

Procedimientos almacenados (librería):

SP 1: Crear autor

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows the 'libreria_buscable' database with a table 'autor' and a stored procedure 'sp_crearAutor'. The main pane shows the SQL code for creating the procedure. The code includes comments in Spanish and a call to the procedure. The 'Result Grid' at the bottom shows the output of the procedure, which is a list of authors.

Stored Procedures Code:

```
4
5 -- Llamar al procedimiento almacenado utilizando CALL
6 -- CALL sp_crearAutor('6', '1998-02-06', 'Colombiano', 'David Beckham');
7 -- Select * from autor
8 -- Definir el procedimiento almacenado con el delimitador correcto
9 DELIMITER //
10
11 CREATE PROCEDURE sp_crearAutor
12 (IN Id INT,
13  IN FechaNacimiento DATE,
14  IN Nacionalidad NVARCHAR(100),
15  IN Nombre NVARCHAR(100))
16 BEGIN
17     INSERT INTO autor (id, `fecha de nacimiento`, nacionalidad, nombre)
18     VALUES (Id, FechaNacimiento, Nacionalidad, Nombre);
19 END //
20
21 -- Restaurar el delimitador por defecto
22 DELIMITER ;
```

Result Grid:

id	fecha de nacimiento	nacionalidad	nombre
1	1999-01-01	Colombiano	Johan Cifuentes
2	1993-10-15	Peruano	Jorge Rodriguez
3	2001-01-09	Colombiano	David Gomez
4	2000-10-11	Ingles	David White
5	1980-01-01	Americano	Allan Walker
6	1998-02-06	Colombiano	David Beckham

Annotations in the image:

- ejemplo de uso (example of use) pointing to the CALL statement.
- procedimiento almacenado (stored procedure) pointing to the CREATE PROCEDURE statement.
- dato insertado (data inserted) pointing to the row with id 6.
- resultado (result) pointing to the entire result grid.

SP 2: Actualizar autor

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'Schemas' pane shows the 'libreria_buscablibre' database with various tables and views. The 'Stored Procedures' folder is expanded, and 'sp_editarAutor' is highlighted, with a red arrow pointing to it and the text 'sp creado'. The main pane shows the SQL code for the stored procedure, with a red box highlighting the procedure definition and another red box highlighting an example usage comment. The bottom pane shows the 'Result Grid' with a table of authors, where the row for 'David Salazar' is highlighted, and a red box is drawn around it. Red arrows point from the text labels to the corresponding elements in the image.

Stored Procedures

```
19  END //
20
21  -- Restaurar el delimitador por defecto
22  DELIMITER ;
23
24  -- SP 2: EDITAR AUTOR
25  -- Llamar al procedimiento almacenado utilizando CALL
26  -- CALL sp_editarAutor('6', '2003-02-06', 'Peruano', 'David Salazar');
27  -- SELECT * FROM AUTOR
28  -- Definir el procedimiento almacenado con el delimitador correcto
29  DELIMITER //
```

CREATE PROCEDURE sp_editarAutor

```
31  CREATE PROCEDURE sp_editarAutor
32  (
33      IdCompare VARCHAR(10),
34      FechaNacimiento VARCHAR(100),
35      Nacionalidad NVARCHAR(100),
36      Nombre NVARCHAR(100))
37  BEGIN
38      UPDATE autor
39      SET
40          `fecha de nacimiento` = FechaNacimiento, nacionalidad = Nacionalidad, nombre = Nombre
41      WHERE
42          id = IdCompare;
43  END //
```

Result Grid

	id	fecha de nacimiento	nacionalidad	nombre
▶	1	1999-01-01	Colombiano	Johan Cifuentes
	2	1993-10-15	Peruano	Jorge Rodriguez
	3	2001-01-09	Colombiano	David Gomez
	4	2000-10-11	Ingles	David White
	5	1980-01-01	Americano	Allan Walker
	6	2003-02-06	Peruano	David Salazar

SP 3: Consultar autor

hospital

library_db

libreria_busc Libre

Tables

Views

Stored Procedures

sp_buscarAutor

sp_crea Autor

sp_editar Autor

Functions

sakila

sys

world

sp creado

Administration Schemas

Information

Unable to retrieve node description.

```
49  -- =====
50
51  -- SP 3: CONSULTAR AUTOR
52  -- Llamar al procedimiento almacenado utilizando CALL
53  -- CALL sp_buscarAutor('6');
54  -- SELECT * FROM AUTOR
55  -- Definir el procedimiento almacenado con el delimitador correcto
56  DELIMITER //
57
58  CREATE PROCEDURE sp_buscarAutor
59      (IdCompare VARCHAR(10))
60  BEGIN
61      SELECT
62          id
63          ,`fecha de nacimiento`
64          ,nacionalidad
65          ,nombre
66      FROM autor
67      WHERE id = IdCompare;
68  END //
69
70  -- Restaurar el delimitador por defecto
71  DELIMITER ;
72
```

ejemplo de uso

cuerpo sp

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	id	fecha de nacimiento	nacionalidad	nombre
▶	6	2003-02-06	Peruano	David Salazar

resultado

SP 4: Eliminar autor

The screenshot displays the SQL Server Enterprise Manager interface on the left and the SQL Server Enterprise Edition query editor on the right. The Enterprise Manager shows the 'libreria_buscable' database with a table 'autor' and a stored procedure 'sp_eliminarAutor'. The query editor shows the SQL code for creating and executing the stored procedure. The 'Result Grid' shows the current state of the 'autor' table, and the 'Output' window shows the execution results.

SQL Code:

```
-- SP 3: ELIMINAR AUTOR
-- Llamar al procedimiento almacenado utilizando CALL
-- CALL sp_eliminarAutor('6');
-- SELECT * FROM AUTOR
-- Definir el procedimiento almacenado con el delimitador correcto
DELIMITER //

CREATE PROCEDURE sp_eliminarAutor
    (IdCompare VARCHAR(10))
BEGIN
    DELETE FROM autor
    WHERE id = IdCompare;
END //

-- Restaurar el delimitador por defecto
DELIMITER ;
```

Result Grid:

id	fecha de nacimiento	nacionalidad	nombre
1	1999-01-01	Colombiano	Johan Cifuentes
2	1993-10-15	Peruano	Jorge Rodriguez
3	2001-01-09	Colombiano	David Gomez
4	2000-10-11	Ingles	David White
5	1980-01-01	Americano	Allan Walker
6	NULL	NULL	NULL

Output:

#	Time	Action	Message
85	17:44:35	CREATE PROCEDURE sp_buscador (IdCompare VARCHAR(10)) BEGIN SELECT i...	0 row(s) affected
86	17:44:47	CALL sp_editarAutor('6')	Error Code: 1318. Incom...
87	17:45:38	CALL sp_buscador('6')	1 row(s) returned
88	18:02:36	CREATE PROCEDURE sp_buscador (IdCompare VARCHAR(10)) BEGIN DELETE F...	Error Code: 1304. PRO...
89	18:02:47	CREATE PROCEDURE sp_eliminarAutor (IdCompare VARCHAR(10)) BEGIN DELETE ...	0 row(s) affected
90	18:04:15	CALL sp_eliminarAutor('6')	1 row(s) affected
91	18:04:33	SELECT * FROM AUTOR LIMIT 0, 1000	5 row(s) returned

Annotations:

- Unable to retrieve node description. sp creado
- ejemplo de uso
- cuerpo sp
- consulta donde no esta el dato
- resultado ejecutar sp

TRIGGERS:

TRIGGER 1: Control de inserción en tabla autor:

hospital

library_db

libreriaescalibre

Tables

autor

Columns

id

fecha de nacimiento

nacionalidad

nombre

Indexes

Foreign Keys

Triggers

controlInsercionAutor

cliente

control_de_cambios_libreria

editorial

libro

libro_autor

libro_cliente

telefono_cliente

Views

Stored Procedures

sp_buscarAutor

sp_crearAutor

sp_editarAutor

Administration

Schemas

Information

Schema: libreriaescalibre

```
10
11 DELIMITER //
12 -- Creacion 1er Trigger:
13 -- Ejemplo uso:
14 -- CALL sp_crearAutor('6', '1998-02-06', 'Colombiano', 'David Beckham');
15 -- SELECT * FROM control_de_cambios_libreria
16 CREATE TRIGGER controlInsercionAutor
17 AFTER INSERT ON autor
18 FOR EACH ROW
19 BEGIN
20 INSERT INTO control_de_cambios_libreria (usuario, accion, fecha)
21 VALUES (USER(), 'Insert Autor', NOW());
22 END;
23 //
24
25 DELIMITER ;
26
27
28
```

CUERPO TRIGGER

usuario	accion	fecha
root@localhost	Insert Autor	2024-03-27 18:16:07

RESULTADO TABLA DE AUDITORIAS

Output

Action Output

#	Time	Action	Message
90	18:04:15	CALL sp_eliminarAutor('6')	1 row(s) affected
91	18:04:33	SELECT * FROM AUTOR LIMIT 0, 1000	5 row(s) returned
92	18:11:40	USE 'libreriaescalibre'	0 row(s) affected
93	18:12:42	CREATE TABLE control_de_cambios_libreria (usuario VARCHAR(100), accion VAR...	0 row(s) affected
94	18:15:08	-- Creacion 1er Trigger: CREATE TRIGGER controlInsercionAutor AFTER INSERT ON aut...	0 row(s) affected
95	18:16:07	CALL sp_crearAutor('6', '1998-02-06', 'Colombiano', 'David Beckham')	1 row(s) affected
96	18:16:13	SELECT * FROM control_de_cambios_libreria LIMIT 0, 1000	1 row(s) returned

EJECUCION DEL SP

TRIGGER 2: Control de eliminación de autores

The screenshot displays a database IDE interface with the following components:

- Left Panel (Schema Explorer):** Shows the database structure for 'libreria_buscable'. The 'Triggers' folder is expanded, showing 'controlInsercionAutor' and 'controlEliminacionAutor'. A red box highlights 'controlEliminacionAutor' with the annotation 'trigger creado'.
- Top Panel (Code Editor):** Contains the SQL script for creating the trigger. A red box highlights the script with annotations: 'ejemplo de uso' pointing to the example comments and 'cuerpo trigger' pointing to the trigger body. The script is as follows:

```
28 DELIMITER //
29
30 -- Creacion 2er Trigger: Control de eliminación
31 -- Ejemplo uso:
32 -- CALL sp_eliminarAutor('6');
33 -- SELECT * FROM control_de_cambios_libreria
34 CREATE TRIGGER controlEliminacionAutor
35 AFTER DELETE ON autor
36 FOR EACH ROW
37 BEGIN
38     INSERT INTO control_de_cambios_libreria (usuario, accion, fecha)
39     VALUES (USER(), 'Delete Autor', NOW());
40 END;
41 //
42
43 DELIMITER ;
```
- Middle Panel (Result Grid):** Shows the execution results of the trigger. A red box highlights the grid with the annotation 'tabla de auditorias'. The data is as follows:

	usuario	accion	fecha
▶	root@localhost	Insert Autor	2024-03-27 18:16:07
▶	root@localhost	Delete Autor	2024-03-27 18:23:25
- Bottom Panel (Action Output):** Shows the execution log. A red box highlights the log with the annotation 'resultado de ejecuciones'. The log entries are as follows:

#	Time	Action	Message
92	18:11:40	USE 'libreria_buscable'	0 row(s) affected
93	18:12:42	CREATE TABLE control_de_cambios_libreria (usuario VARCHAR(100), accion VARCHAR(100), fecha DATETIME)	0 row(s) affected
94	18:15:08	-- Creacion 1er Trigger: CREATE TRIGGER controlInsercionAutor AFTER INSERT ON autor FOR EACH ROW BEGIN INSERT INTO contro...	0 row(s) affected
95	18:16:07	CALL sp_crearAutor('6', '1998-02-06', 'Colombiano', 'David Beckham')	1 row(s) affected
96	18:16:13	SELECT * FROM control_de_cambios_libreria LIMIT 0, 1000	1 row(s) returned
97	18:22:49	-- Creacion 2er Trigger: Control de eliminación -- Ejemplo uso: -- CALL CALL sp_eliminarAutor('6'); -- SELECT * FROM control_de_cambios_libreria...	0 row(s) affected
98	18:23:07	CALL CALL sp_eliminarAutor('6')	Error Code: 1064. Y
99	18:23:10	SELECT * FROM control_de_cambios_libreria LIMIT 0, 1000	1 row(s) returned
100	18:23:25	CALL sp_eliminarAutor('6')	1 row(s) affected
101	18:23:36	SELECT * FROM control_de_cambios_libreria LIMIT 0, 1000	2 row(s) returned

HOSPITAL

STORED PROCEDURES (SP):

SP 1: Insertar Médico

The screenshot displays a database management interface with a left-hand sidebar showing a tree view of the 'hospital' database. The 'Stored Procedures' folder is expanded, and 'sp_InsertarMedico' is highlighted. The main window shows the SQL script for creating and using the procedure. Red boxes and arrows highlight specific parts of the script and the resulting data.

SQL Script:

```
-- SP 1: CREAMAR MEDICO
-- Llamar al procedimiento almacenado utilizando CALL
-- CALL sp_InsertarMedico(11, 'Julian', 'Alvarez', 'Cardiología');
-- SELECT * FROM Medico
-- Definir el procedimiento almacenado con el delimitador correcto
DELIMITER //

CREATE PROCEDURE sp_InsertarMedico
(IN IdMedico INT,
 IN NombreMedico VARCHAR(100),
 IN ApellidoMedico VARCHAR(100),
 IN Especialidad VARCHAR(50))
BEGIN
    INSERT INTO Medico (IdMedico, NombreMedico, ApellidoMedico, Especialidad)
    VALUES (IdMedico, NombreMedico, ApellidoMedico, Especialidad);
END //

-- Restaurar el delimitador por defecto
DELIMITER ;

-- SP 2: EDITAR AUTOR
-- Llamar al procedimiento almacenado utilizando CALL
```

Result Grid:

IdMedico	NombreMedico	ApellidoMedico	Especialidad
6	Dr. Miguel	Sánchez	Oftalmología
7	Dra. Ana	Fernández	Ginecología
8	Dr. Luis	Hernández	Urología
9	Dra. Sofia	Díaz	Endocrinología
10	Dr. Diogo	Romero	Ortopedia
11	Julian	Alvarez	Cardiología

Action Output:

#	Time	Action	Message
✓ 103	19:09:37	SELECT * FROM Medico LIMIT 0, 1000	10 row(s) returned
✓ 104	19:11:42	CREATE PROCEDURE sp_InsertarMedico ((IN IdMedico INT, IN NombreMedico VARCHAR(100), IN ApellidoMedico VARCHAR(100), ...	0 row(s) affected
✓ 105	19:12:01	CALL sp_InsertarMedico(11, 'Julian', 'Alvarez', 'Cardiología')	1 row(s) affected
✓ 106	19:12:04	SELECT * FROM Medico LIMIT 0, 1000	11 row(s) returned

Annotations in the image:

- dato de ejemplo**: Points to the example data in the CALL statement.
- ejemplo de uso**: Points to the CALL statement.
- cuerpo sp**: Points to the body of the stored procedure.
- dato insertado**: Points to the newly inserted row in the Result Grid.

SP 2: Actualizar medico

The screenshot displays the SQL Server Enterprise Manager interface with the 'Stored Procedures' tasklist selected. The left pane shows the 'hospital' database schema, with 'sp_editarMedico' highlighted under 'Stored Procedures'. A red box around it is labeled 'sp creado'. The main pane shows the SQL script for creating and executing the procedure. Red annotations highlight specific parts: 'Dato de ejemplo' points to the example ID '11', 'Ejemplo de uso' points to the example name 'David', and 'Cuerpo Sp' points to the procedure definition. The 'Result Grid' shows a list of doctors, with the row for 'David Alvarez' (ID 11) highlighted and labeled 'Dato actualizado'. The 'Output' pane shows the execution log, with the call to 'sp_editarMedico' highlighted and labeled 'Ejecución sp'.

Stored Procedures

```
22 DELIMITER ;
23
24 -- =====
25
26 -- SP 2: EDITAR MEDICO
27 -- Llamar al procedimiento almacenado utilizando CALL
28 -- CALL sp_editarMedico(11, 'David', 'Alvarez', 'Cardiología');
29 -- SELECT * FROM Medico
30 -- Definir el procedimiento almacenado con el delimitador correcto
31 DELIMITER //
32
33 CREATE PROCEDURE sp_editarMedico
34 (
35     IN IdMedicoCompare INT,
36     IN NombreMedicoNuevo VARCHAR(100),
37     IN ApellidoMedicoNuevo VARCHAR(100),
38     IN EspecialidadNuevo VARCHAR(50))
39 BEGIN
40     UPDATE Medico
41     SET
42         NombreMedico = NombreMedicoNuevo, ApellidoMedico = ApellidoMedicoNuevo, Especialidad = EspecialidadNuevo
43     WHERE
44         idMedico = IdMedicoCompare;
45 END //
```

Result Grid

IdMedico	NombreMedico	ApellidoMedico	Especialidad
6	Dr. Miguel	Sánchez	Oftalmología
7	Dra. Ana	Fernández	Ginecología
8	Dr. Luis	Hernández	Urología
9	Dra. Sofia	Díaz	Endocrinología
10	Dr. Diego	Romero	Ortopedia
11	David	Alvarez	Cardiología

Output

#	Time	Action	Message
114	19:28:05	CREATE PROCEDURE sp_InsertarMedico (IN IdMedico INT, IN NombreMedico VARCHAR(100), IN ApellidoMedico VARCHAR(100), ...	0 row(s) affected
115	19:28:09	CREATE PROCEDURE sp_editarMedico (IN IdMedicoCompare INT, IN NombreMedicoNuevo VARCHAR(100), IN ApellidoMedicoNuev...	0 row(s) affected
116	19:28:15	CALL sp_editarMedico(11, 'David', 'Alvarez', 'Cardiología')	1 row(s) affected
117	19:28:18	SELECT * FROM Medico LIMIT 0, 1000	11 row(s) returned

SP 3: Consultar Medico:

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'hospital' database is expanded, showing its schema. The 'Stored Procedures' folder is selected, and the 'sp' schema is active. The main pane shows the SQL script for creating and calling the stored procedure 'sp_buscarMedico'. The script is as follows:

```
50
51 -- SP 3: CONSULTAR MEDICO
52 -- Llamar al procedimiento almacenado utilizando CALL
53 -- CALL sp_buscarMedico(6);
54 -- Definir el procedimiento almacenado con el delimitador correcto
55 DELIMITER //
56
57 CREATE PROCEDURE sp_buscarMedico
58 (IdMedicoCompare INT)
59 BEGIN
60     SELECT
61         `IdMedico`
62         , `NombreMedico`
63         , `ApellidoMedico`
64         , `Especialidad`
65     FROM Medico
66     WHERE idMedico = IdMedicoCompare;
67 END //
68
69 -- Restaurar el delimitador por defecto
70 DELIMITER ;
71
72 -- =====
```

Below the script, the 'Result Grid' shows the output of the stored procedure call:

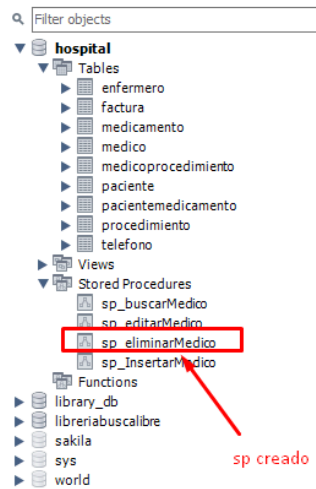
	IdMedico	NombreMedico	ApellidoMedico	Especialidad
▶	6	Dr. Miguel	Sánchez	Oftalmología

The 'Output' pane at the bottom shows the execution log, including the successful execution of the stored procedure call:

#	Time	Action
119	19:34:55	CALL sp_buscarMedico(6)
120	19:35:14	DROP PROCEDURE 'hospital'. 'sp_buscarMedico'
121	19:35:16	CREATE PROCEDURE sp_buscarMedico (IdMedicoCompare INT) BEGIN SELECT 'IdMedico'
122	19:35:21	CALL sp_buscarMedico(6)

Red arrows and labels highlight key elements: 'ejemplo de uso' points to the 'CALL' statement in the script; 'cuerpo' points to the procedure body; 'resultado sp' points to the 'Result Grid'; and 'ejecución sp' points to the execution log entry for the stored procedure call.

SP 4: Eliminar Medico



```
73
74 -- SP 4: ELIMINAR MEDICO
75 -- Llamar al procedimiento almacenado utilizando CALL
76 -- CALL sp_eliminarMedico(11); id de medico a eliminar
77 -- SELECT * FROM Medico
78 -- Definir el procedimiento almacenado con el delimitador correcto
79 DELIMITER //
80
81 CREATE PROCEDURE sp_eliminarMedico
82     (IdMedicoCompare INT)
83 BEGIN
84     DELETE FROM Medico
85     WHERE IdMedico = IdMedicoCompare;
86 END //
87
88 -- Restaurar el delimitador por defecto
89 DELIMITER ;
90
91
92
93
94
95
96
```

IdMedico	NombreMedico	ApellidoMedico	Especialidad
5	Dr. Pablo	Rodríguez	Neurología
6	Dr. Miguel	Sánchez	Oftalmología
7	Dra. Ana	Fernández	Ginecología
8	Dr. Luis	Hernández	Urología
9	Dra. Sofia	Díaz	Endocrinología
10	Dr. Diego	Romero	Ortopedia
* NULL	NULL	NULL	NULL

#	Time	Action	Message
123	19:37:48	SELECT * FROM Medico LIMIT 0, 1000	11 row(s) returned
124	19:38:37	CREATE PROCEDURE sp_eliminarMedico (IdMedicoCompare INT) BEGIN DELETE FROM Medico WHERE IdMedico = IdMedicoCompa...	0 row(s) affected
125	19:38:45	CALL sp_eliminarMedico(11)	elimino un registro 1 row(s) affected
126	19:38:48	SELECT * FROM Medico LIMIT 0, 1000	10 row(s) returned

TRIGGERS HOSPITAL

Creación de la tabla:

```
2
3  -- Creacion de la tabla para las auditorias
4  CREATE TABLE control_de_cambios_hospital(
5      usuario VARCHAR(100),
6      accion VARCHAR(100),
7      fecha DATETIME
8  );
```

TRIGGER 1: Control de inserción tabla medico

```
--
11 DELIMITER //
12 -- Creacion 1er Trigger: Control de inserción
13 -- Ejemplo uso:
14 -- CALL sp_InsertarMedico(11, 'Julian', 'Alvarez', 'Dermatología').
15 -- SELECT * FROM control_de_cambios_hospital
16 CREATE TRIGGER controlInsercionMedico
17 AFTER INSERT ON Medico
18 FOR EACH ROW
19 BEGIN
20     INSERT INTO control_de_cambios_hospital (usuario, accion, fecha)
21     VALUES (USER(), 'Insert Medico', NOW());
22 END;
23 //
```

usuario	accion	fecha
root@localhost	Insert Medico	2024-03-27 20:22:32

Resultado tabla
de auditoria

Output			
Action Output			
#	Time	Action	Message
131	20:22:23	-- Creacion 1er Trigger: Control de inserción -- Ejemplo uso: -- CALL sp_InsertarMedico(11, '...	Error Code: 1359. Trigger already exists
132	20:22:32	CALL sp_InsertarMedico(11, 'Julian', 'Alvarez', 'Dermatología')	1 row(s) affected
133	20:22:37	SELECT * FROM control_de_cambios_hospital LIMIT 0, 1000	1 row(s) returned

ejecución sp de inserción

TRIGGER 2: Control de eliminación de médico

```
29 DELIMITER //
30 -- Creacion 2er Trigger: Control de eliminación
31 -- Ejemplo uso:
32 -- CALL sp_eliminarMedico (11);
33 -- SELECT * FROM control_de_cambios_hospital
34 CREATE TRIGGER controlEliminacionMedico
35 AFTER DELETE ON Medico
36 FOR EACH ROW
37 BEGIN
38     INSERT INTO control_de_cambios_hospital (usuario, accion, fecha)
39     VALUES (USER(), 'Delete Medico', NOW());
40 END;
41 //
```

ejemplo de uso

cuerpo trigger

usuario	accion	fecha
root@localhost	Insert Medico	2024-03-27 20:22:32
root@localhost	Delete Medico	2024-03-27 20:28:38

registro de eliminación

registros de auditoria

control_de_cambios_hospital 2 x

Output

ejecución sp de eliminación

#	Time	Action	Message
134	20:28:32	-- Creacion 2er Trigger: Control de eliminación -- Ejemplo uso: -- CALL sp_eliminarMedico (1...	0 row(s) affected
135	20:28:38	CALL sp_eliminarMedico (11)	1 row(s) affected
136	20:28:41	SELECT * FROM control_de_cambios_hospital LIMIT 0, 1000	2 row(s) returned