¡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)