| GUÍA 2.3.2:
Construcción de Triggers para control de acciones a nivel de sentencia

Sigla	Asignatura	Experiencia de Aprendizaje
BDY1103	Taller de Base de Datos	EA2: Implementa programas PL/SQL en la base de datos para construir una solución integral de procesamiento y generación de información o ser usados en otros procesos y/o aplicaciones.
Tiempo	**Modalidad de Trabajo**	**Indicadores de logro**
4h	Individual	IL2.3

Antecedentes generales

En esta guía encontrarás los contenidos asociados a la construcción de triggers para el control de acciones a nivel de sentencia, junto con ejemplos y actividades prácticas a desarrollar.

Requerimientos para esta actividad

En esta actividad, los y las estudiantes deberán utilizar SQL Developer y seguir las instrucciones indicadas por el/la docente.

Sesión 1: Triggers a nivel de sentencia

Objetivos de Aprendizaje

1. Comprender qué son los triggers y sus tipos.
2. Aprender a crear triggers a nivel de sentencia.
3. Aplicar triggers a nivel de sentencia en un escenario de negocio.

1. Introducción a los Triggers

1.1 ¿Qué son los Triggers?

- **Triggers**: Son bloques PL/SQL que se ejecutan automáticamente en respuesta a ciertos eventos en una tabla, tales como **INSERT, UPDATE o DELETE**. Los triggers pueden utilizarse para completar la integridad referencial, también para imponer reglas de negocio complejas o para auditar cambios en los datos

1.2 Tipos de Triggers

- **Before**: Ejecutados antes de que el evento ocurra.
- **After**: Ejecutados después de que el evento ocurra.
- **Instead Of**: Ejecutados en lugar de la operación DML (usualmente para vistas).

1.3 Niveles de Triggers

- **A Nivel de Sentencia**: Ejecutados una vez por cada sentencia DML.
- **A Nivel de Filas**: Ejecutados una vez por cada fila afectada por la sentencia DML.

2. Triggers a Nivel de Sentencia

2.1 Concepto

- **Triggers a Nivel de Sentencia**: Son aquellos que se ejecutan una sola vez por cada sentencia DML, sin importar cuántas filas sean afectadas.

2.2 Sintaxis Básica

```
CREATE OR REPLACE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE | DELETE} ON table_name
BEGIN
   -- Código PL/SQL
END;
/
```

3. Ejemplos Prácticos

Ejemplo 1: Registro de Cambios en la Tabla EMP

Objetivo: Crear un trigger que registre cada modificación en la tabla `emp` en una tabla de auditoría `emp_audit`.

```
-- Crear tabla de auditoría
CREATE TABLE emp_audit (
   audit_id NUMBER PRIMARY KEY,
   empno NUMBER,
   operation VARCHAR2(10),
   change_date DATE
);

-- Crear secuencia para generar IDs únicos
CREATE SEQUENCE audit_seq START WITH 1 INCREMENT BY 1;

-- Crear trigger a nivel de sentencia
CREATE OR REPLACE TRIGGER emp_audit_trigger
   • AFTER INSERT OR UPDATE OR DELETE ON emp
   • BEGIN
   •    if INSERTING then
   •      INSERT INTO emp_audit (audit_id, empno, operation,
      change_date)
   •      VALUES (audit_seq.NEXTVAL, NULL, 'INSERT', sysdate);
   •    elsif UPDATING then
   •      INSERT INTO emp_audit (audit_id, empno, operation,
      change_date)
   •      VALUES (audit_seq.NEXTVAL, NULL, 'UPDATE', sysdate);
   •    elsif DELETING then
   •      INSERT INTO emp_audit (audit_id, empno, operation,
      change_date)
   •      VALUES (audit_seq.NEXTVAL, NULL, 'DELETE', sysdate);
   •    end if;
   • END;
   • /
```

Ejemplo 2: Trigger a Nivel de Sentencia en la Tabla ORDERS

Objetivo: Registrar cada operación en la tabla `orders` en una tabla de auditoría `orders_audit`.

```
-- Crear tabla de auditoría
CREATE TABLE orders_audit (
   audit_id NUMBER PRIMARY KEY,
   order_id NUMBER,
   operation VARCHAR2(10),
   change_date DATE
);

-- Crear secuencia para IDs únicos
CREATE SEQUENCE orders_audit_seq START WITH 1 INCREMENT BY 1;

-- Crear trigger a nivel de sentencia
CREATE OR REPLACE TRIGGER orders_audit_trigger
   • AFTER INSERT OR UPDATE OR DELETE ON orders
   • BEGIN
   •    if INSERTING then
```

```
      INSERT INTO orders_audit(audit_id, order_id, operation,
change_date)
      VALUES (orders_audit_seq.NEXTVAL, NULL, 'INSERT', sysdate);
   elsif UPDATING then
      INSERT INTO orders_audit(audit_id, order_id, operation,
change_date)
      VALUES (orders_audit_seq.NEXTVAL, NULL, 'UPDATE', sysdate);
   elsif DELETING then
      INSERT INTO orders_audit(audit_id, order_id, operation,
change_date)
      VALUES (orders_audit_seq.NEXTVAL, NULL, 'DELETE', sysdate);
   end if;
END;
/
```

4. Actividad Práctica

Actividad 1: Crear un Trigger en la Tabla CUSTOMERS

- **Objetivo**: Crear un trigger que registre cada operación en la tabla `customers` en una tabla de auditoría `customers_audit`.
- **Pasos**:
  - Crear la tabla `customers_audit`.
  - Crear la secuencia `customers_audit_seq`.
  - Crear el trigger `customers_audit_trigger`.

Conclusión de la Sesión 1

- **Resumen**: Los triggers a nivel de sentencia son útiles para registrar y auditar operaciones a nivel global de la tabla.

Evaluación y Mejores Prácticas

1. **Validación**: Siempre validar los datos en los triggers BEFORE.
2. **Auditoría**: Usar triggers AFTER para auditar cambios importantes.
3. **Pruebas**: Realizar pruebas exhaustivas para asegurar el funcionamiento correcto de los triggers en diferentes escenarios.

Recursos Adicionales

Candel, C. J. F., Molina, J., Ruiz, F. J. B., Barceló, J. R. H., Ruiz, D. S., & Viera, B. J. C. (2019). Developing a model-driven reengineering approach for migrating PL/SQL triggers to Java: A practical experience. *The Journal of systems and software*, *151*, 38–64. https://doi.org/10.1016/J.JSS.2019.01.068

INFORMATICONFIG [@informaticonfig333]. (2021, enero 14). *Curso de Oracle PLSQL en español desde cero | TRIGGERS, FOR EACH ROW / BEFORE INSERT video(17)*. Youtube. https://www.youtube.com/watch?v=xkhGEFEH_Ec