

























```
FOR EACH ROW
WHEN ((NEW.edad <= 0) OR (NEW.edad > 150))
BEGIN
  RAISE_APPLICATION_ERROR(-20001, 'La edad introducida no es válida');
END;
```

Veamos algunos aspectos importantes sobre este trigger:

- Como se ve, se indican los dos eventos en el mismo trigger.
- No se usa la cláusula `REFERENCING` para indicar los nombres de correlación para los valores nuevo y viejo de la tupla que se está insertando o modificando, usando el valor `NEW` para la nueva tupla (no necesitamos acceder al valor viejo en este caso).
- Dado que la condición es simple y no necesita subconsultas, podemos usar la cláusula `WHEN`.
- Elevamos una excepción en la parte de la acción, con lo que se hace un rollback y además el usuario obtiene un mensaje de error. Si usásemos simplemente `ROLLBACK` se deshazarían los cambios silenciosamente.

Como ejercicio adicional, se puede comprobar qué pasa si se intenta borrar la tupla que se acaba de insertar.

Esto podría resolver ambos eventos en un solo trigger. Sin embargo, para el caso de la actualización, la transacción puede continuar normalmente si asignamos el valor anterior de la edad (que sería válido) en un trigger de tipo `BEFORE`. Por ello podríamos eliminar el evento de actualización del trigger anterior y añadir el siguiente:

```
CREATE OR REPLACE TRIGGER ora_edad_val i da_upd
  BEFORE UPDATE OF EDAD ON EMPL
  REFERENCING NEW AS tupl a_nueva
               OLD AS tupl a_vi ej a
  FOR EACH ROW
  WHEN ((tupl a_nueva.Edad <= 0) OR (tupl a_nueva.Edad > 150))
BEGIN
  : tupl a_nueva.Edad := : tupl a_vi ej a.Edad;
END;
```

Como se ve, en este caso usamos los nombres de correlación, que en la parte de la acción (es decir, del bloque PL/SQL) van precedidos de dos puntos. Dado que la sentencia de asignación es válida, la actualización continúa, pero con el valor viejo que le hemos asignado, por lo que realmente no se actualiza la edad del empleado.

Una alternativa a usar estos dos triggers es la utilización, en un único trigger, de los predicados `INSERTING` y `UPDATING`, de la siguiente forma:

```
CREATE OR REPLACE TRIGGER ora_edad_val i da_un_solo_trigger
  AFTER INSERT OR UPDATE OF EDAD ON EMPL
  FOR EACH ROW
  WHEN ((NEW.edad <=0 ) OR (NEW.edad > 150))
BEGIN
```

```

IF UPDATING THEN
    :@71 .Edad := :A>6.Edad; ELSIF
INSERTING THEN
    RAISE_APPLICATION_ERROR(-20001, 'La edad introducida no es válida');
END IF;
END;

```

**Ejercicio 1.4** *Un empleado no puede ganar más que su jefe.*

**Solución:** Aunque también podríamos implementar esta regla con un único trigger para ambos eventos, usaremos de nuevo dos triggers, con lo que la transacción que incluya la actualización podrá continuar, como en el ejercicio anterior.

**Control de inserciones:** Una posible definición del trigger que controla la inserción de empleados con salario inválido es la siguiente:

```

CREATE OR REPLACE TRIGGER ora_emp_val i do_i ns
BEFORE INSERT ON EMPL
REFERENCING NEW AS tupla_nueva
FOR EACH ROW
DECLARE
    NumJefesCobranMenos    INT;
BEGIN
    SELECT COUNT(*) INTO NumJefesCobranMenos
    FROM EMPL
    WHERE Codi go=: tupla_nueva. Jefe
    AND Sal ari o < : tupla_nueva. Sal ari o;
    IF NumJefesCobranMenos=1
    THEN
        RAISE_APPLICATION_ERROR(-20002, 'El salario del empleado no es válido');
    END IF;
END;

```

Como se ve, este trigger es bastante diferente de los anteriores. No existe la cláusula WHEN, debido a que la condición requiere una subconsulta, y esto no es válido para Oracle. Por lo tanto, la comprobación se hace mediante un IF en el bloque PL/SQL de la acción. Además, este bloque ahora empieza con la cláusula DECLARE ya que necesitamos una variable local (NumJefesCobranMenos) que usamos para almacenar el número de jefes que cobran menos que el empleado que estamos insertando. Si este valor es 1, se hace un rollback para evitar la inserción.

En el siguiente apartado veremos los errores que se producen (no al crear el trigger sino al activarse) si declaramos el trigger de tipo AFTER INSERT o si especificamos una sentencia ROLLBACK en la acción del trigger.

**Control de actualizaciones:** Para controlar la modificación tendríamos de nuevo una condición compleja, por lo que no usamos la cláusula WHEN y trasladamos la comprobación de la condición al bloque de la acción.

```

CREATE TRIGGER ora_emp_val i do_upd  -- No funcionar!!!
BEFORE UPDATE ON EMPL

```



























