16

Creación de Disparadores de Base de Datos

ORACLE

Objetivos

Al finalizar esta lección, debería estar capacitado para hacer lo siguiente:

- **Describir diferentes tipos de disparadores**
- **Describir los disparadores de base de datos y su uso**
- **Crear disparadores de base de datos**
- **Describir las reglas para arrancar un disparador de base de datos**
- **Eliminar disparadores de base de datos**

ORACLE

Finalidad de la Lección

En esta lección aprenderá a crear y utilizar disparadores de base de datos.

Tipos de Disparadores

Un disparador:

- **Es un bloque PL/SQL o un procedimiento PL/SQL asociado a una tabla, una vista, un esquema o la base de datos**
- **Se ejecuta implícitamente siempre que se produce un evento en particular**
- **Puede ser:**
  - **Un disparador de aplicaciones: Arranca siempre que se produce un evento con una aplicación en particular**
  - **Un disparador de bases de datos: Arranca cuando se produce un evento de datos (como, por ejemplo, DML) o un evento del sistema (como, por ejemplo, la conexión o el cierre) en un esquema o en la base de datos**

ORACLE

Tipos de Disparadores

Los disparadores de aplicaciones se ejecutan implícitamente siempre que se produce un evento DML (Lenguaje de Manipulación de Datos) en particular dentro de una aplicación. Una aplicación desarrollada con Oracle Forms Developer sería un ejemplo de una aplicación que utiliza muchos disparadores.

Los disparadores de base de datos se ejecutan implícitamente cuando se emiten eventos de datos como, por ejemplo, una operación DML en una tabla (una sentencia disparadora `INSERT`, `UPDATE` o `DELETE`), un disparador `INSTEAD OF` en una vista o sentencias DDL (Lenguaje de Definición de Datos) como, por ejemplo, `CREATE` y `ALTER`, con independencia de qué usuario esté conectado o qué aplicación se utilice. Los disparadores de base de datos también se ejecutan implícitamente cuando se producen algunas acciones de usuario o del sistema de base de datos como, por ejemplo, cuando se conecta un usuario o cuando el DBA cierra la base de datos.

Nota: Los disparadores de base de datos se pueden definir en las tablas y en las vistas. Si se emite una operación DML en una vista, el disparador `INSTEAD OF` define qué acciones se realizan. Si estas acciones incluyen operaciones DML en tablas, entonces arranca cualquier disparador de las tablas base.

Los disparadores de base de datos pueden ser disparadores del sistema en una base de datos o en un esquema. Con una base de datos, los disparadores arrancan por cada evento para todos los usuarios . Con un esquema, los disparadores arrancan por cada evento para cada usuario específico.

Este curso cubre la creación de disparadores de base de datos. La creación de disparadores de base de datos basados en eventos del sistema se explica en la lección “Otros Conceptos sobre Disparadores”.

Instrucciones para Diseñar Disparadores

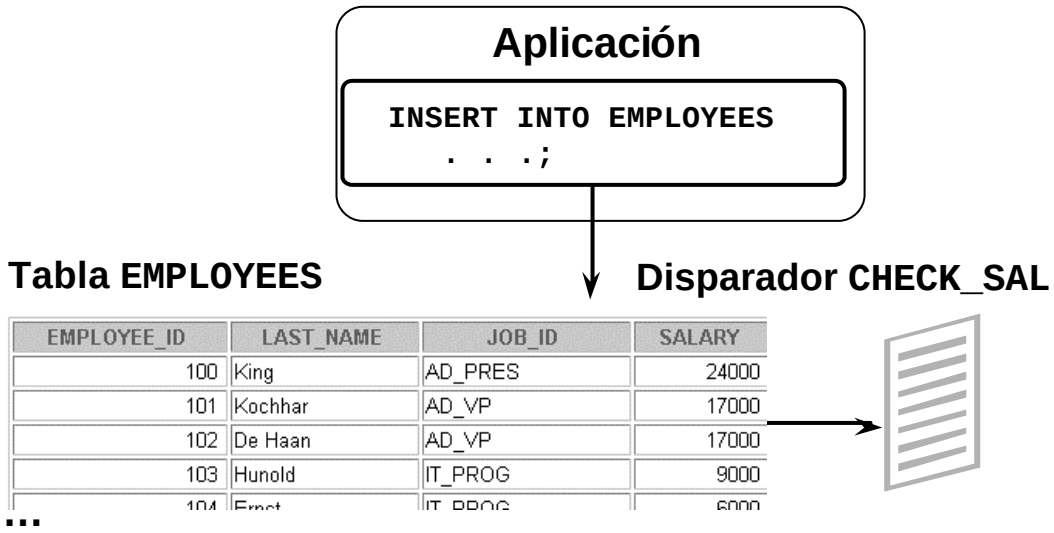
- **Diseñe disparadores para:**
  - **Realizar acciones relacionadas**
  - **Centralizar operaciones globales**
- **No diseñe disparadores:**
  - **Donde la funcionalidad ya esté incorporada en Oracle Server**
  - **Que dupliquen otros disparadores**
- **Cree procedimientos almacenados y llámelos en un disparador si el código PL/SQL es muy largo.**
- **El uso excesivo de disparadores puede producir interdependencias muy complejas que pueden ser difíciles de mantener en aplicaciones de gran tamaño.**

ORACLE

Instrucciones para Diseñar Disparadores

- Utilice disparadores para garantizar que, cuando se realice una operación específica, también se lleven a cabo una serie de acciones relacionadas.
- Utilice disparadores de base de datos sólo para las operaciones globales y centralizadas que deberían arrancar para la sentencia disparadora, con independencia de qué usuario o aplicación emita la sentencia.
- No defina disparadores que dupliquen o sustituyan la funcionalidad que ya está incorporada en la base de datos de Oracle. Por ejemplo, no defina disparadores para implementar reglas de integridad, ya que se puede hacer utilizando restricciones declarativas. Una manera muy fácil de recordar el orden de diseño de una regla de negocio es:
  - Utilizar restricciones incorporadas en Oracle Server como, por ejemplo, de clave primaria, de clave ajena, etc.
  - Desarrollar un disparador de base de datos o desarrollar una aplicación como, por ejemplo, un servlet o un EJB (Enterprise JavaBean) en la capa intermedia.
  - Utilizar una interfaz de presentación como, por ejemplo, Oracle Forms, HTML dinámico, JSP (Java ServerPages), etc., si no puede desarrollar la regla de negocio anteriormente mencionada, que puede ser una regla de presentación.
- El uso excesivo de disparadores puede producir interdependencias muy complejas que pueden ser difíciles de mantener en aplicaciones de gran tamaño. Utilice disparadores únicamente cuando sea necesario y tenga cuidado con los efectos en cascada y recursivos.
- Si la lógica del disparador es muy larga, cree procedimientos almacenados con la lógica y llámelos en el cuerpo del disparador.
- Tenga en cuenta que los disparadores de base de datos se arrancan para todos los usuarios cada vez que se produce un evento sobre el cual se haya creado el disparador.

Disparador de Base de Datos: Ejemplo



ORACLE

Ejemplo de un Disparador de Base de Datos

En este ejemplo, el disparador de base de datos CHECK_SAL comprueba los valores de los sueldos siempre que una aplicación intenta insertar una fila en la tabla EMPLOYEES. Los valores que están fuera de rango de acuerdo con la categoría del trabajo se pueden rechazar o se pueden permitir y registrar en una tabla de auditoría.

Creación de Disparadores DML

Una sentencia disparadora contiene:

- **La temporización del disparador**
  - **Para tablas: `BEFORE`, `AFTER`**
  - **Para vistas: `INSTEAD OF`**
- **Evento disparador: `INSERT`, `UPDATE` o `DELETE`**
- **Nombre de tabla: En tabla, vista**
- **Tipo de disparador: Fila o sentencia**
- **Cláusula `WHEN`: Condición de restricción**
- **Cuerpo del disparador: Bloque PL/SQL**

ORACLE

Disparador de Base de Datos

Antes de codificar el cuerpo del disparador, decida los valores de los componentes del disparador: la temporización del disparador, el evento disparador y el tipo de disparador.

Parte	Descripción	Valores Posibles

Temporización del disparador	Cuando el disparador arranca en relación con el evento disparador	`BEFORE`
		`AFTER`
		`INSTEAD OF`

Evento disparador	La operación de manipulación de datos en la tabla o la vista que arranca el disparador	`INSERT`
		`UPDATE`
		`DELETE`

Tipo de disparador	Cuántas veces se ejecuta el cuerpo del disparador	Sentencia Fila
Cuerpo del disparador	Qué acción realiza el disparador	Bloque PL/SQL completo

Si se definen varios disparadores para una tabla, recuerde que el orden en que arrancan varios disparadores del mismo tipo es arbitrario. Para asegurarse de que los disparadores del mismo tipo arrancan en un orden específico, combine los disparadores en uno solo que llame a procedimientos distintos en el orden deseado.

Componentes de los Disparadores DML

Temporización del disparador: ¿Cuándo debería arrancar el disparador?

- **BEFORE: Ejecuta el cuerpo del disparador antes del evento DML disparador en una tabla.**
- **AFTER: Ejecuta el cuerpo del disparador después del evento DML disparador en una tabla.**
- **INSTEAD OF: Ejecuta el cuerpo del disparador en lugar de la sentencia disparadora. Se utiliza en las vistas que no se pueden modificar de otra manera.**

Disparadores BEFORE

Este tipo de disparador se utiliza con frecuencia en las siguientes situaciones:

- Para determinar si se debería permitir que se completase la sentencia disparadora. (Esta situación permite eliminar el procesamiento innecesario de la sentencia disparadora y su eventual operación de rollback en los casos en los que se provoca una excepción en la acción disparadora.)
- Para derivar valores de columna antes de completar una sentencia `INSERT` o `UPDATE` disparadora.
- Para inicializar variables globales o indicadores y para validar complejas reglas de negocio.

Disparadores AFTER

Este tipo de disparador se utiliza con frecuencia en las siguientes situaciones:

- Para completar la sentencia disparadora antes de ejecutar la acción disparadora.
- Para realizar diferentes acciones en la misma sentencia disparadora si ya existe un disparador `BEFORE`.

Disparadores INSTEAD OF

Este tipo de disparador se utiliza para modificar de una manera transparente las vistas que no se pueden modificar directamente mediante sentencias DML de SQL, porque las vistas no son modificables de manera inherente.

Puede escribir sentencias `INSERT`, `UPDATE` y `DELETE` en la vista. El disparador `INSTEAD OF` funciona de manera invisible en segundo plano, realizando la acción codificada del cuerpo del disparador directamente en las tablas subyacentes.

Componentes de Disparadores DML

Evento disparador de usuario: ¿Qué sentencia DML hace que se ejecute el disparador? Puede utilizar:

- **INSERT**
- **UPDATE**
- **DELETE**

ORACLE

El Evento Disparador

La sentencia o el evento disparador puede ser una sentencia `INSERT`, `UPDATE` o `DELETE` en una tabla.

- Si el evento disparador es una sentencia `UPDATE`, se puede incluir una lista de columnas para identificar qué columnas hay que cambiar para arrancar el disparador. No obstante, no se puede especificar una lista de columnas para una sentencia `INSERT` o `DELETE`, porque estas sentencias afectan siempre a filas enteras.

```
. . . UPDATE OF salary . . .
```

- El evento disparador puede contener una, dos o las tres operaciones DML.

```
. . . INSERT o UPDATE o DELETE

. . . INSERT o UPDATE OF job_id . . .
```

Componentes de Disparadores DML

Tipo de disparador: ¿Se debería ejecutar el cuerpo del disparador por cada fila a la que afecte la sentencia o sólo una vez?

- **Sentencias: El cuerpo del disparador se ejecuta una vez para el evento disparador. Esta es la opción por defecto. Los disparadores de sentencias arrancan una vez, aunque no haya ninguna fila afectada.**
- **Filas: El cuerpo del disparador se ejecuta una vez por cada fila afectada por el evento disparador. Los disparadores de filas no se ejecutan si el evento disparador no afecta a ninguna fila.**

ORACLE

Disparadores de Sentencias y Disparadores de Filas

Puede especificar que el disparador se ejecute una vez por cada fila afectada por la sentencia disparadora (como, por ejemplo, una sentencia UPDATE de varias filas) o una vez para la sentencia disparadora sin importar a cuántas filas afecte.

Disparador de Sentencias

Los disparadores de sentencias arrancan una vez cuando se produce el evento disparador, aunque no haya ninguna fila afectada.

Los disparadores de sentencias son útiles si la acción del disparador no depende de los datos de las filas afectadas ni de los datos que proporciona el propio evento disparador. Un ejemplo sería un disparador que realiza una comprobación de seguridad compleja en el usuario actual.

Disparador de Filas

Los disparadores de filas arrancan cada vez que el evento disparador afecta a la tabla. Si el evento disparador no afecta a ninguna fila, el disparador de filas no se ejecuta.

Los disparadores de filas son útiles si la acción del disparador depende de los datos de las filas afectadas o de los datos que proporciona el propio evento disparador.

Componentes de Disparadores DML

Cuerpo del disparador: ¿Qué acción debería realizar el disparador?

El cuerpo del disparador es un bloque PL/SQL o una llamada a un procedimiento.

ORACLE

Cuerpo del Disparador

La acción del disparador define qué hay que hacer cuando se emite el evento disparador. El bloque PL/SQL puede contener sentencias SQL y PL/SQL y puede definir construcciones PL/SQL como, por ejemplo, variables, cursores, excepciones, etc. También se puede llamar a un procedimiento PL/SQL o a un procedimiento Java.

Además, los disparadores de filas utilizan nombres de correlación para acceder a nuevos y antiguos valores de columna de la fila que está procesando el disparador.

Nota: Los disparadores no pueden tener más de 32 K.

Secuencia de Arranque

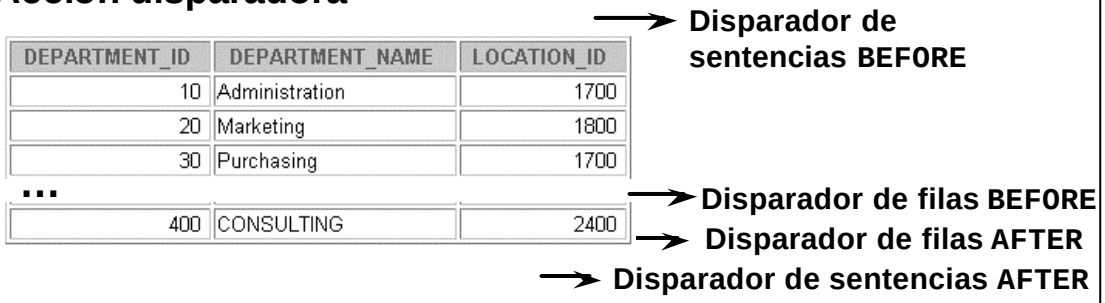
Cuando se manipule una sola fila, utilice la siguiente secuencia de arranque para un disparador en una tabla :

Sentencia DML

```
INSERT INTO departments (department_id,
                         department_name, location_id)
VALUES (400, 'CONSULTING', 2400);
```

1 row created.

Acción disparadora



ORACLE

Creación de Disparadores de Filas o de Sentencias

Cree un disparador de sentencias o de filas, teniendo en cuenta que el disparador se debe arrancar una vez por cada fila a la que afecte la sentencia disparadora, o bien sólo una vez para la sentencia disparadora, con independencia del número de filas a las que afecte.

Si la sentencia de manipulación de datos disparadora afecta a una sola fila, tanto el disparador de sentencias como el disparador de filas arrancan una sola vez.

Ejemplo

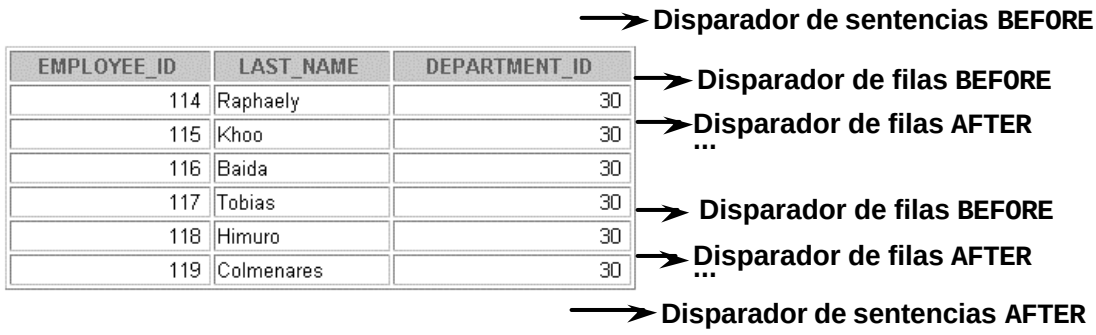
Esta sentencia SQL no diferencia los disparadores de sentencias de los disparadores de filas, ya que se ha insertado una sola fila en la tabla utilizando esta sintaxis.

Secuencia de Arranque

Cuando se manipulen muchas filas, utilice la siguiente secuencia de arranque para un disparador en una tabla:

```
UPDATE employees
    SET salary = salary * 1.1
    WHERE department_id = 30;
```

6 rows updated.



ORACLE

Creación de Disparadores de Filas o de Sentencias (continuación)

Si la sentencia de manipulación de datos disparadora afecta a muchas filas, el disparador de sentencias arranca sólo una vez y el disparador de filas una vez por cada fila a la que afecte la sentencia.

Ejemplo

La sentencia SQL de la transparencia anterior hace que el disparador de nivel de fila arranque un número de veces equivalente al número de filas que cumplen la cláusula `WHERE`, es decir, al número de empleados que pertenecen al departamento 30.

Sintaxis para Crear Disparadores de Sentencias DML

Sintaxis:

```
CREATE [OR REPLACE] PROCEDURE nombre_disparador
   temporización
      evento1 [OR evento2 OR evento3]
         ON nombre_tabla
cuerpo_disparador
```

Nota: Los nombres de los disparadores deben ser únicos con respecto a otros disparadores del mismo esquema.

ORACLE

Sintaxis para Crear un Disparador de Sentencias

nombre_disparador	Es el nombre del disparador
temporización	Indica el momento en el que arranca el disparador en relación con el evento disparador: BEFORE AFTER
evento	Identifica la operación de manipulación de datos que hace arrancar el disparador: INSERT UPDATE [OF *columna*] DELETE
tabla/nombre_vist	Indica la tabla asociada al disparador
cuerpo_del_disparador	Es el cuerpo del disparador que define la acción que realiza el disparador, que comienza con DECLARE o BEGIN y que termina con END, o una llamada a un procedimiento

Los nombres de los disparadores deben ser únicos con respecto a otros disparadores del mismo esquema. No obstante, no tienen que ser únicos con respecto a otros objetos del esquema como, por ejemplo, tablas, vistas y procedimientos.

Si se utilizan nombres de columnas junto con la cláusula UPDATE en el disparador, el rendimiento mejora, ya que el disparador arranca sólo cuando se actualiza esa columna en concreto y, así, evita un arranque involuntario al actualizar cualquier otra columna.

Creación de Disparadores de Sentencias DML

Ejemplo:

```
CREATE OR REPLACE TRIGGER secure_emp
BEFORE INSERT ON employees
BEGIN
  IF (TO_CHAR(SYSDATE,'DY') IN ('SAT','SUN')) OR
     (TO_CHAR(SYSDATE,'HH24:MI')
              NOT BETWEEN '08:00' AND '18:00')
   THEN RAISE_APPLICATION_ERROR (-20500,'You may
             insert into EMPLOYEES table only
                     during business hours.');
  END IF;
END;
/
```

```
Trigger created.
```

ORACLE

Creación de Disparadores de Sentencias DML

Puede crear un disparador de sentencias `BEFORE` para evitar que la operación disparadora se lleve a cabo cuando se viola una condición determinada.

Por ejemplo, puede crear un disparador que restrinja las inserciones en la tabla `EMPLOYEES` a ciertas horas de oficina, de lunes a viernes.

Si un usuario intenta insertar una fila en la tabla `EMPLOYEES` el sábado, ese usuario verá el mensaje, el disparador fallará y se realizará un rollback en la sentencia disparadora. Recuerde que `RAISE_APPLICATION_ERROR` es un procedimiento incorporado del servidor que devuelve un error al usuario y hace que el bloque PL/SQL falle.

Cuando un disparador de base de datos falla, Oracle Server realiza un rollback automáticamente en la sentencia disparadora.

Prueba de SECURE_EMP

```
INSERT INTO employees (employee_id, last_name,
            first_name, email, hire_date,
            job_id, salary, department_id)
VALUES (300, 'Smith', 'Rob', 'RSMITH', SYSDATE,
        'IT_PROG', 4500, 60);
```

```
INSERT INTO employees (employee_id, last_name, first_name, email,
      *
ERROR at line 1:
ORA-20500: You may insert into EMPLOYEES table only during business hours.
ORA-06512: at "PLSQL.SECURE_EMP", line 4
ORA-04088: error during execution of trigger 'PLSQL.SECURE_EMP'
```

ORACLE

Ejemplo

Se inserta una fila en la tabla EMPLOYEES fuera de las horas de oficina. Cuando la fecha y la hora no se corresponden con la temporización especificada en el disparador, se obtiene el mensaje de error que se muestra en la transparencia.

Uso de Predicados Condicionales

```
CREATE OR REPLACE TRIGGER secure_emp
BEFORE INSERT OR UPDATE OR DELETE ON employees
BEGIN
IF (TO_CHAR (SYSDATE,'DY') IN ('SAT','SUN')) OR
    (TO_CHAR (SYSDATE, 'HH24') NOT BETWEEN '08' AND '18')
THEN
   IF  DELETING THEN
     RAISE_APPLICATION_ERROR (-20502,'You may delete  from
            EMPLOYEES table only during business hours.');
   ELSIF  INSERTING THEN
     RAISE_APPLICATION_ERROR (-20500,'You may insert into
            EMPLOYEES table only during business hours.');
   ELSIF   UPDATING ('SALARY') THEN
     RAISE_APPLICATION_ERROR (-20503,'You may update
                SALARY only during business hours.');
   ELSE
     RAISE_APPLICATION_ERROR (-20504,'You may update
            EMPLOYEES table only during normal hours.');
   END IF;
  END IF;
END;
```

ORACLE

Combinación de Eventos Disparadores

Puede combinar varios eventos disparadores en uno utilizando los predicados condicionales especiales `INSERTING`, `UPDATING` y `DELETING` en el cuerpo del disparador.

Ejemplo

Se crea un disparador que restringe todos los eventos de manipulación de datos de la tabla `EMPLOYEES` a ciertas horas laborables, de lunes a viernes.

Creación de un Disparador de Filas DML

Sintaxis:

```
CREATE [OR REPLACE] TRIGGER nombre_disparador
  temporización
    evento1 [OR evento2 OR evento3]
      ON nombre_tabla
  [REFERENCING OLD AS antiguo | NEW AS nuevo]
FOR EACH ROW
  [WHEN (condición)]
cuerpo_disparador
```


Sintaxis para Crear un Disparador de Filas

nombre_disparador	Es el nombre del disparador
temporización	Indica el momento en el que arranca el disparador en relación con el evento disparador: BEFORE AFTER INSTEAD OF
evento	Identifica la operación de manipulación de datos que hace arrancar el disparador: INSERT UPDATE [OF *columna*] DELETE
nombre_tabla	Indica la tabla asociada al disparador
REFERENCING	Especifica los nombres de correlación de los valores nuevos y antiguos de la fila actual (los valores por defecto son OLD y NEW)
FOR EACH ROW	Designa que el disparador es un disparador de filas
WHEN	Especifica la restricción del disparador (este predicado condicional debe ir entre paréntesis y se evalúa por cada fila para determinar si el cuerpo del disparador se ejecuta o no)
cuerpo_del_disparador	Es el cuerpo del disparador que define la acción que realiza el disparador, que comienza con DECLARE o BEGIN y termina con END, o una llamada a un procedimiento

Creación de Disparadores de Filas DML

```
CREATE OR REPLACE TRIGGER restrict_salary
  BEFORE  INSERT OR UPDATE OF salary ON employees
    FOR EACH ROW
    BEGIN
      IF  NOT (:NEW.job_id IN ('AD_PRES', 'AD_VP'))
          AND :NEW.salary > 15000
      THEN
        RAISE_APPLICATION_ERROR (-20202,'Employee
                          cannot earn this amount');
     END IF;
END;
/
```

```
Trigger created.
```

ORACLE

Creación de un Disparador de Filas

Puede crear un disparador de filas BEFORE para evitar que la operación disparadora se lleve a cabo cuando se viola una condición determinada.

Puede crear un disparador que permita que sólo ciertos empleados puedan ganar un sueldo superior a 15.000.

Si el usuario intenta hacer esto, el disparador emite un error.

```
UPDATE employees
SET salary = 15500
WHERE last name = 'Russell';
```

```
UPDATE EMPLOYEES
       *
ERROR at line 1:
ORA-20202: Employee can not earn this amount
ORA-06512: at "PLSQL.RESTRICT_SALARY", line 5
ORA-04088: error during execution of trigger 'PLSQL.RESTRICT_SALARY'
```

Uso de los Cualificadores OLD y NEW

```
CREATE OR REPLACE TRIGGER audit_emp_values
AFTER DELETE OR INSERT OR UPDATE ON employees
FOR EACH ROW
BEGIN
  INSERT INTO audit_emp_table (user_name, timestamp,
     id, old_last_name, new_last_name, old_title,
     new_title, old_salary, new_salary)
  VALUES (USER, SYSDATE, :OLD.employee_id,
       :OLD.last_name, :NEW.last_name, :OLD.job_id,
       :NEW.job_id, :OLD.salary, :NEW.salary );
END;
/
```

```
Trigger created.
```

ORACLE

Uso de los Cualificadores OLD y NEW

En un disparador ROW, haga referencia al valor de una columna antes y después de que los datos cambien prefijándolo con los cualificadores OLD y NEW.

Operación de Datos	Valor Antiguo	Valor Nuevo
INSERT	NULL	Valor insertado
UPDATE	Valor antes de la actualización	Valor después de la actualización
DELETE	Valor antes de la supresión	NULL

- Los cualificadores OLD y NEW sólo están disponibles en los disparadores ROW.
- Prefije con dos puntos (:) estos cualificadores en todas las sentencias SQL y PL/SQL.
- No utilice los dos puntos (:) si se hace referencia a los cualificadores en la condición de restricción WHEN.

Nota: Los disparadores de filas pueden reducir el rendimiento si se realizan muchas actualizaciones en las tablas de mayor tamaño.

Uso de los Cualificadores OLD y NEW: Ejemplo del Uso de Audit_Emp_Table

```
INSERT INTO employees
       (employee_id, last_name, job_id, salary, ...)
VALUES (999, 'Temp emp', 'SA_REP', 1000, ...);

UPDATE employees
SET salary = 2000, last_name = 'Smith'
WHERE employee_id = 999;
```

1 row created.
1 row updated.

```
SELECT user_name, timestamp, ... FROM audit_emp_table
```

USER_NAME	TIMESTAMP	ID	OLD_LAST_N	NEW_LAST_N	OLD_TITLE	NEW_TITLE	OLD_SALARY	NEW_SALARY
PLSQL	28-SEP-01			Temp emp		SA_REP		1000
PLSQL	28-SEP-01	999	Temp emp	Smith	SA_REP	SA_REP	1000	2000

ORACLE

Uso de los Cualificadores OLD y NEW: Ejemplo del Uso de AUDIT_EMP_TABLE

Se crea un disparador en la tabla EMPLOYEES para agregar filas a una tabla de usuario, AUDIT_EMP_TABLE, y se registra la actividad de un usuario sobre la tabla EMPLOYEES. El disparador registra los valores de varias columnas antes y después de que los datos cambien utilizando los cualificadores OLD y NEW con su respectivo nombre de columna.

Existe la columna adicional COMMENTS en la tabla AUDIT_EMP_TABLE que no se muestra en esta transparencia.

Restricción de un Disparador de Filas

```
CREATE OR REPLACE TRIGGER derive_commission_pct
  BEFORE INSERT OR UPDATE OF salary ON employees
  FOR EACH ROW
  WHEN (NEW.job_id = 'SA_REP')
BEGIN
  IF  INSERTING
     THEN :NEW.commission_pct := 0;
  ELSIF :OLD.commission_pct IS NULL
     THEN :NEW.commission_pct := 0;
  ELSE
    :NEW.commission_pct := :OLD.commission_pct + 0.05;
  END IF;
END;
/
```

```
Trigger created.
```

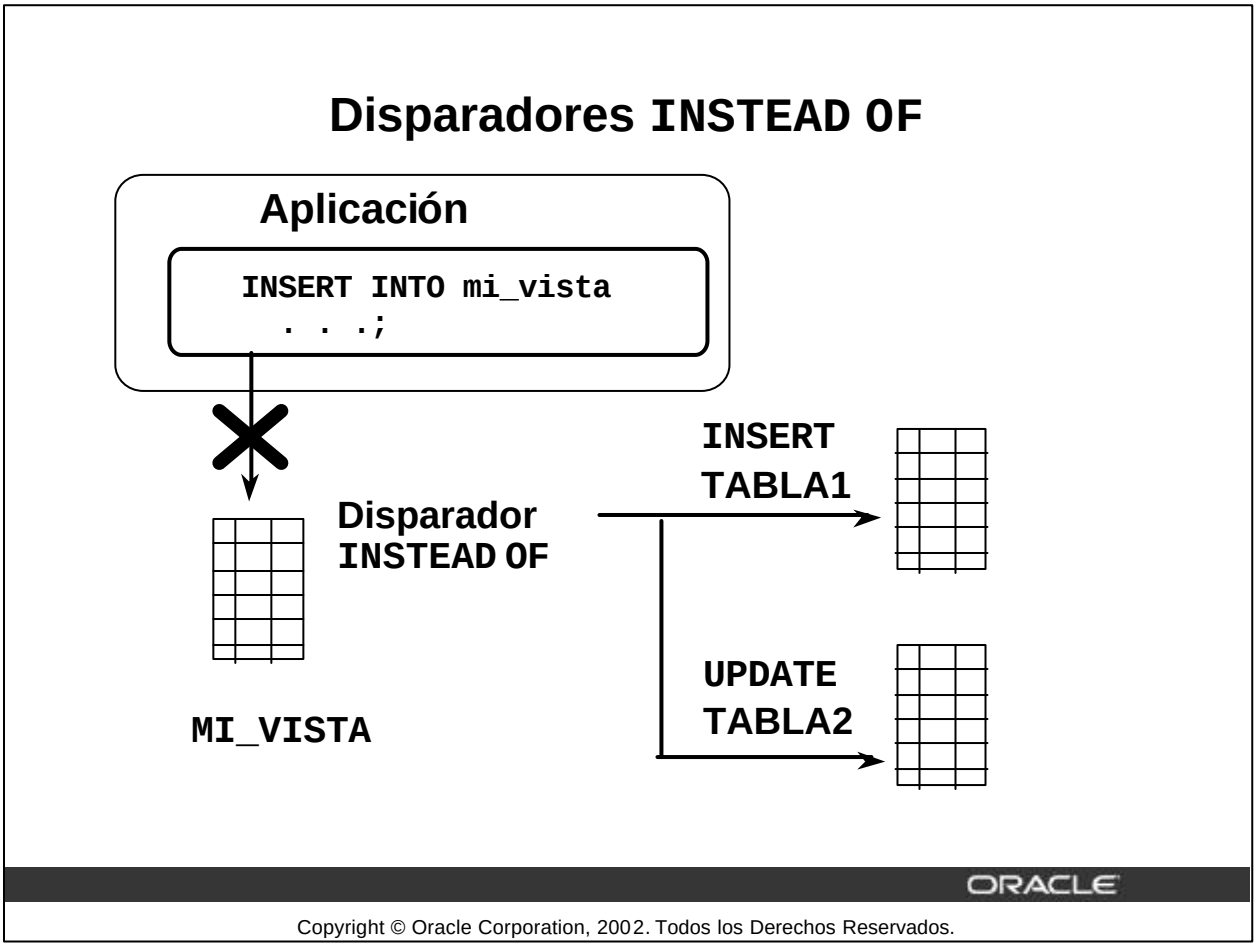
ORACLE

Ejemplo

Para restringir una acción del disparador a aquellas filas que cumplan una condición concreta, utilice una cláusula `WHEN`.

Se crea un disparador en la tabla `EMPLOYEES` para calcular la comisión de un empleado cuando se agrega una fila a la tabla `EMPLOYEES` o cuando se modifica el sueldo de un empleado.

El cualificador `NEW` no se puede prefijar con dos puntos (:) en la cláusula `WHEN` porque dicha cláusula se encuentra fuera de los bloques PL/SQL.



Disparadores INSTEAD OF

Utilice disparadores INSTEAD OF para modificar los datos en los cuales se ha emitido la sentencia DML para una vista inherentemente no actualizable. Estos disparadores se denominan disparadores INSTEAD OF porque, a diferencia de otros, Oracle Server arranca el disparador en lugar de ejecutar la sentencia disparadora. Este disparador sirve para realizar una operación INSERT, UPDATE o DELETE directamente en las tablas subyacentes.

Si escribe sentencias INSERT, UPDATE o DELETE para la vista, el disparador INSTEAD OF trabaja de manera invisible, en segundo plano, para encargarse de que se realicen las acciones correctas.

¿Por Qué Usar Disparadores INSTEAD OF?

Las vistas no se pueden modificar por medio de sentencias DML normales si la consulta de la vista contiene operadores de juegos, funciones de grupos, cláusulas como GROUP BY, CONNECT BY, START, el operador DISTINCT, o uniones. Si, por ejemplo, una vista se compone de más de una tabla, puede que una inserción en la vista conlleve una inserción en una tabla y una actualización en otra. Por lo tanto, escriba un disparador INSTEAD OF que arranque cuando escriba una inserción para la vista. En lugar de la inserción original, se ejecuta el cuerpo del disparador, lo que produce una inserción de datos en una tabla y una actualización en otra tabla.

Nota: Si una vista es inherentemente actualizable y tiene disparadores INSTEAD OF, los disparadores tienen prioridad. Los disparadores INSTEAD OF son disparadores de filas.

Cuando las inserciones o las actualizaciones de la vista se realizan con disparadores INSTEAD OF, no se fuerza la opción CHECK de las vistas. El cuerpo del disparador INSTEAD OF debe forzar la comprobación.

Creación de un Disparador INSTEAD OF

Sintaxis:

```
CREATE [OR REPLACE] TRIGGER nombre_disparador
  INSTEAD OF
    evento1 [OR evento2 OR evento3]
      ON nombre_vista
   [REFERENCING OLD AS antiguo | NEW AS nuevo]
[FOR EACH ROW]
cuerpo_disparador
```

ORACLE

Sintaxis para Crear un Disparador INSTEAD OF

nombre_disparador	Es el nombre del disparador.
INSTEAD OF	Indica que el disparador pertenece a una vista
evento	Identifica la operación de manipulación de datos que hace arrancar el disparador: INSERT UPDATE [OF columna] DELETE
nombre_vista	Indica la vista asociada al disparador
REFERENCING	Especifica los nombres de correlación de los valores nuevos y antiguos de la fila actual (los valores por defecto son OLD y NEW)
FOR EACH ROW	Designa que el disparador es un disparador de filas; los disparadores INSTEAD OF sólo pueden ser disparadores de filas: si se omite, el disparador seguirá estando definido como disparador de filas.
cuerpo_del_disparador	Es el cuerpo del disparador que define la acción que realiza el disparador, que comienza con DECLARE o BEGIN y termina con END, o una llamada a un procedimiento

Nota: Los disparadores INSTEAD OF sólo se pueden escribir para las vistas. Las opciones BEFORE y AFTER no son válidas.

Creación de un Disparador INSTEAD OF

Ejemplo:

En el siguiente ejemplo se crean dos nuevas tablas, NEW_EMPS y NEW_DEPTS, basadas en las tablas EMPLOYEES y DEPARTMENTS, respectivamente. También se crea la vista EMP_DETAILS a partir de las tablas EMPLOYEES y DEPARTMENTS. En el ejemplo también se crea un disparador INSTEAD OF, NEW_EMP_DEPT. Para insertar una fila en la vista EMP_DETAILS, en lugar de insertar la fila directamente en la vista, se agregan filas a las tablas NEW_EMPS y NEW_DEPTS, basándose en los datos de la sentencia INSERT. Igualmente, cuando se modifica o se suprime una fila por medio de la vista EMP_DETAILS, las filas correspondientes de las tablas NEW_EMPS y NEW_DEPTS resultan afectadas.

```
CREATE TABLE new_emps AS
    SELECT employee_id, last_name, salary, department_id,
           email, job_id, hire_date
    FROM employees;

CREATE TABLE new_depts AS
    SELECT d.department_id, d.department_name, d.location_id,
       sum(e.salary) tot_dept_sal
    FROM employees e, departments d
    WHERE e.department_id = d.department_id
    GROUP BY d.department_id, d.department_name, d.location_id;

CREATE VIEW emp_details AS
     SELECT e.employee_id, e.last_name, e.salary, e.department_id,
            e.email, e.job_id, d.department_name, d.location_id
     FROM employees e, departments d
     WHERE e.department_id = d.department_id;

CREATE OR REPLACE TRIGGER new_emp_dept
INSTEAD OF INSERT OR UPDATE OR DELETE ON emp_details
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    INSERT INTO new_emps
    VALUES (:NEW.employee_id, :NEW.last_name, :NEW.salary,
            :NEW.department_id, :NEW.email, :New.job_id, SYSDATE);
    UPDATE new_depts
    SET tot_dept_sal = tot_dept_sal + :NEW.salary
    WHERE department_id = :NEW.department_id;
  ELSIF DELETING THEN
    DELETE FROM new_emps
    WHERE employee_id = :OLD.employee_id;
    UPDATE new_depts
    SET tot_dept_sal = tot_dept_sal - :OLD.salary
    WHERE department_id = :OLD.department_id;
```

Creación de un Disparador INSTEAD OF (continuación)

Ejemplo:

```
  ELSIF UPDATING ('salary')
  THEN
    UPDATE new_emps
    SET salary = :NEW.salary
    WHERE employee_id = :OLD.employee_id;
    UPDATE new_depts
    SET tot_dept_sal = tot_dept_sal + (:NEW.salary - :OLD.salary)
    WHERE department_id = :OLD.department_id;
  ELSIF UPDATING ('department_id')
  THEN
    UPDATE new_emps
    SET department_id = :NEW.department_id
    WHERE employee_id = :OLD.employee_id;
    UPDATE new_depts
    SET tot_dept_sal = tot_dept_sal - :OLD.salary
    WHERE department_id = :OLD.department_id;
    UPDATE new_depts
    SET tot_dept_sal = tot_dept_sal + :NEW.salary
    WHERE department_id = :NEW.department_id;
  END IF;
END;
/
```

Nota: Este ejemplo se explica en la siguiente página por medio de gráficos.

Creación de un Disparador INSTEAD OF

INSERT en EMP_DETAILS que está basada en las tablas EMPLOYEES y DEPARTMENTS

1
```
INSERT INTO emp_details(employee_id, ... )
VALUES(9001,'ABBOTT',3000,10,'abbott.mail.com','HR_MAN');
```

INSTEAD OF INSERT en EMP_DETAILS →

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	EMAIL	JOB_
100	King	90	SKING	AD_PRE
101	Kochhar	90	NKOCHHAR	AD_VP
102	De Haan	90	LDEHAAN	AD_VP

...

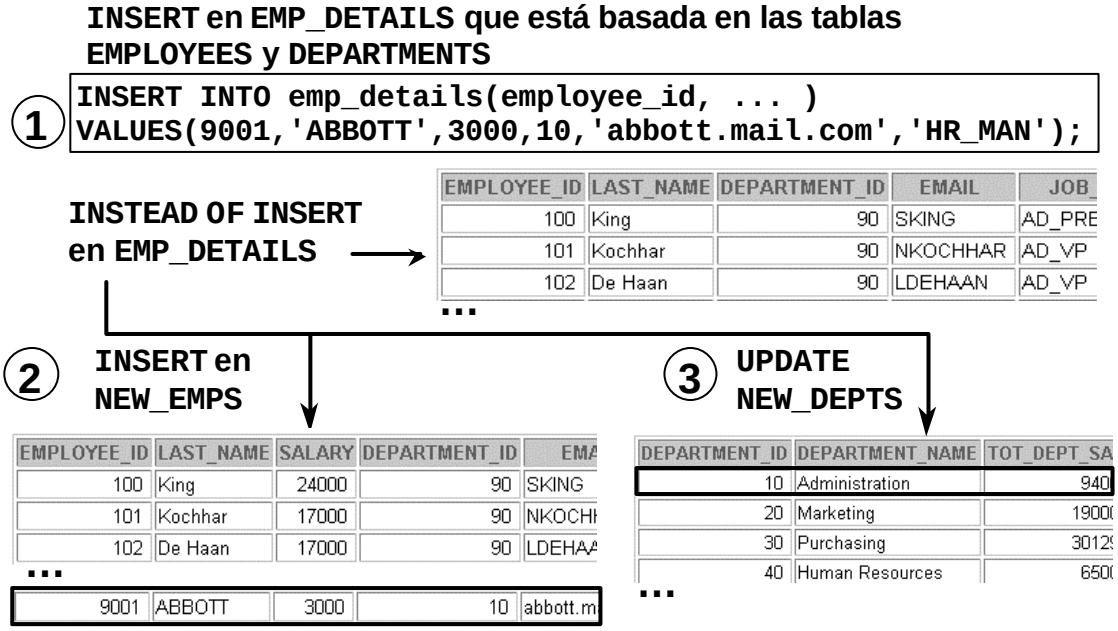
ORACLE

Creación de un Disparador INSTEAD OF

Los disparadores INSTEAD OF se crean para mantener las tablas base en las cuales se basa una vista.

Suponga que se va a insertar el nombre de un empleado utilizando la vista EMP_DETAILS que se crea basándose en las tablas EMPLOYEES y DEPARTMENTS. Se crea un disparador cuyo resultado sea el INSERT y UPDATE adecuados en las tablas base. La transparencia de la página siguiente explica cómo se comporta el disparador INSTEAD OF TRIGGER en esta situación.

Creación de un Disparador INSTEAD OF



Creación de un Disparador INSTEAD OF

A causa del disparador INSTEAD OF TRIGGER de la vista EMP_DETAILS, en lugar de insertar el nuevo registro de empleado en la tabla EMPLOYEES:

- Se inserta una fila en la tabla NEW_EMPS.
- Se actualiza la columna TOTAL_DEPT_SAL de la tabla NEW_DEPTS. El valor del sueldo que se ha proporcionado para el nuevo empleado se agrega al sueldo total existente del departamento al cual se ha asignado el nuevo empleado.

Diferenciación entre Disparadores de Base de Datos y Procedimientos Almacenados

Disparadores	Procedimientos
Definidos con `CREATE TRIGGER`	Definidos con `CREATE PROCEDURE`
El diccionario de datos contiene el código de origen de `USER_TRIGGERS`	El diccionario de datos contiene el código de origen de `USER_SOURCE`
Se llaman implícitamente	Se llaman explícitamente
`COMMIT`, `SAVEPOINT` y `ROLLBACK` no están permitidos	`COMMIT`, `SAVEPOINT` y `ROLLBACK` están permitidos

ORACLE

Disparadores de Base de Datos y Procedimientos Almacenados

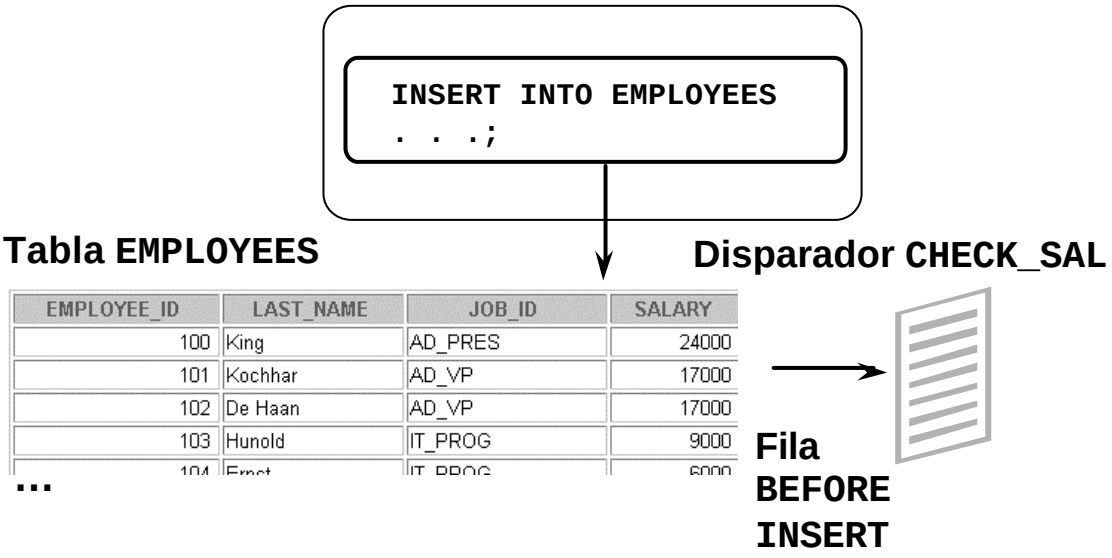
Existen una serie de diferencias entre los disparadores de base de datos y los procedimientos almacenados:

Disparador de Base de Datos	Procedimiento Almacenado
Se llama implícitamente	Se llama explícitamente
No se permiten sentencias `COMMIT`, `ROLLBACK` y `SAVEPOINT` en el interior del cuerpo del disparador. Se pueden realizar validaciones y rollback indirectamente llamando al procedimiento, pero no se recomienda hacerlo por los efectos secundarios que tiene en las transacciones	Se permite utilizar sentencias `COMMIT`, `ROLLBACK` y `SAVEPOINT` en el interior del cuerpo del procedimiento.

Los disparadores se compilan por completo cuando se emite el comando `CREATE TRIGGER` y el código `P` se almacena en el diccionario de datos.

El disparador se crea aunque se produzca un error durante su compilación.

Diferenciación entre Disparadores de Base de Datos y Disparadores de Form Builder



ORACLE

Diferencias entre un Disparador de Base de Datos y un Disparador de Form Builder

Los disparadores de base de datos son diferentes a los disparadores de Form Builder.

Disparador de Base de Datos	Disparador de Form Builder
Se ejecutan por las acciones de cualquier herramienta o aplicación de la base de datos	Se ejecutan sólo en una aplicación concreta de Form Builder
Siempre se disparan por una sentencia DML o DDL de SQL, o por una acción determinada de la base de datos	Pueden dispararse por navegar entre los campos, por pulsar una tecla o por muchas otras acciones
Se distingue entre disparadores de sentencias y de filas	Se distingue entre disparadores de sentencias y de filas
Cuando se produce un fallo, hace que la sentencia disparadora realice un rollback	Cuando se produce un fallo, el cursor se congela y puede hacer que la transacción entera realice un rollback
Arranca con independencia de los disparadores de Form Builder, y además de ellos	Arranca con independencia de los disparadores de base de datos, y además de ellos
Se ejecuta en el dominio de seguridad del autor del disparador	Se ejecuta en el dominio de seguridad del usuario de Form Builder

Gestión de Disparadores

Para desactivar o volver a activar un disparador de base de datos:

```
ALTER TRIGGER nombre_disparador  DISABLE | ENABLE
```

Para desactivar o volver a activar todos los disparadores de una tabla:

```
ALTER TABLE nombre_tabla   DISABLE | ENABLE  ALL TRIGGERS
```

Para recompilar un disparador de una tabla:

```
ALTER TRIGGER nombre_disparador COMPILE
```

ORACLE

Modos de los Disparadores: Activado o Desactivado

- El disparador se activa automáticamente en el momento de ser creado.
- Oracle Server comprueba las restricciones de integridad para los disparadores activados y garantiza que dichos disparadores no puedan comprometerlas. Asimismo, Oracle Server proporciona vistas de lectura consistente para las consultas y las restricciones, gestiona las dependencias y proporciona un proceso de validación de dos fases si un disparador actualiza tablas remotas en una base de datos distribuida.
- Para desactivar un disparador específico, utilice la sintaxis `ALTER TRIGGER`, y para desactivar *todos* los disparadores de una tabla, utilice la sintaxis `ALTER TABLE`.
- Desactive un disparador para mejorar el rendimiento o para evitar comprobaciones de la integridad de los datos cuando cargue cantidades masivas de datos mediante utilidades como, por ejemplo, SQL*Loader. Puede que también sea conveniente desactivar el disparador cuando hace referencia a un objeto de base de datos que en esos momentos no está disponible, debido a un fallo en la conexión de red, a un fallo de disco, a un archivo de datos fuera de línea o a un tablespace fuera de línea.

Compilación de un Disparador

- Utilice el comando `ALTER TRIGGER` para recompilar explícitamente un disparador que no es válido.
- Cuando se emite la sentencia `ALTER TRIGGER` con la opción `COMPILE`, el disparador se recompila, con independencia de si es válido o no.

Sintaxis de DROP TRIGGER

Para eliminar un disparador de la base de datos, utilice la sintaxis de DROP TRIGGER:

```
DROP TRIGGER nombre_disparador;
```

Ejemplo:

```
DROP TRIGGER secure_emp;
```

```
Trigger dropped.
```

Nota: Cuando se borra una tabla, también se borran todos los disparadores de esa tabla.

ORACLE

Eliminación de Disparadores

Cuando un disparador ya no es necesario, puede utilizar una sentencia SQL en *i*SQL*Plus para borrarlo.

Casos de Prueba de Disparadores

- **Pruebe todas las operaciones de datos disparadoras y no disparadoras.**
- **Pruebe todos los casos de la cláusula `WHEN`.**
- **Haga que el disparador arranque directamente desde una operación de datos básica, así como indirectamente desde un procedimiento.**
- **Compruebe el efecto del disparador en otros disparadores.**
- **Compruebe el efecto de otros disparadores en el disparador.**

ORACLE

Pruebas de Disparadores

- Asegúrese de que el disparador funciona correctamente probando por separado una serie de casos.
- Utilice los procedimientos `DBMS_OUTPUT` para depurar los disparadores. También puede utilizar Procedure Builder como herramienta de depuración de los disparadores. El uso de Procedure Builder se explica en el Apéndice F, "Creación de Unidades de Programa con Procedure Builder."

Modelo de Ejecución de Disparadores y Comprobaciones de Restricciones

1. Ejecute todos los disparadores `BEFORE STATEMENT`.
2. Realice un bucle por cada fila afectada:
   a. Ejecute todos los disparadores `BEFORE ROW`.
   b. Ejecute todos los disparadores `AFTER ROW`.
3. Ejecute la sentencia DML y realice comprobaciones de las restricciones de integridad.
4. Ejecute todos los disparadores `AFTER STATEMENT`.

ORACLE

Modelo de Ejecución de Disparadores

Una sola sentencia DML puede arrancar cuatro tipos de disparadores: las sentencias `BEFORE` y `AFTER` y los disparadores de fila. Un evento o una sentencia disparadora dentro del disparador puede hacer que se comprueben una o varias restricciones de integridad. Los disparadores también pueden arrancar otros disparadores (disparadores en cascada).

Todas las acciones y las comprobaciones derivadas de una sentencia SQL se deben realizar correctamente. Si se provoca una excepción en un disparador y la excepción no se maneja explícitamente, se realiza un rollback en todas las acciones que se han realizado debido a la sentencia SQL original. Esto incluye las acciones que se realizan al arrancar los disparadores. De esta manera, se garantiza que los disparadores nunca podrán comprometer las restricciones de integridad.

Cuando un disparadorarranca, la tablas a las que se hace referencia en la acción del disparador pueden sufrir cambios a causa de las transacciones de otros usuarios. En todos los casos, se garantiza una imagen de lectura consistente para los valores modificados que necesita leer (consultar) o escribir (actualizar) el disparador.

Modelo de Ejecución de Disparadores y Comprobaciones de Restricciones: Ejemplo

```
UPDATE employees SET department_id = 999
WHERE employee_id = 170;
-- Error de violación de restricción de integridad
```

```
CREATE OR REPLACE TRIGGER constr_emp_trig
AFTER UPDATE ON employees
  FOR EACH ROW
BEGIN
  INSERT INTO departments
    VALUES (999, 'dept999', 140, 2400);
END;
/
```

```
UPDATE employees SET department_id = 999
WHERE employee_id = 170;
-- Se realiza correctamente después de arrancar el disparador
```

ORACLE

Modelo de Ejecución de Disparadores y Comprobaciones de Restricciones: Ejemplo

El ejemplo de la transparencia explica una situación en la que es posible controlar la restricción de integridad utilizando un disparador. La tabla EMPLOYEES tiene una restricción de clave ajena en la columna DEPARTMENT_ID de la tabla DEPARTMENTS.

En la primera sentencia SQL, el DEPARTMENT_ID del empleado con EMPLOYEE_ID 170 se cambia por 999.

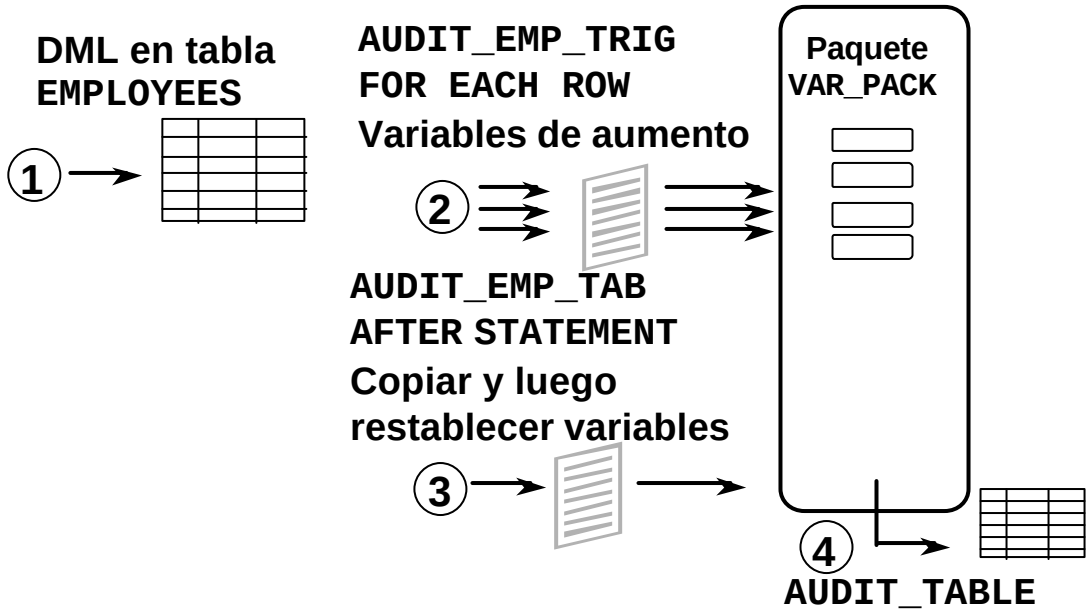
Dado que dicho departamento no existe en la tabla DEPARTMENTS, la sentencia provoca la excepción -2292 por la violación de la restricción de integridad.

Se crea el disparador CONSTR_EMP_TRIG que inserta el nuevo departamento 999 en la tabla DEPARTMENTS.

Cuando se emite la sentencia UPDATE que cambia el departamento del empleado 170 a 999, el disparador arranca. Luego, se comprueba la restricción de clave ajena. Dado que el disparador ha insertado el departamento 999 en la tabla DEPARTMENTS, la comprobación de la restricción de clave ajena se realiza correctamente y no se provoca ninguna excepción.

Este proceso funciona con Oracle 8*i* y las versiones posteriores. El ejemplo que se describe en la transparencia produce un error en tiempo de ejecución en las versiones anteriores a Oracle8*i*.

Demostración de Ejemplo de Disparadores que Utilizan Construcciones de Paquetes



ORACLE

Demostración de Ejemplo

Las siguientes páginas de subprogramas PL/SQL son un ejemplo de la interacción entre los disparadores, los procedimientos empaquetados, las funciones y las variables globales.

La secuencia de eventos:

1. Se emite un comando `INSERT`, `UPDATE` o `DELETE` que puede manipular una o varias filas.
2. `AUDIT_EMP_TRIG`, el disparador `AFTER ROW`, llama al procedimiento empaquetado para aumentar las variables globales del paquete `VAR_PACK`. Puesto que es un disparador de filas, éste arranca una vez por cada fila que se haya actualizado.
3. Cuando la sentencia termina, `AUDIT_EMP_TAB`, el disparador `AFTER STATEMENT`, llama al procedimiento `AUDIT_EMP`.
4. Este procedimiento asigna los valores de las variables globales a variables locales utilizando las funciones empaquetadas, actualiza `AUDIT_TABLE` y luego restablece las variables globales.

Disparadores de Sentencias AFTER y Disparadores de Filas AFTER

```
CREATE OR REPLACE TRIGGER audit_emp_trig
AFTER    UPDATE or INSERT or DELETE on EMPLOYEES
FOR EACH ROW
BEGIN
  IF      DELETING    THEN  var_pack.set_g_del(1);
  ELSIF   INSERTING   THEN  var_pack.set_g_ins(1);
  ELSIF   UPDATING ('SALARY')
                      THEN  var_pack.set_g_up_sal(1);
  ELSE    var_pack.set_g_upd(1);
  END IF;
END audit_emp_trig;
/
```

```
CREATE OR REPLACE TRIGGER audit_emp_tab
AFTER    UPDATE or INSERT or DELETE on employees
BEGIN
  audit_emp;
END audit_emp_tab;
/
```

ORACLE

Disparadores de Sentencias AFTER y Disparadores de Filas AFTER

El disparador AUDIT_EMP_TRIG es un disparador de filas que arranca después de manipular cada fila. Este disparador llama a los procedimientos de paquetes, dependiendo del tipo de operación DML que se realice. Por ejemplo, si la operación DML actualiza el sueldo de un empleado, el disparador llama al procedimiento SET_G_UP_SAL. Este procedimiento de paquete, a su vez, llama a la función G_UP_SAL. Esta función aumenta la variable de paquete GV_UP_SAL que controla el número de filas que se cambian debido a la actualización del sueldo.

Una vez terminada la sentencia, arrancará el disparador AUDIT_EMP_TAB. Este disparador llama al procedimiento AUDIT_EMP, que se encuentra en las siguientes páginas. El procedimiento AUDIT_EMP actualiza la tabla AUDIT_TABLE. Se realiza una entrada en la tabla AUDIT_TABLE con información como, por ejemplo, el usuario que ha realizado la operación DML, la tabla en la que se ha realizado y el número total de manipulaciones de datos realizadas hasta ese momento en la tabla (que indica el valor de la columna correspondiente de la tabla AUDIT_TABLE). Al final, el procedimiento AUDIT_EMP restablece las variables de paquete en 0.

Demostración: Especificación del Paquete VAR_PACK

var_pack.sql

```
CREATE OR REPLACE PACKAGE var_pack
IS
-- estas funciones se utilizan para devolver los
-- valores de las variables de paquetes
  FUNCTION g_del RETURN NUMBER;
  FUNCTION g_ins RETURN NUMBER;
  FUNCTION g_upd RETURN NUMBER;
  FUNCTION g_up_sal RETURN NUMBER;
-- estos procedimientos se utilizan para modificar los
-- valores de las variables de paquetes
  PROCEDURE set_g_del    (p_val  IN  NUMBER);
  PROCEDURE set_g_ins    (p_val  IN  NUMBER);
  PROCEDURE set_g_upd    (p_val  IN  NUMBER);
  PROCEDURE set_g_up_sal (p_val  IN  NUMBER);
END var_pack;
/
```

ORACLE

Demostración: Cuerpo del Paquete VAR_PACK

var_pack_body.sql

```
CREATE OR REPLACE PACKAGE BODY var_pack IS
  gv_del     NUMBER  := 0;  gv_ins     NUMBER  := 0;
  gv_upd     NUMBER  := 0;  gv_up_sal  NUMBER  := 0;
FUNCTION  g_del RETURN NUMBER IS
  BEGIN
    RETURN gv_del;
  END;
FUNCTION  g_ins RETURN NUMBER IS
  BEGIN
    RETURN gv_ins;
  END;
FUNCTION  g_upd RETURN NUMBER IS
  BEGIN
    RETURN gv_upd;
  END;
FUNCTION g_up_sal RETURN NUMBER IS
  BEGIN
    RETURN gv_up_sal;
  END;
```

(continúa en la siguiente página)

Cuerpo del Paquete VAR_PACK (continuación)

```
   PROCEDURE set_g_del (p_val  IN NUMBER) IS
   BEGIN
        IF p_val = 0   THEN
           gv_del := p_val;
        ELSE gv_del := gv_del +1;
        END IF;
   END set_g_del;
   PROCEDURE set_g_ins (p_val  IN NUMBER) IS
   BEGIN
        IF p_val = 0   THEN
           gv_ins := p_val;
        ELSE gv_ins := gv_ins +1;
        END IF;
   END set_g_ins;
   PROCEDURE set_g_upd (p_val  IN NUMBER) IS
   BEGIN
        IF p_val = 0   THEN
           gv_upd := p_val;
        ELSE gv_upd := gv_upd +1;
        END IF;
   END set_g_upd;
   PROCEDURE set_g_up_sal (p_val  IN NUMBER) IS
   BEGIN
        IF p_val = 0   THEN
          gv_up_sal := p_val;
        ELSE gv_up_sal := gv_up_sal +1;
        END IF;
   END set_g_up_sal;
END var_pack;
/
```

Demostración: Uso del Procedimiento AUDIT_EMP

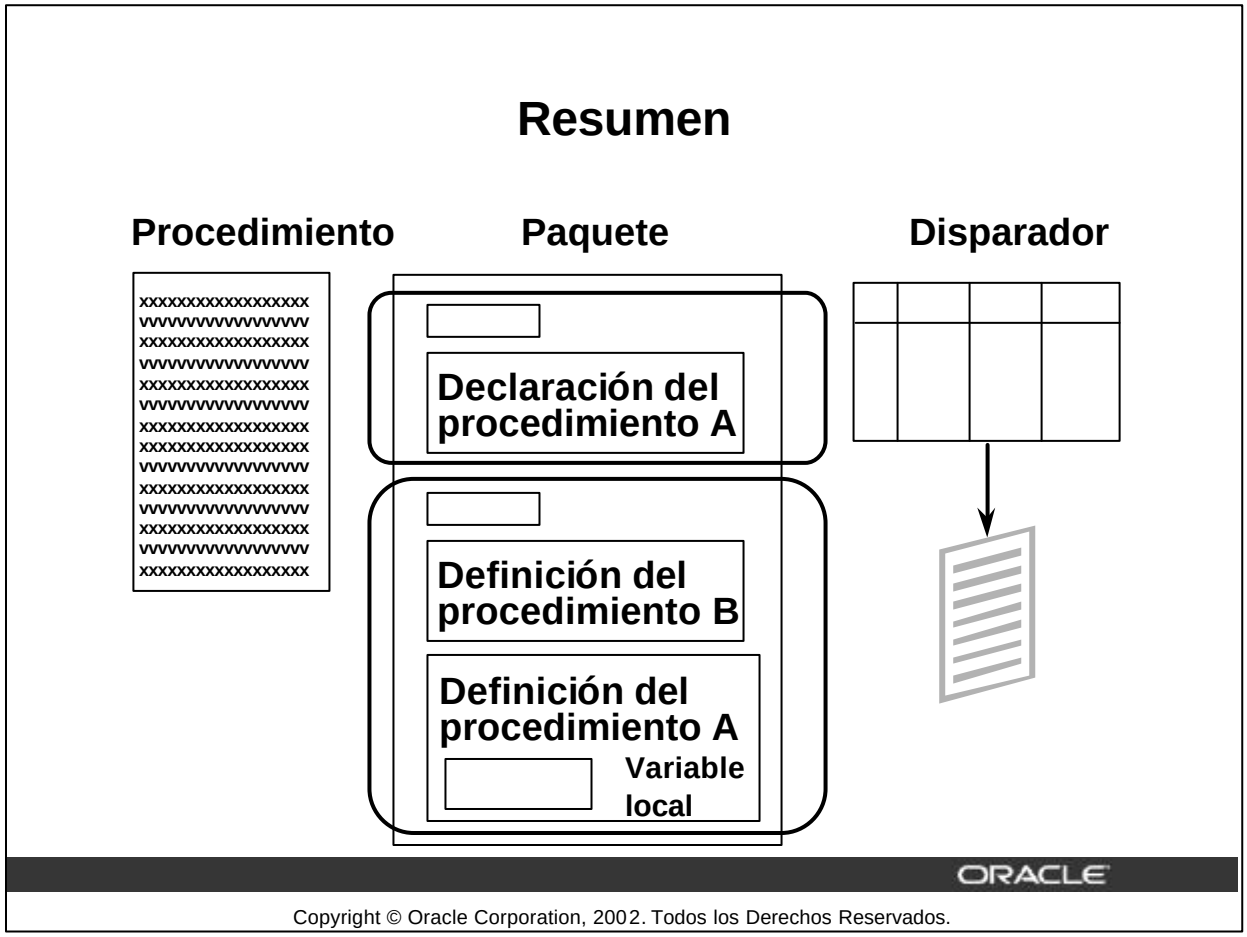
```
CREATE OR REPLACE PROCEDURE audit_emp  IS
  v_del     NUMBER   := var_pack.g_del;
  v_ins     NUMBER   := var_pack.g_ins;

  v_upd     NUMBER   := var_pack.g_upd;
  v_up_sal  NUMBER   := var_pack.g_up_sal;
BEGIN
  IF  v_del + v_ins + v_upd != 0  THEN
    UPDATE audit_table SET
      del = del + v_del, ins = ins + v_ins,
      upd = upd + v_upd
    WHERE user_name=USER AND tablename='EMPLOYEES'
    AND   column_name IS NULL;
  END IF;
  IF v_up_sal != 0   THEN
    UPDATE audit_table SET upd = upd + v_up_sal
    WHERE user_name=USER AND tablename='EMPLOYEES'
    AND   column_name = 'SALARY';
  END IF;
-- restablecimiento de las variables globales en el paquete VAR_PACK
  var_pack.set_g_del (0); var_pack.set_g_ins (0);
  var_pack.set_g_upd (0); var_pack.set_g_up_sal (0);
END audit_emp;
```

ORACLE

Actualización de AUDIT_TABLE con el procedimiento AUDIT_EMP

El procedimiento AUDIT_EMP actualiza AUDIT_TABLE y llama a las funciones del paquete VAR_PACK que restablecen las variables de paquetes, y está listo para la siguiente sentencia DML.



Hay que desarrollar diferentes tipos de construcciones de base de datos procedurales en función de su uso.

Construcción	**Uso**
Procedimiento	Bloque de programación PL/SQL que se almacena en la base de datos para repetir su ejecución
Paquete	Grupo de procedimientos, funciones, variables, cursores, constantes y excepciones relacionadas
Disparador	Bloque de programación PL/SQL que se ejecuta implícitamente por medio de una sentencia de manipulación de datos

Visión General de la Práctica 16

Esta práctica cubre los siguientes temas:

- **Creación de disparadores de sentencias y de filas**
- **Creación de disparadores avanzados para aumentar las capacidades de la base de datos de Oracle**

ORACLE

Visión General de la Práctica 16

En esta práctica deberá crear disparadores de sentencias y de filas. También creará procedimientos que se llamarán desde los disparadores.

Práctica 16

1. Sólo está permitido realizar cambios en las tablas durante las horas de oficina normales, entre las 8:45 a.m y las 5:30 p.m., de lunes a viernes.

   Cree un procedimiento almacenado llamado SECURE_DML que evite que la sentencia DML se ejecute fuera de las horas normales de oficina y que devuelva el mensaje, "You may only make changes during normal office hours."

2. a. Cree un disparador de sentencias en la tabla JOBS que llame al procedimiento anterior.

   b. Pruebe el procedimiento modificando temporalmente las horas de dicho procedimiento e intentando insertar un nuevo registro en la tabla JOBS. (Ejemplo: reemplace 08:45 por 16:45. Este intento devuelve un mensaje de error).

Después de las pruebas, restablezca las horas del procedimiento que se especifican en la pregunta 1 y vuelva a crear el procedimiento de la misma manera que en la pregunta 1 anterior.

Si tiene tiempo:

3. Los empleados deberían recibir un aumento de sueldo automáticamente si se aumenta el sueldo mínimo de un puesto de trabajo. Implemente este requisito por medio de un disparador en la tabla JOBS.
   a. Cree un procedimiento almacenado llamado UPD_EMP_SAL para actualizar la cantidad del sueldo. Este procedimiento acepta dos parámetros: el identificador del puesto de trabajo cuyo sueldo se va a actualizar y el nuevo sueldo mínimo para este identificador de puesto de trabajo. Este procedimiento se ejecuta desde el disparador en la tabla JOBS.
   b. Cree un disparador de filas denominado UPDATE_EMP_SALARY en la tabla JOBS que llame al procedimiento UPD_EMP_SAL, cuando se actualice el sueldo mínimo de la tabla JOBS para un identificador de puesto de trabajo especificado.
   c. Consulte la tabla EMPLOYEES para ver el sueldo actual de los empleados que sean programadores.

LAST_NAME	FIRST_NAME	SALARY
Austin	David	5280
Hunold	Alexander	9000
Ernst	Bruce	6000
Pataballa	Valli	5280
Lorentz	Diana	4620

   d. Aumente el sueldo mínimo de los programadores de 4.000 a 5.000.
   e. El empleado Lorentz (identificador de empleado 107) tenía un sueldo inferior a 4.500. Verifique que su sueldo se haya aumentado hasta el nuevo mínimo de 5.000.

LAST_NAME	FIRST_NAME	SALARY
Lorentz	Diana	5000

17

Otros Conceptos sobre Disparadores

ORACLE

Objetivos

Al finalizar esta lección, debería estar capacitado para hacer lo siguiente:

- **Crear disparadores de base de datos adicionales**
- **Explicar las reglas que rigen los disparadores**
- **Implementar disparadores**

ORACLE

Finalidad de la Lección

En esta lección aprenderá a crear más disparadores de base de datos y conocerá las reglas que rigen los disparadores. También aprenderá muchas aplicaciones de los disparadores.

Creación de Disparadores de Base de Datos

- **Evento disparador de usuario:**
  - **CREATE, ALTER o DROP**
  - **Conexión o desconexión**
- **Evento disparador del sistema o de la base de datos:**
  - **Cierre o inicio de la base de datos**
  - **Emisión de un error específico (o cualquier error)**

ORACLE

Creación de Disparadores de Base de Datos

Antes de codificar el cuerpo del disparador, decida cuáles van a ser sus componentes.

Los disparadores basados en eventos del sistema se pueden definir al nivel de la base de datos o al nivel del esquema. Por ejemplo, los disparadores de cierre de la base de datos se definen al nivel de la base de datos. Los disparadores basados en sentencias DDL (Lenguaje de Definición de Datos), o en la conexión o desconexión de un usuario, también se pueden definir al nivel de la base de datos o al nivel del esquema. Los disparadores que están basados en sentencias DML se definen en una tabla específica o en una vista.

Los disparadores que están definidos al nivel de la base de datos se arrancan para todos los usuarios, mientras que los que están definidos al nivel del esquema o de la tabla sólo se arrancan cuando el evento disparador implica ese esquema o esa tabla.

Estos son los eventos disparadores que pueden arrancar un disparador:

- Una sentencia de definición de datos en un objeto de la base de datos o del esquema
- La conexión o desconexión de un usuario específico (o cualquier usuario)
- El cierre o inicio de la base de datos
- Un error específico o cualquier tipo de error

Creación de Disparadores en Sentencias DDL

Sintaxis:

```
CREATE [OR REPLACE] TRIGGER nombre_disparador
   temporización
      [ddl_evento1 [OR ddl_evento2 OR ...]]
         ON {DATABASE|SCHEMA}
cuerpo_disparador
```

ORACLE

Sintaxis para Crear Disparadores

Evento_DDL	Valores Posibles
CREATE	Hace que Oracle Server arranque el disparador siempre que una sentencia CREATE agrega un nuevo objeto de base de datos al diccionario
ALTER	Hace que Oracle Server arranque el disparador siempre que una sentencia ALTER modifica un objeto de base de datos en el diccionario de datos
DROP	Hace que Oracle Server arranque el disparador siempre que una sentencia DROP elimina un objeto de base de datos del diccionario de datos

El cuerpo del disparador representa un bloque PL/SQL completo.

Se pueden crear disparadores para estos eventos en DATABASE o SCHEMA. También hay que especificar BEFORE o AFTER para la temporización del disparador.

Los disparadores DDL sólo se arrancan si el objeto que se está creando es un agrupamiento, un índice, un paquete, un procedimiento, un rol, una secuencia, un sinónimo, una tabla, un tablespace, un disparador, un tipo, una vista o un usuario.

Creación de Disparadores Basados en Eventos del Sistema

```
CREATE [OR REPLACE] TRIGGER nombre_disparador
   temporización
      [evento1_basedatos [OR evento2_basedatos OR ...]]
         ON {DATABASE|SCHEMA}
cuerpo_disparador
```


Sintaxis para Crear Disparadores

Evento_basedatos	Valores Posibles
AFTER SERVERERROR	Hace que Oracle Server arranque el disparador siempre que se registra un mensaje de error del servidor
AFTER LOGON	Hace que Oracle Server arranque el disparador siempre que un usuario se conecta a la base de datos
BEFORE LOGOFF	Hace que Oracle Server arranque el disparador siempre que un usuario se desconecta de la base de datos
AFTER STARTUP	Hace que Oracle Server arranque el disparador siempre que se abre la base de datos
BEFORE SHUTDOWN	Hace que Oracle Server arranque el disparador siempre que se cierra la base de datos

Se pueden crear crear disparadores para estos eventos en DATABASE o SCHEMA, pero no en SHUTDOWN y STARTUP, que sólo se aplican a DATABASE.

Ejemplo de Disparadores LOGON y LOGOFF

```
CREATE OR REPLACE TRIGGER logon_trig
AFTER LOGON  ON  SCHEMA
BEGIN
INSERT INTO log_trig_table(user_id, log_date, action)
VALUES (USER, SYSDATE, 'Logging on');
END;
/
```

```
CREATE OR REPLACE TRIGGER logoff_trig
BEFORE LOGOFF  ON  SCHEMA
BEGIN
INSERT INTO log_trig_table(user_id, log_date, action)
VALUES (USER, SYSDATE, 'Logging off');
END;
/
```

ORACLE

Ejemplo de Disparadores LOGON y LOGOFF

Puede crear este disparador para controlar la frecuencia con la que se conecta y se desconecta, o bien puede que quiera escribir un informe que controle el tiempo que está conectado. Al especificar ON SCHEMA, el disparador se arranca para el usuario específico. Si especificara ON DATABASE, el disparador se arrancaría para todos los usuarios.

Sentencias CALL

```
CREATE [OR REPLACE] TRIGGER nombre_disparador
  temporización
    evento1 [OR evento2 OR evento3]
      ON nombre_tabla
  [REFERENCING OLD AS antiguo | NEW AS nuevo]
[FOR EACH ROW]
  [WHEN condición]
CALL nombre_procedimiento
```

```
CREATE OR REPLACE TRIGGER log_employee
BEFORE INSERT ON EMPLOYEES
  CALL log_execution
/
```

ORACLE

Sentencias CALL

Las sentencias CALL permiten llamar a un procedimiento almacenado, en lugar de codificar el cuerpo PL/SQL en el propio disparador. El procedimiento se puede implementar en PL/SQL, en C o en Java.

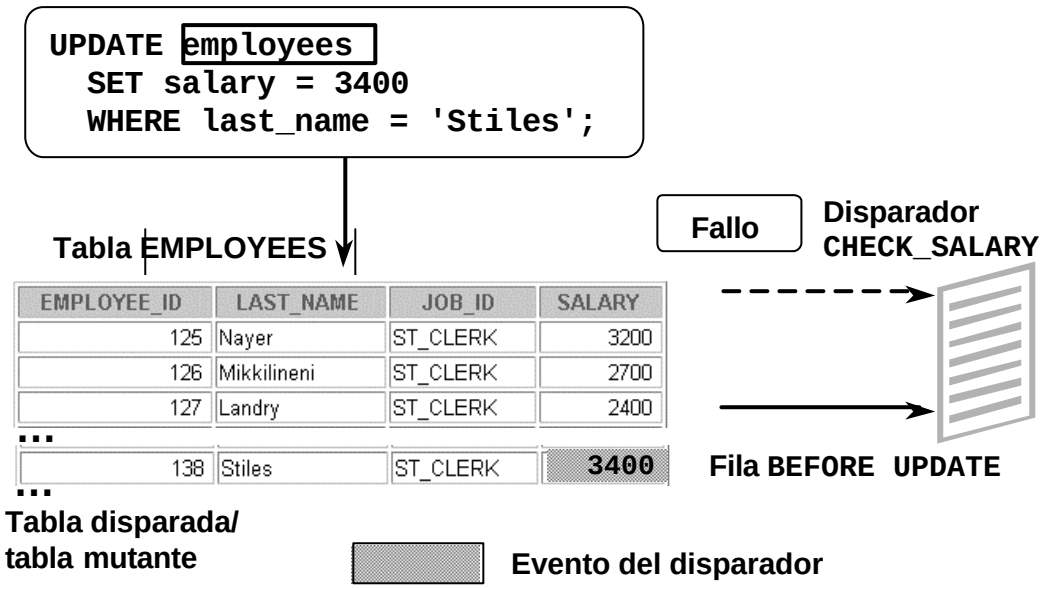
La llamada puede hacer referencia a los atributos :NEW y :OLD del disparador como parámetros, como muestra el siguiente ejemplo:

```
CREATE TRIGGER salary_check
  BEFORE UPDATE OF salary, job_id ON employees
  FOR EACH ROW
  WHEN (NEW.job_id <> 'AD_PRES')
    CALL check_sal(:NEW.job_id, :NEW.salary)
/
```

Nota: Al final de la sentencia CALL no se utiliza el punto y coma (;).

En el ejemplo anterior, el disparador llama al procedimiento check_sal. Este procedimiento compara el nuevo sueldo con el rango de sueldos del nuevo identificador de puesto de trabajo desde la tabla JOBS.

Lectura de Datos desde una Tabla Mutante



ORACLE

Reglas Que Rigen los Disparadores

La lectura y la escritura de datos con disparadores está sujeta a ciertas reglas. Las restricciones sólo se aplican a los disparadores de filas, a menos que se arranque un disparador de sentencias como resultado de `ON DELETE CASCADE`.

Tabla Mutante

Una tabla mutante es una tabla que se está modificando actualmente por medio de una sentencia `UPDATE`, `DELETE` o `INSERT`, o bien una tabla que puede que se deba actualizar por el efecto de una acción de integridad referencial de una sentencia `DELETE CASCADE` declarativa. No se considera que una tabla sea mutante para los disparadores `STATEMENT`.

La tabla disparada es una tabla mutante, así como cualquier tabla que haga referencia a ella con la restricción `FOREIGN KEY`. Esta restricción evita que un disparador de filas vea un conjunto de datos inconsistente.

Tabla Mutante: Ejemplo

```
CREATE OR REPLACE TRIGGER check_salary
  BEFORE INSERT OR UPDATE OF salary, job_id
  ON employees
  FOR EACH ROW
  WHEN (NEW.job_id <> 'AD_PRES')
DECLARE
  v_minsalary employees.salary%TYPE;
  v_maxsalary employees.salary%TYPE;
BEGIN
  SELECT MIN(salary), MAX(salary)
   INTO  v_minsalary, v_maxsalary
   FROM  employees
   WHERE job_id = :NEW.job_id;
  IF :NEW.salary < v_minsalary OR
     :NEW.salary > v_maxsalary THEN
     RAISE_APPLICATION_ERROR(-20505,'Out of range');
  END IF;
END;
/
```

ORACLE

Tabla Mutante: Ejemplo

El disparador CHECK_SALARY del ejemplo intenta garantizar que siempre que se agregue un nuevo empleado a la tabla EMPLOYEES o que se cambie el sueldo o el identificador de puesto de trabajo de un empleado existente, el sueldo de ese empleado esté en el rango de sueldos establecido para su puesto de trabajo.

Cuando se actualiza el registro de un empleado, el disparador CHECK_SALARY se arranca por cada fila que se actualiza. El código del disparador consulta la misma tabla que se está actualizando. Por eso, se dice que la tabla EMPLOYEES es una tabla mutante.

Tabla Mutante: Ejemplo

```
UPDATE employees
SET salary = 3400
WHERE last_name = 'Stiles';
```

```
UPDATE employees
       *
ERROR at line 1:
ORA-04091: table PLSQL.EMPLOYEES is mutating, trigger/function may not see it
ORA-06512: at "PLSQL.CHECK_SALARY", line 5
ORA-04088: error during execution of trigger 'PLSQL.CHECK_SALARY'
```

ORACLE

Tabla Mutante: Ejemplo (continuación)

Se intenta realizar una lectura desde una tabla mutante.

Si el sueldo de un rango se restringe entre el valor mínimo y el valor máximo existente, se producirá un error en tiempo de ejecución. La tabla `EMPLOYEES` está mutando o en estado de cambio. Por lo tanto, el disparador no puede realizar lecturas en ella.

Recuerde que las funciones también pueden producir un error de tabla mutante si se llaman en una sentencia DML.

Implementación de Disparadores

Los disparadores sirven para:

- **Seguridad**
- **Auditoría**
- **Integridad de los datos**
- **Integridad referencial**
- **Replicación de tablas**
- **Cálculo de datos derivados de manera automática**
- **Registro de eventos**

ORACLE

Implementación de Disparadores

Desarrolle disparadores de base de datos con el fin de mejorar funciones que Oracle Server no puede implementar, o como alternativa a las funciones que proporciona Oracle Server.

Función	Mejora
Seguridad	Oracle Server permite el acceso a las tablas a usuarios o roles. Los disparadores permiten el acceso a las tablas de acuerdo con valores de datos.
Auditorías	Oracle Server realiza un seguimiento de las operaciones de datos en las tablas. Los disparadores realizan un seguimiento de los valores para las operaciones de datos en las tablas.
Integridad de los datos	Oracle Server fuerza restricciones de integridad. Los disparadores implementan reglas de integridad muy complejas.
Integridad referencial	Oracle Server fuerza reglas de integridad referencial estándar. Los disparadores implementan funcionalidades no estándar.
Replicación de tablas	Oracle Server copia tablas de manera asíncrona en instantáneas. Los disparadores copian tablas de manera sincrónica en réplicas.
Datos derivados	Oracle Server calcula manualmente los valores de los datos derivados. Los disparadores calculan los valores de los datos derivados automáticamente.
Registro de eventos	Oracle Server registra los eventos explícitamente. Los disparadores registran los eventos de manera transparente.

Control de la Seguridad en el Servidor

```
GRANT SELECT, INSERT, UPDATE, DELETE
ON    employees
TO    clerk;              -- rol de base de datos
GRANT clerk TO scott;
```

ORACLE

Control de la Seguridad en el Servidor

Se desarrollan esquemas y roles en Oracle Server para controlar la seguridad de las operaciones de datos en las tablas de acuerdo con la identidad del usuario.

- Los privilegios se basan en el nombre de usuario proporcionado cuando el usuario se conecta a la base de datos.
- Se determina el acceso a las tablas, las vistas, los sinónimos y las secuencias.
- Se determina también los privilegios de consultas, manipulaciones de datos y definiciones de datos.

Control de la Seguridad con un Disparador de Base de Datos

```
CREATE OR REPLACE TRIGGER secure_emp
  BEFORE INSERT OR UPDATE OR DELETE ON employees
DECLARE
  v_dummy VARCHAR2(1);
BEGIN
IF (TO_CHAR (SYSDATE, 'DY') IN ('SAT','SUN'))
  THEN RAISE_APPLICATION_ERROR (-20506,'You may only
        change data during normal business hours.');
END IF;
SELECT COUNT(*) INTO v_dummy FROM holiday
WHERE holiday_date = TRUNC (SYSDATE);
IF v_dummy > 0 THEN RAISE_APPLICATION_ERROR(-20507,
           'You may not change data on a holiday.');
END IF;
END;
/
```

ORACLE

Control de la Seguridad con un Disparador de Base de Datos

Se desarrollan disparadores para gestionar requisitos de seguridad más complejos.

- Los privilegios se basan en cualquier valor de la base de datos como, por ejemplo, la hora del día, el día de la semana, etc.
- Sólo se determina el acceso a las tablas.
- Sólo se determinan privilegios de manipulación de datos.

Uso de la Utilidad del Servidor para Auditar Operaciones de Datos

```
AUDIT INSERT, UPDATE, DELETE
  ON  departments
  BY ACCESS
WHENEVER SUCCESSFUL;
```

Audit succeeded.

Oracle Server almacena la información de auditoría en un tabla de diccionario de datos o en un archivo del sistema operativo.

ORACLE

Auditoría de Operaciones de Datos

En Oracle Server, se puede auditar las operaciones de datos. Las auditorías de base de datos sirven para controlar y recopilar datos acerca de actividades específicas de la base de datos. El DBA puede recoger estadísticas acerca de qué tablas se están actualizando, cuántas operaciones de E/S se realizan, cuántos usuarios simultáneos se conectan en las horas de más actividad, etc.

- Se auditan usuarios, sentencias u objetos.
- También se auditan las sentencias de recuperación de datos, de manipulación de datos y de definición de datos.
- La pista de auditoría se escribe en una tabla de auditoría centralizada.
- Se generan registros de auditoría una vez por sesión o una vez por cada intento de acceso.
- Se capturan los intentos correctos, incorrectos o ambos.
- Se realizan activaciones y desactivaciones dinámicamente.

La ejecución de SQL a través de unidades de programa PL/SQL puede generar varios registros de auditoría, porque es posible que las unidades de programa hagan referencia a otros objetos de base de datos.

Auditorías con un Disparador

```
CREATE OR REPLACE TRIGGER audit_emp_values
AFTER DELETE OR INSERT OR UPDATE ON employees
FOR EACH ROW
BEGIN
IF (audit_emp_package.g_reason IS NULL) THEN
    RAISE_APPLICATION_ERROR (-20059, 'Specify a reason
     for the data operation through the procedure SET_REASON
     of the AUDIT_EMP_PACKAGE before proceeding.');
ELSE
    INSERT INTO audit_emp_table (user_name, timestamp, id,
      old_last_name, new_last_name, old_title, new_title,
      old_salary, new_salary, comments)
    VALUES (USER, SYSDATE, :OLD.employee_id, :OLD.last_name,
      :NEW.last_name, :OLD.job_id, :NEW.job_id, :OLD.salary,
      :NEW.salary, audit_emp_package.g_reason);
END IF;
END;
```

```
CREATE OR REPLACE TRIGGER cleanup_audit_emp
AFTER INSERT OR UPDATE OR DELETE ON employees
BEGIN
  audit_emp_package.g_reason := NULL;
END;
```

ORACLE

Auditoría de Valores de Datos

Utilice disparadores para auditar los valores de datos reales.

Puede:

- Auditar sólo sentencias de manipulación de datos
- Escribir la pista de auditoría en una tabla de auditoría definida por el usuario
- Generar registros de auditoría una vez para la sentencia o una vez para cada fila
- Capturar sólo los intentos correctos
- Realizar activaciones y desactivaciones dinámicamente

Con Oracle Server, puede realizar auditorías de base de datos. Las auditorías de base de datos no pueden registrar los cambios realizados en valores de columna específicos. Si hay que realizar un seguimiento de los cambios realizados en las columnas de tabla y almacenar los valores de columna por cada cambio, utilice auditorías de aplicaciones. Las auditorías de aplicaciones se pueden realizar a través de procedimientos almacenados o de disparadores de base de datos, tal y como se muestra en el ejemplo de la transparencia.

Forzado de la Integridad de Datos en el Servidor

```
ALTER TABLE employees ADD
  CONSTRAINT ck_salary CHECK (salary >= 500);
```

Table altered.

ORACLE

Forzado de la Integridad de Datos en el Servidor

Puede forzar la integridad de datos en Oracle Server y desarrollar disparadores para gestionar reglas de integridad de datos más complejas.

Las reglas de integridad de datos estándar son no nulo, único, clave primaria y clave ajena.

Utilícelas para:

- Proporcionar valores por defecto a las constantes
- Forzar restricciones estáticas
- Realizar activaciones y desactivaciones dinámicamente

Ejemplo

El ejemplo de código de la transparencia garantiza que el sueldo es al menos $500.

Protección de la Integridad de Datos con un Disparador

```
CREATE OR REPLACE TRIGGER check_salary
  BEFORE UPDATE OF salary ON employees
  FOR EACH ROW
  WHEN (NEW.salary < OLD.salary)
BEGIN
  RAISE_APPLICATION_ERROR (-20508,
      'Do not decrease salary.');
END;
/
```

ORACLE

Protección de la Integridad de Datos con un Disparador

Proteja la integridad de los datos con un disparador y fuerce comprobaciones de integridad de datos no estándar.

- Se proporcionan valores por defecto a las variables.
- Se fuerzan restricciones dinámicas.
- Se realizan activaciones y desactivaciones dinámicamente.
- Se incorporan restricciones declarativas en la definición de una tabla para proteger la integridad de los datos.

Ejemplo

El ejemplo de código de la transparencia garantiza que el sueldo nunca se reduce.

Forzado de la Integridad Referencial en el Servidor

```
ALTER TABLE employees
  ADD CONSTRAINT emp_deptno_fk
  FOREIGN KEY (department_id)
     REFERENCES departments(department_id)
ON DELETE CASCADE;
```

ORACLE

Forzado de la Integridad Referencial en el Servidor

Incorpore restricciones de integridad referencial en la definición de una tabla para evitar datos inconsistentes y para forzar la integridad referencial en el servidor.

- Se restringen las actualizaciones y las supresiones.
- Se realizan supresiones en cascada.
- Se realizan activaciones y desactivaciones dinámicamente.

Ejemplo

Cuando se elimina un departamento de la tabla principal `DEPARTMENTS`, se realiza una supresión en cascada de las filas correspondientes en la tabla secundaria `EMPLOYEES`.

Protección de la Integridad Referencial con un Disparador

```
CREATE OR REPLACE TRIGGER cascade_updates
AFTER UPDATE OF department_id ON departments
FOR EACH ROW
BEGIN
  UPDATE employees
   SET employees.department_id=:NEW.department_id
   WHERE employees.department_id=:OLD.department_id;
  UPDATE job_history
   SET department_id=:NEW.department_id
   WHERE department_id=:OLD.department_id;
END;
/
```

ORACLE

Protección de la Integridad Referencial con un Disparador

Desarrolle disparadores para implementar reglas de integridad referencial que no estén permitidas en las restricciones declarativas.

- Se realizan actualizaciones en cascada.
- Se definen valores `NULL` para las actualizaciones y las supresiones.
- Se definen valores por defecto en las actualizaciones y las supresiones.
- Se fuerza la integridad referencial en los sistemas distribuidos.
- Se realizan activaciones y desactivaciones dinámicamente.

Ejemplo

Se fuerza la integridad referencial con un disparador. Cuando el valor de `DEPARTMENT_ID` cambia en la tabla primaria `DEPARTMENTS`, se realiza una actualización en cascada de las filas correspondientes en la tabla secundaria `EMPLOYEES`.

Para obtener una solución completa para la integridad referencial, un solo disparador no es suficiente.

Replicación de una Tabla en el Servidor

```
CREATE SNAPSHOT emp_copy AS
  SELECT * FROM employees@ny;
```

ORACLE

Creación de una Instantánea

Una instantánea es una copia local de los datos de una tabla cuyo origen se encuentra en una o varias tablas maestras remotas. Una aplicación puede consultar los datos de una instantánea de tabla de sólo lectura, pero no puede realizar inserciones, actualizaciones ni supresiones de filas en dicha instantánea. Para mantener los datos de la instantánea actualizados con respecto a los datos maestros, Oracle Server debe refrescar periódicamente la instantánea.

Cuando se utiliza esta sentencia en SQL, Oracle Server realiza la replicación implícitamente con disparadores internos. Con este método de replicación se obtiene un mejor rendimiento que con los disparadores PL/SQL definidos por el usuario.

Copia de Tablas con Instantáneas de Servidor

Copie una tabla con una instantánea.

- Las tablas se copian de forma asíncrona, a intervalos definidos por el usuario.
- Las instantáneas se basan en varias tablas maestras.
- Sólo se pueden realizar lecturas desde las instantáneas.
- Se mejora el rendimiento de las manipulaciones de datos en la tabla maestra, especialmente si la red falla.

También puede replicar las tablas utilizando disparadores.

Ejemplo

En San Francisco, se crea una instantánea de la tabla remota `EMPLOYEES` de Nueva York.

Replicación de una Tabla con un Disparador

```
CREATE OR REPLACE TRIGGER emp_replica
BEFORE INSERT OR UPDATE ON employees
FOR EACH ROW
BEGIN /*Sólo continúa si el usuario inicia una operación de datos,
        NO a través del disparador en cascada.*/
   IF INSERTING THEN
     IF :NEW.flag IS NULL THEN
       INSERT INTO employees@sf
       VALUES(:new.employee_id, :new.last_name,..., 'B');
       :NEW.flag := 'A';
     END IF;
     ELSE   /* Actualización. */
      IF :NEW.flag = :OLD.flag THEN
        UPDATE employees@sf
        SET ename = :NEW.last_name, ...,
            flag = :NEW.flag
        WHERE employee_id = :NEW.employee_id;
      END IF;
      IF :OLD.flag = 'A' THEN :NEW.flag := 'B';
      ELSE :NEW.flag := 'A';
      END IF;
    END IF;
END;
```

ORACLE

Replicación de una Tabla con un Disparador

Replique una tabla con un disparador.

- Las tablas se copian de forma síncrona, en tiempo real.
- Las réplicas se basan en una sola tabla maestra.
- Se pueden realizar lecturas y escrituras en las réplicas.
- Se empeora el rendimiento de las manipulaciones de datos en la tabla maestra, especialmente si la red falla.

Se mantienen copias de las tablas automáticamente utilizando instantáneas, especialmente en nodos remotos.

Ejemplo

En Nueva York, replique la tabla local `EMPLOYEES` para San Francisco.

Cálculo de Datos Derivados en el Servidor

```
UPDATE departments
SET total_sal=(SELECT SUM(salary)
                FROM employees
                WHERE employees.department_id =
                     departments.department_id);
```

ORACLE

Cálculo de Datos Derivados en el Servidor

Calcule los valores derivados en un trabajo por lotes.

- Los valores de columna derivados se calculan de forma asíncrona, a intervalos definidos por el usuario.
- Los valores derivados sólo se almacenan en tablas de base de datos.
- Los datos se modifican en una sola transferencia a la base de datos y los datos derivados se calculan en una segunda transferencia.

También puede utilizar disparadores para continuar ejecutando los cálculos de datos derivados.

Ejemplo

El sueldo total de cada departamento se mantiene en una columna `TOTAL_SALARY` especial de la tabla `DEPARTMENTS`.

Cálculo de Valores Derivados con un Disparador

```
CREATE OR REPLACE PROCEDURE increment_salary
  (p_id     IN departments.department_id%TYPE,
   p_salary IN departments.total_sal%TYPE)
IS
BEGIN
   UPDATE departments
   SET    total_sal = NVL (total_sal, 0)+ p_salary
   WHERE  department_id = p_id;
END increment_salary;
```

```
CREATE OR REPLACE TRIGGER compute_salary
AFTER INSERT OR UPDATE OF salary OR DELETE ON employees
FOR EACH ROW
BEGIN
IF DELETING THEN
    increment_salary(:OLD.department_id,(-1*:OLD.salary));
ELSIF  UPDATING THEN
increment_salary(:NEW.department_id,(:NEW.salary-:OLD.salary));
ELSE increment_salary(:NEW.department_id,:NEW.salary);--INSERT
END IF;
END;
```

ORACLE

Cálculo de Valores Derivados con un Disparador

Calcule valores derivados con un disparador.

- Las columnas derivadas se calculan de forma síncrona, en tiempo real.
- Los valores derivados se almacenan en tablas de base de datos o en variables globales de paquetes.
- Se modifican los datos y se calculan los datos derivados en una sola transferencia a la base de datos.

Ejemplo

El sueldo total actualizado de cada departamento se mantiene en una columna `TOTAL_SALARY` especial de la tabla `DEPARTMENTS`.

Registro de Eventos con un Disparador

```
CREATE OR REPLACE TRIGGER notify_reorder_rep
BEFORE UPDATE OF quantity_on_hand, reorder_point
ON inventories  FOR EACH ROW
DECLARE
v_descrip product_descriptions.product_description%TYPE;
v_msg_text VARCHAR2(2000);
stat_send  number(1);
BEGIN
  IF  :NEW.quantity_on_hand <= :NEW.reorder_point THEN
   SELECT product_description INTO v_descrip
   FROM product_descriptions
   WHERE product_id = :NEW.product_id;
   v_msg_text := 'ALERT: INVENTORY LOW ORDER:'||CHR(10)||
   ...'Yours,' ||CHR(10) ||user || '.'|| CHR(10)|| CHR(10);
  ELSIF
   :OLD.quantity_on_hand < :NEW.quantity_on_hand THEN NULL;
  ELSE
   v_msg_text := 'Product #'||... CHR(10);
  END IF;
  DBMS_PIPE.PACK_MESSAGE(v_msg_text);
  stat_send := DBMS_PIPE.SEND_MESSAGE('INV_PIPE');
END;
```

ORACLE

Registro de Eventos con un Disparador

En el servidor, puede registrar eventos consultando los datos y realizando operaciones manualmente. Este código envía un mensaje utilizando un canal cuando el inventario de un producto específico desciende por debajo del límite aceptable. El disparador utiliza el paquete de Oracle `DBMS_PIPE` para enviar el mensaje.

Registro de Eventos en el Servidor

- Se consultan los datos explícitamente para determinar si una operación es necesaria.
- En un segundo paso, se realiza la operación como, por ejemplo, enviar un mensaje.

Uso de Disparadores para Registrar Eventos

- Se realizan operaciones implícitamente como, por ejemplo, el lanzamiento de un memorándum electrónico.
- Se modifican los datos y se realiza su operación dependiente en un solo paso.
- Los eventos se registran automáticamente mientras los datos cambian.

Registro de Eventos con un Disparador (continuación)

Registro de Eventos de Manera Transparente

En el código del disparador:

- `CHR(10)` es un retorno de carro
- `Reorder_point` es no nulo
- Otra transacción puede recibir y leer el mensaje en el canal

Ejemplo

```
 CREATE OR REPLACE TRIGGER notify_reorder_rep
 BEFORE UPDATE OF amount_in_stock, reorder_point
 ON inventory  FOR EACH ROW
 DECLARE
   v_descrip product.descrip%TYPE;
   v_msg_text VARCHAR2(2000);
   stat_send  number(1);
 BEGIN
   IF  :NEW.amount_in_stock <= :NEW.reorder_point THEN
     SELECT descrip INTO  v_descrip
     FROM PRODUCT WHERE prodid = :NEW.product_id;
     v_msg_text := 'ALERT: INVENTORY LOW ORDER:'||CHR(10)||
     'It has come to my personal attention that, due to recent'
     ||CHR(10)||'transactions, our inventory for product # '||
     TO_CHAR(:NEW.product_id)||'-- '||v_descrip ||
     ' -- has fallen below acceptable levels.' || CHR(10) ||
     'Yours,' ||CHR(10) ||user || '.'|| CHR(10)|| CHR(10);
   ELSIF
     :OLD.amount_in_stock<:NEW.amount_in_stock THEN NULL;
   ELSE
     v_msg_text := 'Product #'|| TO_CHAR(:NEW.product_id)
     ||' ordered. '|| CHR(10)|| CHR(10);    END IF;
   DBMS_PIPE.PACK_MESSAGE(v_msg_text);
   stat_send := DBMS_PIPE.SEND_MESSAGE('INV_PIPE');
END;
```

Ventajas de los Disparadores de Bases de Datos

- **Mayor seguridad de los datos:**
  - **Proporcionan comprobaciones de seguridad mejores y más complejas**
  - **Proporcionan auditorías mejores y más complejas**
- **Mayor integridad de los datos:**
  - **Fuerzan restricciones de integridad de datos dinámicas**
  - **Fuerzan restricciones de integridad referencial complejas**
  - **Garantizan que las operaciones relacionadas se realizan conjuntamente de manera implícita**

ORACLE

Ventajas de los Disparadores de Bases de Datos

Puede utilizar disparadores de base de datos:

- Como alternativa a las funciones que proporciona Oracle Server
- Si sus requisitos son más complejos o más sencillos que los que proporciona Oracle Server
- Si sus requisitos no están incluidos en Oracle Server

Gestión de Disparadores

Estos son los privilegios del sistema necesarios para gestionar los disparadores:

- **El privilegio `CREATE/ALTER/DROP (ANY) TRIGGER` le permite crear un disparador en cualquier esquema**
- **El privilegio `ADMINISTER DATABASE TRIGGER` le permite crear un disparador en `DATABASE`**
- **El privilegio `EXECUTE` (si su disparador hace referencia a cualquier objeto que no esté en su esquema)**

Nota: Las sentencias del cuerpo del disparador funcionan bajo el privilegio del propietario del disparador, no del usuario del disparador.

ORACLE

Gestión de Disparadores

Para poder crear un disparador en un esquema, se necesita el privilegio del sistema CREATE TRIGGER, ser el propietario de la tabla que se especifica en la sentencia disparadora, tener el privilegio ALTER para la tabla de la sentencia disparadora o tener el privilegio del sistema ALTER ANY TABLE. Para modificar o borrar los disparadores no es necesario tener ningún otro privilegio.

Si utiliza la palabra clave ANY, podrá crear, modificar o borrar sus propios disparadores y los que se encuentran en otro esquema, así como asociarlos a la tabla de cualquier usuario.

No necesita ningún privilegio para llamar a un disparador del esquema. Las sentencias DML que se emiten llaman a un disparador. No obstante, si su disparador hace referencia a cualquier objeto que no esté en el esquema, el usuario que crea el disparador debe tener el privilegio EXECUTE sobre los procedimientos, las funciones o los paquetes a los que se hace referencia, y no mediante roles. Al igual que con los procedimientos almacenados, la sentencia situada en el cuerpo del disparador funciona bajo el dominio del privilegio del propietario del disparador, no del usuario que emite la sentencia disparadora.

Para crear un disparador en DATABASE, debe tener el privilegio ADMINISTER DATABASE TRIGGER. Si, más adelante, este privilegio se revoca, puede borrar el disparador, pero no modificarlo.

Visualización de la Información de los Disparadores

Puede ver la siguiente información de los disparadores:

- **Vista de diccionario de datos `USER_OBJECTS`: información del objeto**
- **Vista de diccionario de datos `USER_TRIGGERS`: el texto del disparador**
- **Vista de diccionario de datos `USER_ERRORS`: Errores de sintaxis PL/SQL (errores de compilación) del disparador**

ORACLE

Visualización de la Información del Disparador

La transparencia muestra las vistas de diccionario de datos a las que puede acceder para obtener información acerca de los disparadores.

La vista `USER_OBJECTS` contiene el nombre y el estado del disparador y la fecha y la hora en la que fue creado.

La vista `USER_ERRORS` contiene los detalles de los errores de compilación que se producen durante la compilación de un disparador. El contenido de estas vistas es similar al de los subprogramas.

La vista `USER_TRIGGERS` contiene detalles como, por ejemplo, el nombre, el tipo, el evento disparador, la tabla en la se crea el disparador y el cuerpo del disparador.

La sentencia `SELECT NombreUsuario FROM USER_USERS;` proporciona el nombre del propietario del disparador, no el nombre del usuario que está actualizando la tabla.

Uso de USER_TRIGGERS*

Columna	Descripción de la Columna
TRIGGER_NAME	Nombre del disparador
TRIGGER_TYPE	El tipo es BEFORE, AFTER, INSTEAD OF
TRIGGERING_EVENT	La operación DML que arranca el disparador
TABLE_NAME	Nombre de la tabla de base de datos
REFERENCING_NAMES	Nombre utilizado para :OLD y :NEW
WHEN_CLAUSE	La cláusula when utilizada
STATUS	El estado del disparador
TRIGGER_BODY	La acción que se va a realizar

* Lista de columnas resumida

ORACLE

Uso de USER_TRIGGERS

Si el archivo de origen no está disponible, puede utilizar *i*SQL*Plus para regenerarlo desde USER_TRIGGERS. También puede examinar todas las vistas ALL_TRIGGERS y DBA_TRIGGERS, cada una de las cuales contiene la columna OWNER adicional para el propietario del objeto.

Listado del Código de los Disparadores

```
SELECT trigger_name, trigger_type, triggering_event,
        table_name, referencing_names,
        status, trigger_body
FROM   user_triggers
WHERE  trigger_name = 'RESTRICT_SALARY';
```

TRIGGER_NAME	TRIGGER_TYPE	TRIGGERING_EVENT	TABLE_NAME	REFERENCING_NAMES	WHEN_CLAUS	STATUS	TRIGGER_BODY
RESTRICT_SALARY	BEFORE EACH ROW	INSERT OR UPDATE	EMPLOYEES	REFERENCING NEW AS NEW OLD AS OLD		ENABLED	BEGIN IF NOT (:NEW.JOB_ ID IN ('AD_PRES ', 'AD_VP')) AND :NE W.SAL

ORACLE

Ejemplo

Utilice la vista de diccionario de datos `USER_TRIGGERS` para ver la información acerca del disparador `RESTRICT_SAL`.

Resumen

En esta lección, ha aprendido a:

- **Utilizar disparadores de base de datos avanzados**
- **Enumerar las reglas de mutación y de restricción de los disparadores**
- **Describir las aplicaciones reales de los disparadores**
- **Gestionar disparadores**
- **Ver la información de los disparadores**

ORACLE

Visión General de la Práctica 17

Esta práctica cubre la creación de disparadores avanzados para aumentar las capacidades de la base de datos de Oracle.

ORACLE

Visión General de la Práctica 17

En esta práctica, tiene que decidir cómo implementar una serie de reglas de negocio. Deberá crear disparadores para esas reglas que se deberían implementar como disparadores. Los disparadores ejecutarán procedimientos que habrá colocado en un paquete.

Práctica 17

Más adelante se enumeran una serie de reglas de negocio que se aplican a las tablas EMPLOYEES y DEPARTMENTS.

Decida cómo implementar cada una de estas reglas de negocio, por medio de restricciones declarativas o utilizando disparadores.

¿Qué restricciones o disparadores se necesitan? ¿Hay que esperar que se produzca algún problema?

Implemente las reglas de negocio definiendo los disparadores o las restricciones que haya decidido crear.

En el archivo `lab17_1.sql` se proporciona un paquete parcial al cual debería agregar todos los procedimientos o funciones necesarias que se vayan a llamar desde los disparadores que haya decidido crear para las siguientes reglas.

(Los disparadores deberían ejecutar los procedimientos o las funciones que haya definido en el paquete).

Reglas de Negocio

Regla 1. Los jefes de ventas y los representantes de ventas deberían recibir siempre una comisión. Los empleados que no sean jefes de ventas ni representantes de ventas no deberían recibir nunca una comisión. Asegúrese de que esta restricción no valide los registros existentes de la tabla EMPLOYEES. Sólo debería ser efectiva para las inserciones y las actualizaciones posteriores que se realicen en la tabla.

Regla 2. La tabla EMPLOYEES sólo debería contener un presidente.

Compruebe su respuesta insertando un registro de empleado con los siguientes detalles: identificador de empleado `400`, apellido `Harris`, nombre `Alice`, identificador de correo electrónico AHARRIS, identificador de puesto de trabajo AD_PRES, fecha de contratación SYSDATE, sueldo `20000` e identificador de departamento `20`.

Nota: No es necesario que implemente una regla para la sensibilidad a mayúsculas/minúsculas; en su lugar, tendrá que comprobar el número de personas que tienen el puesto de trabajo de Presidente.

Regla 3. Un empleado nunca puede ser jefe de más de 15 empleados.

Compruebe su respuesta insertando los siguientes registros en la tabla EMPLOYEES (realice una consulta para contar el número de empleados que trabajan actualmente para el jefe 100 antes de insertar estas filas):

i Identificador de empleado `401`, apellido `Johnson`, nombre `Brian`, identificador de correo electrónico BJOHNSON, identificador de puesto de trabajo SA_MAN, fecha de contratación SYSDATE, sueldo `11000`, identificador de jefe `100` e identificador de departamento `80`. (Esta inserción debería ser correcta, porque hasta ahora sólo hay 14 empleados que trabajan para el jefe 100).

ii. Identificador de empleado `402`, apellido `Kellogg`, nombre `Tony`, identificador de correo electrónico TKELLOG, identificador de puesto de trabajo ST_MAN, fecha de contratación SYSDATE, sueldo `7500`, identificador de jefe `100` e identificador de departamento `50`. (Esta inserción debería ser incorrecta, porque ya hay 15 empleados trabajando para el jefe 100).

Regla 4. Los sueldos sólo pueden aumentar, nunca reducirse.

El sueldo actual del empleado 105 es 5000. Compruebe su respuesta reduciendo el sueldo del empleado 105 hasta 4500.

Práctica 17 (continuación)

Regla 5. Si un departamento se traslada de lugar, cada empleado de ese departamento recibe automáticamente un aumento de sueldo del 2 por ciento.

Visualice los sueldos actuales de los empleados del departamento 90.

LAST_NAME	SALARY	DEPARTMENT_ID
King	24000	90
Kochhar	17000	90
De Haan	17000	90

Compruebe su respuesta trasladando el departamento 90 a la ubicación 1600. Consulte los nuevos sueldos de los empleados del departamento 90.

LAST_NAME	SALARY	DEPARTMENT_ID
King	24480	90
Kochhar	17340	90
De Haan	17340	90