**BLOQUES ANONIMOS**

No necesito inizilzar variables, PERO PUEDO HACERLO SI QUIERO EN EL DECLARE

Se usa comillas simples

NUMBER ADMITE DECIMALES

El parámetro i en wile y for minucsculas

SE PEUDE HACER DIVISIONES TIPO 1/12

 **Bloque anónimo** → Usa **variables**, pero no tiene parámetros de entrada.

 **Procedimientos y funciones** → Pueden tener **parámetros de entrada** (IN), de salida (OUT) o ambos (IN OUT).

**no es necesario poner las restricciones dentro de los procedimientos o funciones** si ya las has definido al momento de crear o modificar la tabla. Las restricciones, como NOT NULL, PRIMARY KEY, FOREIGN KEY, CHECK, entre otras, se aplican **a nivel de la tabla**, y la base de datos las manejará automáticamente cuando intentes insertar, actualizar o eliminar registros.

Sí, los **temas sintácticos** y las **reglas** básicas de **PL/SQL** siguen siendo en gran medida iguales, ya sea que estés escribiendo un **bloque anónimo**, un **procedimiento almacenado** o una **función almacenada**. Las diferencias surgen principalmente en la forma en que se definen y se invocan estos bloques de código, así como en algunos aspectos relacionados con la persistencia y la reutilización.

**Similitudes en la sintaxis de bloques anónimos y procedimientos/funciones almacenadas:**

1. **Declaración de variables**:
   * En **bloques anónimos**, **procedimientos** y **funciones almacenadas**, la sintaxis para declarar variables es la misma:

sql

CopiarEditar

DECLARE

variable\_name DATA\_TYPE;

BEGIN

-- Lógica aquí

END;

**bloque anónimo**:

DECLARE

v\_saludo VARCHAR2(50) := 'Hola, mundo';

BEGIN

DBMS\_OUTPUT.PUT\_LINE(v\_saludo);

END;

**procedimiento almacenado**:

CREATE OR REPLACE PROCEDURE saludar IS

v\_saludo VARCHAR2(50) := 'Hola, mundo';

BEGIN

DBMS\_OUTPUT.PUT\_LINE(v\_saludo);

END saludar;

1. **Estructuras de control**:
   * Las estructuras como IF, LOOP, FOR, y WHILE son igualmente válidas en **bloques anónimos**, **procedimientos** y **funciones almacenadas**. La sintaxis sigue siendo la misma.

Ejemplo con IF:

IF condicion THEN

-- Acción si es verdadera

ELSE

-- Acción si es falsa

END IF;

1. **Manejo de excepciones**:
   * La gestión de excepciones es idéntica para los bloques anónimos y los procedimientos/funciones almacenadas. Usamos BEGIN...EXCEPTION...END.

Ejemplo con manejo de excepciones:

BEGIN

-- Código que puede generar error

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Manejo de error

END;

1. **Instrucciones SQL**:
   * Puedes utilizar **sentencias SQL** como SELECT INTO, INSERT, UPDATE, DELETE, etc., tanto en bloques anónimos como en procedimientos/funciones almacenadas.

Ejemplo de SELECT INTO:

DECLARE

v\_nombre VARCHAR2(100);

BEGIN

SELECT nombre INTO v\_nombre FROM empleados WHERE emp\_id = 1;

DBMS\_OUTPUT.PUT\_LINE('Nombre: ' || v\_nombre);

END;

**Diferencias clave en la sintaxis y reglas:**

1. **Definición y uso** de CREATE OR REPLACE:
   * **Procedimientos** y **funciones almacenadas** requieren el uso de CREATE OR REPLACE para definirlos. Esto les da **persistencia** en la base de datos.
   * Los **bloques anónimos** no requieren CREATE OR REPLACE. Son definidos y ejecutados directamente dentro de un entorno de ejecución.

**procedimiento almacenado**:

CREATE OR REPLACE PROCEDURE saludar IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('¡Hola!');

END saludar;

1. **Invocación**:
   * Los **bloques anónimos** se ejecutan directamente cuando se envían a la base de datos.
   * Los **procedimientos y funciones almacenadas** se invocan explícitamente desde otro bloque PL/SQL, o se pueden llamar desde aplicaciones externas, usando EXECUTE o dentro de otros bloques de código.

Ejemplo de invocación de un **procedimiento almacenado**:

BEGIN

saludar; -- Llamada a un procedimiento almacenado

END;

1. **Declaración de parámetros**:
   * **Procedimientos y funciones almacenadas** pueden **declarar parámetros** de entrada (IN), salida (OUT) o ambos (IN OUT), lo cual no es aplicable a bloques anónimos.

Ejemplo de **procedimiento almacenado con parámetros**:

CREATE OR REPLACE PROCEDURE saludar(nombre IN VARCHAR2) IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('¡Hola, ' || nombre || '!');

END saludar;

**Bloque anónimo** (sin parámetros):

DECLARE

v\_nombre VARCHAR2(100) := 'Juan';

BEGIN

DBMS\_OUTPUT.PUT\_LINE('¡Hola, ' || v\_nombre || '!');

END;

1. **Persistencia**:
   * Los **procedimientos** y **funciones almacenadas** se **guardan en la base de datos** para su reutilización posterior, mientras que los bloques anónimos **no tienen persistencia**. Esto significa que, en el caso de los procedimientos/funciones almacenadas, no necesitas reescribirlos cada vez que los uses.

Ejemplo de uso persistente:

CREATE OR REPLACE PROCEDURE saludar IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('¡Hola!');

END saludar;

1. **Accesibilidad**:
   * Los **procedimientos almacenados** pueden ser **accedidos y ejecutados por otros usuarios** si tienen los permisos adecuados, mientras que los bloques anónimos son **locales** al usuario que los ejecuta y no son accesibles por otros usuarios.

**Resumen de las diferencias clave:**

| **Característica** | **Bloque Anónimo** | **Procedimiento/Función Almacenada** |
| --- | --- | --- |
| **Persistencia** | No persiste después de la ejecución | Se guarda en la base de datos hasta que se elimine |
| **Invocación** | Se ejecuta directamente | Se invoca explícitamente con EXECUTE o desde otro bloque |
| **Definición de parámetros** | No se pueden usar parámetros | Se pueden definir parámetros de entrada, salida o ambos |
| **Accesibilidad** | Local, solo se puede ejecutar desde el bloque | Global, puede ser ejecutado por otros usuarios y aplicaciones |
| **Uso de CREATE OR REPLACE** | No se necesita | Requiere CREATE OR REPLACE para ser definido |