PUCP

1INF33 - Base de Datos

Triggers

2024

Profesores del curso

ÍNDICE

Objetivos

Objetivos

- Explicar qué son los *triggers* en Oracle y por qué son importantes en el contexto de las bases de datos relacionales.
- Conocer los diferentes tipos de *triggers* que Oracle admite
- Detallar la sintaxis y la estructura básica de un *trigger* en Oracle, incluyendo cómo definirlos, activarlos y desactivarlos.
- Identificar los eventos que pueden desencadenar un *trigger*
- Detallar las acciones que un *trigger* puede llevar a cabo, como insertar, actualizar, eliminar o validar datos en tablas relacionadas.
- Proporcionar ejemplos prácticos y escenarios comunes donde los *triggers* pueden ser útiles, como auditoría de datos, aplicaciones de integridad referencial y automatización de tareas.

Triggers en Oracle

Definición

Disparador (*Trigger*)

- Un *trigger* es como un procedimiento almacenado que el servidor de Base de datos invoca automáticamente cada vez que ocurre un evento específico.

- Al igual que un procedimiento almacenado, un *trigger* es una unidad PL/SQL con nombre que se almacena en la base de datos y se puede invocar repetidamente. A diferencia de un procedimiento, un *trigger* se puede habilitar y deshabilitar, pero no se puede invocar explícitamente.

Disparador (*Trigger*)

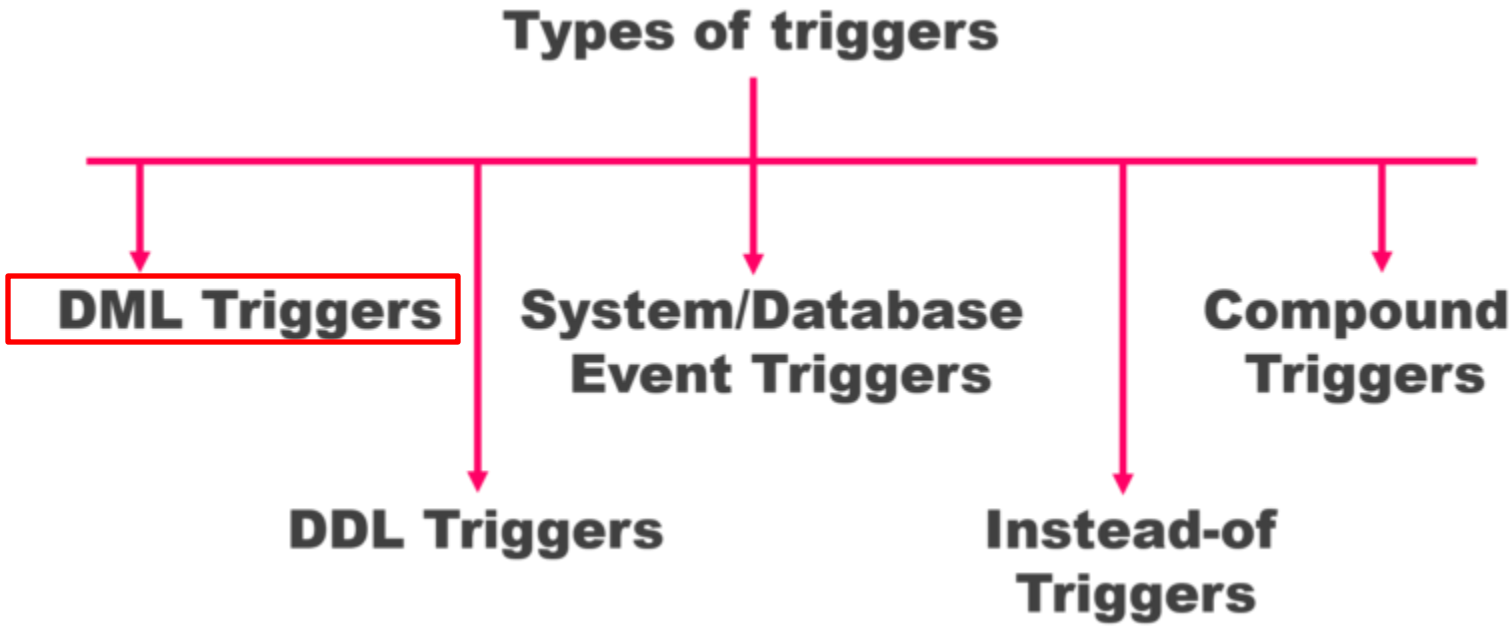
- Ejemplo:

Caso propuesto:

> Se están realizando operaciones DML en una tabla después del horario comercial habitual ¿Cómo se puede identificar quién realiza estas operaciones y desde cuál equipo?

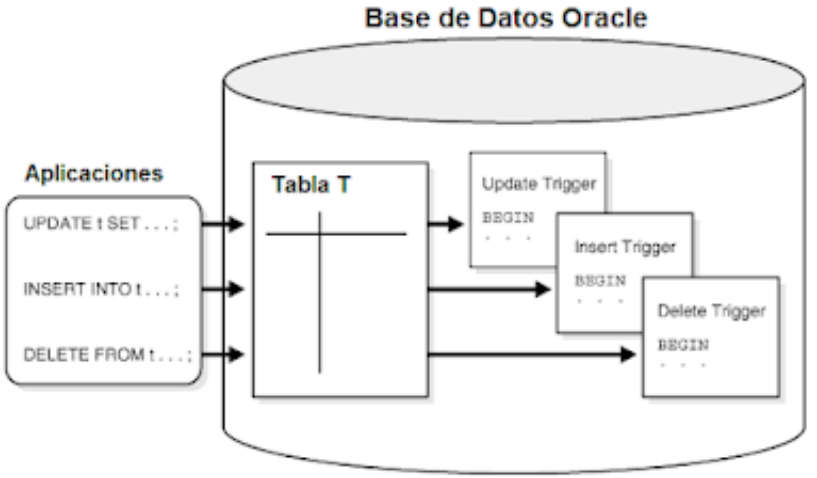
Disparador (*Trigger*)

Existen cinco tipos de *trigger*:



Disparador DML (DML *Triggers*)

- Es un bloque PL/SQL que se ejecuta de forma implícita cuando se ejecuta cierta operación DML: INSERT, DELETE o UPDATE. DML *Triggers* pueden ser de sentencia (*statements*) o de fila (*row-level activities*)
- Contrariamente, a los procedimientos y las funciones se ejecutan sin hacer una llamada explícita a ellos.
- Un disparador no admite argumentos.



Tipos

Disparador DML (DML *Triggers*)

Los *triggers* pueden ser a nivel de registro (row) o a nivel de sentencia (*statement*).

- A nivel de registro o fila el *trigger* se ejecuta cada vez que un registro es actualizado, insertado o borrado
- A nivel de sentencia, el *trigger* se ejecuta una vez que la sentencia INSERT, UPDATE o INSERT se completa. Obviamente en este caso el *trigger* sólo puede ser ejecutado después de que se ejecute dicha sentencia

Disparador DML (DML *Triggers*)

```
CREATE [OR REPLACE]  TRIGGER trigger_name
     {BEFORE | AFTER }
     {INSERT | UPDATE | UPDATE OF column1 [, columna(n+1)] | DELETE
      ON table_name
     [FOR EACH ROW]
     [FOLLOWS | PRECEDES another_trigger]
     [ENABLE /DISABLE]
     [WHEN condition]
```

Trigger header

```
[DECLARE]
         declaration statements;
BEGIN
         executable statements;
EXCEPTION
         exception_handling statements;
END [trigger_name];
```

Trigger body

Disparador DML (DML *Triggers*)

Usos:

- Implementar controles de auditoría y dejar rastro ("*logs*") de transacciones
- Implementar replicación
- Evitar transacciones no válidas
- Mantener estadísticas de acceso a datos
- Implementar integridad referencial en bases de datos distribuidas

Disparador DML (DML *Triggers*)

Consideraciones

- Un disparador puede disparar a otro ***trigger*** y así sucesivamente formando una cascada. Sin embargo, su uso excesivo puede generar interdependencias dificultando su mantenimiento. NO DEBEN SER RECURSIVOS.
- A diferencia de las restricciones de integridad, que se aplican sobre datos existentes y cualquier sentencia sobre la tabla correspondiente, un *trigger* restringe la acción de transacciones sobre los datos que fueron o serán manipulados después de la ejecución del *trigger*.
- Sirven para restricciones transitorias o en función de condiciones dinámicas.

Estructura

Disparador DML (DML *Triggers*)

Partes

Un *trigger* tiene tres partes básicas:

- El Evento desencadenante
- La Restricción de activación
- La Acción desencadenante

Disparador DML (DML *Triggers*)

El Evento

El MOMENTO cuando se disparará el *trigger* en relación con el Evento, puede ser BEFORE (antes) o AFTER (después).

La sentencia SQL que "dispara" al *trigger* : **INSERT, DELETE o UPDATE**.

UPDATE puede tener la cláusula **OF** para indicar lista de columnas.

Puede especificarse combinaciones con el operador **OR**.

La Restricción

Expresión lógica para usarse como condición para ejecutar el *trigger*.

Es opcional y usada en el *trigger* "para cada fila". Cláusula **WHEN**.

La Acción

Procedimiento (bloque PL/SQL) a ejecutarse cuando ocurra el evento y la condición (si es especificada) se cumpla. Puede llamar otros procedimientos.

Disparador DML (DML *Triggers*)

Ejemplo de implementación de un *trigger*:

```
CREATE TRIGGER orden_automatica
        /* Evento que acciona el trigger */
        AFTER UPDATE OF stock ON inventario
        FOR EACH ROW
        /* restricción del trigger */
        WHEN (new.indica_automatico = 'T')
BEGIN
        /* acción del trigger */
        IF :new.stock < :new.stock_minimo   THEN
                INSERT INTO orden_pendiente  VALUES
                        (:new.id_parte, :new.cantidad_reorden, SYSDATE);
        END IF;
END;
```

Disparador DML (DML *Triggers*)

El momento de ejecución ("timing") se especifica con las cláusulas:

BEFORE

Para ejecutar la acción antes de la sentencia especificada en el evento.

Se usa :

- Cuando la acción será la que determine si la sentencia especificada en el evento se completará (se evita procesamiento innecesario y el ROLLBACK).
- Cuando se requiere obtener valores para una columna necesaria en una actualización con INSERT o UPDATE.

Disparador DML (DML *Triggers*)

El momento de ejecución ("timing") se especifica con las cláusulas :

AFTER

Para ejecutar la acción después de ejecutar la sentencia especificada en el evento.

Se usa :

- Cuando se desea la conclusión de la sentencia del evento antes de la acción.
- Cuando existe un BEFORE *trigger* para la misma sentencia y se desea ejecutar alguna acción después de la sentencia del evento.

Disparador DML (DML *Triggers*)

Ejemplo:

Muestra un *trigger* que inserta un registro en la tabla PRECIOS_PRODUCTOS cada vez que insertamos un nuevo registro en la tabla PRODUCTOS.

```
CREATE OR REPLACE TRIGGER TR_PRODUCTOS_01
  AFTER INSERT ON PRODUCTOS
  FOR EACH ROW
DECLARE
  -- local variables
BEGIN
  INSERT INTO PRECIOS_PRODUCTOS
  (CO_PRODUCTO,PRECIO,FX_ACTUALIZACION)
  VALUES
  (:NEW.CO_PRODUCTO,100,SYSDATE);
END ;
```

1

2

El *trigger* se ejecutará cuando sobre la tabla PRODUCTOS se ejecute una sentencia INSERT.

```
INSERT INTO PRODUCTOS (CO_PRODUCTO, DESCRIPCION) VALUES  ('000100','PRODUCTO
  000100');
```

Disparador DML (DML *Triggers*)

El Evento - Sintaxis

[AFTER|BEFORE]
    UPDATE ON NombreTabla
    INSERT ON NombreTabla
    DELETE ON NombreTabla
    UPDATE OF ListaColumnas ON NombreTabla
    INSERT OR UPDATE OR DELETE ON NombreTabla

Disparador DML (DML *Triggers*)

La Acción

La cantidad de ejecuciones de la acción determina el tipo de *trigger* :

Disparador "para cada fila" (Row trigger)

Es ejecutado cada vez que se ejecuta la sentencia especificada en el evento, cuando esta involucra un conjunto de filas.

Se especifica con la cláusula **FOR EACH ROW.**

Tanto la expresión de la Restricción, como las sentencias dentro de la Acción pueden usar los valores de columnas de la fila (:old y :new) siendo procesada anteriores y posteriores al evento.

Para estos se proporcionan los siguientes "nombres correlacionados“ para ser usados como calificadores:

- **new** y **old** para usarse en la Restricción
- **:new** y **:old** para usarse en la Acción

Disparador DML (DML *Triggers*)

La Acción

Modificación de los pseudos-registros:

- **:new** no se puede modificar en un *trigger* AFTER a nivel de fila
- **:old** nunca se puede modificar, sólo se puede leer

Sentencia DML	:old	:new
INSERT	No definido: NULL	Valores nuevos a insertar
UPDATE	Valores originales (antes de la orden)	Valores actualizados (después)
DELETE	Valores originales (antes de la orden)	No definido: NULL

Disparador DML (DML *Triggers*)

Resumen

- Puede especificarse el momento en el cual disparar la ejecución, existen dos posibilidades: antes (BEFORE) que se lleve a cabo la instrucción o después (AFTER).
- Dependiendo del momento de disparo, existen variables especiales (NEW y OLD) que nos permiten acceder a los datos de la tupla en cuestión antes y luego de la ejecución de la instrucción.

Disparador DML (DML *Triggers*)

Resumen

- Las variables **NEW** y **OLD** no están disponibles en todos los momentos ni tipos de eventos. La variable **NEW** existe solo para un *trigger* del tipo **INSERT/UPDATE**, accediendo a **NEW**.columna, accedemos al valor. Si es un *trigger* de tipo **BEFORE** se puede utilizar para asignar un valor nuevo a la columna, si es de tipo **AFTER** solo podemos leer el valor. Y la variable **OLD** existe solo para un *trigger* del tipo **AFTER UPDATE/ AFTER DELETE,** pero solo para lectura.
- Dentro del *trigger* podremos saber qué tipo de evento disparó la ejecución mediante la lectura de INSERTING, UPDATING y DELETING. Esto es especialmente útil cuando se definen condiciones de ejecución para varios tipos de eventos, por ejemplo, UPDATE OR DELETE .

Disparador DML (DML *Triggers*)

La Acción

Disparador "para la sentencia" (*Statement trigger*)

Es ejecutado una sola vez, al ejecutar la sentencia especificada en el evento, independientemente de la cantidad de filas afectadas por esta (aún cero).

Nota:

Recuerde que el *trigger* se activa para una tabla, pero las instrucciones no deben actualizar esa misma tabla.

Disparador DML (DML *Triggers*)

La Acción

```
UPDATE EMP
SET SAL = SAL * 1.5
WHERE DEPTNO = ‘81’
```

Afecta a 50 filas (50 empleados en el departamento ‘81’).

```
CREATE TRIGGER T1
AFTER UPDATE ON EMP
[FOR EACH ROW]
```

- Un TRIGGER con la opción FOR EACH ROW, se ejecutaría 50 veces (una vez por cada fila afectada).
- Si no se usa la opción mencionada, el TRIGGER se ejecutaría 1 sola vez sin importar la cantidad de filas afectadas.

Disparador DML (DML *Triggers*)

La Acción

```
CREATE TRIGGER T1
AFTER UPDATE ON EMP   --evento
FOR EACH ROW
BEGIN
        UPDATE DEP --acción
        SET TOTALSUELDO = TOTALSUELDO - :OLD.SAL + :NEW.SAL
        WHERE DEPTNO =  ‘81’;
END;
```

```
UPDATE EMP
SET SAL = SAL * 1.5
WHERE DEPTNO = '81'
```

AFTER

Para ejecutar la acción después de ejecutar la sentencia especificada en el evento

Se ejecuta 50 veces y se va actualizando el sueldo total del departamento ‘81’

Disparador DML (DML *Triggers*)

La Acción

```
CREATE TRIGGER T2
AFTER UPDATE ON EMP

BEGIN
     UPDATE DEP
     SET TOTALSUELDO = (SELECT SUM(SAL) FROM EMP
                              WHERE DEPTNO = ‘81’);
END;
```

Solo se ejecuta una vez y cambia el sueldo total del departamento ‘81’

Disparador DML (DML *Triggers*)

La Acción - más sobre nombres correlacionados

Algunos nombres correlacionados no tienen sentido dependiendo del *trigger*:

- Los valores antiguos de un *trigger* disparado al insertar son nulos
- Los nuevos valores de un *trigger* disparado al borrar son nulos
- El valor de una columna NEW no puede asignarse/modificarse en un **AFTER** *trigger* pues el *triggering statement* se ejecuta antes de que se dispare un *trigger* de este tipo.

```
…
AFTER UPDATE ON EMP
FOR EACH ROW
BEGIN
        ...
        :NEW.DEPTNO := ‘95’;
        …
END;
```

X no puede hacerse esto

Disparador DML (DML *Triggers*)

La Acción – múltiples eventos

Detectan el tipo de *triggering statement* que disparó el *trigger* en el caso de sentencias de accionamiento múltiples. Esto permite tomar diferentes acciones según el caso.

```
CREATE TRIGGER ...
... INSERT OR UPDATE OR DELETE ON NombreTabla
BEGIN
    IF INSERTING THEN ... END IF;

    IF UPDATING THEN ... END IF;

    IF DELETING THEN ... END IF;
END;
```

Disparador DML (DML *Triggers*)

La Acción – lista de campos

En un **UPDATE** *trigger* se puede especificar una columna en el predicado condicional para determinar si esta columna específica es actualizada.

```
CREATE TRIGGER…
…  UPDATE OF COL1, COL2 ON TABLA1…
BEGIN
   …
   IF UPDATING('COL1') THEN … END IF;
   …
END;
```

El código en la cláusula THEN se ejecuta si la sentencia de disparo **UPDATE** actualiza la columna COL1. La siguiente sentencia dispararía el *trigger* anterior y causaría que el predicado condicional se evaluara como verdadero:

```
UPDATE TABLA1 SET COL1 = COL1 + 100;
```

Disparador DML (DML *Triggers*)

Restricciones al crear *triggers*

Número de *triggers* por tabla: solo hay 12 combinaciones posibles:

[AFTER | BEFORE]
X
[ROW | STATEMENT]
X
[INSERT | UPDATE | DELETE]

Si se duplica cualquier tipo de *trigger* para una tabla se da un error en tiempo de compilación. Sin embargo, cada tabla puede tener más de 4 UPDATE *triggers* pues se puede especificar su activación por una columna específica.

Disparador DML (DML *Triggers*)

Sentencias SQL válidas en cuerpos de triggers

Sólo se permiten sentencias DML (Data Manipulation Language).

No se permite:

- Sentencias DDL
- Sentencias ROLLBACK, COMMIT ni SAVEPOINT
- Llamar a procedimientos que usen las sentencias mencionadas

Disparador DML (DML *Triggers*)

Sentencias SQL válidas en cuerpos de *triggers*

- El cuerpo de un *trigger* puede incluir cualquier instrucción del DML SQL, incluyendo SELECT (que debe ser un SELECT-INTO o un SELECT en la definición de un cursor), INSERT, UPDATE y DELETE.

- Una instrucción SQL dentro de un *trigger* puede insertar datos en una columna de tipo LONG o LONG RAW. Sin embargo, no se pueden declarar variables de estos tipos en el cuerpo del disparador. Además, ni **:NEW** ni **:OLD** pueden ser usados con columnas de tipo LONG o LONG RAW.

- Cuando una instrucción de un *trigger* produce que se dispare otro *trigger*, se dice que estos están “en cascada”. ORACLE permite hasta 32 *triggers* en cascada.

Disparador DML (DML *Triggers*)

Tablas Mutantes

Una tabla mutante es aquella que actualmente está siendo actualizada por una instrucción UPDATE, DELETE o INSERT.

Limitaciones en *triggers* de filas:

- Sus sentencias no pueden leer ni modificar una tabla mutante. Esto evita que el ROW *trigger* vea data inconsistente.
- Esta limitación no se da en un *trigger* de sentencia.

Estados

Disparador DML (DML *Triggers*)

Estados

Habilitado

Ejecuta la Acción cuando ocurre el evento y la condición (si es especificada) se cumple.

1. Forma implícita: CREATE [OR REPLACE] TRIGGER *trigger* ...
2. Forma explícita: ALTER TRIGGER *trigger* ENABLE
3. De tabla: ALTER TABLE *tabla* ENABLE ALL TRIGGERS

Disparador DML (DML *Triggers*)

Estados

Inhabilitado

No ejecuta la Acción, aunque ocurra el evento y se cumpla la condición. Puede inhabilitarse por :

- No encontrarse disponible un objeto que referencia
- Para realizar cargas de datos voluminosas o recargas

1. Forma explícita : ALTER TRIGGER *trigger* DISABLE

2. De tabla : ALTER TABLE *tabla* DISABLE ALL TRIGGERS

Importante

Para borrar un *trigger* usamos:

- DROP TRIGGER nombreTrigger;

Manejo de errores

Disparador DML (DML *Triggers*)

***Triggers* y Manejo de Errores.**

- Oracle permite que los errores o controles definidos por el usuario en un *trigger* se manejen de modo que se devuelvan al procedimiento que contiene al *triggering statement* que lo llamó.

- Los manejos de error se devuelven con:

RAISE_APPLICATION_ERROR(NumError, 'Texto')

- NumError va de -20000 a -20999

Disparador DML (DML *Triggers*)

```
PROCEDURE fire_emp (emp_id NUMBER) IS
        invalid_empid EXCEPTION;
        PRAGMA EXCEPTION_INIT(invalid_empid, -20101);
BEGIN
        DELETE FROM emp WHERE empno = empid;
EXCEPTION
        WHEN invalid_empid THEN
                INSERT INTO emp_audit
                        VALUES (empid, 'Employee fired before probation
ended');
END;
```

El número de error es retornado al procedimiento que disparó el trigger

```
TRIGGER emp_probation
BEFORE DELETE ON emp
FOR EACH ROW
BEGIN
        IF (sysdate - old.hiredate) < 30 THEN
                raise_application_error(-20101, 'Employee on probation');
        END IF;
END;
```

Disparador DML (DML *Triggers*)

Conclusiones:

- Solo hemos revisado el uso de DML *Triggers*
- Dos tipos de *trigger*:
  - De sentencia y de fila
- Tener en cuenta la estructura de un *trigger*: el evento, la restricción y la acción

Conclusiones

- Solo hemos revisado el uso de DML *Triggers*
- Hay dos tipos de *trigger*: (1) sentencia y (2) fila
- Tener en cuenta la estructura de un *trigger*:
  - el evento
  - la restricción
  - la acción

Referencias

- AR. Elmasri y S.B. Navathe. (2007). Fundamentos de Sistema de Base de Datos, 5ta edición
- Oracle Help Center. (02 de noviembre de 2024). Database PL/SQL User's Guide and Reference. https://docs.oracle.com/cd/B19306_01/appdev.102/b14261/overview.htm#sthref283

¡Gracias!

PUCP