ADMINISTRACIÓN DE BASE DE DATOS.

Tema Nº13:Programación PL/SQL IX.

Indicador de logro Nº13:Elabora disparadores avanzados en programación PL/SQL.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº13:**

Programación PL/SQL IX.

**Subtema 13.1:**

Implementa disparadores avanzados, utilizando lenguaje PL/SQL, en beneficio de la lógica del negocio.

**Un disparador (o trigger) es un procedimiento almacenado asociado a una tabla que se ejecuta al realizar una operación “básica” (INSERT, un DELETE o un UPDATE)** sobre ésta. La operación básica que despierta al trigger es conocida como sentencia disparadora. La ejecución del disparador puede ser antes o después de llevar a cabo la sentencia disparadora. Es posible especificar condiciones adicionales para la ejecución del disparador (restrictores). Dado que una sentencia disparadora puede afectar una o más filas de una tabla, es necesario especificar si se quiere que el disparador se ejecute para cada una de las filas afectadas o para el bloque en general.

Se presentan los siguientes casos:

**Ejemplo 1:**

Crear la tabla EMPLEADO.

CREATE TABLE Empleado (

Dni char (8) PRIMARY KEY,

Nomemp varchar2(50),

Jefe Char (8),

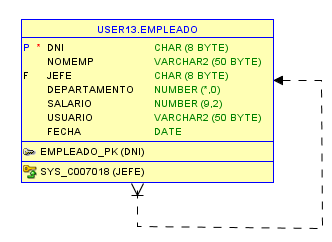
Departamento integer,

Salario Number (9,2) DEFAULT 1000,

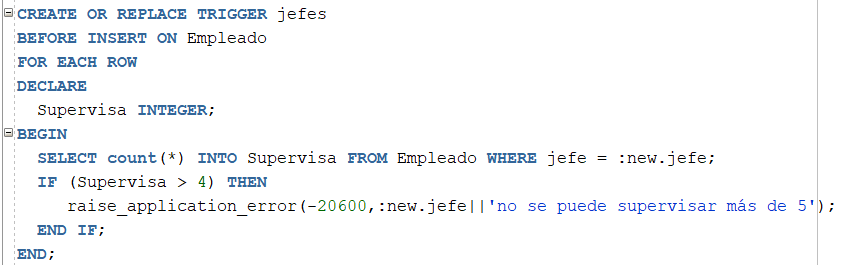
Usuario varchar2(50),

Fecha date,

FOREIGN KEY (Jefe) REFERENCES Empleado (Dni)

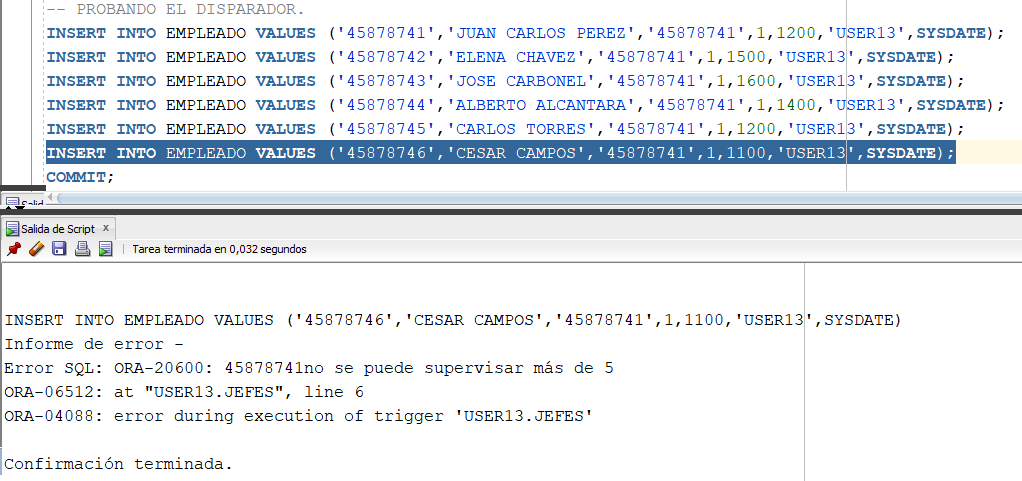
);

Crear un trigger sobre la tabla EMPLEADO para que no se permita que un empleado sea jefe de más de cinco empleados.



Para probar el disparador se debe de insertar registros sobre la tabla Empleado, tal como se observa en la siguiente imagen:

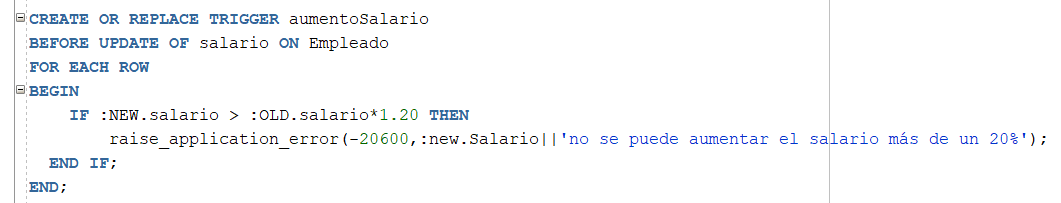
El error se genera en el registro de CESAR CAMPOS, dado que este empleado tiene como jefe a la misma persona por DNI, que los anteriores empleados.



**Ejemplo 2:**

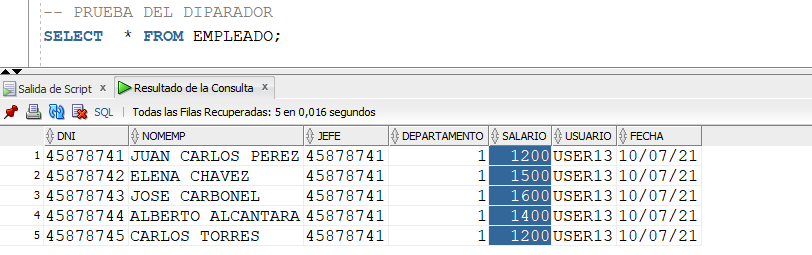
Crear un trigger para impedir que se aumente el salario de un empleado en más de un 20%

* Es necesario comparar los valores **:old.salario** y **:new.salario** cada vez que se modifica el atributo salario BEFORE UPDATE.

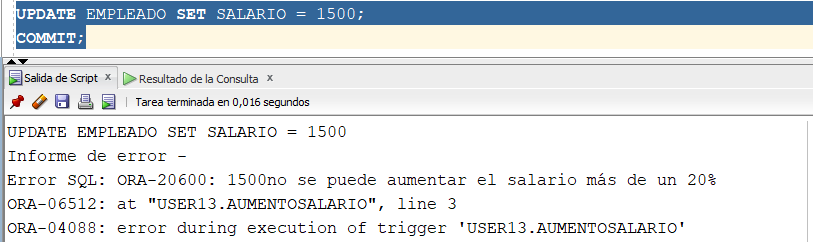


Probando el disparador:

Antes de actualizar:



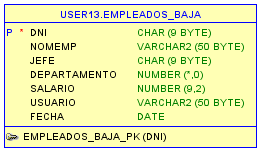
Actualizando registros.



Aquí se genera errores debido a que algunos sueldos no se les puede aumentar más del 20%.

**Ejemplo 3:**

Crear un trigger que inserte una fila en la tabla Empleados\_Baja cuando se borre una fila en la tabla empleado.

CREATE TABLE Empleados\_Baja (

Dni char (8) PRIMARY KEY,

Nomemp varchar2(50),

Jefe Char (8),

Departamento integer,

Salario Number (9,2) DEFAULT 1000,

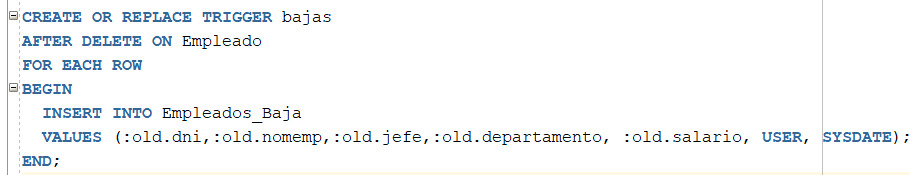
Usuario varchar2(50),

Fecha date,

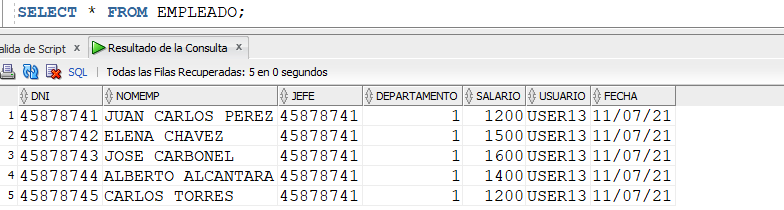
FOREIGN KEY (Jefe) REFERENCES Empleado (Dni)

);

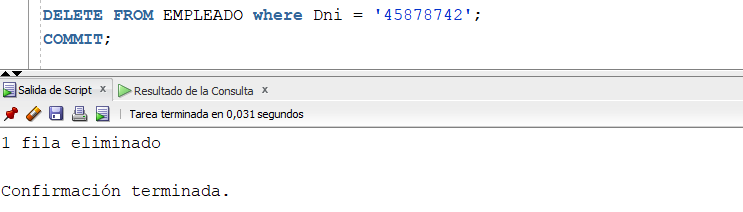
Disparador:



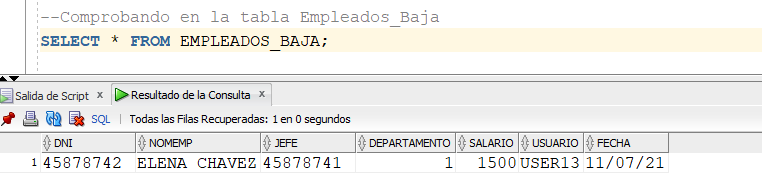
Antes de Eliminar, escogemos un registro, en este caso ‘ELENA CHAVEZ’ cuyo Dni es: 45878742



Proceda a Eliminar:



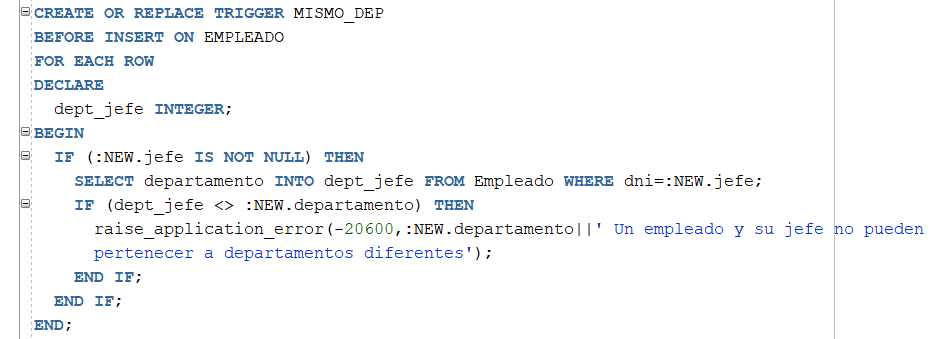
Ahora verificamos en la tabla Empleados\_Baja



**OTROS EJEMPLOS:**

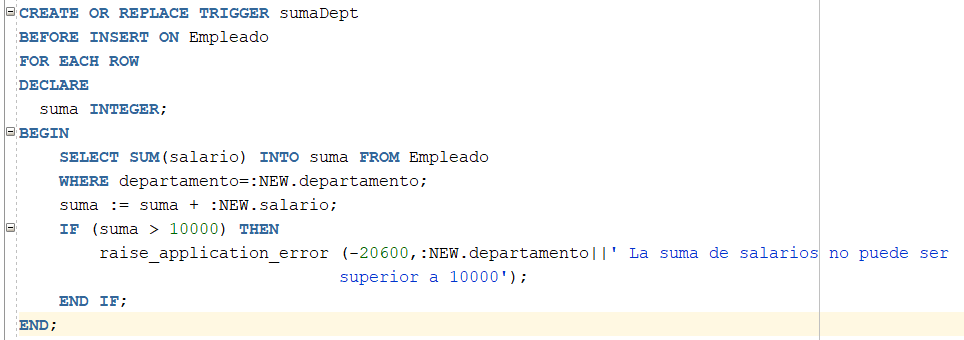
**Ejemplo 4:**

Crear un trigger para impedir que, al insertar un Empleado, el empleado y su jefe puedan pertenecer a departamentos distintos.



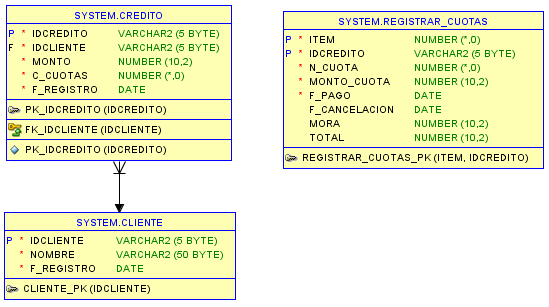
**Ejercicio 5:**

Crear un trigger para impedir que, al insertar un Empleado, la suma de los salarios de los empleados pertenecientes al departamento del empleado insertado supere los 10.000.



**Ejemplo 6:**

Se tiene el siguiente modelo relacional:



Este modelo relacional consiste en registrar a clientes (**CLIENTE**) con sus respectivos prestamos (**CREDITO** y **REGISTRAR\_CUOTAS**), implementar un disparador el cual sea capaz de almacenar las cuotas mensuales (REGISTRAR\_CUOTAS) de un determinado crédito.

Las condiciones son las siguientes:

* Registrar solo 3 Clientes
* Registre solo un crédito, con el primer cliente, con los siguientes elementos
  + Monto de 5000 soles.
  + Se pagará en 5 cuotas mensuales
  + Se aplicará a cada cuota el 5% adicional

**SOLUCIÓN:**

**PASO 1:**

Insertamos registros en la tabla cliente:

**INSERT INTO CLIENTE VALUES** ('C0001','PEREZ GONZALES JUAN','05/05/2019');

**INSERT INTO CLIENTE VALUES** ('C0002','CESPEDES LOPEZ ELIZABETH', '10/05/2019');

**INSERT INTO CLIENTE VALUES** ('C0003','FERNANDEZ CARRION JULIA','08/09/2019');

**COMMIT;**

**PASO 2:**

Creamos el disparador (REGISTRO) para la tabla CREDITO, que permitirá registrar las mensualidades en la tabla REGISTRAR\_CUOTA, después (AFTER) de generar el crédito, según el número de cuotas establecidas.

**CREATE OR REPLACE TRIGGER** REGISTRO

**AFTER INSERT ON CREDITO**

**FOR EACH ROW**

**DECLARE**

CONTEO INT:=0;

NUEVA\_FECHA DATE;

COSTO\_CUOTA NUMBER (10, 2):=0;

**BEGIN**

COSTO\_CUOTA:=(:NEW.MONTO / :NEW.C\_CUOTAS) +

((:NEW.MONTO / :NEW.C\_CUOTAS) \* 0.05);

**FOR** CONTEO **IN** 1..:NEW.C\_CUOTAS LOOP

NUEVA\_FECHA:= add\_months(:NEW.F\_REGISTRO,CONTEO);

INSERT INTO REGISTRAR\_CUOTAS VALUES (CONTEO, :NEW.IDCREDITO,

CONTEO, COSTO\_CUOTA, NUEVA\_FECHA, NULL, NULL, NULL);

**END LOOP;**

**END;**

**PASO 3:**

Insertaremos un solo registro en la tabla CREDITO, este contendrá los datos necesarios para generar las mensualidades en la tabla REGISTRAR\_CUOTAS.

**INSERT INTO** CREDITO **VALUES** ('00001','C0001', 1000.00, 5, SYSDATE);

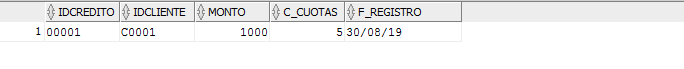
**COMMIT;**

**PASO 4:**

Comprobamos las inserciones en las tablas: **CREDITO** y **REGISTRAR\_CUOTAS** con simples consultas.

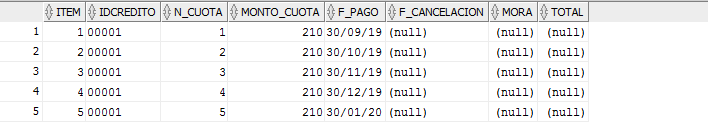
**Aquí se registró el crédito de 1000 soles en 5 cuotas, en fecha 30/08/2019.**

**SELECT \* FROM CREDITO;**



Aquí se han registrado con ayuda del disparador, las mensualidades del primer cliente.

**SELECT \* FROM REGISTRAR\_CUOTAS;**



**FIN DEL EJEMPLO 5.**

**Subtema 13.2:**

Escenarios donde se aplique disparadores en las reglas del negocio.

* **Utilización interna para la explotación o administración de la Bas de Datos:**
  + Implementación de restricciones de integridad.
  + Actualización de vistas materializadas (si no están soportadas de forma transparente por el SGBD) o atributos derivados.
  + Seguridad y auditoría: registro de acciones sobre objetos de la Base de Datos.
  + Gestión de versiones, mantenimiento de datos históricos.
  + Realización automática de copias de los datos
* **Utilización externa para la implementación de reglas de negocio:**
  + Ej.: si pasan 7 días sin que el cliente haya pagado una factura, se le envía un email, se le bloquea la posibilidad de realizar más compras, y se registra como moroso

**CONCLUSIONES Y RECOMENDACIONES DE LA EXPERIENCIA**

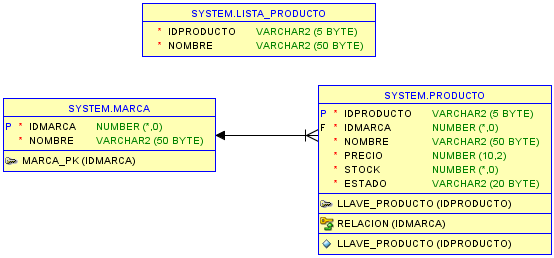
La codificación PL/SQL de Oracle es tan amplia, que ofrece a los estudiantes una alta gama de alternativas para la resolución de problemas en disparadores, combinados con procedimientos y funciones, no difíciles de entender, con la prioridad de manejar los datos con suma responsabilidad por el programador Oracle.

Adicional mente, estos disparadores ofrecen a los administradores en base de datos, el mejor mantenimiento de tablas y datos que la conforman.

**ACTIVIDAD VIRTUAL**

Revisar y analizar el material presentado del Tema 13.

**SE TIENE EL SIGUIENTE MODELO RELACIONAL:**



**SOLUCIONAR LOS CASOS, LUEGO SUBIRLO COMO PRESENTACION DE LA ACTIVIDAD.**

De acuerdo al material presentado:

**RESOLVER:**

Crear las siguientes funciones en PL/SQL.

1. Elabore el Script de Base de datos, del presente modelo relacional.
2. Se pide elaborar un disparador que no permita el ingreso de productos cuyo Stock sea cero.
3. Elabore un disparador el cual permita obtener una copia de los productos que se hayan podido eliminar (crear una tabla Respaldo).
4. Si un Producto es Modificado en su stock, este nunca no debe recibir un cero, ni negativo.