Disparadores

1 Definición de disparadores en ORACLE

Un disparador es un procedimiento almacenado en la base de datos que está asociado a una tabla y que se ejecuta implícitamente ("se dispara") cuando se realiza una instrucción INSERT, UPDATE o DELETE sobre la tabla asociada al disparador.

En un disparador se pueden usar instrucciones SQL e instrucciones PL/SQL y se puede llamar a otros procedimientos.

Los disparadores se almacenan en la base de datos separados de las tablas asociados a ellos.

Para crear un disparador se usa la instrucción CREATE TRIGGER, cuyo formato básico es el siguiente:

```
CREATE   [OR REPLACE] TRIGGER [esquema.] disparador
         {BEFORE | AFTER | INSTEAD OF}
         {DELETE | INSERT | UPDATE [OF columna [, columna] ...]}
         [OR {DELETE | INSERT | UPDATE [OF columna [, columna] ...]}] ...
         ON [esquema.]tabla
         FOR EACH ROW
         [WHEN (condición)]
         bloque pl/sql
```

OR REPLACE:

Permite cambiar la definición de un disparador que ya existe sin tener que borrarlo antes.

esquema:

Hace referencia al nombre del esquema que contiene al disparador. Si se omite, el disparador se crea en el esquema del usuario.

disparador:

es el nombre del disparador. Los nombres de los disparadores deben ser únicos con respecto a otros disparadores en el esquema, aunque no tienen porque ser únicos con respecto a otros objetos del esquema, como tablas, vistas y procedimientos.

BEFORE:

Indica que el disparador se dispara antes de ejecutar la instrucción disparador (INSERT, UPDATE O DELETE).

AFTER:

Indica que el disparador se dispara después de ejecutar la instrucción disparador (INSERT, UPDATE O DELETE).

INSTEAD OF:

Permite transformar una actualización sobre una vista no directamente actualizable en actualizaciones adecuadas sobre las tablas base.

DELETE:

Asocia el disparador con una instrucción DELETE, de tal modo que siempre que se borre una fila de la tabla el disparado se ejecuta.

INSERT:

Asocia el disparador con una instrucción INSERT, de tal modo que siempre que se añada una fila a la tabla el disparado se ejecuta.

UPDATE OF:

Asocia el disparador con una instrucción UPDATE, de tal modo que siempre que se modifique un valor en alguna columna de la tabla o en una de las columnas especificadas 3en la cláusula OF, si se utiliza esta cláusula, el disparador se ejecuta.

ON:

Hace referencia al esquema y a la tabla sobre la que se crea el disparador. Si se omite el nombre esquema, se supone que la tabla se encuentra en el esquema del usuario.

FOR EACH ROW:

Define el disparador como un disparador fila. Un disparador fila se dispara una vez por cada fila afectada por el disparador.

WHEN:

Permite especificar una restricción para el disparador, definiéndola mediante una condición SQL que debe satisfacerse para ejecutar el disparador. Esta condición no puede contener consultas.

Un disparador fila sólo puede tener una restricción que se evaluará para cada fila que se vea afectada por la instrucción disparador

bloque pl/sql:

Es el bloque PL/SQL que ORACLE ejecuta cuando activa el disparador. No puede contener las instrucciones SQL: COMMIT, ROLLBACK, y SAVEPOINT.

Cuando se crea un disparador, es *obligatorio* terminar la creación del mismo con una barra inclinada / como primer y único carácter de la última línea.

Crear un disparador cuya sintaxis sea correcta no dando errores al ser compilado por ORACLE, no significa que sea semánticamente correcto. De hecho, el error más frecuente en el uso de disparadores aparece al ejecutarlos y tratar de acceder o referenciar directa o indirectamente con algún tipo de operación a la misma fila de la tabla que activó el disparador, cosa que está *totalmente prohibida*.

1.1 Partes de un disparador

Un disparador consta de tres partes básicas: una instrucción disparador, una restricción disparador y una acción disparador.

Una *instrucción disparador* define las instrucciones SQL (INSERT, UPDATE o DELETE) que cundo se ejecutan para una tabla específica causan que el disparador asociado a dicha tabla sea activado.

Por ejemplo, si se crea un disparado que contiene la instrucción disparador:

```
... UPDATE OF nombre ON empleados...
```

siempre que se modifique la columna nombre de una fila en la tabla empleados se activa el disparador.

Para incluir en una instrucción disparador varias iinstrucciones DML se utiliza el formato siguiente:

```
... INSERT OR UPDATE OR DELETE ON empleados...
```

Lo que especifica que el disparador se activa siempre que se ejecute una nstrucción INSERT, UPDATE o DELETE sobre la tabla empleados. En este caso, para detectar qué instrucción ha activado el disparador se utilizan los predicados condicionales INSERTING, UPDATING y DELETING. Por lo tanto, se pueden crear disparadores que ejecutarán fragmentos de código distinto según la instrucción disparador que se ejecute.

La *restricción disparador* especifica la expresión booleana (condición SQL) que debe ser verdadera (TRUE) para que se ejecute la acción disparador. Si la expresión booleana se evalúa falsa (FALSE) la acción disparador no se realiza.

La restricción disparador se utiliza para los disparadores fila, utilizando la cláusla WHEN.

Una *acción disparador* contiene las instrucciones SQL y el código PL/SQL que se ejecutan cuando un disparador es activado.

Una acción disparador puede: contener instrucciones SQL y PL/SQL, definir elementos del lenguaje PL/SQL como variables, constantes, cursores, excepciones, etc., y llamar a procedimientos.

1.2 Modificación de un disparador

Un disparador no se puede modificar explícitamente, debe reemplazarse con una nueva definición de disparador utilizando la opción OR REPLACE en la instrucción CREATE TRIGGER.

Al crear una nueva versión de un disparador que ya existe en la base de datos se reemplaza la versión antigua por la nueva sin que se vean afectados los privilegios que tenía el disparador antiguo.

Otra forma de modificar un disparador consiste en borrar el disparador y volver a crear otro con las modificaciones necesarias, En este caso, odos los privilegios del disparador antiguo se borran y deben darse otra vez para el disparador nuevo.

Un disparador se puede borrar utilizando la instrucción DRO TRIGGER, cuyo formato es el siguiente:

```
DROP TRIGGER [esquema.]disparador
```

1.3 Nombres correlativos

Existen dos nombre correlativos, OLD y NEW, para cada columna de la tabla que se está modificando.

Para referenciar a los valores nuevos de la columna se utiliza el calificador NEW aantes del nombre de la columna, y para los valores viejos se utiliza el calificador OLD antes del nombre de la columna.

Dependiendo del tipo de instrucción disparador puede suceder que alguns nombre correlativos no tengan significado:

- Un disparador activado por una instrucción INSERT tiene acceso solamente a los valores nuevos de la columna. Los valores viejos son NULL.
- Un disparador activado por una instrucción UPDATE tiene acceso a los valores viejos y nuevos de la columna para los disparadores fila BEFORE y AFTER.
- Un disparador activado por una instrucción DELETE tiene acceso solamente a los valores viejos de la columna. Los valores nuevos son NULL.

Los nombres correlativos también se pueden usar en la expresión booleana de uan cláusula WHEN. Cuando los calificadores NEW y OLD se usan en el cuerpo de un disparador (bloque pl/sql) deben ir precedidos por dos puntos (:) que no son necesarios cuando los calificadores se utilizan en la cláusula WHEN.

2 Bloques PL/SQL

Un bloque es la unidad básica de programación en PL/SQL, y para definirlo se usan las siguientes palabras reservadas:

- DECLARE (define la parte declarativa)
- BEGIN (define el comienzo de la parte ejecutable)
- EXCEPTION (define la parte de manejo de excepciones)
- END (indica el final del bloque)

Solamente es necesaria la parte ejecutable. Se puede anida un bloque dentro de otro bloque.

El formato de un bloque es el siguiente

```
[<<etiqueta>>]
[DECLARE    {declaración de variable
             |declaración de cursor
             |declaración de excepción
             |declaración de registro
             |declaración de tabla pl/sql
             |declaración de procedimiento
             |declaración de función} ...]
```

BEGIN
    Secuencia de instrucciones
[EXCEPTION manejador de excepción; [manejador de excepción;]...]
END [etiqueta]

etiqueta:

Opcionalmente se puede identificar un bloque PL/SQL mediante esta etiqueta, que debe ir encerrada entre << >> y debe aparecer al comienzo del bloque. Del mismo modo, la etiqueta puede aparecer al final del bloque.

DECLARE:

Define la parte declarativa de un bloque PL/SQL, donde se declaran los objetos locales, los cuales son visibles solamente dentro del bloque actual y de todos sus sub-bloques.

Para referenciar un objeto en una instrucción debe estar previamente declarado.

declaración de variable:

Declara variables y constantes.

declaración de cursor:

Declara un cursor.

declaración de excepción:

Declara excepciones.

declaración de tabla pl/sql:

Declara tablas PL/SQL

declaración de procedimiento:

Declara un procedimiento.

declaración de función:

Declara una función

BEGIN:

Define la parte ejecutable de un bloque PL/SQL, que contiene las instrucciones de manipulación y control de datos..

Secuencia de instrucciones:

Representa una secuencia de instrucciones, donde cada instrucción puede ser una instrucción PL/SQL incluyendo otro bloque. Cada instrucción debe terminar en punto y coma (;).

EXCEPTION:

Define la parte de manejo de excepciones en el bloque. Cuando un bloque encuentra una excepción detiene su ejecución y transfiere el control al manejador de excepciones adecuado. Una vez que el manejador de excepciones termina, la ejecución continúa en la siguiente instrucción del bloque.

Si PL/SQL no puede encotrar un un manejador de excepción detiene la ejecución del bloque y devuelve un mensaje de error.

manejador de excepción:

Define las instrucciones que se ejecutan cuando se encuentra la excepción.

END;

Indica el final del bloque PL/SQL. Debe ser la última palabra de un bloque.

3 Instrucciones PL/SQL

3.1 Instrucción IF

La instrucción IF ejecuta una sucesión de instrucciones dependiendo del valor de una condición. Su formato es el siguiente:

```
IF condición pl/sql
    THEN secuencia de instrucciones
    [ESLSEIF condición pl/sql THEN secuencia de instrucciones ]...
    [ELSE secuencia de instrucciones]
END IF
```

condición pl/sql:

Define la condición asociada con la secuencia de instrucciones.

Secuencia de instrucciones:

Representa una secuencia de instrucciones.

THEN:

Asocia la condición con la secuencia de instrucciones, de modo que si la condición se evalúa TRUE se ejecuta la secuencia de instrucciones.

ELSEIF:

Define una condición que se evaluará si tidas las condiciones han sido evaluadas FALSE o NULL.

ELSE:

Se ejecuta la secuencia de instrucciones asociada si todas las condiciones anteriores han sido evaluadas FALSE o NULL.

3.1.1 Formatos de IF

Hay tres formatos en la instrucción IF:

1. IF-THEN
2. IF-THEN-ELSE
3. IF-THEN-ELSEIF

1. IF-THEN

Es la forma más simple de la instrucción IF, nos permite asociar una condición con una secuencia de instrucciones.

```
IF condición pl/sql THEN
    secuencia de instrucciones;
END IF;
```

El funcionamiento es el siguiente:

- Si la condición se evalúa TRUE se ejeuta la secuencia de instrucciones.

- Si la condición se evalúa FALSE o NULL el control pasa a la siguiente instrucción.

2. IF-THEN-ELSE

El segundo formato formato de la instrucción IF es

```
IF condición pl/sql THEN
    secuencia de instrucciones 1:
ELSE
    secuencia de instrucciones 2;
END IF;
```

El funcionamiento es el siguiente:

- Si la condición se evalúa TRUE se ejeuta la secuencia de instrucciones 1.
- Si la condición se evalúa FALSE o NULL se ejecuta la secuencia de instrucciones 2.

Las cláusulas THEN y ELSE pueden incluir a otras instrucciones IF.

3. IF-THEN-ELSEIF

El tercer formato formato de la instrucción IF usa la palabra reservada ELSEIF permitiendo que se pueda seleccionar una acción de varias que son mutuamente excluyentes.

```
IF condición 1 pl/sql THEN
    secuencia de instrucciones 1;
ELSEIF condición 2 pl/sql
    secuencia de instrucciones 2;
[ELSE
    secuencia de instrucciones 3;]
END IF;
```

El funcionamiento es el siguiente:

- Se evalúan todas las condiciones empezando por la primera.
- Si una condición es TRUE, se ejecuta la secuencia de instrucciones asociada y el control pasa a la siguiente instrucción después de IF..
- Si todas las condiciones son evaluadas FALSE o NULL, se ejecuta la secuencia de instrucciones aosicada con la cláusula ELSE.

Una instrucción IF puede tener cualquier número de cláusualas ELSEIF; la cláusula ELSE es opcional.

3.2 Instrucción LOOP

La instrucción LOOP permite ejcutar una secuencia de instrucciones múltiples veces. Su formato es el siguiente:

```
[<<etiqueta>>]
[WHILE condición pl/sql]
[FOR {parámetro bucle numérico | parámetro bucle cursor}]
LOOP
    secuencia de instrucciones
```

END LOOP
[etiqueta]

etiqueta:

Opcionalmente este identificador etiqueta al bucle.. Debe aparecer al principio del bucle entre <<>>. También puede al final de la instrucción LOOP. Etiquetar un bucle será especialmente útil cundo se desee referenciar de manera inequívoca el contador del bucle dentro de él; para ello se nombraría como *etiqueta contador*.

Secuencia de instrucciones:

Representa la secuencia de instrucciones que se repite en cada iteración del bucle.

A su vez, e formato de parámetro bucle numérico es:

contador IN [REVERSE] límite inferior.. límite superior

Y el formato de parámetro bucle cursor es el siguiente:

Registro IN {cursor [(parámetro act [, parámetro act] ...)] | (instrucción select)}

Los elementos que aparecen en estos dos últimos tipos de parámetros se detallan al explicar seguidamente los distintos tipos de bucles:

3.2.1 Formatos de LOOP

PL/SQL soporta cuatro tipos de bucles:

1. Bucle básico
2. Bucle WHILE
3. Bucle FOR numérico
4. Bucle FOR para cursores

1. Bucle básico

Es la forma más simple de una instrucción LOOP que permite ejecutar la secuencia de instrucciones comprendida entre las palabras LOOP y END LOOP.

```
LOOP
    secuencia de instrucciones:
END LOOP;
```

El funcionamiento es el siguiente:

- Se ejcuta la secuencia de instrucciones en cada iteración del bucle.
- Después de ejecutarse la última instrucción de la secuencia el control pasa de nuevo al principio del bucle.
- Para terminar la ejecución del bucle se utiliza la instrucción EXIT.

El formato de la instrucción EXIT es el siguiente:

EXIT [etiqueta] [WHEN condición pl/sql]

Hay dos formas de utilizar la instrucción EXIT:

EXIT:

Cuando se encuentra una instrucción EXIT se fuerza la terminación incondicional del bucle, pasando el control a la siguiente instrucción después de la palabra reservada END LOOP.

EXIT WHEN:

Cuando se encuentra una instrucción EXIT WHEN se evalúa la condición de la cláusula WHEN. Si la condición es TRUE, el bucle termina y el control pasa a la siguiente instrucción después de la palabra reservada END LOOP. En caso contrario, se sigue con la ejecución del bucle.

2. Bucle WHILE

La instrucción WHILE LOOP permite ejecutar una secuencia de instrucciones de forma iterativa dependiendo del valor de una condición:

```
WHILE condición pl/sql LOOP
    secuencia de instrucciones:
END LOOP;
```

El funcionamiento es el siguiente:

- En cada iteración se evalúa la condición..
- Si la condición es TRUE, se ejcuta la secuencia de instrucciones y el control pasa a la evaluación de la condición.
- Si la condición se evalúa a FALSE o NULL, no se ejcuta la secuencia de instrucciones y el control pasa a la siguiente instrucción después de la palabra reservada END LOOP.

Puede ocurrir que la secuencia de instrucciones no se ejecute ninguna vez ya que la condición del bucle se evalúa al principio de la ejecución del mismo.

3. Bucle FOR numérico

La instrucción FOR LOOP permite ejecutar una secuencia de instrucciones un número de veces conocido a priori:

```
FOR contador IN [REVERSE] limite inferior… limite superior
LOOP
    secuencia de instrucciones:
END LOOP;
```

El funcionamiento es el siguiente:

- Se evalúa el rango del contador cuando se entra por primera vez al bucle y no se vuelve a evaluar..
- La secuencia de instrucciones se ejecuta tantas veces como indique el rango.
- Después de cada iteración el valor del contador del bucle se incrementa o decrementa.

Hay que tener presente que tanto límite inferior como límite superior representan expresiones cuyo resultado debe ser un valor entero.

Si no se usa la palabra reservada REVERSE, la iteración va del límite inferior al límite superior, incrementándose el valor del contador en cada iteración. Si se utiliza REVERSE, la iteración va del límite superior al límite inferior decrementándose el valor del contador en cada iteración.

Algunas características del contador que hay que tener en cuenta son:

- El contador está declarado implícitamente como una variable local de tipo entero.
- Su ámbito es el bucle y no se puede acceder a su valor fuera fuera del mismo.
- Dentro del bucle el contador puede referenciarse como una constante pero no se le puede asignar un valor.

4. Bucle FOR para cursores

Permite recorrer de forma automática todas las filas de un cursor previamente declarado.

```
FOR registro IN {cursor [(parámetro act [, parámetro act]…)]|(instrucción select)}
LOOP
    secuencia de instrucciones:
END LOOP;
```

parámetro act:

Representa una expresión PL/SQL que es tratada como actual en la llamada al cursor. El tipo de dato resultante debe ser compatible con el tipo de dato del parámetro formal al que sustituye.

instrucción select:

Es una instrucción cuya sintaxis es similar a la de una SELECT INTO, salvo que se omite toda la parte correspondiente a INTO, ya que la información que se obtenga se almacena en registro. PL/SQL gestiona implícitamente y de forma automática un cursor que accede a la información de esta consulta.

El funcionamiento de la instrucción FOR es el siguiente:

- Se crea implícitamente un registro que tiene la misma estructura de campos que las filas que pueden recuperarse con el cursos

  El ámbito del registro es el bucle, por lo que no se pueden referenciar sus campos fuera del mismo..

  Los campos del registro se referencian como: registro.columna
- Se abre implícitamente el cursor al comienzo del bucle.
- En cada iteración el registro índice contiene una nueva fila del cursor.
- Al finalizar el bucle, ya sea por la terminación de las flas del cursor o porque sse use una instrucción EXIT, el cursor se cierra implícitamente.

3.3 Instrucción SELECT INTO

La instrucción SELECT INTO es una variante de la instrucción SELECT que asigna los datos seleccionados a variables. Su formato es el siguiente:

```
SELECT item seleccionado [alias] [, ítem seleccionado [alias]]….
    INTO {variable [, variable] …| registro}
    FROM lista de tablas resto de instrucciones select
```

Ítem seleccionado:

Se le indican las columnas a recuperar e las tablas que intervienen en la selección. Son las columnas que se van a asignar a las variables indicadas o a los campos del registro correspondiente que se emplea para el almacenamiento.

INTO variable:

Se definen qué variables almacenarán los datos recuperador. Estas variables deben estar previamente declaradas con el mismo tipo al tipo de datos de la columna correspondiente. Debe haber tantas variables omo columnas se van a selccionar.

INTO registro:

Se indica que el dato recuperado se almacenará en una variable registro que previamente ha sido declarada utilizando el atributo %ROWTYPE Se accede a los valores de los campos mediante la siguiente notación: registro.campo.

lista de tablas:

Se especifica uno o más nombres de tablas o vistas separadas por comas, que tienen que estar accesibles en el momento de la ejecución de la instrucción SELECT.

alias:

Permite nombrar las columnas, tablas o vistas de otra forma, y puede usarse en la cláusula WHERE.

resto de instrucciones select:

Se hace referencia a cualquier estructura válida que pueda seguir a la cláusula FROM de una instrucción SELECT.

3.4 Instrucción RAISE

Mediante esta instrucción se puede transferir el control al manejador de excepciones correspondiente, finalizando la ejecución normal de un bloque PL/SQL o de un subprograma.

Las excepciones predefinidas son manejadas implícitamente por el sistema en tiempo de ejecución. Si se quiee ejecutar explícitamente una excepción definida por el usuario hay que utilizar la instrucción RAISE con el siguiente formato:

RAISE [excepción]

excepción:

Define el nombre de una excepción predefinida o definida por el usuario.

Si se omite excepción: dentro del ámbito de un manejador de excepciones, la instrucción RAISE vuelve a ejecutar la excepción actual.

La instrucción RAISE solamente se debe utilizar en un bloquue o subprograma cuando es imposible seguir ejecutando el bloque o subprograma debido a la ocurrencia de un error.

3.4.1 Procedimiento raise_application_error

El procedimiento raise_application_error es un procedimiento proporcionado por ORACLE, que permite que el usuario defin mensajes de error. El formato de llamada al procedimiento es el siguiente:

Raise_applocatio_error (número de error, mensaje de error)

número de error:

Número entero negativo en el rango -20000 a -999.

mensaje de error:

Cadena de caracteres de una longitud máxima de 512 bytes.

El procedimiento raise_application_error sólo se puede ejecutar desde un subprograma o un bloque, su funcionamiento es el siguiente:

- El subprograma o bloque finaliza su ejecución.
- No se guarda ningún cambio realizado en la base de datos.
- Se devuelve el número de error y el mensaje de error definidos por el usuario.