A. Introducción

Además de los bloques PL/SQL anónimos utilizados por SQL*PLus o por las herramientas de desarrollo (Oracle*FORMS, Oracle*Reports...), se puede emplear código PL/SQL en determinados objetos de la base de datos, como los procedimientos almacenados (PROCEDURE, FUNCTION, PACKAGE) y los disparadores (TRIGGER) de la base de datos.

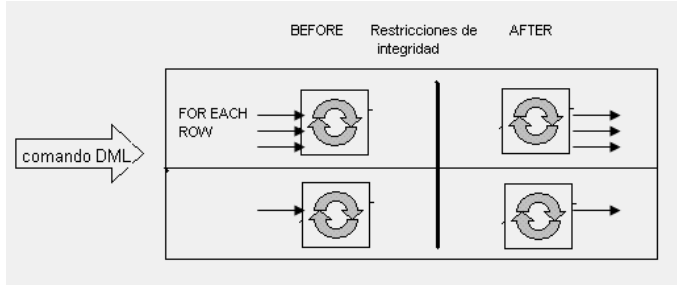
B. Los DATABASES TRIGGERS

Un disparador es un bloque PL/SQL asociado a una tabla. Este bloque se ejecutará cuando se aplique a la tabla una instrucción DML (INSERT, UPDATE, DELETE).

El bloque PL/SQL que constituye el disparador puede ejecutarse antes o después de la actualización y por la tanto, antes o despues de la verificación de las restricciones de integridad.

Los disparadores ofrecen una solución procedimental para definir restricciones complejas o que tengan en cuenta datos procedentes de varias filas o de varias tablas, como por ejemplo, para garantizar el hecho de que un cliente no pueda tener más de dos pedidos no pagados. Sin embargo, los disparadores no deben emplearse cuando sea posible establecer una restricción de integridad. En efecto, las restricciones de integridad se definen en el nivel de tabla y forman parte de la estructura de la propia tabla, por lo que la verificación de estas restricciones es mucho más rápida. Además, las restricciones de integridad garantizan que todas las filas de las tablas respetan dichas restricciones, mientras que los disparadores no tienen en cuenta los datos ya contenidos en la tabla en el momento de definirlos.

El bloque PL/SQL asociado a un disparador se puede ejecutar para cada fila afectada por la instrucción DML (opción FOR EACH ROW), o una sola vez para cada instrucción DML ejecutada (opción predeterminada).



Ejecución antes o después de la verificación de las restricciones de integridad para cada fila o cada instrucción.

En los disparadores BEFORE y FOR EACH ROW, se pueden modificar los datos que van a ser insertados en la tabla, de modo que respeten las restricciones de integridad. También es posible ejecutar consultas de tipo SELECT sobre la tabla a la que se aplica la instrucción DML, aunque únicamente en el marco de un disparador BEFORE INSERT. Todas estas operaciones no se pueden realizar en los disparadores AFTER, ya que después de la verificación de las restricciones de integridad no es posible modificar los datos y, dado que la modificación (adición o eliminación) de la fila no se ha terminado, no es posible ejecutar consultas de tipo SELECT sobre la tabla.

También se pueden incluir disparadores en las vistas (VIEW), con el fin de capturar las instrucciones DML que se pueden ejecutar sobre ellas. Estos disparadores permiten controlar todas las operaciones realizadas sobre las vistas y, para el usuario final, la vista es completamente similar a una tabla, ya que puede realizar sobre ella operaciones INSERT, UPDATE y DELETE. Estos disparadores son de tipo INSTEAD OF, es decir, que su ejecución va a reemplazar a la instrucción DML a la que estén asociados. Este tipo de disparador sólo puede definirse en vistas y es el único tipo de disparador que puede implementarse en ellas.

Principio de funcionamiento de los disparadores instead of

Sintaxis

```
CREATE [OR REPLACE] TRIGGER nombre_disparador
{BEFORE/AFTER/INSTEAD OF}
{INSERT/UPDATE[OF col,...]/DELETE}
ON Nombre_tabla [FOR EACH ROW]
[FOLLOWS nombre_otro_trigger[,…]]
[ENABLE/DISABLE]
[WHEN (condition)]
Bloc PL/SQLv
```

`OR REPLACE`

Reemplaza la descripción del disparador (TRIGGER) si ya existe.

`BEFORE`

El bloque PL/SQL se ejecuta ANTES de la verificación de las restricciones de tabla y de actualizar los datos almacenados en la misma.

`AFTER`

El bloque PL/SQL se ejecuta DESPUÉS de la actualización de los datos contenidos en la tabla.

`INSTEAD OF`

El bloque PL/SQL siguiente reemplaza el procesamiento estándar asociado a la instrucción que ha activado al disparador (sólo por una vista).

`INSERT/UPDATE [OF col, ...]/DELETE`

Instrucción asociada a la activación del disparador. Varias instrucciones pueden activar un mismo disparador y se combinan mediante el operador OR.

`FOR EACH ROW`

El disparador se ejecuta para cada fila tratada por la instrucción asociada.

`FOLLOWS nombre_otro_trigger[,…]`

Oracle permite definir varios disparadores para la misma tabla y el mismo evento. En este caso, el orden relativo de ejecución de estos disparadores es indeterminado. Si el orden de ejecución de estos disparadores es importante en su aplicación, puede utilizar la cláusula FOLLOWS disponible a partir de la versión 11. Esta cláusula permite indicar que el disparador debe ejecutarse después de los disparadores enumerados.

`ENABLE/DISABLE`

Esta cláusula permite indicar si el disparador está o no activo desde el momento de su creación; por defecto, un disparador de nueva creación está activo. Crear un disparador desactivado, permite verificar que se compila correctamente antes de ponerlo realmente en servicio. Un disparador creado desactivado, puede activarse más adelante utilizando una sentencia ALTER TRIGGER...ENABLE.

`WHEN (condición)`

La condición especificada debe ser verificada para que se ejecute el código.

Los datos de la tabla a la que está asociado el disparador son inaccesibles desde las instrucciones del bloque. Sólo la fila que se está modificando es accesible a través de dos variables de tipo RECORD: OLD y NEW, las cuales poseen la estructura de la tabla o de la vista asociada. Estas variables pueden ser utilizadas en la cláusula WHEN del disparador (TRIGGER) o en el bloque de instrucciones. En este último caso, se referencian como variables host mediante el prefijo ":" (:OLD.nombre_campo, :NEW.nombre_campo).

La palabra OLD permite conocer qué fila se va a eliminar en un disparador DELETE o la fila que se va a modificar en un disparador UPDATE. La palabra NEW permite conocer cuál es la nueva fila insertada en un disparador INSERT o la fila después de su modificación en un disparador UPDATE.

Los nombres OLD y NEW están definidos de manera predeterminada, aunque es posible utilizar otros nombres empleando la cláusula `REFERENCING OLD AS nuevo_nombre NEW AS nuevo_nombre`.
Esta cláusula se incluye justo antes de la cláusula FOR EACH ROW (si existe) en la definición del disparador.

Ejemplo

Ejecución de un bloque PL/SQL antes de una eliminación en la tabla CLIENTES por parte del usuario FLORENCIO:

```
CREATE TRIGGER antes_elim_cli
      BEFORE    DELETE
      ON FLORENCIO.CLIENTES
      DECLARE
         ...
      BEGIN
         ...
      END;
```

Ejecución de un bloque PL/SQL después de actualizar cada fila de la tabla ARTICULOS, cuando el precio antiguo es mayor que el nuevo:

```
create or replace trigger post_actprecio
  after update of precio
  on articulos
  for each row
  when (old.precio > new.precio)
declare
  ...
begin
  ...
end;
```

Para cada pedido, se desea conocer el nombre del usuario de Oracle que lo ha introducido. La primera etapa consiste en añadir una nueva columna a la tabla de pedidos. Esta columna debe aceptar el valor NULL ya que, para las filas de los pedidos existentes, el nombre del usuario de Oracle es desconocido.

Modificación de la tabla PEDIDOS:

```
SQL> alter table pedidos
  2    add (usuario varchar2(30));

Tabla modificada.

SQL>
```

En el disparador, se cambia el nombre de la nueva fila, y el disparador se ejecuta antes de la verificación de las restricciones de integridad para cada fila insertada en la tabla de pedidos.
Definición del disparador:

```
SQL> create or replace trigger bf_ins_pedidos
  2     before insert
  3     on pedidos
  4     referencing new as nuevo
  5     for each row
  6  begin
  7     select user into :nuevo.usuario from dual;
  8  end;
  9  /

Disparador creado.

SQL>
```

Se desea saber el número de pedidos introducido por el usuario de Oracle. Para evitar escribir una consulta que recorra la tabla de pedidos completa, operación que puede resultar muy pesada, el disparador se limita a actualizar una tabla de estadísticas.
Creación de la tabla de estadísticas:

```
SQL> create table stat_util(
  2    usuario varchar2(30),
  3    numero integer);

Tabla creada.

SQL>
```

El disparador debe asegurarse de que el usuario existe en la tabla de estadísticas y, si todavía no existe, debe crearlo. El disparador se ejecuta después de cada inserción de fila, por razones de optimización en caso de que se violen las restricciones de integridad.

Código del disparador:

```
SQL> create or replace trigger af_ins_pedidos
  2     after insert
  3     on pedidos for each row
  4  declare
  5     vnumero stat_util.numero%type;
  6  begin
  7     -- obtener el número del usuario actual
  8     select numero into vnumero
  9     from stat_util
 10     where usuario=:new.usuario;
 11     -- actualizar el valor
 12     update stat_util
 13     set numero=vnumero+1
 14     where usuario=:new.usuario;
 15  exception
 16     when no_data_found then
 17        insert into stat_util (usuario, numero)
 18        values (:new.usuario,1);
 19  end;
 20  /

Disparador creado.

SQL>
```