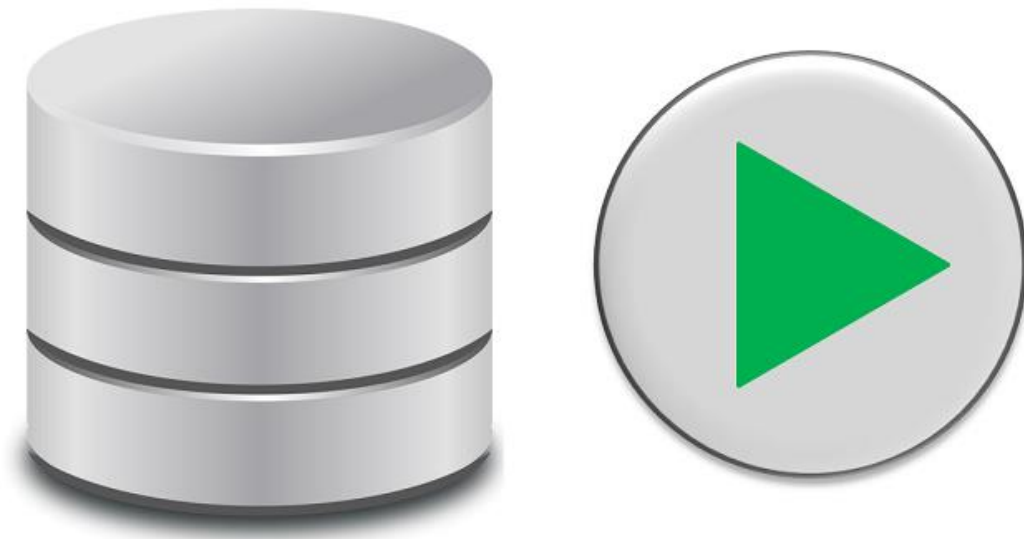




# ORACLE PL/SQL



MÓDULO 11

## DESENCADENANTES

**GUSTAVO CORONEL**  
*desarrollasoftware.com*



# ORACLE PL/SQL



## Contenido

¿QUÉ ES UN DESENCADENANTES? .....	3
TIPOS DE DESENCADENANTES .....	3
SINTAXIS GENERAL .....	4
DESENCADENANTES DML .....	4
FUNCIONES LÓGICAS: INSERTING, DELETING, UPDATING. ....	5
USO DE :OLD Y :NEW .....	6
DESENCADENANTES DE SUSTITUCIÓN .....	9
DESENCADENANTES DEL SISTEMA.....	10
CURSOS VIRTUALES .....	11
JAVA ORIENTADO A OBJETOS .....	11
PROGRAMACIÓN DE BASE DE DATOS CON JAVA JDBC .....	11



# ORACLE PL/SQL



## ¿QUÉ ES UN DESENCADENANTES?

Los desencadenantes son similares a los procedimientos y funciones, en el sentido de que son bloques nominados de PL/SQL con secciones declarativa, ejecutable y de tratamiento de excepciones, se almacenan en la base de datos como objetos independientes, y no pueden ser locales a un bloque o paquete.

Los desencadenantes difieren de los procedimientos y funciones en la forma en que se ejecutan, mientras que el desencadenante se ejecuta de manera implícita, los procedimientos y las funciones se ejecutan de manera explícita.

Los desencadenantes se activan cuando ocurre un suceso. El suceso que lo activa puede ser una operación DML (INSERT, UPDATE o DELETE) sobre una tabla o ciertos tipos de vista.

También es posible asociar desencadenantes con sucesos del sistema, como el inicio y finalización de una sesión, y ciertos tipos de operaciones del lenguaje de definición de datos (DDL).

## TIPOS DE DESENCADENANTES

Los desencadenantes se agrupan en las siguientes categorías:

- Desencadenantes DML
- Desencadenantes de Sustitución: Se aplica a vistas.
- Desencadenantes del Sistema



# ORACLE PL/SQL



## SINTAXIS GENERAL

```
CREATE OR REPLACE TRIGGER nombre_desencadenante
{BEFORE | AFTER | INSTEAD OF} suceso_disparo
[cláusula_referencia]
[WHEN condición_dispro]
[FOR EACH ROW]
BEGIN
    -----
    -----
    -----
END;
```

## DESENCADENANTES DML

CATEGORÍA	VALORES	COMENTARIOS
Instrucción	INSERT, DELETE, UPDATE	Define qué tipo de instrucción DML causa la activación del desencadenante.
Temporización	Antes o Después	
Nivel	Fila o Instrucción	



## FUNCIONES LÓGICAS: INSERTING, DELETING, UPDATING.

Estas funciones se utilizan cuando el evento de un desencadenante es compuesto, es decir, queremos que el desencadenante se active ante diferentes operaciones DML pero no queremos que haga lo mismo para cualquiera de los eventos activadores. Con lo visto hasta ahora, la única solución sería diseñar un desencadenante para cada una de las acciones DML de activación. Sin embargo, con las funciones inserting, deleting y updating, podremos hacer todo en un único desencadenante.

### Script 1

```
CREATE OR REPLACE TRIGGER tr_test_emp
AFTER INSERT OR DELETE OR UPDATE ON emp
BEGIN
    if inserting then
        dbms_output.put_line( 'nuevo empleado se ha insertado' );
    Elsif updating then
        dbms_output.put_line( 'un empleado se ha modificado' );
    Elsif deleting then
        dbms_output.put_line( 'un empleado se ha eliminado' );
    end if;
END tr_test_emp;
```



# ORACLE PL/SQL



## USO DE :OLD Y :NEW

Instrucción de Disparo	:old	:new
INSERT	NULL	Valores que se insertan en la tabla.
UPDATE	Valores originales.	Nuevos valores.
DELETE	Valores originales.	NULL

### Script 2

Realizar un desencadenante que registre los cambios de salario de un empleado, el usuario que lo realiza y la fecha.

Tabla para registrar los cambios de salario:

```
Create Table Sal_History(  
  EmpNo Number(4) not null,  
  SalOld Number(7,2) null,  
  SalNew Number(7,2) null,  
  StartDate Date not null,  
  SetUser Varchar2(30) not null  
);
```

El desencadenante para registrar los cambios de salario:

```
Create or Replace Trigger tr_UpdateEmpSal  
After Insert OR Update ON Emp  
For Each Row  
Begin  
  Insert Into Sal_History(EmpNo, SalOld, SalNew, StartDate, SetUser)  
    Values( :New.EmpNo, :Old.Sal, :New.Sal, sysdate, USER );  
End tr_UpdateEmpSal;
```



# ORACLE PL/SQL



## Prueba 1:

```
SQL> update emp
  2  set sal = 1200
  3  where empno = 7369;
```

1 row updated.

```
SQL> select * from Sal_History;
```

EMPNO	SALOLD	SALNEW	STARTDAT	SETUSER
7369	1000	1200	19/02/05	SCOTT

```
SQL> rollback;
```

Rollback complete.

## Prueba 2:

```
SQL> update emp
  2  set sal = 5000
  3  where empno in (7788, 7902);
```

2 rows updated.

```
SQL> select * from Sal_History;
```

EMPNO	SALOLD	SALNEW	STARTDAT	SETUSER
7902	3000	5000	19/02/05	SCOTT
7788	3000	5000	19/02/05	SCOTT

```
SQL> rollback;
```

Rollback complete.



# ORACLE PL/SQL



Prueba 3:

```
SQL> conn / as sysdba
Connected.
```

```
SQL> update scott.emp
  2  set sal = 20000
  3  where empno = 7844;
```

1 row updated.

```
SQL> select * from scott.sal_history;
```

EMPNO	SALOLD	SALNEW	STARTDAT	SETUSER
7844	1500	20000	19/02/05	SYS

## Script 3

En este script se ilustra como deshabilitar un desencadenante.

```
SQL> alter trigger tr_UpdateEmpSal disable;
```

Trigger altered.

```
SQL> update emp
  2  set sal = 5000
  3  where empno = 7788 ;
```

1 row updated.

```
SQL> select * from Sal_History;
```

no rows selected

```
SQL> rollback;
```

Rollback complete.





## DESENCADENANTES DE SUSTITUCIÓN

### Script 4

Vista para consultar los empleados:

```
Create Or Replace View v_empleados As
Select e.empno, e.ename, d.deptno, d.dname
From emp e Inner Join dept d
On e.deptno = d.deptno;
```

Desencadenante para insertar nuevos registros:

```
Create Or Replace Trigger tr_vista
  Instead Of Insert Or Delete On v_empleados
  For Each Row
Declare
  cuenta Number;
Begin
  If Inserting Then
    Select Count(*) Into cuenta From dept Where deptno = :new.deptno;
    If cuenta = 0 Then
      Insert Into dept(deptno,dname)
      Values(:New.deptno, :New.dname);
    End If;

    Select Count(*) Into cuenta From emp Where empno = :new.empno;
    If cuenta = 0 Then
      Insert Into emp(empno,ename,deptno)
      Values(:New.empno, :New.ename, :New.deptno);
    End If;
  Elsif Deleting Then
    Delete From emp Where empno = :old.empno;
  End If;
End tr_vista;
```



## DESENCADENANTES DEL SISTEMA

### Script 5

Desencadenante que impide borrar cualquier objeto del esquema actual.

```
Create Or Replace Trigger tr_deny_drop
  before Drop ON Schema
Begin

  raise_application_error( -20000, 'No es posible borrar objetos. ');

End tr_deny_drop;
```



# ORACLE PL/SQL



## CURSOS VIRTUALES

En esta URL tienes cupones de descuento:

<http://gcoronelc.blogspot.com/p/cursos-virtuales.html>

## Java Orientado a Objetos



### **JAVA ORIENTADO A OBJETOS**

---

Eric Gustavo Coronel Castillo  
[www.desarrollasoftware.com](http://www.desarrollasoftware.com)  
INSTRUCTOR

<https://www.udemy.com/java-orientado-a-objetos/>

## Programación de Base de Datos con Java JDBC



### **PROGRAMACIÓN DE BASE DE DATOS ORACLE CON JAVA JDBC**

---

Eric Gustavo Coronel Castillo  
[www.desarrollasoftware.com](http://www.desarrollasoftware.com)  
INSTRUCTOR

[https://www.udemy.com/java-jdbc-oracle /](https://www.udemy.com/java-jdbc-oracle/)