



Invoice

Invoice_id
Customer_id
Order_id
Product_id
Date_time
Status
Total
Remark

UT10. SQL MODO PROGRAMACIÓN

Módulo: BASES DE DATOS

Curso 2022/2023. 1º DAM

Ruth Lospitao Ruiz



CONTENIDOS

- Introducción. Conceptos generales
- Variables, operadores, expresiones
- Bloques instrucciones de código
- Estructuras de control.
- Procedimientos y funciones almacenados.
 - Sintaxis y consideraciones generales
 - Comandos para mostrar, modifica y eliminar procedimientos
 - Uso de DDL y DML en procedimientos
- Cursores
- Manejo de errores



SINTAXIS Y CONSIDERACIONES GENERALES



PERMISOS

- Los procedimientos almacenados requieren que la tabla proc en la Base de datos.
- Esta tabla se crea durante la instalación de MySQL 5.9 y versiones posteriores
 - Permiso CREATE ROUTINE se necesita para crear procedimientos almacenados
 - Permiso ALTER ROUTINE se necesita para alterar o borrar procedimientos almacenados. Este permiso se da automáticamente al creador de una rutina.
 - Permiso EXECUTE se requiere para ejecutar procedimientos almacenados. Sin embargo, este permiso se da automáticamente al creador de la rutina. También, la característica SQL SECURITY por defecto para una rutina es DEFINER; lo que permite a los usuarios que tienen acceso a la base de datos ejecutar la rutina asociada.



SINTAXIS

```
CREATE PROCEDURE nombre_procedimiento ([parametro1[,...]])  
  [LANGUAGE SQL]  
  [ [NOT] DETERMINISTIC]  
  [ {CONTAINS SQL | MODIFIES SQL DATA | READS SQL DATA | NO SQL} ]  
  [SQL SECURITY {DEFINER | INVOKER} ]  
  [COMMENT comentario]  
  bloque_de_instrucciones_del_procedimiento
```



ASPECTOS A TENER EN CUENTA

- El nombre del procedimiento cumple con las normas ya expuestas para nombrar variables.
 - Los nombres de scripts no distinguen entre mayúsculas y minúsculas.
 - No puede haber dos scripts con el mismo nombre en la misma base de datos de MySQL
- La lista de argumentos suministrados al procedimiento también se ha expuesto con anterioridad
- La **cláusula LANGUAGE SQL** indica que el lenguaje utilizado en los procedimientos cumple el estándar SQL:PSM. Es una cláusula innecesaria en este momento pues los procedimientos en MySQL sólo soportan este estándar. Si MySQL en un futuro aceptara la escritura de procedimientos almacenados en otros lenguajes como C o Java entonces ya sería necesario indicar el lenguaje de programación utilizado en el procedimiento
- **[NOT] DETERMINISTIC.** Referido al comportamiento del procedimiento. Si un procedimiento es DETERMINISTIC, siempre ante una misma entrada devuelve una misma salida. Funciones numéricas como el valor absoluto, cuadrado, raíz cuadrada son DETERMINISTIC pues siempre devuelven el mismo resultado ante un mismo valor. Por otro lado, una función que devolviera el número de días transcurridos desde 1900 hasta la fecha sería NOT DETERMINISTIC pues cambia según el día que se ejecuta. Por defecto la opción es NOT DETERMINISTIC. Al igual que la anterior **se puede prescindir de su utilización debido a que por el momento no es tomada en cuenta por el servidor.**



ASPECTOS A TENER EN CUENTA

- **[{CONTAINS SQL | MODIFIES SQL DATA | READS SQL DATA | NO SQL}]** Indica el tipo de acceso que realizará el procedimiento a la base de datos, si sólo se va a leer datos se especificará la cláusula READ SQL DATA, si además de ello los modifica entonces la cláusula MODIFIES SQL DATA será la que se emplee. Si el procedimiento no realiza ningún tipo de acceso a la base de datos puede utilizarse la cláusula NO SQL. Por defecto se utiliza la opción CONTAINS SQL que indica que el procedimiento contiene consultas SQL. Estos parámetros se utilizan para mejorar el rendimiento.
- **[SQL SECURITY {DEFINER | INVOKER}]** Indica si el procedimiento almacenado se ejecutará con los permisos del usuario que lo creó (DEFINER) o con los permisos del usuario que llama al procedimiento (INVOKER) La opción por defecto es DEFINER. Hay que tener muy en cuenta que con la opción DEFINER un usuario que lance el procedimiento podrá acceder a los datos aunque no posea los privilegios sobre las tablas que almacenan dichos datos; se trata de un mecanismo de acceso a los procedimientos sin dar directamente acceso a los datos.
- **[COMMENT comentario]** Comentario sobre el procedimiento que puede ayudar al usuario a conocer/recordar el funcionamiento del procedimiento. Esta información puede consultarse como se ha visto anteriormente en el diccionario de datos. Puede prescindirse de dicha cláusula y realizar la escritura de los comentarios aclaratorios al principio del bloque de instrucciones del procedimiento utilizando los caracteres /* */ o --



SINTAXIS DE PROCEDIMIENTO ALMACENADO

```
DELIMITER //  
  CREATE PROCEDURE miProcedimiento (IN id INT, ...)  
  BEGIN  
    SELECT * FROM miTabla WHERE miCampo=id;  
  END //  
DELIMITER ; ← Espacio entre "DELIMITER" y ";"
```

```
CALL miProcedimiento (parámetro);  
SOURCE miArchivoDeProcedimientos.sql;  
SHOW CREATE PROCEDURE miProcedimiento;
```

```
DROP PROCEDURE miProcedimiento;
```

→ Los parámetros pueden ser IN, OUT o INOUT



SINTAXIS DE FUNCIÓN ALMACENADA

```
DELIMITER //  
CREATE FUNCTION miFuncion (id INT) RETURNS FLOAT  
BEGIN  
    DECLARE resultado FLOAT DEFAULT 0;  
    SET resultado=id*2;  
    RETURN resultado;  
END //  
DELIMITER ; ← Espacio entre "DELIMITER" y ";"
```

```
SELECT miFuncion(3);  
SOURCE miArchivoDeFunciones.sql;  
SHOW CREATE FUNCTION miFuncion;
```

```
DROP FUNCTION miFuncion;  
→ Los parámetros sólo pueden ser IN
```



SCRIPTS ALMACENADOS

Procedimientos almacenados	Funciones almacenadas
CREATE PROCEDURE	CREATE FUNCTION
Se invocan con el comando CALL	Se pueden utilizar en expresiones así como en el interior de sentencias SQL
Tienen parámetros IN, OUT e INOUT	Sólo tienen parámetros IN
Pueden devolver varios valores a través de sus parámetros OUT e INOUT	Devuelven valores a través de la instrucción RETURN
Pueden llamar a otras rutinas almacenadas	Pueden llamar a otras rutinas almacenadas
Están asociados a una base de datos concreta	Están asociados a una base de datos concreta



COMANDOS PARA MOSTRAR, MODIFICAR ELIMINAR PROCEDIMIENTOS



CONOCER INFORMACIÓN DE LOS PROCEDIMIENTOS ALMACENADOS

Sintaxis:

```
SHOW {PROCEDURE | FUNCTION} STATUS [LIKE patrón]
```

■ Ejemplo

- 1 • `use curso;`
- 2 • `show procedure status;`
- 3

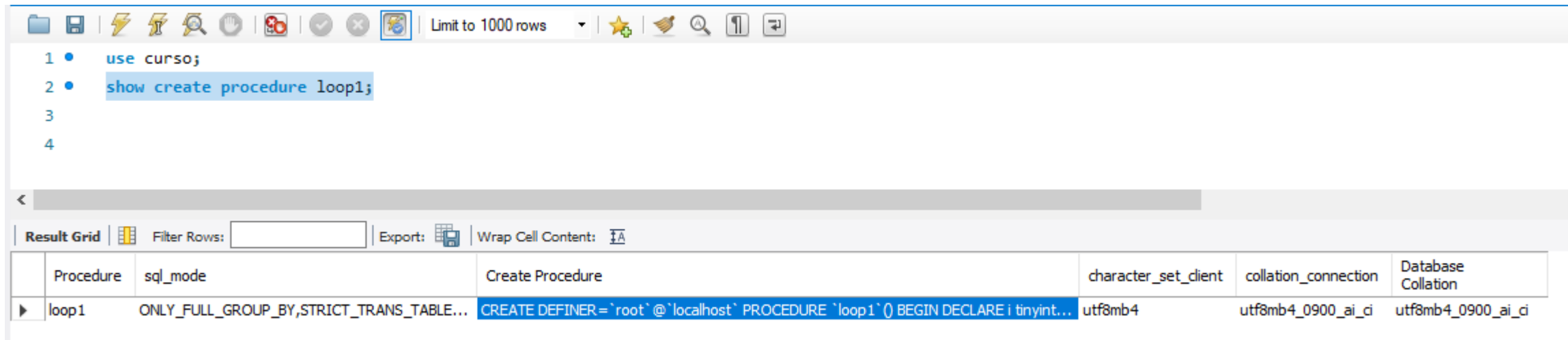
	Db	Name	Type	Definer	Modified	Created	Security_ty	Commer	character_set_client	collation_co
▶	curso	bloque1	PROCEDURE	root@localhost	2023-05-04 12:08:33	2023-05-04 12:08:33	DEFINER		utf8mb4	utf8mb4_09i
	curso	bloque2	PROCEDURE	root@localhost	2023-05-04 12:10:33	2023-05-04 12:10:33	DEFINER		utf8mb4	utf8mb4_09i
	curso	bloque3	PROCEDURE	root@localhost	2023-05-04 12:16:35	2023-05-04 12:16:35	DEFINER		utf8mb4	utf8mb4_09i
	curso	budesanidados	PROCEDURE	root@localhost	2023-05-04 12:56:40	2023-05-04 12:56:40	DEFINER		utf8mb4	utf8mb4_09i
	curso	case01	PROCEDURE	root@localhost	2023-05-04 12:31:38	2023-05-04 12:31:38	DEFINER		utf8mb4	utf8mb4_09i
	curso	ejemplo1	PROCEDURE	root@localhost	2023-05-03 12:46:07	2023-05-03 12:46:07	DEFINER		utf8mb4	utf8mb4_09i
	curso	example_variable02	PROCEDURE	root@localhost	2023-05-04 11:19:29	2023-05-04 11:19:29	DEFINER		utf8mb4	utf8mb4_09i
	curso	example_variables	PROCEDURE	root@localhost	2023-05-04 11:13:41	2023-05-04 11:13:41	DEFINER		utf8mb4	utf8mb4_09i
	curso	example_varUser	PROCEDURE	root@localhost	2023-05-04 11:29:46	2023-05-04 11:29:46	DEFINER		utf8mb4	utf8mb4_09i
	curso	example_varUser2	PROCEDURE	root@localhost	2023-05-04 11:36:59	2023-05-04 11:36:59	DEFINER		utf8mb4	utf8mb4_09i
	curso	HolaMundo	PROCEDURE	root@localhost	2023-05-04 10:56:14	2023-05-04 10:56:14	DEFINER		utf8mb4	utf8mb4_09i
	curso	if01	PROCEDURE	root@localhost	2023-05-04 12:22:12	2023-05-04 12:22:12	DEFINER		utf8mb4	utf8mb4_09i
	curso	if02	PROCEDURE	root@localhost	2023-05-04 12:25:32	2023-05-04 12:25:32	DEFINER		utf8mb4	utf8mb4_09i

Result 1 x

CONOCER INFORMACIÓN DE LOS PROCEDIMIENTOS ALMACENADOS

- Ver información de los procedimientos y funciones así como las líneas de código

```
SHOW CREATE [PROCEDURE | FUNCTION] nombre
```



The screenshot shows a MySQL IDE interface. The top toolbar includes icons for file operations, execution, and search. Below the toolbar, a SQL editor contains the following code:

```
1 • use curso;  
2 • show create procedure loop1;  
3  
4
```

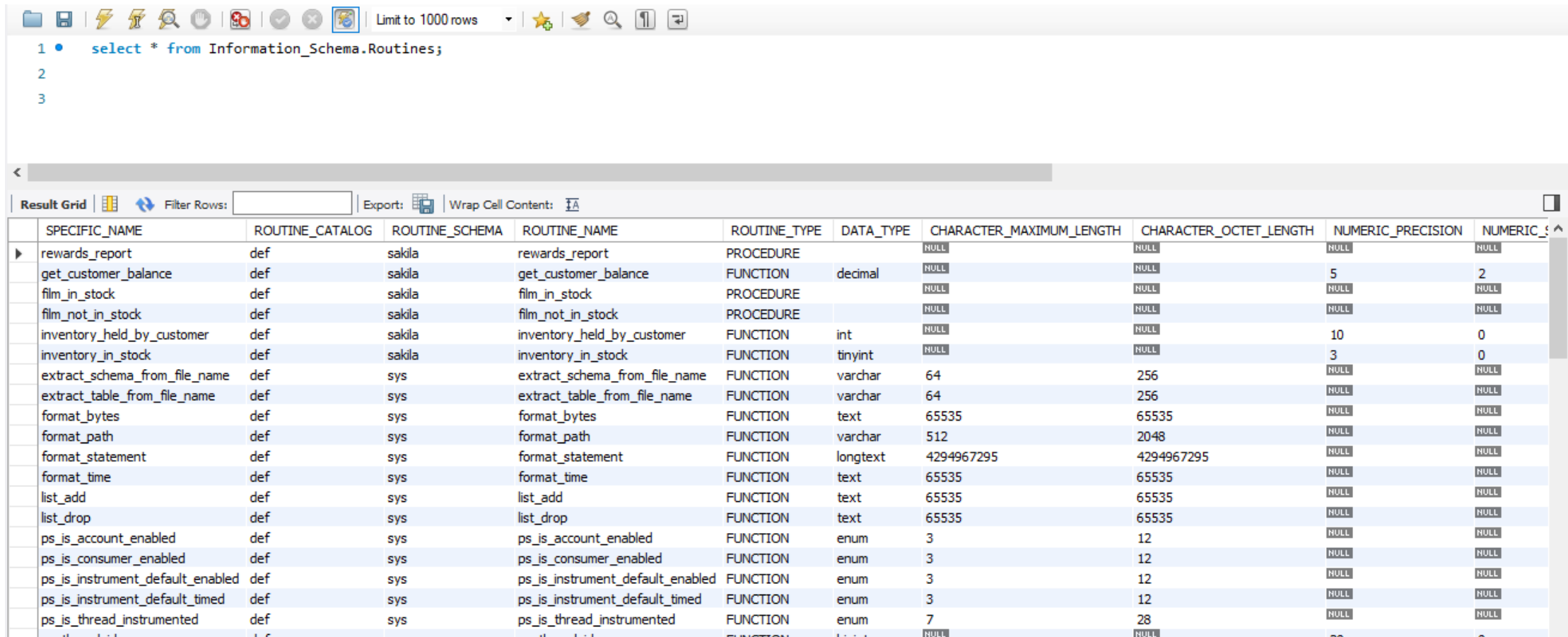
The bottom section displays the 'Result Grid' with a table of results. The table has columns for Procedure, sql_mode, Create Procedure, character_set_client, collation_connection, and Database Collation. The first row shows the details for the 'loop1' procedure.

Procedure	sql_mode	Create Procedure	character_set_client	collation_connection	Database Collation
loop1	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	CREATE DEFINER='root'@"localhost" PROCEDURE `loop1`() BEGIN DECLARE i tinyint...	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci



CONOCER INFORMACIÓN DE LOS PROCEDIMIENTOS ALMACENADOS

- Otra forma de consultar es con `SELECT * FROM Information.Schema.Routines;`



The screenshot shows a database client interface with a SQL editor at the top containing the query `select * from Information_Schema.Routines;`. Below the editor, the 'Result Grid' displays the query results as a table with 11 columns: `SPECIFIC_NAME`, `ROUTINE_CATALOG`, `ROUTINE_SCHEMA`, `ROUTINE_NAME`, `ROUTINE_TYPE`, `DATA_TYPE`, `CHARACTER_MAXIMUM_LENGTH`, `CHARACTER_OCTET_LENGTH`, `NUMERIC_PRECISION`, and `NUMERIC_SCALE`. The table lists various routines from the 'sakila' and 'sys' schemas, including procedures like `rewards_report` and `film_in_stock`, and functions like `get_customer_balance` and `inventory_held_by_customer`. It also includes system routines like `extract_schema_from_file_name` and `format_bytes`.

SPECIFIC_NAME	ROUTINE_CATALOG	ROUTINE_SCHEMA	ROUTINE_NAME	ROUTINE_TYPE	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH	CHARACTER_OCTET_LENGTH	NUMERIC_PRECISION	NUMERIC_SCALE
rewards_report	def	sakila	rewards_report	PROCEDURE		NULL	NULL	NULL	NULL
get_customer_balance	def	sakila	get_customer_balance	FUNCTION	decimal	NULL	NULL	5	2
film_in_stock	def	sakila	film_in_stock	PROCEDURE		NULL	NULL	NULL	NULL
film_not_in_stock	def	sakila	film_not_in_stock	PROCEDURE		NULL	NULL	NULL	NULL
inventory_held_by_customer	def	sakila	inventory_held_by_customer	FUNCTION	int	NULL	NULL	10	0
inventory_in_stock	def	sakila	inventory_in_stock	FUNCTION	tinyint	NULL	NULL	3	0
extract_schema_from_file_name	def	sys	extract_schema_from_file_name	FUNCTION	varchar	64	256	NULL	NULL
extract_table_from_file_name	def	sys	extract_table_from_file_name	FUNCTION	varchar	64	256	NULL	NULL
format_bytes	def	sys	format_bytes	FUNCTION	text	65535	65535	NULL	NULL
format_path	def	sys	format_path	FUNCTION	varchar	512	2048	NULL	NULL
format_statement	def	sys	format_statement	FUNCTION	longtext	4294967295	4294967295	NULL	NULL
format_time	def	sys	format_time	FUNCTION	text	65535	65535	NULL	NULL
list_add	def	sys	list_add	FUNCTION	text	65535	65535	NULL	NULL
list_drop	def	sys	list_drop	FUNCTION	text	65535	65535	NULL	NULL
ps_is_account_enabled	def	sys	ps_is_account_enabled	FUNCTION	enum	3	12	NULL	NULL
ps_is_consumer_enabled	def	sys	ps_is_consumer_enabled	FUNCTION	enum	3	12	NULL	NULL
ps_is_instrument_default_enabled	def	sys	ps_is_instrument_default_enabled	FUNCTION	enum	3	12	NULL	NULL
ps_is_instrument_default_timed	def	sys	ps_is_instrument_default_timed	FUNCTION	enum	3	12	NULL	NULL
ps_is_thread_instrumented	def	sys	ps_is_thread_instrumented	FUNCTION	enum	7	28	NULL	NULL

CONOCER INFORMACIÓN DE LOS PROCEDIMIENTOS ALMACENADOS

- Ejemplo usando un patrón para afinar el resultado

```
1 • use curso;
2 • show procedure status like 'l%';
3
```

	Db	Name	Type	Definer	Modified	Created	Security_ty	Commer	character_set_client	collation_connection	Datab Collat
►	curso	loop1	PROCEDURE	root@localhost	2023-05-04 12:38:41	2023-05-04 12:38:41	DEFINER		utf8mb4	utf8mb4_0900_ai_ci	utf8mb
	curso	loop2	PROCEDURE	root@localhost	2023-05-04 12:42:54	2023-05-04 12:42:54	DEFINER		utf8mb4	utf8mb4_0900_ai_ci	utf8mb



MODIFICAR PROCEDIMIENTOS ALMACENADOS

Sintaxis:

```
ALTER PROCEDURE nombre_procedimiento  
    {CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA}  
    | SQL SECURITY {DEFINER|INVOKER}  
    | COMMENT comentario
```

- Nota: Se debe poseer el privilegio de ALTER ROUTINE para poder modificar procedimientos y funciones



ELIMINAR PROCEDIMIENTOS ALMACENADOS

Sintaxis:

```
DROP PROCEDURE [IF EXISTS] nombre_procedimiento
```

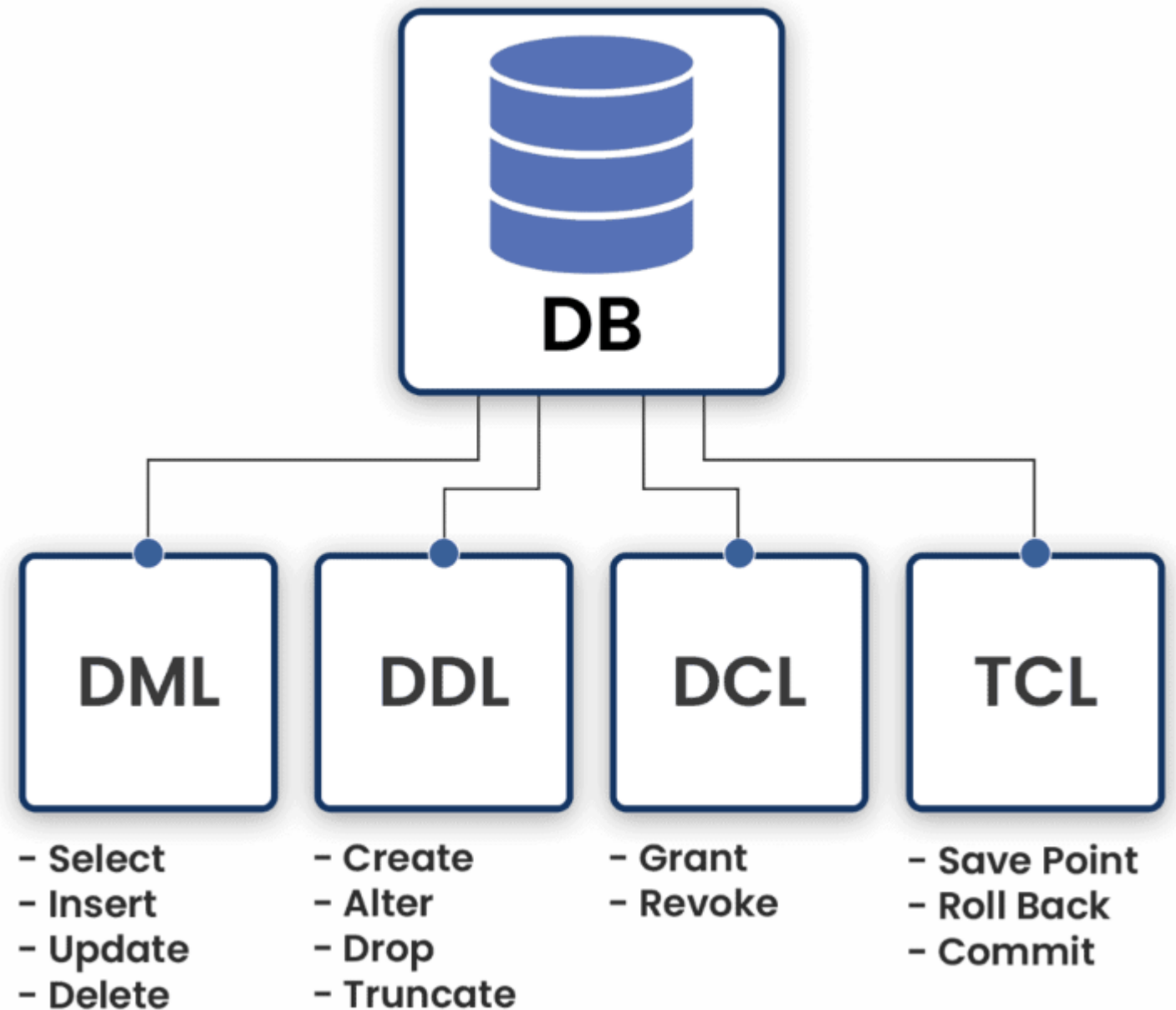
- Nota: se debe poseer el privilegio de ALTER ROUTINE para poder borrar procedimientos y funciones



USO DE DML Y DDL EN PROCEDIMIENTOS





RECORDEMOS



EJEMPLO CON DDL Y DML

```
2 • USE CURSO;
3 • DROP PROCEDURE IF EXISTS procDML;
4 DELIMITER //
5 • CREATE PROCEDURE procDML ()
6 BEGIN
7     DROP TABLE IF exists dptosMadrid;
8     create table dptosMadrid(
9         id int NOT NULL AUTO_INCREMENT primary KEY,
10        dnombre varchar(25),
11        localidad varchar(45)
12    );
13
14    insert into dptosMadrid (DNOMBRE, LOCALIDAD)
15    SELECT DNOMBRE, LOCALIDAD FROM DEPARTAMENTOS WHERE LOCALIDAD='MADRID';
16    SELECT * FROM dptosMadrid;
17 END //
18 DELIMITER ;
19 • call procDML();
20
21
22
```

<			
Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
	id	dnombre	localidad
▶	1	VENTAS	MADRID



GUARDAR EN VARIABLES

```
1  /* Ejemplo guardar registro en variables, cuando es solo uno podemos hacerlo pero si la select nos devuelve más de un registro nos dará error */
2  •  USE CURSO;
3  •  DROP PROCEDURE IF EXISTS procDML2;
4  DELIMITER //
5  •  CREATE PROCEDURE procDML2 ()
6  BEGIN
7      declare dnombre_aux, localidad_aux varchar(25);
8      SELECT DNOMBRE, LOCALIDAD
9      INTO dnombre_aux, localidad_aux
10     FROM DEPARTAMENTOS where localidad='MADRID' ;    -- Nos devolverá solo un registro porque solo tiene uno
11     select dnombre_aux, localidad_aux;
12 END //
13 DELIMITER ;
14 •  call procDML2();
15
16 •  DROP PROCEDURE IF EXISTS procDML3;
17 DELIMITER //
18 •  CREATE PROCEDURE procDML3 () -- Sin embargo, en este procedimiento nos devolverá más de un registro y no podremos guardar en variables
19 BEGIN
20
21     declare dnombre_aux, localidad_aux varchar(25);
22     SELECT DNOMBRE, LOCALIDAD
23     INTO dnombre_aux, localidad_aux
24     FROM DEPARTAMENTOS ;    -- Nos devolverá más de un registro
25     select dnombre_aux, localidad_aux; -- Provocará un error 1172 Result Consisted of more than one row
26 END //
27 DELIMITER ;
28 •  call procDML3();
```

CUIDADO: Cuando la select nos devuelve más de un registro no podremos almacenarlo en variables. Será necesario usar un Cursor.





Invoice

Invoice_id
Customer_id
Order_id
Product_id
Date_time
Status
Total
Remark

UT10. SQL MODO PROGRAMACIÓN

Módulo: BASES DE DATOS

Curso 2022/2023. 1º DAM

Ruth Lospitao Ruiz

