Laboratorio 10

Enlace directo: Video Laboratorio 9 (este enlace se abre en una nueva ventana / pestaña)

En este laboratorio trabajaremos **triggers** en PL/SQL,

Triggers

Un trigger es un procedimiento almacenado que el motor de base de datos ejecuta **automáticamente** cuando ocurre un **evento DML** (insert, update, o delete) en una tabla. De acuerdo a como se haya definido, el trigger puede ejecutarse antes o después de ejecutar el evento.

La sintaxis básic apara crear un trigger es:

```
CREATE [OR REPLACE] TRIGGER nombre_trigger
momento evento ON tabla
[FOR EACH ROW [WHEN condición]]
bloque de PL/SQL
-- momento puede ser BEFORE o AFTER
-- evento puede ser INSERT, UPDATE o DELETE
-- FOR EACH ROW indica que el trigger se ejecuta por cada fila que se inserte/actualice/borre. Si no se especifica, el trigger se ejecuta una sola vez (antes o después) de que se ha ejecutado la sentencia (independiente de cuantos registros afecte).
-- WHEN permite especificar una condición que se debe cumplir para que el trigger se ejecute
-- El bloque comienza con DECLARE o BEGIN, y finaliza con END;
```

En el bloque de código de los triggers de nivel **FOR EACH ROW** se pueden usar las variables **:NEW** y **:OLD**, estas son variables de tipo registro con los mismos atributos de la tabla para la que se define el trigger. Por ejemplo, si el trigger es para la tabla Compra, :NEW y :OLD tienen los mismos atributos de la tabla compra. :NEW tiene los valores del registro que va a quedar en la tabla después de que se ejecute el evento y :OLD tiene los valores del registro que estaba en la tabla antes de ejecutar el evento.

Ejemplos:

- Trigger para mantener actualizada la tabla totalSalarios cada vez que se modifica la tabla empleado

```
CREATE OR REPLACE TRIGGER trCambioSalarios
AFTER INSERT OR UPDATE OR DELETE ON Empleado
BEGIN
      UPDATE totalSalarios
      SET    total = (SELECT SUM(salario) FROM empleado);
END;
```

- Trigger para evitar que se registren salarios menores al mínimo actual

```
CREATE OR REPLACE TRIGGER trValor
BEFORE INSERT ON Empleado FOR EACH ROW
DECLARE minSal NUMBER(10) DEFAULT 0;
BEGIN
  SELECT MIN(salario) INTO minSal FROM empleado;
  IF :NEW.salario < minSal THEN
    Raise_application_error(-20000, 'Salario fuera de rango: ' ||:NEW.salario);
```

```
  END IF;
END;
```

- Tener registro del usuario que realiza las modificaciones en los datos de la tabla empleado (en este caso "Usuario" es un atributo de la tabla empleado que indica cual fue el último usuario que cambió --insertó o actualizó-- un registro)

```
CREATE OR REPLACE TRIGGER trUser
BEFORE INSERT OR UPDATE ON Empleado
FOR EACH ROW
BEGIN
  SELECT USER INTO :NEW.Usuario FROM DUAL;
END;
```

- Evitar que se asigne un salario menor que el actual

```
CREATE OR REPLACE TRIGGER trUser
BEFORE UPDATE ON Empleado
FOR EACH ROW
BEGIN
  IF :NEW.salario < :OLD.salario THEN
Raise_application_error(-20100, 'El nuevo salario ('||:NEW.salario||') debe ser mayor que el anterior ('||:OLD.salario||')');
  END IF;
END;
```

Ejercicios

Usando el esquema de base de datos creado y modificado en los laboratorios anteriores, escriba los comandos para dar respuesta a los requerimientos especificados en esta sección.

Para cada ejercicio incluya en el script que va a entregar, el código para crear el trigger y las sentencias que permiten hacer las pruebas de su funcionamiento. En esas sentencias debe haber al menos una que permite verificar que el trigger hace la función solicitada, y en los puntos 1 y 2, sentencias que muestren que los casos válidos operan sin problema.

1. En la tienda, cuando se crea una compra esta toma los productos del carrito de compras. Cree un trigger que evite crear una compra cuando el carrito de compras está vacio.
2. Actualice el trigger del punto anterior, para que además evite que se hagan compras que incluyan productos que ingresaron al carrito de compras hace más de seis meses.
3. Cree un trigger que garantice que cuando se incluyen los productos en la compra, el precio del producto es el que está registrado en la tabla carro. Este trigger no evita que se haga la compra, pero cambia el precio si está desactualizado.
4. Cree la tabla Bonificaciones(codUsuario, fechaBonificacion, valorBono, estado) . El

4. estado de un bono puede ser Vigente (cuando el usuario no ha gastado el bono) o Usado (cuando ya lo ha gastado).
5. Cree un trigger que genere un bono por el 15% del valor de la compra cuando el cliente realiza su primera compra.
6. Cree uno o varios trigger(s) que garanticen que en los atributos del modelo que son fechas, cuando se insertan datos, se guarde la fecha actual del sistema. Este trigger no evita que se haga la inserción, pero cambia fecha si está desactualizada.

Al finalizar la sesión, cada estudiante debe enviar el script a Entrega Laboratorio 10