

# PROCEDIMIENTOS ALMACENADOS / FUNCIONES

Sistemas de Bases de Datos 1

# PROCEDIMIENTOS ALMACENADOS

Es un programa o conjunto de sentencias SQL que son almacenados en una base de datos para su uso posterior.

Ventajas:

- ❑ Mayor Seguridad
- ❑ Reutilización de código
- ❑ Reduce trafico de red entre cliente y servidor.
- ❑ Rendimiento Mejorado

Tipos

- ❑ Definidos por el Usuario
- ❑ Del sistema
  - ❑ Ejemplo: Now(), Cast(), Convert()

# SINTAXIS (MySQL)

```
CREATE PROCEDURE sp_name ([parameter[,...]])  
    [characteristic ...] routine_body
```

```
CREATE FUNCTION sp_name ([parameter[,...]])  
    RETURNS type  
    [characteristic ...] routine_body
```

*parameter:*

```
[ IN | OUT | INOUT ] param_name type
```

*type:*

*Any valid MySQL data type*

*characteristic:*

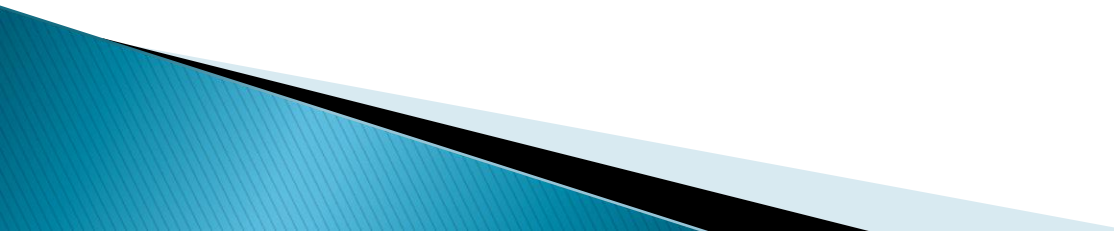
```
    LANGUAGE SQL  
    | [NOT] DETERMINISTIC  
    | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }  
    | SQL SECURITY { DEFINER | INVOKER }  
    | COMMENT 'string'
```

*routine\_body:*

*procedimientos almacenados o comandos SQL válidos*

# INSTRUCCIONES BASICAS

Se pueden ejecutar N cantidad de sentencias en un procedimiento almacenado, desde sentencias DML, sentencias de control de flujo, hasta llamadas a otros procedimientos almacenados.



# DECLARE

Permite declarar variables para almacenar información. Pueden ser de los mismos tipos que los atributos de tablas (int, varchar(50), date).

Sintaxis:

*DECLARE var\_name type [DEFAULT value]*

NOTA: Si no se coloca la clausula DEFAULT, el valor inicial es NULL.

# ASIGNAR VALOR A VARIABLE

Se puede asignar valor a una variable de varias formas.

*DECLARE a int DEFAULT 0;*

Se puede asignar un valor constante.

*SET a = 100;*

O se le puede asignar el resultado de una consulta.

*Select carnet into a from Estudiante LIMIT 1;*



# SENTENCIAS DE CONTROL DE FLUJO

Se puede trabajar con varias sentencias de control de flujo (IF, WHILE, REPEAT, SELECT CASE).

La lógica de estas es la misma que en los lenguajes de programación.



# IF

Instrucción que permite evaluar una o varias condiciones, y ejecutar acciones según el resultado de estas evaluaciones.

Sintaxis:

```
IF search_condition THEN statement_list  
    [ELSEIF search_condition THEN statement_list] ...  
    [ELSE statement_list]  
END IF
```



# WHILE

Una sentencia que define un ciclo, el cual seguirá ejecutándose mientras no se de una condición de salida.

```
[begin_label:] WHILE search_condition DO  
    statement_list  
END WHILE [end_label]
```

# CURSORES

Un cursos es una variable que nos permite recorrer un conjunto de resultados obtenidos a través de una sentencia SELECT fila por fila.

Sintaxis:

```
DECLARE nombre_cursor CURSOR FOR  
sentencia select
```

# USO DE CURSOR

Uso de cursores:

*OPEN nombre\_cursor*

*REPEAT*


*FETCH nombre\_cursor INTO variable;*

*UNTIL done*

*END REPEAT;*

*CLOSE nombre\_cursor;*

**IMPORTANTE:** La o las variables utilizadas en **FETCH** NO debe tener el mismo nombre que los atributos obtenidos del **SELECT** en la asignación del cursor.

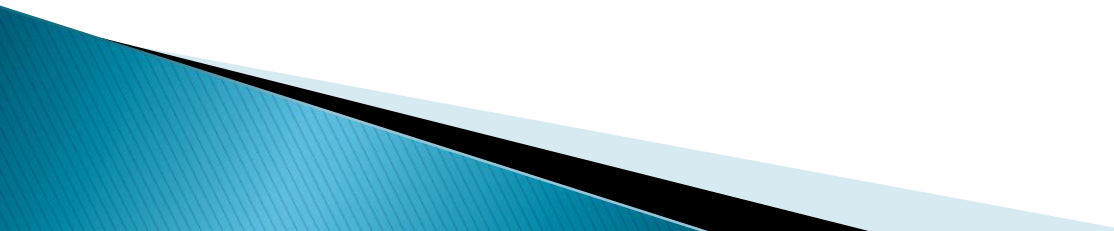


# SENTENCIAS DML

Se pueden ejecutar sentencias DML dentro de un procedimiento almacenado.

Los INSERT, DELETE y UPDATE se ejecutan de igual manera.

Los SELECT nos dan la opción de almacenar el resultado de los mismo en una variable.



# VARIABLES DE ENTRADA/SALIDA

Un procedimiento almacenado puede tener N variables de entrada y N variables de salida.

Sintaxis (MySQL):

```
Create procedure nombre_procedimiento (  
    IN/OUT nombre_variable tipo_variable, ...  
)
```

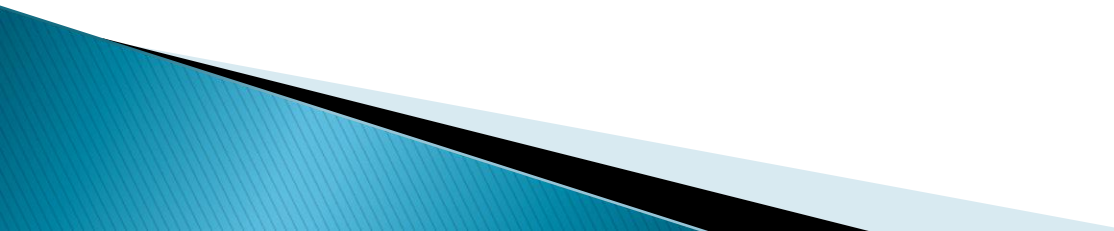
# LLAMADA A PROCEDIMIENTOS

Las llamadas se realizan con la función CALL.

Sintaxis:

```
CALL nombre_procedimiento(  
    valor / variable, ...  
)
```

Si la variable del procedimiento es IN se debe colocar un valor que cumpla con el tipo de dato, si es OUT se debe colocar una variable (local o global) que reciba el valor de retorno.




# FUNCIONES

Igual que un procedimiento almacenado en cuanto a sentencias aceptadas.

No tiene variables IN o OUT, solo tiene N variables de entrada y un tipo de dato a retornar.

Sintaxis:

```
create function nombre_funcion(  
    nombre_variable tipo_variable,  
    ...  
)  
RETURNS tipo_dato
```



# VARIABLE DE RETORNO

Se puede definir uno o mas RETURN, dependiendo de las sentencias de control de flujo utilizadas.

En el return se debe 'devolver' una variable que sea del mismo tipo que el tipo definido en el encabezado de la funcion.

Sintaxis:

*return variable;*





# EJEMPLO RETURN

*Create function nombre\_funcion()*

*RETURNS tipo\_dato1*

*BEGIN*

*IF <condicion> THEN*

*return valor\_uno; /\*Tipo tipo\_dato1\*/*

*ELSE*

*return valor\_dos; /\*Tipo tipo\_dato1\*/*

*END IF;*

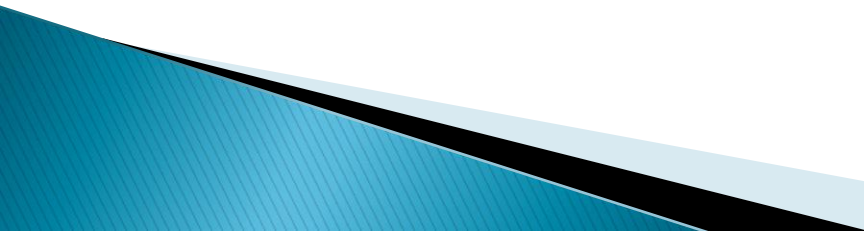
*END*



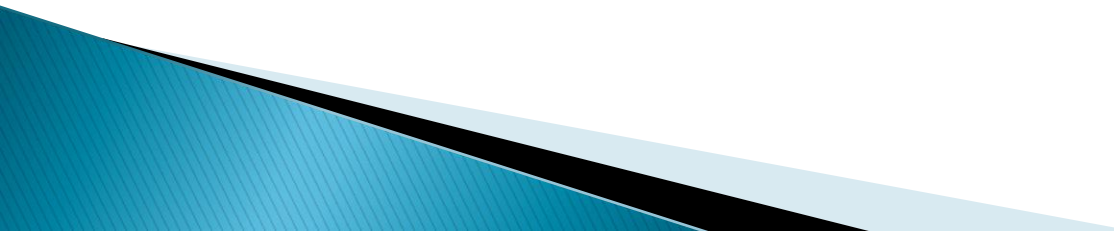
# LLAMADA A FUNCIONES

Las funciones se pueden llamar en sentencias DML.

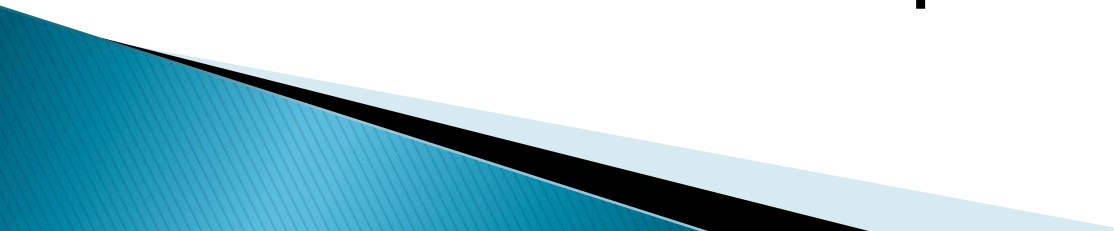
## Sintaxis:

- ❑ *insert into tabla values (funcion(parametro), 1)*
  - ❑ *select funcion(parametro)*
  - ❑ *update tabla set parametro = 1 where atributo = funcion(parametro)*
  - ❑ *delete from tabla where parametro = funcion(parametro)*
- 

# TAREA

- ▶ Crear un Usuario llamado **Emperador** , uno llamado **Vader** y uno llamado **Stormtrooper**.
  - ▶ Dar los permisos necesarios para que el usuario Emperador ejecute una sentencia **SELECT** sobre una tabla determinada ( a discreción del estudiante).
  - ▶ Dar permisos a usuario Vader para ejecutar sentencias **DML** sobre una tabla determinada.
  - ▶ Dar todos los permisos al usuario **Emperador**.
- 

# TAREA

- ▶ Realizar un SELECT sobre la tabla elegida con LOGUEADO con usuario **Stormtrooper**.
  - ▶ Realizar un INSERT sobre la tabla elegida LOGUEADO con usuario **Vader**.
  - ▶ Realizar un ALTER TABLE sobre la tabla elegida con usuario **Emperador**.
  - ▶ Realizar un DROP TABLE sobre la tabla elegida con usuario **Vader**.
  - ▶ Realizar un DELETE sobre la tabla elegida con el usuario **Stormtrooper**.
- 

# TAREA

- ▶ Enviar un screenshot por cada instrucción que se debe ejecutar.
- ▶ En el screenshot se debe poder ver con que usuario ingresaron.
- ▶ Para las sentencias que no son permitidas, desplegar el error respectivo.
- ▶ DBMS → SQL Server
- ▶ Entrega: Lunes 17 de agosto hasta las 11:59 pm
- ▶ Asunto: [BD1]Tarea3
- ▶ Archivo Tarea3\_carnet