Pràctica de programació de Procediments i/o funcions.

1.- Crear un procedimiento llamado actualiza_job_history, que use 3 parámetros (p_id_empleado, p_id_cargo, p_id_departmento). Con los valores recibidos en los parámetros, el procedimiento ha de insertar correctamente el registro en la tabla job_history, controlando las excepciones que se puedan producir (valores duplicados, nulos, otros errores, ...): (4 Ptos) También se debe en cuenta, que:

• Si es la primera vez que se inserta el registro del empleado en job_history, start_date debe ser la hire_date del empleado y end_date debe ser la fecha actual del sistema – 1 dia.

• Si no es la primera vez, start_date debe ser la última fecha que consta en la tabla job_history de ese empleado + 1 dia y end_date debe ser la fecha actual del sistema – 1 dia.

Código:

```
-- 1) Creación de procedimento: actualiza_job_history
CREATE OR REPLACE PROCEDURE actualiza_job_history (
    p_id_empleado    IN employees.employee_id%TYPE,
    p_id_cargo       IN employees.job_id%TYPE,
    p_id_departamento IN employees.department_id%TYPE
) IS
    v_count         NUMBER; --contamos si hay registros en job_history
    v_last_end_date DATE; --start - end date
    v_start_date    DATE;
    v_end_date      DATE := trunc(sysdate) - 1; --TRUNC para quitar la hora y
quedarnos solo con la hora y -1 porque así será el dia de ayer
BEGIN
  --Hay registros?
    SELECT
        COUNT(*)
    INTO v_count
    FROM
        job_history
    WHERE
        employee_id = p_id_empleado;

    IF v_count = 0 THEN
  --Nuevo empleado (hire_date = v_start_date)
```

```
        SELECT
            hire_date
        INTO v_start_date
        FROM
            employees
        WHERE
            employee_id = p_id_empleado;

    ELSE
  --No es nuevo (start_date = end_date +1)
        SELECT
            MAX(end_date)
        INTO v_last_end_date
        FROM
            job_history
        WHERE
            employee_id = p_id_empleado;

        v_start_date := v_last_end_date + 1;
    END IF;

  --Insertar en job_history los datos
    INSERT INTO job_history (
        employee_id,
        start_date,
        end_date,
        job_id,
        department_id
    ) VALUES ( p_id_empleado,
               v_start_date,
               v_end_date,
               p_id_cargo,
               p_id_departamento );

    COMMIT; --Guardams los datos en la base de datos

  --Excepciones: duplicado, no existe y ni idea de que error es
EXCEPTION
    WHEN dup_val_on_index THEN
        dbms_output.put_line('Error: registro duplicado para employee_id = '
|| p_id_empleado);
        ROLLBACK; --si esta mal volvemos atras con un rollback como si no
hubiesemos hecho nada
    WHEN no_data_found THEN
        dbms_output.put_line('Error: empleado '
                             || p_id_empleado
                             || ' no existe en tabla employees');
```

```
        ROLLBACK;
    WHEN OTHERS THEN
        dbms_output.put_line('Error desconocido: ' || sqlerrm); --SQLERRM te
devuelve el error
        ROLLBACK;
END actualiza_job_history;
/
```

Resultado:

```
Procedure ACTUALIZA_JOB_HISTORY compilado
```

Eric López Guerrero

2. Crear un bloque PL/SQL anónimo, para poder hacer las pruebas oportunas en el procedimiento actualiza_job_history: (2 Ptos)
   Para ello se debe pedir por teclado los valores del Id_empleado, id_cargo e id_departamento, en la parte ejecutiva (dentro del begin end) y después llamar al procedure actualiza_job_history,. pasándole los parámetros que se han pedido por teclado.

Debe mostrar por pantalla si la prueba ha tenido éxito o no. En este bloque anónimo se deben controlar los errores que se puedan producir, se pueden usar los paquetes:
DBMS_UTILITY.FORMAT_ERROR_STACK
DBMS_UTILITY.FORMAT_ERROR_BACKTRACE

Código:

```
--2) Bloque anonimo para probar actualiza_job_history pidiendo datos por
teclado
DECLARE
    v_emp_id  employees.employee_id%TYPE;
    v_job_id  employees.job_id%TYPE;
    v_dept_id employees.department_id%TYPE;
BEGIN
    -- Pedir los datos al usuario
    v_emp_id := TO_NUMBER ( &&p_emp_id );--convertimos el numero introducido a
numero y lo recogemos con teclado  con && variable
    v_job_id := '&&P_job_id';
    v_dept_id := TO_NUMBER ( &&p_dept_id );

    -- Llamada al procedimiento
    actualiza_job_history(v_emp_id, v_job_id, v_dept_id);
    dbms_output.put_line('Procedimiento ejecutado correctamente para
employee_id=' || v_emp_id);

    --Excepcion de
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('¡Ha fallado la prueba!: '
                             || dbms_utility.format_error_stack
                             || ' // '
                             || dbms_utility.format_error_backtrace);
END;

/
```

Resultado:
He introducido estos campos: 107, IT_PROG, 60

```
Procedimiento PL/SQL terminado correctamente.
```

3. Crear un bloque anónimo que recorra todos los empleados de la empresa (usando cursores o arrays), teniendo en cuenta que al recorrerlo: (3 ptos)
   OJO: Para evitar entrar en conflicto, desactivar el disparador (trigger) UPDATE_JOB_HISTORY

   • Si el empleado es del departamento 50 y empezó a trabajar antes del año 97 (el AÑO 97 no cuenta) se le debe cambiar al departamento 160.

   • Si el empleado tiene el cargo SA_REP y cobra 3200 € o más, se le debe cambiar al cargo SH_CLERK En ambos casos se ha de utilizar el procedimiento actualiza_job_history, para registrar el cambio en la tabla job_history

   Código:

```
DECLARE
    CURSOR cur_emp IS
    SELECT
        employee_id,
        job_id,
        department_id,
        hire_date,
        salary
    FROM
        employees;

    v_rec cur_emp%rowtype; --rowtype es un tipo de dato que representa una
fila entera de una tabla
BEGIN
    OPEN cur_emp;
    LOOP
        FETCH cur_emp INTO v_rec; --esto hace que guarde la proxima fila del
cursor en la variable v-rec
        EXIT WHEN cur_emp%notfound; --con esto vemos si no hay más filas por
leer, si no encuentra otra más sale del bucle

        -- Caso 1: departamento 50 y hire_date < '01-JAN-1997'
        IF
            v_rec.department_id = 50
            AND v_rec.hire_date < DATE '1997-01-01'
```

```
        THEN
            -- Cambiar departamento a 160
            UPDATE employees
            SET
                department_id = 160
            WHERE
                employee_id = v_rec.employee_id;

            COMMIT;
            actualiza_job_history(v_rec.employee_id, v_rec.job_id, 160);
        END IF;

        -- Caso 2: cargo 'SA_REP' y salario >= 3200
        IF
            v_rec.job_id = 'SA_REP'
            AND v_rec.salary >= 3200
        THEN
            -- Cambiar cargo a 'SH_CLERK'
            UPDATE employees
            SET
                job_id = 'SH_CLERK'
            WHERE
                employee_id = v_rec.employee_id;

            COMMIT;
            actualiza_job_history(v_rec.employee_id, 'SH_CLERK',
v_rec.department_id);
        END IF;

    END LOOP;

    CLOSE cur_emp;
    dbms_output.put_line('Proceso completado para todos los empleados.');

    --excepción para encontrar algún error por si acaso
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error en el procesamiento: ' || sqlerrm);
        ROLLBACK;
END;
```

Resultado:

```
Proceso completado para todos los empleados.

Procedimiento PL/SQL terminado correctamente.
```

Eric López Guerrero