



# MODELOS DE DATOS

*PL/SQL - FUNCIONES*

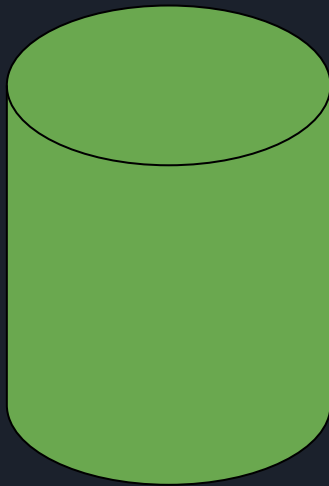


RECORDERIS...





# ¿QUÉ APRENDEREMOS HOY?



- Introducción a PL/SQL
- Definición de "función"
  - Partes de una función
  - Palabras clave
- Manejo de condicionales



# DEFINICIÓN: PL/SQL





# DEFINICIÓN: PL/SQL

**PL/SQL = Procedural Language / Structured Query Language**

**Lenguaje de programación basado en SQL, soporta parámetros, variables, estructuras de control (condicionales, ciclos) y manejo de excepciones**

**Soportado por varios dbms populares (oracle, sqlserver, mysql, mariadb, postgresql...)**

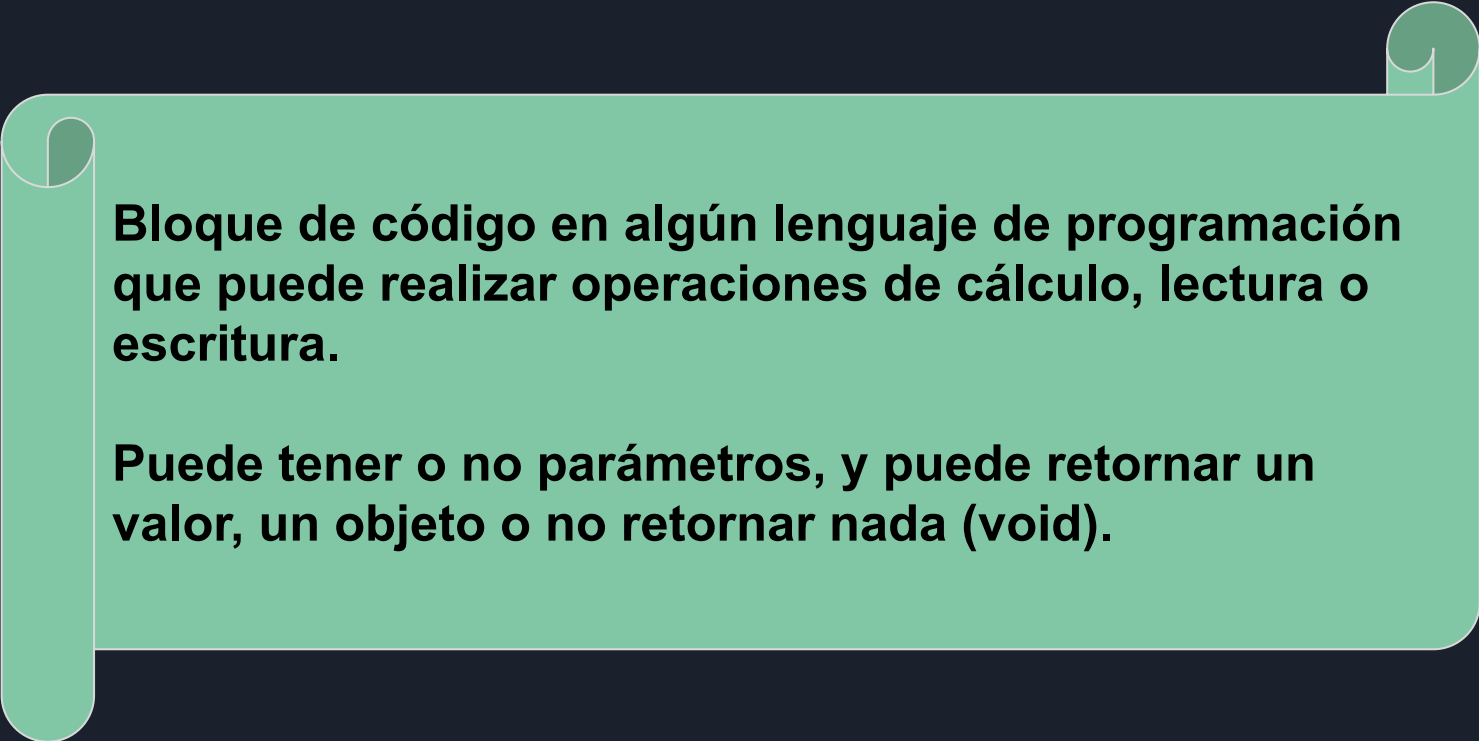


# DEFINICIÓN: FUNCIÓN





# DEFINICIÓN: FUNCIÓN



**Bloque de código en algún lenguaje de programación que puede realizar operaciones de cálculo, lectura o escritura.**

**Puede tener o no parámetros, y puede retornar un valor, un objeto o no retornar nada (void).**



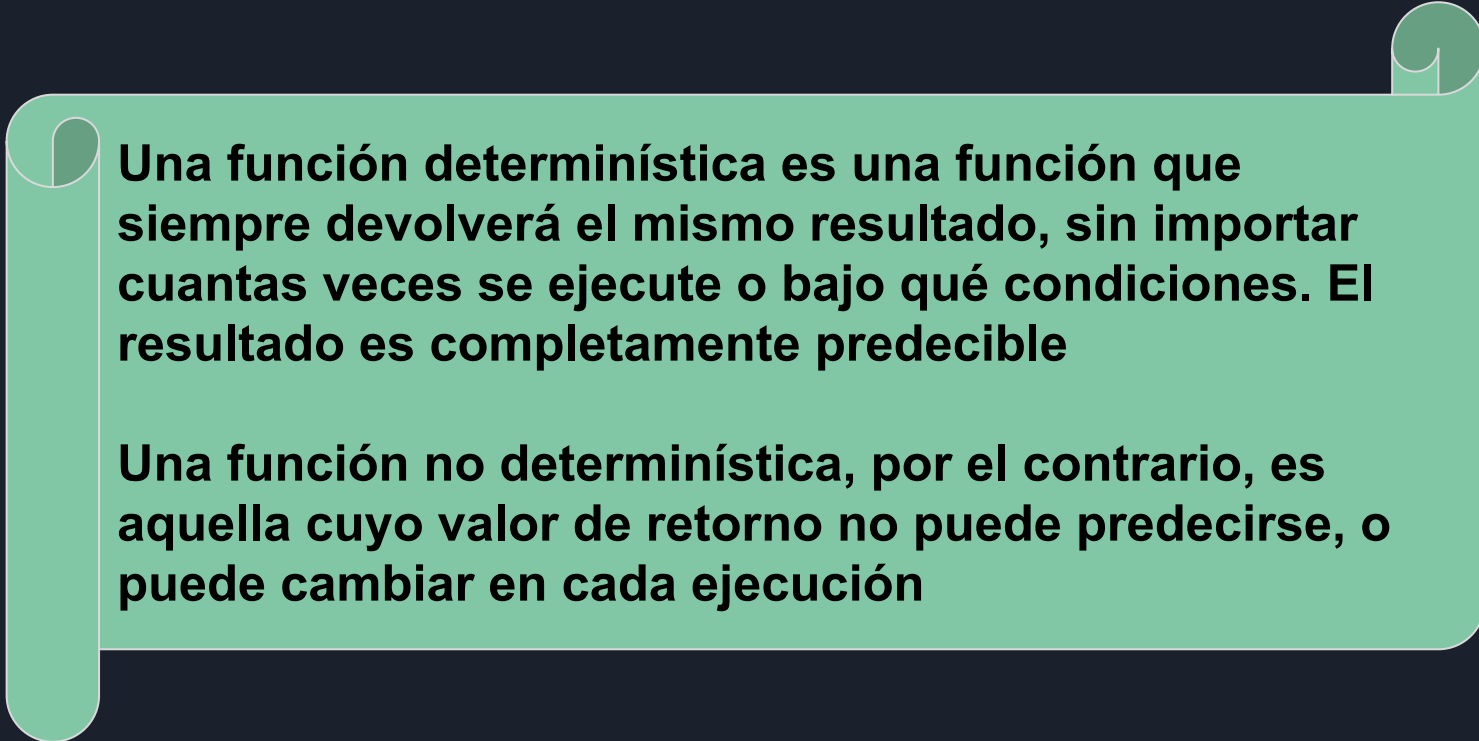
# DEFINICIÓN: Determinística







# DEFINICIÓN: Determinística



Una función determinística es una función que siempre devolverá el mismo resultado, sin importar cuantas veces se ejecute o bajo qué condiciones. El resultado es completamente predecible

Una función no determinística, por el contrario, es aquella cuyo valor de retorno no puede predecirse, o puede cambiar en cada ejecución



# Ejemplos ¿?¿?¿?

Mencionar ejemplos de funciones  
determinísticas y de no determinísticas



# CONCEPTO: Función pl/sql





# CONCEPTO: *Función pl/sql*

**Una función en pl/sql se asemeja mucho a lo que es en cualquier otro lenguaje de programación.**

**En mysql, una función es una secuencia de pasos que realiza operaciones para retornar un valor, y generalmente no modifica datos en la base de datos (es posible que lo haga pero no es lo usual).**



# *Funciones en mysql*

**Consejo muy personal:**

**"Las funciones en MySQL o MariaDB deberían usarse sólo para operaciones de lectura"**



# Funciones nativas en mysql

Algunas  
funciones  
por tipo

Texto

concat, concat\_ws, instr, left, right, lpad,  
trim, replace, reverse, substr

Numéricas

+, -, \*, /, %, abs, conv, exp, ceil, floor,  
pow, rand, round, trunc, sqrt, truncate

Tiempo

now, month, get\_format, last\_day,  
date\_diff, convert\_tz, add\_date, utc\_time

# Estructura de una función

Función

Encabezado

También llamado "definición".

- Nombre de la función (obligatorio)
- Parámetros de entrada (0 ó mas)
- Tipo de dato de retorno (obligatorio)

Cuerpo

Es la funcionalidad o la lógica como tal. Qué es lo que hará la función para calcular lo que va a retornar

Documentación

Realmente no hace parte de la estructura. Pero... si ayuda a entender para qué sirve!



# Ejemplo: función nativa *instr*

Nombre de la función	instr
Parámetro(s) de entrada	texto original, texto a buscar
Tipo de dato de retorno	int
Descripción	Busca la posición del "texto a buscar" dentro de la cadena "texto original"
Ejemplo	instr('hola mundo', 'a') ⇒ Retorna 4





# *¿Para qué crear funciones en mysql?*

- Funcionalidad específica para cierto modelo relacional
- Funcionalidad que no existe con las funciones nativas
- Exponer a la capa de aplicación lógica que es propia de bases de datos
- Abstraer la complejidad de una operación (o serie de operaciones)



# ¿Como crear funciones en mysql?

```
delimiter $$
```

Encabezado

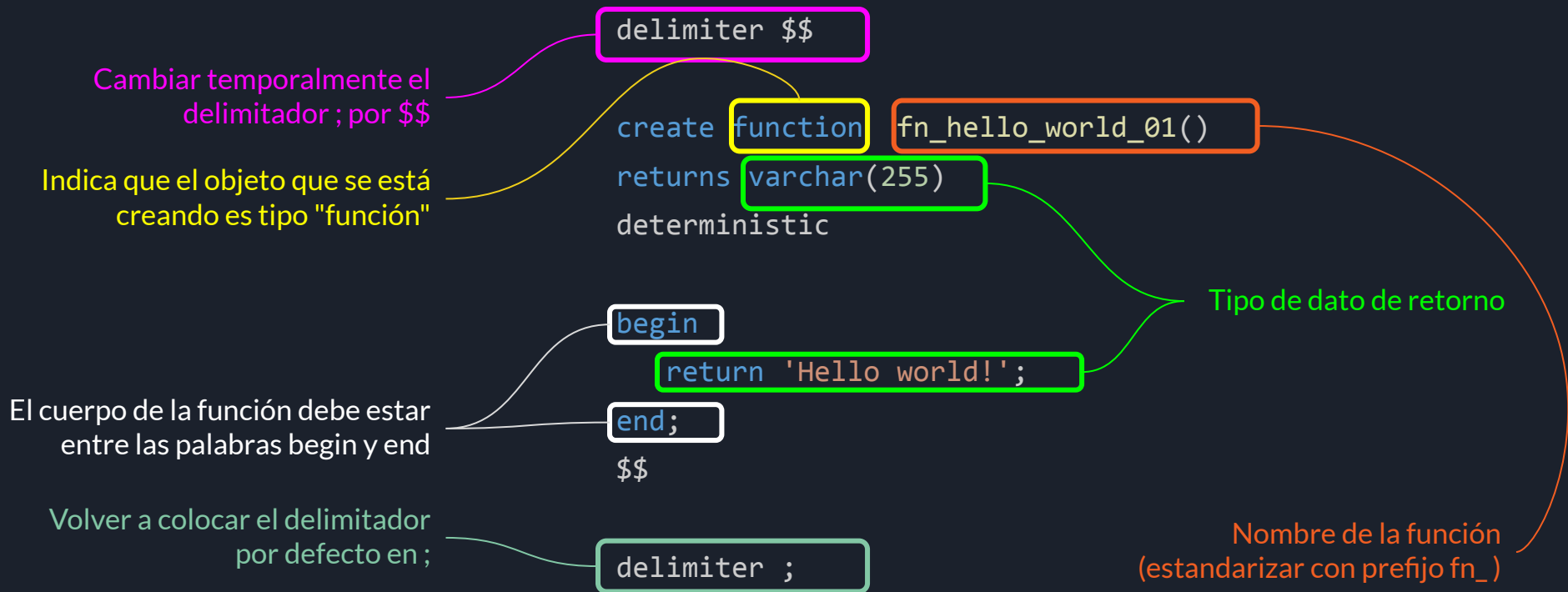
```
create function fn_hello_world_01()  
returns varchar(255)  
deterministic
```

Cuerpo

```
begin  
    return 'Hello world!';  
end;  
$$
```

```
delimiter ;
```

# ¿Como crear funciones en mysql?





# Función con parámetros

Encabezado

Cuerpo

```
delimiter $$
```

```
create function fn_sum_two_integer_numbers(  
    p_number_1 int,  
    p_number_2 int  
)  
returns int  
deterministic  
begin  
    declare v_result int;  
  
    set v_result := p_number_1 + p_number_2;  
  
    return v_result;  
end;  
$$
```

# Función con parámetros

delimiter \$\$

Nombre de los parámetros  
(estandarizar con prefijo p\_)

Nombre de variable  
(estandarizar con prefijo v\_)

Palabra clave para declarar  
variables nuevas

Palabra clave para asignar valor a  
una variable

Operador para asignar un valor a  
una variable

```
create function fn_sum_two_integer_numbers(  
  p_number_1 int,  
  p_number_2 int  
)  
  returns int  
  deterministic  
begin  
  declare v_result int;  
  set v_result := p_number_1 + p_number_2;  
  return v_result;  
end;  
$$
```



# Condicionales: if-then-elseif-else

```
(...)  
begin  
    declare v_result varchar(10);  
  
    if p_color = 'BLACK' then  
        set v_result := 'NEGRO';  
    elseif p_color = 'WHITE' then  
        set v_result := 'BLANCO';  
    elseif p_color = 'YELLOW' then  
        set v_result := 'AMARILLO';  
    else  
        set v_result := 'N/A';  
    end if;  
  
    return v_result;  
end;  
(...)
```

# Condicionales: if-then-elseif-else

```
(...)  
begin  
    declare v_result varchar(10);  
  
    if p_color = 'BLACK' then  
        set v_result := 'NEGRO';  
    elseif p_color = 'WHITE' then  
        set v_result := 'BLANCO';  
    elseif p_color = 'YELLOW' then  
        set v_result := 'AMARILLO';  
    else  
        set v_result := 'N/A';  
    end if;  
  
    return v_result;  
end;  
(...)
```

Todo if debe cerrarse con end if

Los elseif y else son opcionales y dependen únicamente de la lógica



# Condicionales: case-when-else

```
(...)  
begin  
    declare v_result varchar(10);  
  
    case p_color  
    when 'BLACK' then  
        set v_result := 'NEGRO';  
    when 'WHITE' then  
        set v_result := 'BLANCO';  
    when 'YELLOW' then  
        set v_result := 'AMARILLO';  
    else  
        set v_result := 'N/A';  
    end case;  
  
    return v_result  
end  
(...)
```



# Condicionales: case-when-else

Todo case debe cerrarse con end case

Debe existir al menos un when. El else es opcional. Esto depende de la lógica de cada función

```
(...)  
begin  
    declare v_result varchar(10);  
  
    case p_color  
        when 'BLACK' then  
            set v_result := 'NEGRO';  
        when 'WHITE' then  
            set v_result := 'BLANCO';  
        when 'YELLOW' then  
            set v_result := 'AMARILLO';  
        else  
            set v_result := 'N/A';  
        end case;  
  
    return v_result  
end  
(...)
```



# Llamar a una función desde otra

(...)

```
declare v_hello varchar(30);  
declare v_result varchar(30);  
  
set v_hello := 'hello world!';  
set v_result := substr(v_hello, 1, 5); // hello  
set v_result := substr(v_hello, 7);    // world!  
set v_result := substr(v_hello, -6);   // world!  
set v_result := substr(v_hello, -6);   // wor
```

(...)

# Llamar a una función desde otra

(...)

```
declare v_hello varchar(30);  
declare v_result varchar(30);
```

Se coloca el nombre de la función y los  
parámetros.

Sirve tanto para funciones nativas  
como para funciones creadas por el  
usuario

```
set v_hello := 'hello world!';  
set v_result := substr(v_hello, 1, 5); // hello  
set v_result := substr(v_hello, 7);    // world!  
set v_result := substr(v_hello, -6);   // world!  
set v_result := substr(v_hello, -6);   // wor
```

(...)



# *¿Cómo probar una función?*

```
select fn_my_function();
```

```
select fn_my_function(x);
```

```
select fn_my_function(x, y);
```



# RESUMEN HASTA AHORA

delimiter	Modificar el delimitador por defecto ( ; )
declare	Crear variables con su tipo de dato
set	Asignar valor a variable
create function	Crear una función
return	Indicar a la función que debe retornar un valor
returns	Especifica que tipo de dato debe retornar la función
deterministic	Indica que la función es determinística
if / elseif / else / end if	Condicionales
case / when / else / end case	Condicionales



# EJERCICIOS

Escribir una función en pl/sql con las siguientes características:

- Nombre: fn\_addition\_01
- Parámetros: valor inferior, valor\_superior (p\_min, p\_max)
- Tipo de dato de retorno: int
- Descripción: Realizar la suma de p\_min y p\_max
- Ejemplo: Para los valores de entrada 1 y 10 el cálculo sería  $1+10 = 11$



# EJERCICIOS

Escribir una función en pl/sql con las siguientes características:

- Nombre: fn\_triangle\_area
- Parámetros: lado, altura (p\_side, p\_height)
- Tipo de dato de retorno: real
- Descripción: Calcular el área de un triángulo, la cual se hace con la fórmula  $(base * altura) / 2$
- Ejemplo: Para los valores de entrada 3 y 5, el valor de retorno es 7.5



# EJERCICIOS

Escribir una función en pl/sql con las siguientes características:

- Nombre: fn\_arithmetic\_operation\_01
- Parámetros: valor 1, valor 2, operación (p\_val1, p\_val2, p\_operation)
- Tipo de dato de retorno: real
- Descripción: Realizar la operación especificada en "p\_operation" para los números p\_val1 y p\_val2. Los posibles valores para p\_operation son: "addition", "subtraction", "multiplication", "division". Si se especifica cualquier otro valor entonces la función debe retornar 0.
- Ejemplo: Para los valores de entrada 1, 10, "addition" debe retornar 11. Para los valores 10, 5, "subtraction" debe retornar 5.





# EJERCICIOS

Escribir una función en pl/sql con las siguientes características:

- Nombre: fn\_number\_type\_01
- Parámetros: número (p\_number)
- Tipo de dato de retorno: varchar
- Descripción: Si el número de entrada es par debe retornar "EVEN", en caso contrario debe retornar "ODD".
- Ejemplo: Para el valor 2 debe retornar EVEN. Para el valor 3, ODD.



# EJERCICIOS

Escribir una función en pl/sql con las siguientes características:

- Nombre: fn\_is\_triangle\_area\_even\_01
- Parámetros: número (p\_side, p\_height)
- Tipo de dato de retorno: varchar
- Descripción: Si el área del triángulo es un número par, debe retornar "YES". En caso contrario, debe retornar "NO". Debe apoyarse en las funciones de los ejercicios 1 y 3.
- Ejemplo: Para los valores de entrada 3 y 5, el valor de retorno es NO



# ¿PREGUNTAS?

