ÍNDICE

Contenido

Antolín·Muñoz·Chaparro¶
Jefe·de·Proyecto·Sistemas·Informáticos·¶
División·III·de·Aplicaciones·de·Costes·de·Personal·y·Pensiones·Públicas¶
Oficina·de·Informática·Presupuestaria·(DIR3:·EA0027952)¶
Intervención·General·de·la·Administración·del·Estado¶

Antolín·Muñoz·Chaparro¶
Jefe·de·Proyecto·Sistemas·Informáticos·¶
División·III·de·Aplicaciones·de·Costes·de·Personal·y·Pensiones·Públicas¶
Oficina·de·Informática·Presupuestaria·(DIR3:·EA0027952)¶
Intervención·General·de·la·Administración·del·Estado¶
GOBIERNO DE ESPAÑA MINISTERIO DE HACIENDA

BUENAS PRÁCTICAS (TRIGGERS)

Aunque los triggers son una herramienta potente de automatización de tareas en base de datos, tienen sus limitaciones de uso que hay que tener en cuenta cuando se diseñan tanto en el uso de código SQL como PL/SQL .

Uso de COMMIT y ROLLBACK

No se puede utilizar las instrucciones COMMIT y ROLLBACK dentro de la definición del propio trigger.

Por ejemplo, este trigger aunque no diera errores de compilación, produciría un error en tiempo de ejecución del tipo ORA-04092: cannot COMMIT in a trigger porque Oracle no admite que se realicen operaciones del tipo COMMIT o ROLLBACK dentro del propio trigger.

```
CREATE OR REPLACE TRIGGER EJEMPLO
      BEFORE INSERT OF codigo ON departamento
      FOR EACH ROW
BEGIN
      IF INSERTING THEN
            INSERT INTO t_pruebas(sysdate);
            COMMIT;
      END IF;
EXCEPTION
      WHEN OTHERS THEN
            ROLLBACK;
END;
```

Pero además, tampoco podemos realizar operaciones COMMIT o ROLLBACK en un procedimiento/función/paquete que se invoque desde el propio trigger.

Por ejemplo, este trigger que llama al procedimiento EJEMPLO1 aunque no diera errores de compilación, produciría un error en tiempo de ejecución del tipo ORA-04092: cannot COMMIT in a trigger porque Oracle no admite que se realicen operaciones del tipo COMMIT o ROLLBACK dentro del propio trigger o en llamadas a paquetes/procedimientos/funciones que las incluyan.

```
CREATE OR REPLACE PROCEDURE EJEMPLO1 AS
BEGIN
      INSERT INTO t_pruebas(sysdate);
      COMMIT;
EXCEPTION
      WHEN OTHERS THEN
            ROLLBACK;
END;


CREATE OR REPLACE TRIGGER EJEMPLO
      BEFORE INSERT OF codigo ON departamento
      FOR EACH ROW
BEGIN
      IF INSERTING THEN
            EJEMPLO1;
      END IF;
EXCEPTION
      WHEN OTHERS THEN
            ROLLBACK;
END;
```

COMO REALIZAR OPERACIONES COMMIT Y ROLLBACK DESDE TRIGGERS

La única opción que permite Oracle, para realizar operaciones COMMIT o ROLLBACK desde triggers es invocando a un paquete/procedimiento/función que se haya definido como PRAGMA AUTONOMOUS_TRANSACTION;

Si tomamos como ejemplo el caso anterior en el que el trigger invoca a un procedimiento para ejecutar un COMMIT, la solución para que Oracle no devuelva el error ORA-04092: cannot COMMIT in a trigger en tiempo de ejecución, es incluir la cláusula que se indica en sombreado amarillo dentro del código de creación del procedimiento.

```
CREATE OR REPLACE PROCEDURE EJEMPLO1 AS
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
      INSERT INTO t_pruebas(sysdate);
      COMMIT;
EXCEPTION
      WHEN OTHERS THEN
            ROLLBACK;
END;


CREATE OR REPLACE TRIGGER EJEMPLO
      BEFORE INSERT OF codigo ON departamento
      FOR EACH ROW
BEGIN
      IF INSERTING THEN
            EJEMPLO1;
      END IF;
EXCEPTION
      WHEN OTHERS THEN
            ROLLBACK;
END;
```

PRAGMA AUTONOMOUS_TRANSACTION

Como su nombre indica este Pragma hace autónoma una transacción. AUTONOMOUS_TRANSACTION cambia la forma en que funciona un subprograma dentro de una transacción. Un subprograma marcado con este pragma puede realizar operaciones SQL y hacer *COMMIT* o *ROLLBACK* de esas operaciones, sin hacer *COMMIT* o *ROLLBACK* en los datos en la transacción principal.

El término "transacción autónoma" se refiere a la capacidad de PL/SQL de suspender temporalmente la transacción actual y comenzar otra transacción totalmente independiente (que no se revertirá (ROLLBACK) si se anula el código externo). La segunda transacción se conoce como una transacción autónoma. La transacción autónoma funciona independientemente del código principal.

De esta forma aplicando esta filosofía al ejemplo definido anteriormente, cuando se lanza un trigger que invoca al procedimiento con PRAGMA AUTONOMOUS_TRANSACTION se dispone de dos transacciones independientes:

1. La transacción asociada al trigger.
2. La transacción asociada al procedimiento.

No puede aplicar este pragma a un paquete completo o un tipo de objeto completo. En cambio, puede aplicar el pragma a cada subprograma o método de objeto empaquetado.

Puede codificar el pragma en cualquier lugar de la sección DECLARE.

Una vez iniciada, una transacción autónoma es totalmente independiente. No comparte bloqueos, recursos o dependencias de **COMMIT** con la transacción principal. Puede registrar eventos, incrementar los contadores de reintentos, etc., incluso si la transacción principal se revierte.

A diferencia de los Triggers normales, los autónomos pueden contener sentencias de control de transacciones como **COMMIT** y **ROLLBACK**, y pueden emitir sentencias **DDL** (como *CREATE* y *DROP*) a través de la sentencia **EXECUTE INMEDIATE**.

Los cambios realizados por una transacción autónoma se hacen visibles para otras transacciones cuando la transacción autónoma se hace COMMIT. Los cambios también se vuelven visibles para la transacción principal cuando se reanuda, pero solo si su nivel de aislamiento se establece en READ COMMITTED (el valor predeterminado). Si establece el nivel de aislamiento de la transacción principal en SERIALIZABLE, los cambios realizados por sus transacciones autónomas no serán visibles para la transacción principal cuando se reanude.

En la transacción principal, retroceder a un punto de guardado ubicado antes de la llamada al subprograma autónomo no revierte la transacción autónoma. Recuerde, las transacciones autónomas son totalmente independientes de la transacción principal.

Si una transacción autónoma intenta acceder a un recurso retenido por la transacción principal (que no puede reanudarse hasta que la rutina autónoma salga), puede producirse un punto muerto. Oracle genera una excepción en la transacción autónoma, que se revierte si la excepción no se maneja.

Si intenta salir de una transacción autónoma activa sin hace COMMIT o ROLLBACK, Oracle genera una excepción. Si la excepción no se controla, o si la transacción finaliza debido a alguna otra excepción no controlada, la transacción se revierte.

USO DE TABLAS ASOCIADAS AL TRIGGER

Cómo ya se indicó en el temario del bloque 5, las tablas mutantes asociadas a los triggers referencian a aquellas operaciones que se intentan realizar sobre la tabla que está asociada al propio trigger.

Por ejemplo, el siguiente ejemplo de trigger asociado a la tabla DEPARTAMENTO, está haciendo un mal uso de la consulta marcada en amarillo porque si bien no produce errores de compilación, si va a producir errores en tiempo de ejecución por TABLA MUTANTE ORA-04091: table name is mutating, trigger/function may not see it

```
CREATE OR REPLACE TRIGGER EJEMPLO
      BEFORE INSERT OF codigo ON departamento
      FOR EACH ROW
DECLARE
      Contador   NUMBER;
BEGIN
      IF INSERTING THEN
            SELECT COUNT(1) INTO Contador FROM departamento;
            DBMS_OUTPUT.PUT_LINE(Contador);
      END IF;
END;
```

Pero este mal uso, también es extensible a cualquier llamada dentro del trigger a procedimientos/funciones/paquetes que están utilizando la tabla asociada al trigger, aunque se haya definido este procedimiento/función con la clásula PRAGM AUTONOMOUS_TRANSACTION;

Por ejemplo, este ejemplo está erróneamente construido (no por errores de compilación), dado que también producirá un error de tabla mutante.

```
CREATE OR REPLACE PROCEDURE EJEMPLO1 AS
BEGIN
      SELECT COUNT(1) INTO Contador FROM departamento;
      DBMS_OUTPUT.PUT_LINE(Contador);
EXCEPTION
      WHEN OTHERS THEN
            ROLLBACK;
END;


CREATE OR REPLACE TRIGGER EJEMPLO
      BEFORE INSERT OF codigo ON departamento
      FOR EACH ROW
BEGIN
      IF INSERTING THEN
            EJEMPLO1;
      END IF;
END;
```

Y el siguiente ejemplo también está erróneamente construido. En este caso no producirá errores por tabla mutante, pero el procedimiento no se enterará del contenido de la tabla departamento que haya podido ser modificado por el trigger dado que la tabla se encuentra bloqueada por el propio trigger hasta que concluya su operación o transacción asociada.

Es decir, si la tabla departamento tenía antes de ejecutar el trigger 10 elementos y cuando salta el evento del trigger se pretende añadir uno más, cuando se ejecute el procedimiento EJEMPLO1 el resultado que seguirá devolviendo será de 10 elementos porque sólo hasta que termine el trigger no habrá 11 elementos en la tabla.

```
CREATE OR REPLACE PROCEDURE EJEMPLO1 AS
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
      SELECT COUNT(1) INTO Contador FROM departamento;
      DBMS_OUTPUT.PUT_LINE(Contador);
EXCEPTION
      WHEN OTHERS THEN
            ROLLBACK;
END;


CREATE OR REPLACE TRIGGER EJEMPLO
      BEFORE INSERT OF codigo ON departamento
      FOR EACH ROW
BEGIN
      IF INSERTING THEN
            EJEMPLO1;
      END IF;
END;
```

Antolín·Muñoz·Chaparro¶
Jefe·de·Proyecto·Sistemas·Informáticos·¶
División·III·de·Aplicaciones·de·Costes·de·Personal·y·Pensiones·Públicas¶
Oficina·de·Informática·Presupuestaria·(DIR3:·EA0027952)¶
Intervención·General·de·la·Administración·del·Estado¶
GOBIERNO DE ESPAÑA MINISTERIO DE HACIENDA