No es necesario realizar un SELECT para obtener los datos dentro del disparador en este caso, porque las referencias :NEW y :OLD ya te proporcionan directamente los valores que necesitas sin la necesidad de hacer una consulta adicional a la base de datos. Te explico por qué:

**Explicación de las referencias :NEW y :OLD:**

* **:NEW**: Se utiliza en un disparador de tipo INSERT o UPDATE para acceder a los valores nuevos que se están insertando o actualizando en la fila de la tabla. Esta referencia te da acceso a las columnas de la fila **después** de que se haya realizado la operación de INSERT o UPDATE.
* **:OLD**: Se utiliza en un disparador de tipo DELETE o UPDATE para acceder a los valores anteriores de la fila antes de la operación de eliminación o actualización. Esta referencia te da acceso a los valores **antes** de que se modificara o eliminara la fila.

**¿Por qué no es necesario hacer un SELECT?**

1. **Disponibilidad directa de los valores con :NEW y :OLD**:
   * Cuando se dispara un trigger en una operación de INSERT, los valores que se están insertando están disponibles a través de :NEW. Del mismo modo, cuando se dispara un DELETE, los valores de la fila eliminada están disponibles a través de :OLD.
   * Por ejemplo, cuando insertas una fila en la tabla EMP, los valores que se insertan (como EMPNO, ENAME, etc.) ya están disponibles en el contexto del trigger a través de :NEW.
2. **No es necesario acceder a la tabla de nuevo**:
   * El SELECT dentro de un trigger sería redundante porque los valores que necesitas (como el número de empleado y apellido) ya están disponibles en las referencias :NEW o :OLD. Al hacer un SELECT, estarías consultando nuevamente la tabla para obtener los mismos valores que ya puedes obtener directamente.
3. **Eficiencia**:
   * Hacer un SELECT adicional dentro del trigger puede aumentar la carga en el sistema, ya que estarías realizando una consulta innecesaria a la base de datos para obtener valores que ya están presentes en el contexto del disparador. Esto podría generar un coste innecesario en cuanto a rendimiento y tiempos de ejecución.

**Ejemplo con :NEW y :OLD (sin SELECT):**

CREATE OR REPLACE TRIGGER TRG\_EMP

AFTER INSERT OR DELETE ON EMP

FOR EACH ROW

BEGIN

-- Si la operación es una inserción

IF INSERTING THEN

INSERT INTO AUDITAREMPLE

VALUES (SYSDATE, :NEW.EMPNO, :NEW.ENAME, 'INSERCIÓN');

-- Si la operación es un borrado

ELSIF DELETING THEN

INSERT INTO AUDITAREMPLE

VALUES (SYSDATE, :OLD.EMPNO, :OLD.ENAME, 'BORRADO');

END IF;

END;

Como ves, con las referencias :NEW y :OLD tienes acceso directo a los valores que necesitas sin necesidad de hacer un SELECT. Esto hace que tu código sea más limpio y eficiente.

**Conclusión:**

No necesitas hacer un SELECT dentro de un trigger en este caso porque las referencias :NEW y :OLD ya proporcionan los valores de las filas modificadas directamente, y obtener esos mismos valores mediante un SELECT sería redundante y menos eficiente.

Si tienes más dudas sobre este tema o necesitas más detalles, ¡avísame!

El uso de un SELECT dentro de un **trigger** es necesario en algunos casos específicos, generalmente cuando necesitas obtener **información adicional** que no está disponible directamente a través de las referencias :NEW y :OLD. Aquí te explico cuándo sería adecuado usar un SELECT dentro de un trigger:

**Casos en los que se debería usar un SELECT dentro de un trigger:**

1. **Obtener datos de otras tablas**: Si necesitas obtener información adicional de otras tablas que no están relacionadas directamente con la fila que se está modificando (por ejemplo, si necesitas acceder a una tabla de configuraciones o a una tabla relacionada con una clave externa), podrías usar un SELECT.

**Ejemplo**: Supón que tienes una tabla DEPARTAMENTOS y quieres registrar en tu tabla de auditoría no solo el nombre del empleado, sino también el nombre del departamento al que pertenece. Si esa información no está directamente en la tabla EMP y tienes que obtenerla desde otra tabla, necesitarás usar un SELECT para recuperar esos datos.

CREATE OR REPLACE TRIGGER TRG\_EMP

AFTER INSERT OR DELETE ON EMP

FOR EACH ROW

DECLARE

v\_departamento VARCHAR2(100);

BEGIN

-- Obtener el nombre del departamento de la tabla DEPARTAMENTOS

IF INSERTING THEN

SELECT d.dnombre

INTO v\_departamento

FROM departamentos d

WHERE d.deptno = :NEW.deptno;

INSERT INTO AUDITAREMPLE

VALUES (SYSDATE, :NEW.EMPNO, :NEW.ENAME, v\_departamento, 'INSERCIÓN');

ELSIF DELETING THEN

SELECT d.dnombre

INTO v\_departamento

FROM departamentos d

WHERE d.deptno = :OLD.deptno;

INSERT INTO AUDITAREMPLE

VALUES (SYSDATE, :OLD.EMPNO, :OLD.ENAME, v\_departamento, 'BORRADO');

END IF;

END;

En este ejemplo, **obtenemos el nombre del departamento** desde la tabla DEPARTAMENTOS, usando un SELECT dentro del trigger, ya que esa información no está disponible directamente en la tabla EMP.

1. **Obtener valores agregados o datos calculados**: Si necesitas calcular o consultar un valor agregado (como un total, un promedio, etc.) a partir de otras filas o tablas, entonces también necesitarías un SELECT.

**Ejemplo**: Supón que necesitas registrar en tu tabla de auditoría la suma total de salarios de los empleados de un departamento cada vez que se realiza una inserción o eliminación. Para hacerlo, tendrías que usar un SELECT para calcular el total de salarios de ese departamento.

CREATE OR REPLACE TRIGGER TRG\_EMP

AFTER INSERT OR DELETE ON EMP

FOR EACH ROW

DECLARE

v\_total\_salarios NUMBER;

BEGIN

-- Obtener el total de salarios por departamento

IF INSERTING THEN

SELECT SUM(salario)

INTO v\_total\_salarios

FROM emp

WHERE deptno = :NEW.deptno;

INSERT INTO AUDITAREMPLE

VALUES (SYSDATE, :NEW.EMPNO, :NEW.ENAME, v\_total\_salarios, 'INSERCIÓN');

ELSIF DELETING THEN

SELECT SUM(salario)

INTO v\_total\_salarios

FROM emp

WHERE deptno = :OLD.deptno;

INSERT INTO AUDITAREMPLE

VALUES (SYSDATE, :OLD.EMPNO, :OLD.ENAME, v\_total\_salarios, 'BORRADO');

END IF;

END;

1. **Cuando no tienes acceso directo a una columna de la tabla**: Si estás trabajando con triggers en tablas y necesitas obtener valores que no están directamente disponibles en las filas modificadas (por ejemplo, si necesitas datos de una vista, un cálculo específico, o la información de una tabla relacionada), un SELECT puede ser útil para recuperar esos datos.

**Cuándo NO deberías usar un SELECT:**

1. **No es necesario hacer una consulta adicional**: Si la información que necesitas está disponible en las referencias :NEW o :OLD, **no es necesario** hacer un SELECT. Por ejemplo, si solo necesitas el valor de una columna que está siendo modificada en la tabla afectada, puedes acceder directamente a :NEW.columna o :OLD.columna.
2. **Evitar sobrecargar el rendimiento**: Realizar un SELECT adicional dentro de un trigger puede afectar el rendimiento de tu base de datos, especialmente si se realizan muchas operaciones de inserción o eliminación. Si no es estrictamente necesario, trata de evitar el uso de consultas adicionales dentro de triggers.

**Conclusión:**

* **Usa un SELECT dentro de un trigger** cuando necesites obtener datos de otras tablas, calcular valores agregados o hacer consultas adicionales que no estén disponibles directamente a través de las referencias :NEW o :OLD.
* **Evita un SELECT** cuando la información que necesitas ya esté disponible en el contexto del trigger a través de :NEW y :OLD.

Si tienes más dudas sobre el uso de SELECT en triggers o necesitas ejemplos adicionales, no dudes en preguntarme.

Cuando se utiliza :NEW o :OLD en un **trigger**, estos representan **todas las columnas** de la fila afectada, no solo la columna que se está modificando. Esto significa que, incluso si solo se actualiza una columna, como el **salario**, el disparador puede acceder a todas las columnas de esa fila, tanto las nuevas (a través de :NEW) como las antiguas (a través de :OLD).

**Detalle de lo que puedes acceder con :NEW y :OLD:**

* **:NEW**: Representa los **valores nuevos** de la fila **después** de la operación de inserción o actualización.
  + Si estás haciendo un UPDATE de la tabla EMP, y solo se modifica la columna SALARY, entonces :NEW.SALARY tendrá el nuevo valor del salario, pero también puedes acceder a las demás columnas de esa fila, como :NEW.EMPNO, :NEW.ENAME, etc.
* **:OLD**: Representa los **valores anteriores** de la fila **antes** de la operación de actualización o eliminación.
  + En el caso de un UPDATE, si solo se cambia la columna SALARY, entonces :OLD.SALARY tendrá el valor anterior del salario, pero también podrás acceder a las demás columnas como :OLD.EMPNO, :OLD.ENAME, etc.

**Ejemplo con un UPDATE que solo cambia el salario:**

Imaginemos que se realiza un UPDATE en la tabla EMP donde solo se actualiza la columna SALARY de un empleado:

UPDATE EMP

SET SALARY = 5000

WHERE EMPNO = 1234;

Si tienes un trigger de la siguiente forma:

CREATE OR REPLACE TRIGGER TRG\_EMP

AFTER UPDATE ON EMP

FOR EACH ROW

BEGIN

-- Accediendo a todos los valores de la fila actualizada

DBMS\_OUTPUT.PUT\_LINE('Empleado: ' || :NEW.EMPNO || ' Nombre: ' || :NEW.ENAME);

DBMS\_OUTPUT.PUT\_LINE('Nuevo Salario: ' || :NEW.SALARY);

DBMS\_OUTPUT.PUT\_LINE('Salario Anterior: ' || :OLD.SALARY);

DBMS\_OUTPUT.PUT\_LINE('Operación: ACTUALIZADO');

END;

En este caso:

* **:NEW** te dará acceso a todos los valores nuevos de la fila, incluyendo el SALARY modificado, así como las demás columnas como EMPNO, ENAME, etc.
* **:OLD** te dará acceso a todos los valores anteriores, incluido el SALARY antes de ser actualizado, y también las demás columnas de la fila, como EMPNO, ENAME, etc.

Entonces, **aunque solo se modifique una columna (como SALARY)**, en el trigger puedes acceder a **todas las columnas de la fila afectada** mediante :NEW y :OLD.

**Resumen:**

* Cuando un UPDATE se realiza sobre una o más columnas, **tienes acceso a todas las columnas** de la fila tanto con :NEW (valores después de la actualización) como con :OLD (valores antes de la actualización).
* No es necesario que una columna sea la que se actualiza para que puedas acceder a los valores de las demás columnas. Puedes acceder a todas las columnas de esa fila.

Si te queda alguna duda o necesitas más ejemplos, ¡avísame!