¡Buena! Esa es una pregunta clave para escribir buen PL/SQL. Aquí van unas **reglas de oro** (con ejemplos) para saber **cuándo conviene usar BEGIN anidado**:

**✅ Regla 1: Si quieres capturar una excepción de un solo bloque de código sin afectar todo el trigger**

Usa un BEGIN...EXCEPTION...END anidado cuando **solo un trozo de código puede fallar** y **quieres controlar ese error sin interrumpir todo lo demás**.

**🧠 Ejemplo clásico: buscar algo que podría no existir**

BEGIN

SELECT DEPTNO INTO v\_deptno FROM DEPT WHERE DNAME = :NEW.DNAME;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Crear departamento si no existe

INSERT INTO DEPT ...;

END;

🎯 Así, si **no existe el DNAME**, el trigger no peta y tú controlas qué hacer.

**✅ Regla 2: Cuando necesitas manejar un "camino alternativo" si algo falla**

Esto se usa mucho en triggers y procedimientos: **si algo puede fallar, haz otra cosa**.

**Ejemplo:**

BEGIN

SELECT precio INTO v\_precio FROM productos WHERE id = :NEW.id;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

v\_precio := 0; -- si no lo encuentra, le asigno un valor por defecto

END;

**✅ Regla 3: Cuando el error que esperas manejar no es un error general (como acceso a NULL, o error lógico), sino uno muy específico**

Como NO\_DATA\_FOUND, TOO\_MANY\_ROWS, ZERO\_DIVIDE, etc.

Si haces esto en todo el trigger:

BEGIN

-- muchas cosas...

EXCEPTION

WHEN OTHERS THEN

-- capturas errores de cualquier parte y no sabes cuál falló

END;

👎 Es muy difícil de depurar y mantener.

💡 Lo mejor es **capturar solo lo que puede fallar en su propio BEGIN...EXCEPTION...END**.

**🚫 ¿Cuándo NO usar BEGIN anidado?**

* Si no vas a usar EXCEPTION, **no lo necesitas**.
* Si todo el código es secuencial y no esperas errores, no hace falta complicar.
* Si puedes evitarlo con un IF THEN mejor, hazlo (por claridad).

**✅ EN RESUMEN (¡para que lo pegues en tus apuntes!):**

| **CUÁNDO USAR BEGIN ANIDADO** | **POR QUÉ** |
| --- | --- |
| Para capturar errores de una consulta específica | Evitas que el trigger entero falle |
| Para ejecutar un "plan B" si algo falla | Como crear un registro si no existe |
| Para evitar usar WHEN OTHERS a lo loco | Mejor manejo de errores |
| Para mantener código limpio y controlado | Fácil de mantener y entender |

¿Quieres que te dé una tabla resumen con los errores más comunes y cómo manejarlos con BEGIN anidados?

¡Muy buena pregunta, Bryan! 🔥  
Te voy a explicar de forma **directa y sencilla** para que te quede claro **cuándo** y **por qué** usar un BEGIN...END dentro de otro BEGIN...END (o sea, "doble BEGIN").

**📌 ¿CUÁNDO debes usar un BEGIN dentro de otro BEGIN?**

Cuando dentro de tu bloque principal necesitas manejar **una excepción especial para una sola operación** (por ejemplo, un SELECT, un INSERT, etc.).

Es decir:

* Si haces un SELECT INTO y **puede fallar** (porque no encuentra datos), pero **NO** quieres que todo tu trigger o tu procedimiento explote, **solo esa parte** debe controlarse.
* Entonces **envuelves esa parte en un BEGIN...EXCEPTION...END independiente**.

**🔥 Tip mental rápido:**

* Si quieres hacer algo **y si falla, que no afecte todo lo demás**, usa BEGIN...EXCEPTION...END por separado.
* Si **todo depende de todo** (y si falla algo no quieres continuar), entonces **NO** hace falta doble BEGIN.

**🎯 Tu caso real: ¿por qué hicimos un BEGIN...END dentro?**

Porque:

* Cuando intentamos SELECT deptno INTO v\_deptno FROM dept WHERE dname = :NEW.dname;, **podía fallar** si no existía el departamento.
* Y queríamos decir: **"si no existe, lo creo"**, no romper todo el trigger.

Por eso, esa parte va en un pequeño **bloque independiente**:

BEGIN

SELECT deptno INTO v\_deptno FROM dept WHERE dname = :NEW.dname;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Acción si no existe

END;

**🧠 Mini regla de oro:**

| **Si quieres que un error en una acción NO corte todo** | **Usa BEGIN...EXCEPTION...END dentro.** |
| --- | --- |

**⚡ Ejemplo muy simple:**

BEGIN

-- Algo que sí o sí tiene que pasar

hacer\_algo();

-- Algo que puede fallar pero no quiero que corte

BEGIN

hacer\_algo\_que\_puede\_fallar();

EXCEPTION

WHEN OTHERS THEN

registrar\_error();

END;

-- Continuar normal

hacer\_otra\_cosa();

END;

¿Quieres que también te prepare un **esquema visual** (tipo diagrama de cajas) para que lo veas aún más claro? 🎯👨‍💻  
(Si quieres te lo paso en 1 minuto)