



Ivana Sánchez Pérez

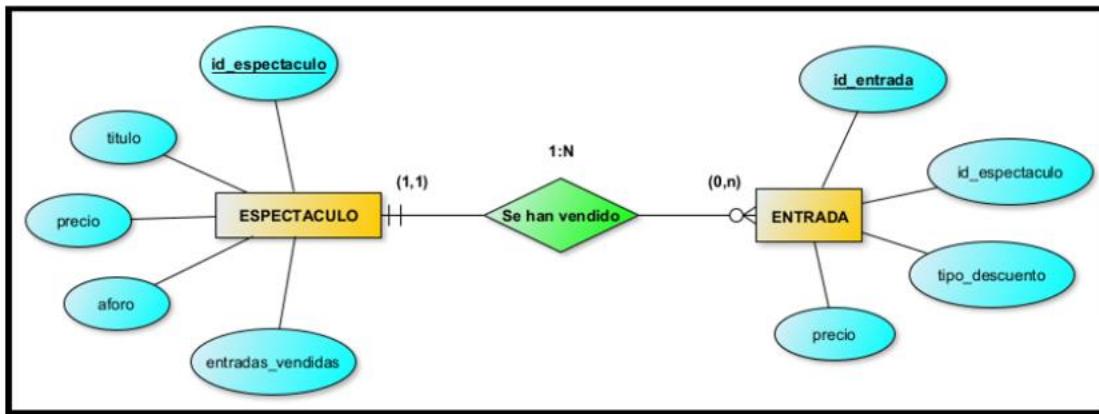
2º ASIR

ÍNDICE

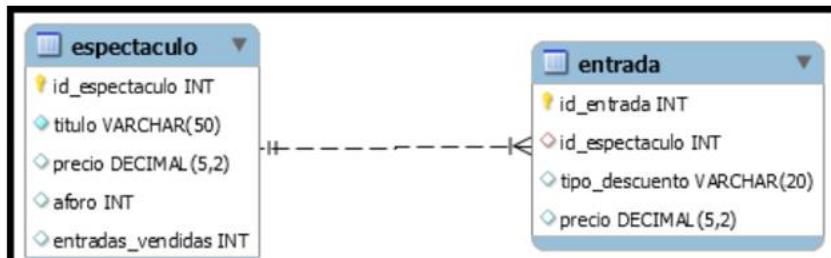
1.- Introducción.....	3
2.- Creación de un trigger.....	3
2.1.- COMPROBACIONES.....	5
a) SELECT DE LA TABLA ESPECTÁCULO.....	5
b) INSERT DE LA TABLA ENTRADA	5
c) NUEVO SELECT DE LA TABLA ESPECTÁCULO	5
3.- Creación de la función “entradas_libres()”	6
3.1.- COMPROBACIONES.....	6
a) SELECT entradas_libres (id_espectaculo).....	6
4.- Creación de la función “precio_espectáculo()”.....	7
4.1.- COMPROBACIONES.....	7
a) SELECT precio_espectaculo(id espectáculo)	7
5.- Procedimiento “venta_entrada”	7
5.1.- COMPROBACIONES.....	8
a) REALIZACIÓN DE TODAS LAS LLAMADAS.....	8
6.- Programar un evento: Recuento diario de caja.....	9
6.1.- COMPROBACIONES.....	10
a) SELECT DE ESPECTÁCULO, ENTRADA Y CAJA ANTES EVENTO	10
b) HACEMOS UN SHOW EVENTS.....	11
c) SELECT DE ESPECTÁCULO, ENTRADA Y CAJA DESPUÉS DEL EVENTO	11
7.- PROPUESTA NUEVA FUNCIONALIDAD.....	12
7.1.- EXPLICACIÓN	12
7.2.- COMPROBACIÓN	15

1.- Introducción

En el presente trabajo se muestra el ejercicio 3 de la asignatura ASGB donde se describe un caso práctico en el que la empresa BK Programación se plantean cómo automatizar la administración sobre el gestor de bases de datos de un proyecto de venta de entradas online según el diagrama-Entidad Relación y modelo relacional siguiente:



Sebastián López. Diagrama E/R sistema de venta de entradas online (CC BY)



2.- Creación de un trigger

El trigger que vamos a crear, actualizará la columna de `entradas_vendidas` en la tabla `espectáculo` cada vez que se inserte una nueva entrada en la tabla `entrada`, incrementando el contador correspondiente al espectáculo al que pertenece la nueva entrada.

Comenzamos descargando el archivo .txt que se nos proporciona en el enunciado de la tarea y, cambiándole la extensión a sólo .sql. entramos en el terminal de MYSQL y lo importamos para crear nuestra base de datos. Hacemos Commit; para guardar.

```

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> DROP DATABASE venta_de_entradas;
Query OK, 2 rows affected (0.21 sec)

mysql> CREATE DATABASE venta_de_entradas;
Query OK, 1 row affected (0.00 sec)

mysql> USE venta_de_entradas;
Database changed
mysql>
mysql> CREATE TABLE espectaculo (
    ->   id_espectaculo INTEGER AUTO_INCREMENT,
    ->   titulo VARCHAR(50) NOT NULL,
    ->   precio DECIMAL(5,2),
    ->   aforo INTEGER,
    ->   entradas_vendidas INTEGER,
    ->   PRIMARY KEY (id_espectaculo)
    -> )ENGINE=InnoDB;
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> CREATE TABLE entrada (
    ->   id_entrada INTEGER AUTO_INCREMENT,
    ->   id_espectaculo INTEGER,
    ->   tipo_descuento VARCHAR(20),
    ->   precio DECIMAL(5,2),
    ->   PRIMARY KEY (id_entrada)
    -> )ENGINE=InnoDB;
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> ALTER TABLE entrada
    -> ADD CONSTRAINT entrada_FK_espectaculo
    -> FOREIGN KEY (id_espectaculo) REFERENCES
    -> espectaculo(id_espectaculo)
    -> ON DELETE RESTRICT
    -> ON UPDATE CASCADE;
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> INSERT INTO espectaculo (id_espectaculo,titulo,precio,aforo,entradas_vendidas) VALUES
    -> ('001','Mamma Tuya',25,200,0),
    -> ('002','El Rey Pelón',40,250,0),
    -> ('003','Romea y Julieta',20,10,7);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

	mysql> SELECT * FROM espectaculo;				
	id_espectaculo	titulo	precio	aforo	entradas_vendidas
entrada	1	Mamma Tuya	25.00	200	0
espectaculo	2	El Rey Pelón	40.00	250	0
+-----+-----+-----+-----+-----+	3	Romea y Julieta	20.00	10	7
2 rows in set (0.01 sec)	3 rows in set (0.01 sec)				

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the 'venta_de_entradas' schema, the 'Tables' section contains the 'entrada' table. In the Query Editor (Query 1), the following SQL code is written:

```

1 DELIMITER //
2 • CREATE TRIGGER after_insert_entrada
3 AFTER INSERT ON entrada
4 FOR EACH ROW
5 BEGIN
6     UPDATE espectaculo
7     SET entrada_vendidas = entradas_vendidas + 1
8     WHERE id_espectaculo = NEW.id_espectaculo;
9 END//
10 DELIMITER ;

```

The Output pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
1	22:05:07	CREATE TRIGGER after_insert_entrada AFTER IN...	0 row(s) affected	0.016 sec

In the Triggers tab of the Information pane, a single trigger is listed:

Name	Event	Table	Timing	Created	SQL Mode	Definer	Client Character...	Connection Coll...	Database
after_insert_entrada	INSERT	entrada	AFTER	2024-11-14 22:0...	NO_ZERO_IN_D...	root@localhost	utf8mb4	utf8mb4_genera...	utf8m

2.1.- COMPROBACIONES

a) SELECT DE LA TABLA ESPECTÁCULO

The screenshot shows the MySQL Workbench interface. In the Navigator pane, the 'Schemas' section lists the 'test' schema. In the Result Grid pane, the following data is displayed:

	id_espectaculo	título	precio	aforo	entradas_vendidas
▶	1	Mamma Tuya	25.00	200	0
▶	2	El Rey Pelón	40.00	250	0
▶	3	Romea y Julieta	20.00	10	7
*	HULL	HULL	HULL	HULL	HULL

b) INSERT DE LA TABLA ENTRADA

The screenshot shows the MySQL Workbench interface. In the Navigator pane, the 'Schemas' section lists the 'test' schema. In the SQL File 11+ pane, the following SQL code is run:

```

1 • INSERT INTO entrada (id_espectaculo, tipo_descuento, precio)
2 VALUES (1, 'General', 25.00);
3

```

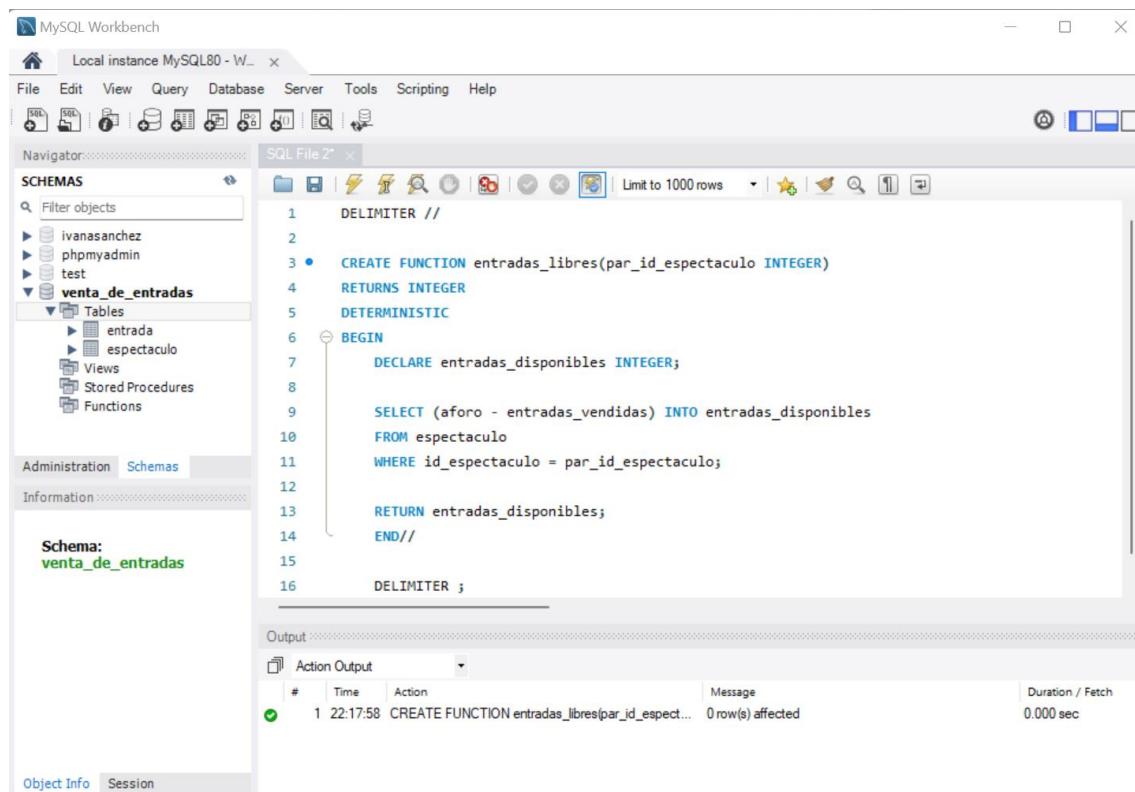
c) NUEVO SELECT DE LA TABLA ESPECTÁCULO

The screenshot shows the MySQL Workbench interface. In the Navigator pane, the 'Schemas' section lists the 'test' schema. In the Result Grid pane, the following data is displayed:

	id_espectaculo	título	precio	aforo	entradas_vendidas
▶	1	Mamma Tuya	25.00	200	1
▶	2	El Rey Pelón	40.00	250	0
▶	3	Romea y Julieta	20.00	10	7
*	HULL	HULL	HULL	HULL	HULL

3.- Creación de la función “entradas_libres()”

Vamos a crear una función llamada entradas_libres que, al tomar como parámetro el ID de un espectáculo, va a devolver el número de entradas libres para dicho espectáculo. El cálculo se basará en el aforo total de éste y la cantidad de entradas ya vendidas.



The screenshot shows the MySQL Workbench interface with the following details:

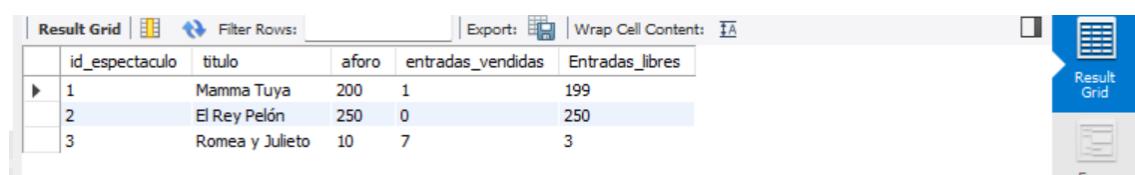
- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Includes icons for SQL, Scripts, Tables, Views, Procedures, Functions, and other database management tools.
- Navigator:** Shows the database schema. Under the 'venta_de_entradas' schema, there are tables 'entrada' and 'espectaculo', and views, stored procedures, and functions.
- SQL Editor:** Titled 'SQL File 2*', contains the SQL code for creating the function:

```
DELIMITER //
CREATE FUNCTION entradas_libres(par_id_espectaculo INTEGER)
RETURNS INTEGER
DETERMINISTIC
BEGIN
    DECLARE entradas_disponibles INTEGER;
    SELECT (aforo - entradas_vendidas) INTO entradas_disponibles
    FROM espectaculo
    WHERE id_espectaculo = par_id_espectaculo;
    RETURN entradas_disponibles;
END//
DELIMITER ;
```
- Output Window:** Titled 'Action Output', shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	22:17:58	CREATE FUNCTION entradas_libres(par_id_espectaculo INTEG... <td>0 row(s) affected</td> <td>0.000 sec</td>	0 row(s) affected	0.000 sec
- Object Info and Session Buttons:** Buttons for Object Info and Session.

3.1.- COMPROBACIONES

a) SELECT entradas_libres (id_espectaculo)



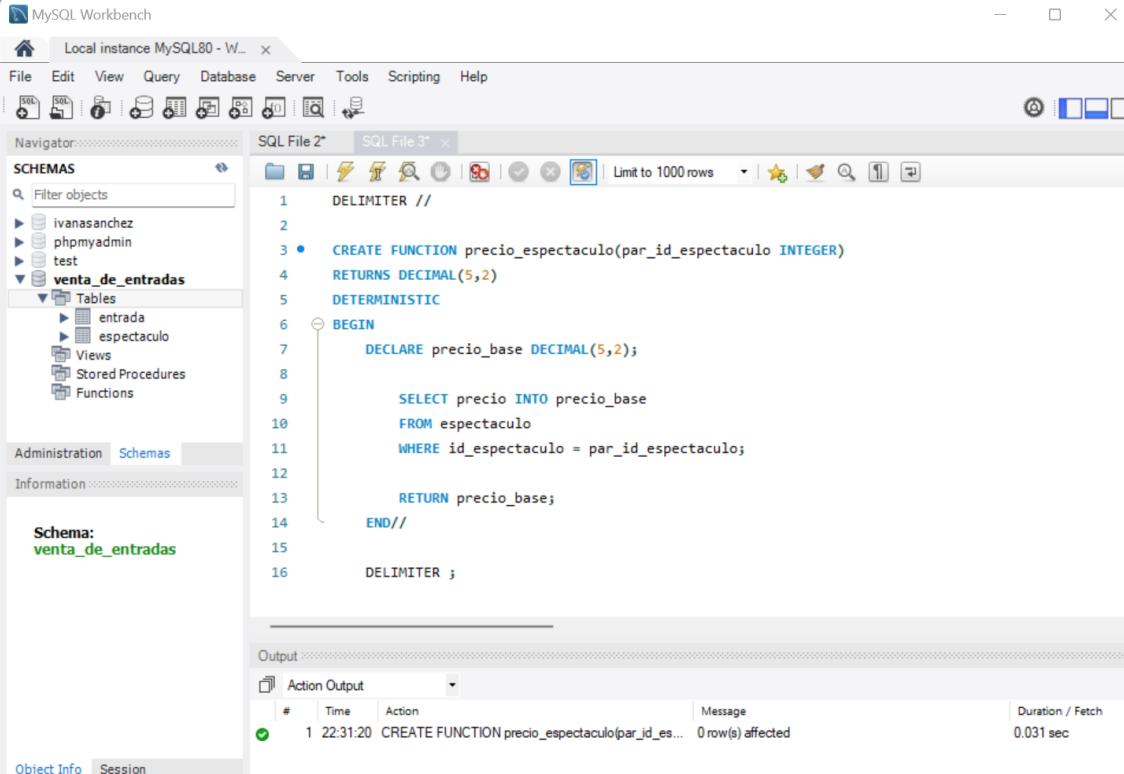
The screenshot shows the results of a query executed in MySQL Workbench:

	id_espectaculo	título	aforo	entradas_vendidas	Entradas_libres
▶	1	Mamma Tuya	200	1	199
	2	El Rey Pelón	250	0	250
	3	Romea y Julieta	10	7	3

On the right side of the results grid, there are two buttons: 'Result Grid' and 'Form'.

4.- Creación de la función “precio_espectáculo()”

Esta función toma como parámetros el ID de un espectáculo y devuelve el precio asociado a ese espectáculo. La función realiza una consulta a la tabla espectáculo para obtener el precio correspondiente al espectáculo identificado por el ID proporcionado.



The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the schema 'venta_de_entradas', there is a 'Tables' folder containing 'entrada', 'espectaculo', 'Views', 'Stored Procedures', and 'Functions'. The 'Functions' item is selected. In the SQL File 2 editor, the following SQL code is written:

```
DELIMITER //
CREATE FUNCTION precio_espectaculo(par_id_espectaculo INTEGER)
RETURNS DECIMAL(5,2)
DETERMINISTIC
BEGIN
    DECLARE precio_base DECIMAL(5,2);

    SELECT precio INTO precio_base
    FROM espectaculo
    WHERE id_espectaculo = par_id_espectaculo;

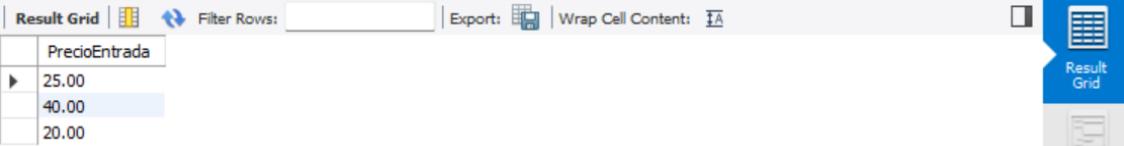
    RETURN precio_base;
END//
DELIMITER ;
```

In the Output pane, a log entry is shown:

#	Time	Action	Message	Duration / Fetch
1	22:31:20	CREATE FUNCTION precio_espectaculo(par_id_es...)	0 row(s) affected	0.031 sec

4.1.- COMPROBACIONES

a) SELECT precio_espectaculo(id espectáculo)



The screenshot shows the Result Grid pane displaying the output of the previous query. The table has one column named 'PrecioEntrada' with three rows containing the values 25.00, 40.00, and 20.00 respectively.

PrecioEntrada
25.00
40.00
20.00

5.- Procedimiento “venta_entrada”

Procedimiento que permite vender entradas para espectáculos, teniendo en cuenta la existencia del espectáculo, la validez del tipo de descuento y la disponibilidad de entradas libres en el aforo del espectáculo.

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'SCHEMAS' section, the 'venta_de_entradas' schema is selected. In the main SQL editor tab, titled 'SQL File 7*', the following SQL code is displayed:

```

1  DELIMITER //
2
3  CREATE PROCEDURE venta_entrada(
4      IN par_id_espectaculo INTEGER,
5      IN par_tipo_descuento VARCHAR(20)
6  )
7  BEGIN
8      DECLARE v_existe_espectaculo INTEGER;
9      DECLARE v_precio_base DECIMAL(5,2);
10     DECLARE v_precio_final DECIMAL(5,2);
11     DECLARE v_entradas_disponibles INTEGER;
12
13     -- Verificar si existe el espectáculo
14     SELECT COUNT(*) INTO v_existe_espectaculo
15     FROM espectaculo
16     WHERE id_espectaculo = par_id_espectaculo;
17

```

In the 'Output' pane at the bottom, there is one entry:

#	Time	Action	Message	Duration / Fetch
1	22:44:54	CREATE PROCEDURE venta_entrada(IN par_id...)	0 row(s) affected	0.000 sec

5.1.- COMPROBACIONES

a) REALIZACIÓN DE TODAS LAS LLAMADAS

- CALL FAMILIA NUMEROSA

Action Output			Message	Duration / Fetch
#	Time	Action		
1	23:51:22	CALL venta_de_entradas(1,"Familia numerosa")	Error Code: 1305. PROCEDURE venta_de_entradas.venta_de_entradas does not exist	0.000 sec

- CALL JUBILADO

Action Output			Message	Duration / Fetch
#	Time	Action		
1	23:49:36	CALL venta_de_entradas(5,"Jubilado")	Error Code: 1305. PROCEDURE venta_de_entradas.venta_de_entradas does not exist	0.000 sec

- CALL GENERAL

Action Output			Message	Duration / Fetch
#	Time	Action		
1	23:54:14	CALL venta_entrada(3,"General")	5 row(s) affected	0.016 sec

- CALL ESTUDIANTE

Action Output			Message	Duration / Fetch
#	Time	Action		
1	23:56:56	CALL venta_entrada(3,"Estudiante")	5 row(s) affected	0.016 sec

- CALL JUBILADO

Action Output			Message	Duration / Fetch
#	Time	Action		
1	23:58:11	CALL venta_entrada(3,"Jubilado")	5 row(s) affected	0.015 sec

- CALL JUBILADO

Action Output		
#	Time	Action
1	23:59:49	CALL venta_entrada(3,"Jubilado")

- CALL JUBILADO

Action Output		
#	Time	Action
1	00:00:56	CALL venta_entrada(1,"Jubilado")

- CALL ESTUDIANTE

Action Output		
#	Time	Action
1	00:02:18	CALL venta_entrada(2,"Estudiante")

6.- Programar un evento: Recuento diario de caja

Como indica el enunciado, hemos creado un evento que se ejecuta cada 24 horas a las 23:30h y que realiza un recuento diario de las entradas vendidas, actualiza la columna entradas_vendidas a cero, elimina todas las filas de la tabla entrada y luego agrega una nueva fila en la tabla llamada caja con la información del recuento diario.

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the schema structure under the 'venta_de_entradas' database, including tables like 'entrada' and 'espectaculo'. The main SQL editor window contains the following code:

```
1 • CREATE TABLE IF NOT EXISTS caja (
2     fecha_apunte DATETIME PRIMARY KEY,
3     entradas_vendidas INT,
4     importe_total DECIMAL(6,2)
5 ) ENGINE=InnoDB;
6
7 -- Habilitar la creación de eventos
8 • SET GLOBAL event_scheduler = ON;
9
10 DELIMITER //
11
12 • -- Crear el evento
13 CREATE EVENT recuento_caja
14 ON SCHEDULE EVERY 1 DAY
15 STARTS CURRENT_DATE + INTERVAL 23 HOUR + INTERVAL 30 MINUTE
16 DO
17 BEGIN
```

The 'Object Info' tab at the bottom shows two actions in the Action Output:

Action Output		
#	Time	Action
5	00:09:37	SET GLOBAL event_scheduler = ON
6	00:09:37	- Crear el evento CREATE EVENT recuento...

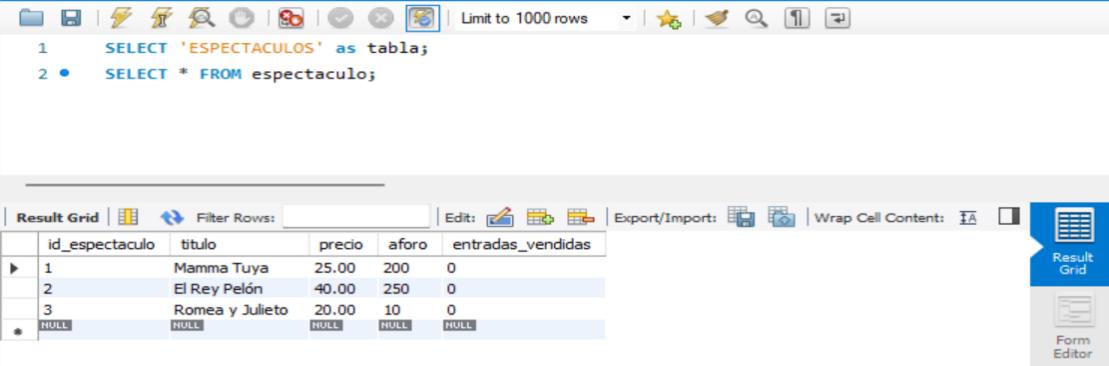
6.1.- COMPROBACIONES

Para hacer las comprobaciones, modificamos el script para que se ejecute cada 5 minutos.



```
1 DELIMITER //
2
3 • DROP EVENT IF EXISTS recuento_caja//
4
5 • CREATE EVENT recuento_caja
6   ON SCHEDULE EVERY 5 MINUTE
7   STARTS CURRENT_TIMESTAMP
8   DO
9     BEGIN
10       DECLARE v_entradas_total INT;
11       DECLARE v_importe_total DECIMAL(6,2);
12
13       -- Calcular totales antes de borrar
14       SELECT COUNT(*), SUM(precio)
15       INTO v_entradas_total, v_importe_total
16       FROM entrada;
17
```

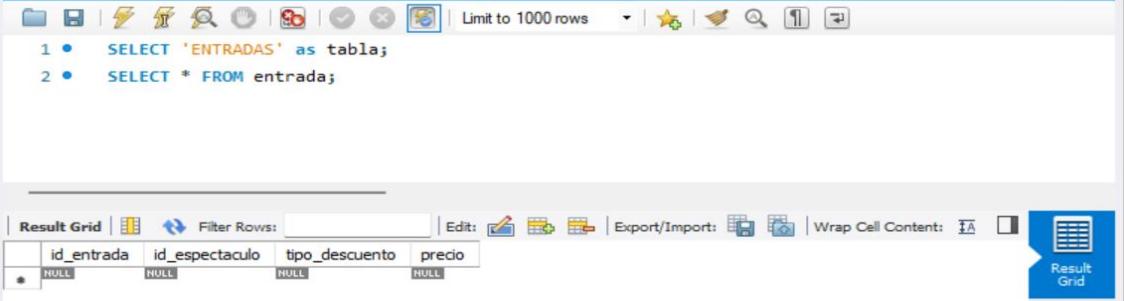
a) SELECT DE ESPECTÁCULO, ENTRADA Y CAJA ANTES EVENTO - ESPECTACULO



```
1 SELECT 'ESPECTACULOS' as tabla;
2 • SELECT * FROM espectaculo;
```

	id_espectaculo	titulo	precio	aforo	entradas_vendidas
▶	1	Mamma Tuya	25.00	200	0
2	El Rey Pelón	40.00	250	0	
3	Romea y Julieta	20.00	10	0	
*	NULL	NULL	NULL	NULL	NULL

- ENTRADAS



```
1 • SELECT 'ENTRADAS' as tabla;
2 • SELECT * FROM entrada;
```

	id_entrada	id_espectaculo	tipo_descuento	precio
*	NULL	NULL	NULL	NULL

- CAJA

The screenshot shows the MySQL Workbench interface. At the top, there is a toolbar with various icons. Below the toolbar, a code editor window displays the following SQL script:

```
1 •  SELECT 'CAJA' as tabla;
2 •  SELECT * FROM caja;
```

Below the script, the results are shown in a "Result Grid". The grid has three columns: "fecha_apunte", "entradas_vendidas", and "importe_total". The data is as follows:

fecha_apunte	entradas_vendidas	importe_total
2024-11-15 00:15:51	7	156.00
2024-11-15 00:20:51	0	0.00
2024-11-15 00:25:51	0	0.00
*	NUL	NUL

On the right side of the results grid, there are two buttons: "Result Grid" and "Form".

b) HACEMOS UN SHOW EVENTS

The screenshot shows the MySQL Workbench interface. At the top, there is a toolbar with various icons. Below the toolbar, a code editor window displays the following SQL script:

```
1 | Result Grid | Filter Rows: | Export: | Wrap Cell Content: | □ | Result Grid | Form
```

Below the script, the results are shown in a "Result Grid". The grid has nine columns: "Db", "Name", "Definer", "Time zone", "Type", "Execute at", "Interval value", "Interval field", and "Result Grid". The data is as follows:

Db	Name	Definer	Time zone	Type	Execute at	Interval value	Interval field	Result Grid
venta_de_entradas	recuento_caja	root@localhost	SYSTEM	RECURRING	NUL	5	MINUTE	

c) SELECT DE ESPECTÁCULO, ENTRADA Y CAJA DESPUÉS DEL EVENTO

Tendremos los mismos resultados, pues no hemos realizado ningún cambio.
Vuelvo a modificar el script para que tenga su configuración de 24 horas.

The screenshot shows the MySQL Workbench interface. At the top, there is a toolbar with various icons. Below the toolbar, a code editor window displays the following SQL script:

```
1     DELIMITER //
2
3 •  DROP EVENT IF EXISTS recuento_caja//
4
5 •  CREATE EVENT recuento_caja
6   ON SCHEDULE EVERY 1 DAY
7   STARTS CURRENT_DATE + INTERVAL 23 HOUR + INTERVAL 30 MINUTE
8   DO
9     BEGIN
10    -- Mismo código del evento...
11  END//
```

Below the script, the results are shown in a "Result Grid". The grid has three columns: "#", "Time", and "Action". The data is as follows:

#	Time	Action
8	00:34:17	DROP EVENT IF EXISTS recuento_caja
9	00:34:17	CREATE EVENT recuento_caja ON SCHED...

At the bottom of the interface, there is an "Output" section with a table titled "Action Output". The table has four columns: "#", "Time", "Action", and "Message". The data is as follows:

#	Time	Action	Message
8	00:34:17	DROP EVENT IF EXISTS recuento_caja	0 row(s) affected
9	00:34:17	CREATE EVENT recuento_caja ON SCHED...	0 row(s) affected

7.- PROPUESTA NUEVA FUNCIONALIDAD

Me he decidido por crear un sistema de reservas anticipadas con gestión de sesiones múltiples por espectáculo. Se trata de una ampliación que permite una gestión de sesiones múltiples (distintas fechas y horas) y llevar un control de ellas (Programada, En venta, Finalizada y Cancelada)

Con este sistema de reservas los clientes podrán reservar entradas proporcionando email y teléfono, tendrán un código único que expirará en 24 horas y, además, contarán con un sistema del estado de su reserva: Pendiente, Confirmada, Cancelada y Completada.

Por último, se ha creado una gestión automática de limpieza de reservas expiradas, de disponibilidad y de generación de códigos de reservas.

7.1.- EXPLICACIÓN

Para que funcione esta ampliación, lo primero que he creado son las 2 tablas que nos van a hacer falta: sesiones y reservas, modificando para ello la tabla entrada para incluir sesiones.

The screenshot shows the MySQL Workbench interface with the SQL editor open. The code in the editor is as follows:

```
12 • CREATE TABLE reservas (
13     id_reserva INTEGER AUTO_INCREMENT,
14     id_sesion INTEGER,
15     email_cliente VARCHAR(100),
16     telefono VARCHAR(15),
17     cantidad_entradas INTEGER,
18     fecha_reserva DATETIME DEFAULT CURRENT_TIMESTAMP,
19     estado ENUM('Pendiente', 'Confirmada', 'Cancelada', 'Completada') DEFAULT 'Pendiente',
20     codigo_reserva VARCHAR(10),
21     fecha_expiracion DATETIME,
22     PRIMARY KEY (id_reserva),
23     FOREIGN KEY (id_sesion) REFERENCES sesiones(id_sesion)
24 ) ENGINE=InnoDB;
25
26 -- Modificar tabla entrada para incluir sesiones
27 • ALTER TABLE entrada ADD COLUMN id_sesion INTEGER;
28 • ALTER TABLE entrada ADD FOREIGN KEY (id_sesion) REFERENCES sesiones(id_sesion);
```

Below the editor, the "Output" pane shows the results of the executed statements:

Action Output
Time Action Message Duration / Fetch
3 01:02:49 ALTER TABLE entrada ADD COLUMN id_sesion ... 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 0.016 sec
4 01:02:49 ALTER TABLE entrada ADD FOREIGN KEY (id_sesion) REFERENCES sesiones(id_sesion) ... 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 0.109 sec

Después de crear las tablas, nos toca los procedimientos para: crear una nueva sesión, validar que el espectáculo existe, validar que la fecha es futura, crear una reserva, obtener id_espectáculo, verificar disponibilidad, generar un código único, crear la reserva por 24 horas, devolver el código de reserva, confirmar una

reserva, obtener datos de la reserva, validaciones, actualizar estado y crear las entradas correspondientes.

```

1  DELIMITER //
2
3 •  -- Procedimiento para crear una nueva sesión
4  CREATE PROCEDURE crear_reserva(
5      IN p_id_espectaculo INTEGER,
6      IN p_fecha_hora DATETIME
7  )
8  BEGIN
9      -- Validar que el espectáculo existe
10     IF NOT EXISTS (SELECT 1 FROM espectaculo WHERE id_espectaculo = p_id_espectaculo) THEN
11         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El espectáculo no existe';
12     END IF;
13
14     -- Validar que la fecha es futura
15     IF p_fecha_hora <= NOW() THEN
16         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'La fecha debe ser futura';
17     END IF;

```

Output:

Action Output	#	Time	Action	Message	Duration / Fetch
	2	01:12:58	CREATE PROCEDURE crear_reserva(IN p...)	0 row(s) affected	0.016 sec
	3	01:12:58	CREATE PROCEDURE confirmar_reserva(...)	0 row(s) affected	0.016 sec

- CONFIRMACIÓN RESERVAS

```

1  DELIMITER //
2
3 •  DROP PROCEDURE IF EXISTS confirmar_reserva//;
4
5  CREATE PROCEDURE confirmar_reserva(
6      IN p_codigo_reserva VARCHAR(10)
7  )
8  BEGIN
9      DECLARE v_id_reserva INTEGER;
10     DECLARE v_id_sesion INTEGER;
11     DECLARE v_cantidad INTEGER;
12     DECLARE v_estado VARCHAR(20);
13     DECLARE v_fecha_exp DATETIME;
14     DECLARE v_id_espectaculo INTEGER;
15     DECLARE v_precio_base DECIMAL(5,2);
16
17     -- Obtener datos de la reserva

```

Output:

Action Output	#	Time	Action	Message	Duration / Fetch
	1	01:38:54	DROP PROCEDURE IF EXISTS confirmar_rese...	0 row(s) affected	0.015 sec
	2	01:38:54	CREATE PROCEDURE confirmar_reserva(I...	0 row(s) affected	0.016 sec

- Modificación del trigger para que respete la variable de control

The screenshot shows the MySQL Workbench interface with a query editor and an output pane.

```

1  DELIMITER //
2
3 •  DROP TRIGGER IF EXISTS after_insert_entrada//
4
5 •  CREATE TRIGGER after_insert_entrada
6    AFTER INSERT ON entrada
7    FOR EACH ROW
8    BEGIN
9      IF @disable_trigger IS NULL THEN
10        UPDATE espectaculo
11          SET entradas_vendidas = entradas_vendidas + 1
12          WHERE id_espectaculo = NEW.id_espectaculo;
13      END IF;
14    END// 
15
16  DELIMITER ;
17

```

Output:

Action Output				
#	Time	Action	Message	Duration / Fetch
1	01:41:37	DROP TRIGGER IF EXISTS after_insert_entrada	0 row(s) affected	0.000 sec
2	01:41:37	CREATE TRIGGER after_insert_entrada AFTE...	0 row(s) affected	0.000 sec

- Por último, crearemos la limpieza de las reservas expiradas

The screenshot shows the MySQL Workbench interface with a query editor and an output pane.

```

1  DELIMITER //
2
3 •  CREATE EVENT limpiar_reservas_expiradas
4    ON SCHEDULE EVERY 1 HOUR
5    DO
6    BEGIN
7      UPDATE reservas
8        SET estado = 'Cancelada'
9        WHERE estado = 'Pendiente'
10       AND fecha_expiracion < NOW();
11    END// 
12
13  DELIMITER ;

```

Output:

Action Output				
#	Time	Action	Message	Duration / Fetch
1	01:14:48	CREATE EVENT limpiar_reservas_expiradas O...	0 row(s) affected	0.016 sec

7.2.- COMPROBACIÓN

- CREAR UNA SESIÓN

The screenshot shows the MySQL Workbench interface. At the top, there is a toolbar with various icons. Below the toolbar, a query window displays the SQL command: `CALL crear_sesion(1, '2024-12-12 20:00:00');`. The status bar at the bottom indicates "Limit to 1000 rows".

Below the query window is the "Action Output" pane. It has columns for #, Time, Action, Message, and Duration / Fetch. One entry is shown:

#	Time	Action	Message	Duration / Fetch
1	01:19:39	CALL crear_sesion(1, '2024-12-12 20:00:00')	1 row(s) affected	0.016 sec

- CREAR UNA RESERVA

The screenshot shows the MySQL Workbench interface. At the top, there is a toolbar with various icons. Below the toolbar, a query window displays the SQL command: `CALL crear_reserva(1, 'cliente@email.com', '123456789', 2);`. The status bar at the bottom indicates "Limit to 1000 rows".

Below the query window is the "Result Grid" pane. It shows a single row of results:

codigo_reserva
ae597d3d

On the right side of the Result Grid pane, there is a "Result Grid" button.

Below the Result Grid is the "Action Output" pane. It has columns for #, Time, Action, Message, and Duration / Fetch. One entry is shown:

#	Time	Action	Message	Duration / Fetch
1	01:47:39	CALL crear_reserva(1, 'cliente@email.com', '12...')	1 row(s) returned	0.000 sec / 0.000 sec

- CONFIRMAR UNA RESERVA

The screenshot shows the MySQL Workbench interface. At the top, there is a toolbar with various icons. Below the toolbar, a query window displays the SQL command: `SELECT codigo_reserva FROM reservas ORDER BY fecha_reserva DESC LIMIT 1;`. The status bar at the bottom indicates "Limit to 1000 rows".

Below the query window is the "Result Grid" pane. It shows a single row of results:

mensaje
Reserva confirmada. Se han generado 2 entradas

On the right side of the Result Grid pane, there is a "Result Grid" button.

Below the Result Grid is the "Action Output" pane. It has columns for #, Time, Action, Message, and Duration / Fetch. One entry is shown:

#	Time	Action	Message	Duration / Fetch
1	01:50:06	CALL confirmar_reserva('ae597d3d')	1 row(s) returned	0.032 sec / 0.000 sec

- VER RESERVAS CONFIRMADAS

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is:

```
1 •  SELECT * FROM reservas;
```

The result grid displays the following data:

	id_reserva	id_sesion	email_cliente	telefono	cantidad_entradas	fecha_reserva	estado	codigo_res
▶	1	1	cliente@email.com	123456789	2	2024-11-15 01:21:33	Confirmada	97d13e63
	2	1	cliente1@email.com	234567890	2	2024-11-15 01:32:07	Confirmada	800449b8
	3	1	cliente2@email.com	56789124	2	2024-11-15 01:34:46	Confirmada	8ad1c092
	4	1	cliente@email.com	123456789	2	2024-11-15 01:47:39	Confirmada	ae597d3d
	5	1	cliente@email.com	123456789	2	2024-11-15 02:14:33	Confirmada	ae071e88
	6	1	cliente@email.com	123456789	2	2024-11-15 02:15:15	Pendiente	9d5ee48a
	7	1	cliente1@email.com	234567891	3	2024-11-15 02:16:03	Pendiente	af72e8f0
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is:

```
1 •  SELECT * FROM reservas WHERE codigo_reserva = 'ae071e88';
```

The result grid displays the following data:

	id_reserva	id_sesion	email_cliente	telefono	cantidad_entradas	fecha_reserva	estado	codigo_res
▶	5	1	cliente@email.com	123456789	2	2024-11-15 02:14:33	Confirmada	ae071e88
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

- VER ENTRADAS GENERADAS

SQL File 47* ×

1 SELECT * FROM entrada;

2

Result Grid | Filter Rows: _____ | Edit: | Export/Import: | Wrap Cell Content: |

	id_entrada	id_espectaculo	tipo_descuento	precio	id_sesion
▶	12	1	General	25.00	1
	13	1	General	25.00	1
	14	3	General	20.00	NULL
	15	3	Estudiante	16.00	NULL
	16	3	Jubilado	18.00	NULL
	17	1	Jubilado	22.50	NULL
	18	2	Estudiante	32.00	NULL
	19	1	General	25.00	1
	20	1	General	25.00	1
*	NULL	NULL	NULL	NULL	NULL

entrad a 1 × Apply Revert

Result Grid

Form Editor

Field Types