v_dep_nue NUMBER)
AS
BEGIN
UPDATE emple SET dept_no = v_dep_nue
WHERE DEPT_NO=v_dep_borrar;
DELETE FROM depart WHERE dept_no = v_dep_borrar;
END borrar_depar;
/*************************************************************/

PROCEDURE visualizar_datos_depart
(v_num_dep NUMBER)
AS
vr_dep depart%ROWTYPE;
v_num_empleados NUMBER(4);
BEGIN
SELECT * INTO vr_dep FROM depart
```

```
WHERE DEPT_NO=v_num_dep;
SELECT COUNT(*) INTO v_num_empleados FROM
EMPLE WHERE DEPT_NO=v_num_dep;

DBMS_OUTPUT.PUT_LINE
('Número de departamento: '||vr_dep.dept_no);
DBMS_OUTPUT.PUT_LINE
('Nombre del departamento: '||vr_dep.dnombre);
DBMS_OUTPUT.PUT_LINE
('Localidad : '||vr_dep.loc);
DBMS_OUTPUT.PUT_LINE
('Numero de empleados : '||v_num_empleados);
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Err departamento no encontrado');
END visualizar_datos_depart;
/************************************************************/

PROCEDURE visualizar_datos_depart /* Versión sobrecargada */
(v_nom_dep VARCHAR2)
AS
v_num_dep depart.dept_no%TYPE;
vr_dep depart%ROWTYPE;
v_num_empleados NUMBER(4);
BEGIN
v_num_dep:=buscar_depart_por_nombre(v_nom_dep);
```

```
SELECT * INTO vr_dep FROM depart
WHERE dept_no=v_num_dep;
SELECT COUNT(*) INTO v_num_empleados FROM EMPLE
WHERE dept_no=v_num_dep;

DBMS_OUTPUT.PUT_LINE
('Número de departamento: '||vr_dep.dept_no);
DBMS_OUTPUT.PUT_LINE
('Nombre del departamento: '||vr_dep.dnombre);
DBMS_OUTPUT.PUT_LINE
('Localidad : '||vr_dep.loc);
DBMS_OUTPUT.PUT_LINE
('Numero de empleados : '||v_num_empleados);

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Err departamento no encontrado');
END visualizar_datos_depart;
/************************************************************/

FUNCTION buscar_depart_por_nombre
(v_nom_dep VARCHAR2)
RETURN NUMBER
AS
v_num_dep depart.dept_no%TYPE;
BEGIN
```

```
SELECT dept_no INTO v_num_dep FROM depart

WHERE DNOMBRE = v_nom_dep;

RETURN v_num_dep;

EXCEPTION

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE('Err departamento no encontrado');

END buscar_depart_por_nombre;

/*************************************************************/


PROCEDURE cambiar_localidad(

v_num_dep NUMBER,

v_loc VARCHAR2)

AS

BEGIN

UPDATE depart

SET LOC=v_loc

WHERE dept_no=v_num_dep;

EXCEPTION

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE('Err departamento no encontrado');

END cambiar_localidad;

END gest_depart;
```