Julián Stevan Torrado Velasco
Angela Marcela Soler López
Bases de datos II
Politécnico Internacional

CASO DE ESTUDIO INTEGRADOR

OBJETIVOS CON ESTE TRABAJO:

1. Crea una base de datos para almacenar y organizar datos, definiendo la relación de cada tabla.
2. Crear bases de datos con códigos que sean abiertos para poder usarlos o modificarlos a la necesidad de cada persona, empresa o cliente.
3. Definir los tipos de datos que se tienen y en casos específicos.
4. Practicidad en los procesos de búsqueda.

Construir los modelos que componen la Base de Datos

Diseño del modelo conceptual

Se desea diseñar una base de datos relacional, la cual a información sobre los préstamos de las películas de un vídeo club. En la actualidad la gestión de esta información se lleva cabo del siguiente modo:

Cuando se hace un préstamo se rellena una ficha en la que se anota el socio que se lleva la película, el teléfono, y su dirección. Esta ficha se deposita en el archivador de películas prestadas. Cuando el socio devuelve la cinta, la ficha se pasa al archivador de películas devueltas. El vídeo club tiene, además, un archivador con fichas de películas ordenadas por título; cada ficha tiene además el género de la película (comedia, terror, ...), su director y los nombres de los actores que intervienen.

También se tiene un archivador con las fichas de los Ejercicios de Diseño de Bases de Datos Relacionales Curso 2001/2002 2 socios, ordenadas por el código que el vídeo club les da

cuando les hace el carné; cada ficha tiene el nombre del socio, su dirección y teléfono, los nombres de sus directores favoritos, los nombres de sus actores favoritos y los géneros cinematográficos de su preferencia. Cuando un socio quiere tomar prestada una película de la que no hay copias disponibles, se le puede anotar en la lista de espera de esa película. Cada vez que se devuelve una película, se comprueba si hay alguien en su lista de espera, y si es así se llama por teléfono al primer socio de la lista para decirle que ya puede pasar a recogerla, borrándolo después de la lista.

Se debe iniciar creando la base de datos con sus respectivas tablas que tienen la conexión con película, ya que será quien tendrá las claves foráneas.

SOCIO (Entidad)

- Nombre.
- Número telefónico.
- Dirección.

ACTORES (Entidad)

- Nombre actor.

GENERO (Entidad)

- Terror.
- Drama.
- Comedia.
- Romance.
- Acción.

CLASIFICACIÓN (Entidad)

- Nombre clasificación.

NACIONALIDAD (Entidad)

- Nombre país.

DIRECTOR (Entidad)

- Nombre director.

PREMIO (Entidad)

- Nombre premio.

PELICULA (Entidad)

- Nombre película.
- Nombre premio.
- Nacionalidad.
- Director.
- Genero.
- Clasificación.
- Actor.

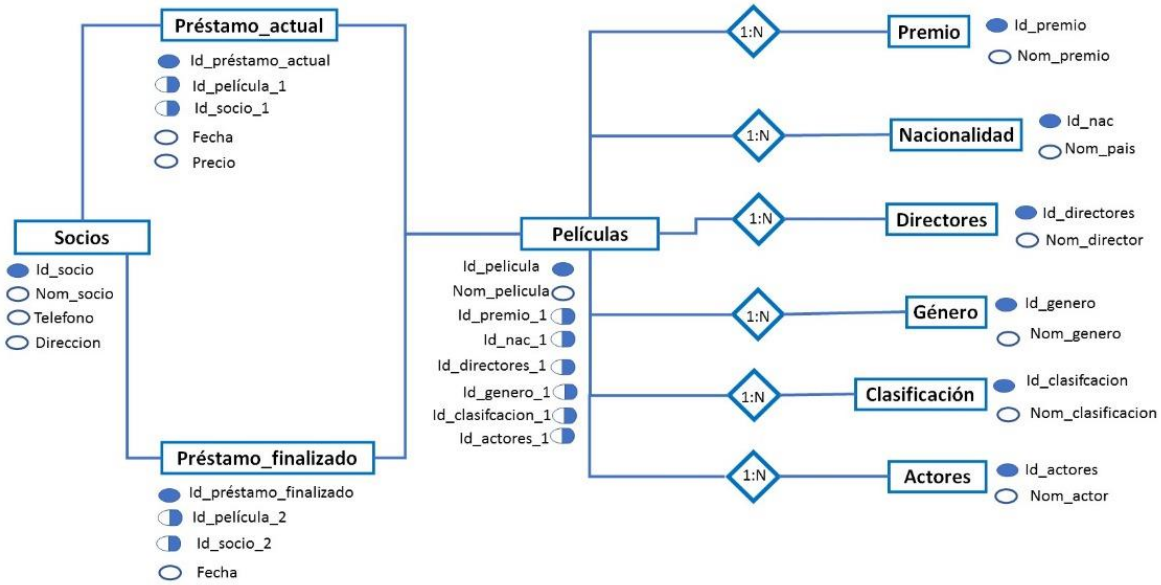
PRESTAMO ACTUAL (Entidad)

- Nombre película.
- Socio.
- Fecha.
- Precio.
- Prestamos actual.

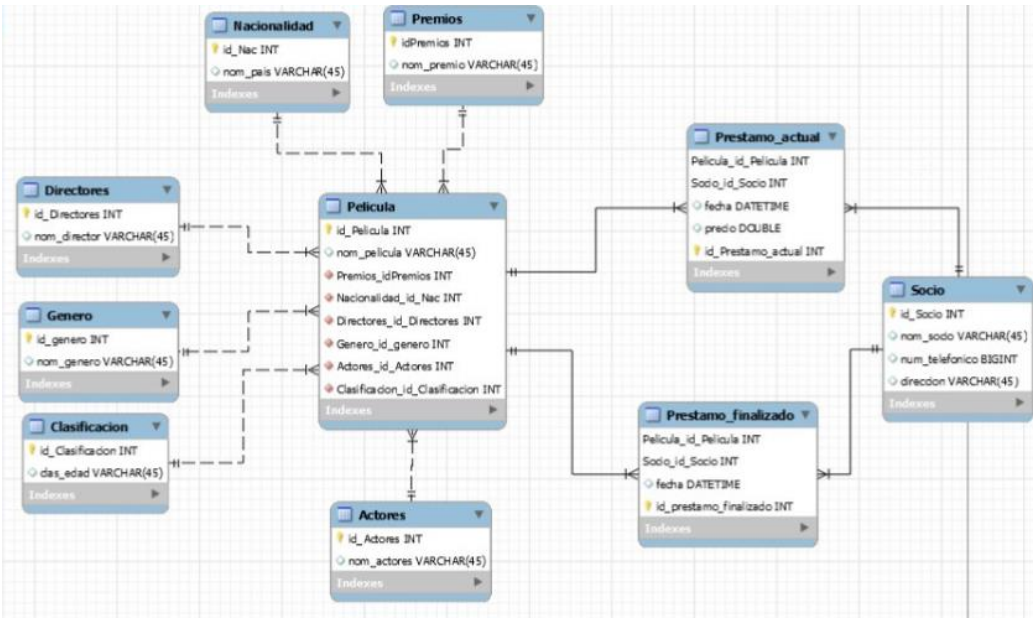
PRESTAMO FINALIZADO (Entidad)

- Nombre película.
- Socio.
- Fecha.
- Préstamo finalizado.

Diseño del modelo entidad relación



Diseño del modelo lógico



Al contener campos repetitivos en películas, como lo son; actor principal, clasificación, genero, director, nacionalidad y premio. Se deberán tener tablas desde donde se envíen claves foráneas a películas y donde se puede referenciar un número indeterminado de veces.

Diseño del modelo físico

```
1 •   CREATE DATABASE Peliculas;
2
3 •   USE Peliculas;
4
5     /* TABLA PREMIOS*/
```

```sql
/* TABLA PREMIOS*/

DROP TABLE IF EXISTS premio;

CREATE TABLE premio (
id_premio int AUTO_INCREMENT NOT NULL,
nom_premio VARCHAR(45),
CONSTRAINT premio_pk PRIMARY KEY (id_premio)
);

/*TABLA NACIONALIDAD*/

DROP TABLE IF EXISTS nacionalidad;

CREATE TABLE nacionalidad (
id_nac int AUTO_INCREMENT NOT NULL,
nom_pais VARCHAR(45),
CONSTRAINT nacionalidad_pk PRIMARY KEY (id_nac)
);

/*DIRECTORES*/


DROP TABLE IF EXISTS director;

CREATE TABLE director (
id_director int AUTO_INCREMENT NOT NULL,
nom_director VARCHAR(45),
CONSTRAINT director_pk PRIMARY KEY (id_director)
);


/*GENERO*/

DROP TABLE IF EXISTS genero;

CREATE TABLE genero (
id_genero int AUTO_INCREMENT NOT NULL,
nom_genero VARCHAR(45),
CONSTRAINT genero_pk PRIMARY KEY (id_genero)
);
```

```
46      /*ACTORES*/
47
48      DROP TABLE IF EXISTS actor;
49
50      CREATE TABLE actor (
51      id_actor int AUTO_INCREMENT NOT NULL,
52      nom_actor VARCHAR(45),
53       CONSTRAINT actor_pk PRIMARY KEY (id_actor)
54      );

56      /*CLASIFICACION*/
57
58      DROP TABLE IF EXISTS clasificacion;
59
60      CREATE TABLE clasificacion (
61      id_clasificacion int AUTO_INCREMENT NOT NULL,
62      nom_clasificacion VARCHAR(45),
63      CONSTRAINT clasificacion_pk PRIMARY KEY (id_clasificacion)
64      );
```

Una vez creadas las tablas con contenido, se procede a realizar la creación de la tabla principal quien es desde la que se tendrá la clave foránea.

```sql
/*PELICULA*/

DROP TABLE IF EXISTS pelicula;

CREATE TABLE pelicula (
id_pelicula INT AUTO_INCREMENT NOT NULL,
nom_pelicula VARCHAR(45),
id_premio_1 int,
id_nac_1 int,
id_director_1 int,
id_genero_1 int,
id_actor_1 int,
id_clasificacion_1 int,
PRIMARY KEY (id_pelicula),
FOREIGN KEY (id_premio_1) REFERENCES premio(id_premio),
FOREIGN KEY (id_nac_1) REFERENCES nacionalidad(id_nac),
FOREIGN KEY (id_genero_1) REFERENCES genero(id_genero),
FOREIGN KEY (id_actor_1) REFERENCES actor(id_actor),
FOREIGN KEY (id_clasificacion_1) REFERENCES clasificacion(id_clasificacion),
FOREIGN KEY (id_director_1) REFERENCES director(id_director)
);
```

Se incluye el socio, el cual solo tiene una clave primaria, esta tabla será de ayuda para la creación de las tablas resultantes.

```sql
/*SOCIO*/

DROP TABLE IF EXISTS socio;

CREATE TABLE socio (
id_socio int AUTO_INCREMENT NOT NULL,
nom_socio VARCHAR(45),
num_telefonico BIGINT,
direccion VARCHAR(45),
 PRIMARY KEY(id_socio)
);
```

Por último, solo queda la realización de las tablas resultantes.

```sql
/*RESULTANTES */

/*PRESTAMO ACTUAL */


DROP TABLE IF EXISTS prestamo_actual;

CREATE TABLE prestamo_actual (
id_prestamo_actual int AUTO_INCREMENT NOT NULL,
id_pelicula_1 int,
id_socio_1 int,
fecha DATETIME,
precio DOUBLE,
PRIMARY KEY (id_prestamo_actual),
FOREIGN KEY (id_pelicula_1) REFERENCES pelicula(id_pelicula),
FOREIGN KEY (id_socio_1) REFERENCES socio(id_socio)
);

 /*PRESTAMO FINALIZADO*/


 DROP TABLE IF EXISTS prestamo_finalizado;

CREATE TABLE prestamo_finalizado (
id_prestamo_finalizado int AUTO_INCREMENT NOT NULL,
id_pelicula_2 int,
id_socio_2 int,
fecha DATETIME,
PRIMARY KEY (id_prestamo_finalizado),
FOREIGN KEY (id_pelicula_2) REFERENCES pelicula(id_pelicula),
FOREIGN KEY (id_socio_2) REFERENCES socio(id_socio)
);
```

Una vez, creadas las tablas pertinentes se procede a realizar las inserciones para más adelante realizar las consultas y mostrar las tablas con su respectivo resultado.

```sql
INSERT INTO premio (nom_premio)
VALUES("Oscar"),
      ("Globo De Oro"),
      ("Directors Guild of America Award"),
      ("Screen Actors Guild Awards"),
      ("New York Film Critics Circle Awards");

INSERT INTO nacionalidad (nom_pais)
VALUES ("Colombia"),
       ("Estados Unidos"),
       ("Mexico"),
       ("Korea"),
       ("Autralia"),
       ("Peru"),
       ("Francia"),
       ("Inglaterra"),
       ("Italia"),
       ("Brasil");

INSERT INTO director(nom_director)
VALUES("Andres"),
      ("Jhon"),
      ("Guillermo"),
      ("Steven"),
      ("George"),
      ("Min ho "),
      ("Scott"),
      ("Peter"),
      ("Stanley"),
      ("Miguel");
```

```sql
INSERT INTO genero(nom_genero)
VALUES ("Terror"),
       ("Drama"),
       ("Comedia"),
       ("Accion"),
       ("Romance"),
       ("Infantil"),
       ("Aventura"),
       ("Suspenso"),
       ("Ciencia ficcion");

INSERT INTO actor(nom_actor)
VALUES("Angelina"),
       ("Tom"),
       ("Ricardo"),
       ("Richard"),
       ("Kim"),
       ("Gary"),
       ("Marlon"),
       ("jhon"),
       ("Daniel"),
       ("Jack");


INSERT INTO clasificacion(nom_clasificacion)
VALUES ("Para todo publico"),
       ("Mayores de quince años"),
       ("mayores de dieciocho");

/*PELICULAS*/
INSERT INTO pelicula(nom_pelicula,id_premio_1,id_nac_1,id_director_1,id_genero_1,id_actor_1,id_clasificacion_1)
VALUES ("IT",1,2,1,1,3,3),
       ("Yo antes de ti",3,5,3,5,1,1),
       ("Tormenta",1,8,8,1,9,3),
       ("Monsters",3,5,3,6,7,1),
       ("Fuerza Elite",3,6,4,4,1,2),
       ("Amelie",2,3,7,9,2,2),
       ("Sueños de libertad",5,2,3,2,4,2),
       ("Tomb rider",1,2,8,7,5,1),
       ("Venganza",3,10,10,9,10,3),
       ("Divergente",4,3,1,4,5,2);
```

```
80      /*SOCIO */
81
82  •   INSERT INTO socio(nom_socio,num_telefonico,direccion)
83      VALUES("Andres","3025632012","cll 1 No. 20-56"),
84            ("Carlos","3052105689","Av 5 No. 30-11"),
85            ("Sandra","2728058956","cra 30 No. 50-150"),
86            ("Diego","4561238256","cll 81 No. 121-50"),
87            ("Zuleima","3694125874","Av 4 No. 21-28"),
88            ("Hernan","3259874563","cra 32 No. 7-15"),
89            ("German","3255236589","cra 15 No. 25-70"),
90            ("Ariel","3597895646","cll 5 No. 10-8"),
91            ("Bianca","3598962356","Av 8 No. 6-11"),
92            ("Carla","3598962356","cra 9 No. 56-120");
```

```
95      /*PRESTAMO ACTUAL */
96
97  •   INSERT INTO prestamo_actual (id_pelicula_1,id_socio_1,fecha,precio)
98      VALUES ("4","1","2019-12-28 00:00:00","10000"),
99             ("1","2","2020-01-05 00:00:00","10000"),
100            ("5","8","2020-01-07 00:00:00","10000"),
101            ("7","5","2020-01-10 00:00:00","10000"),
102            ("3","9","2020-01-13 00:00:00","10000");
103
104 •   INSERT INTO prestamo_finalizado (id_pelicula_2,id_socio_2,fecha)
105     VALUES ("4","1","2020-01-08 00:00:00"),
106            ("1","2","2020-01-15 00:00:00");
```

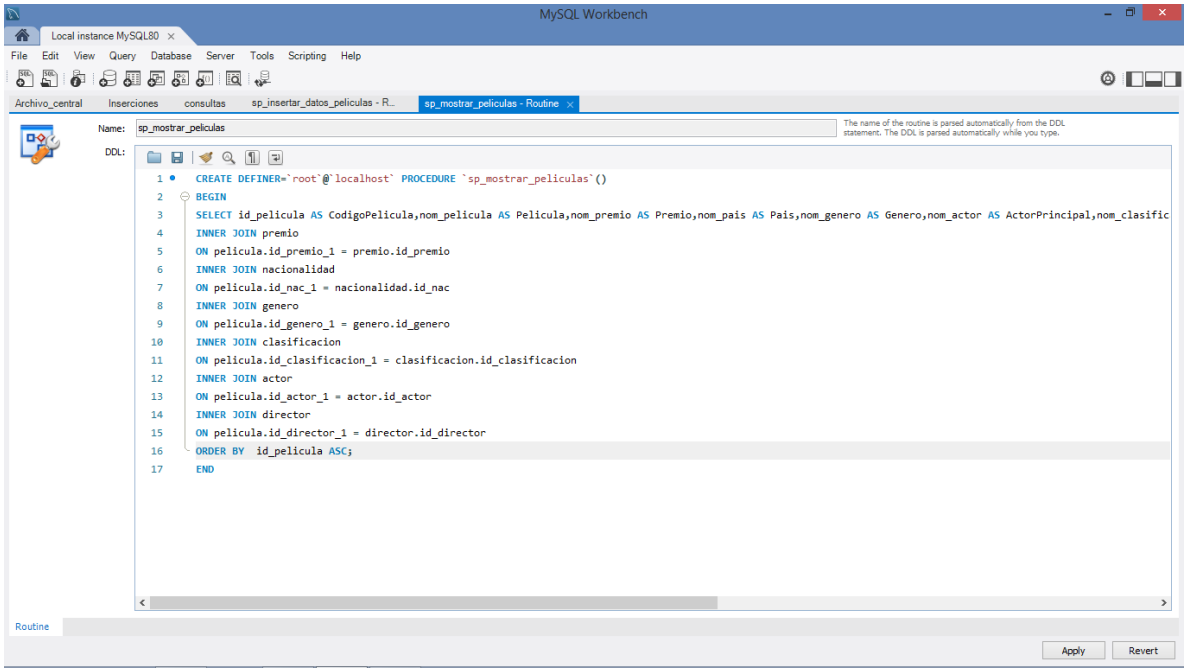
1. Construcción de procedimientos almacenados

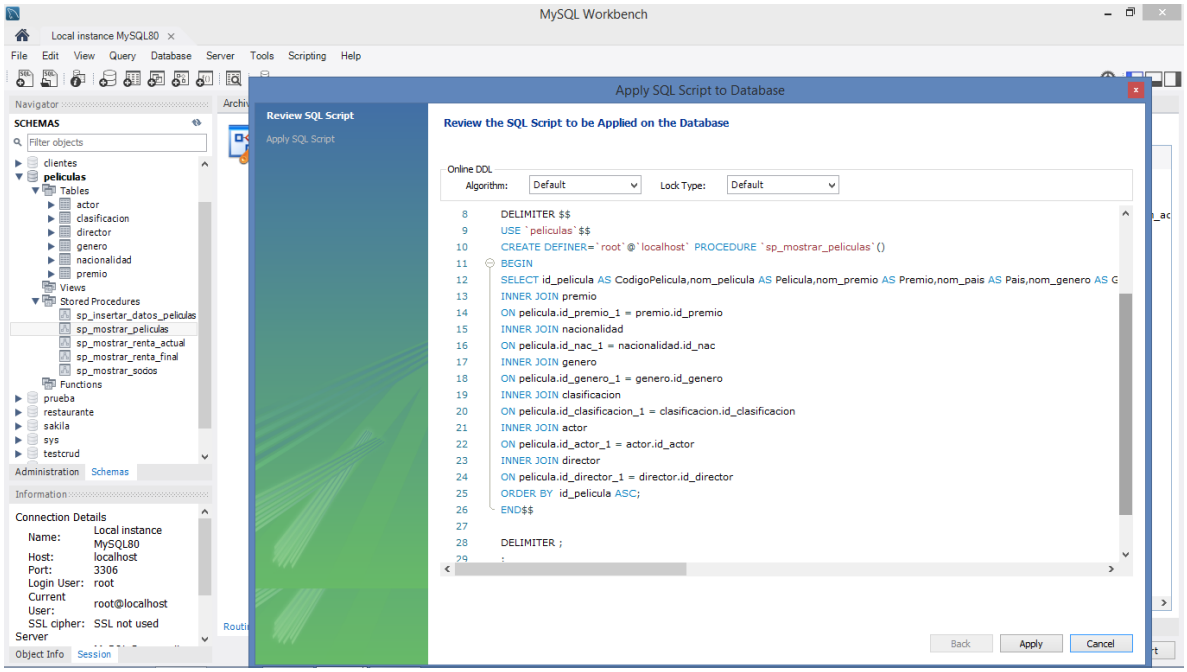
CONSULTAS:

Se utiliza CRUD para cubrir grandes listados, en este caso es una base de datos más pequeña pero que en una tienda real es mucho más amplio.

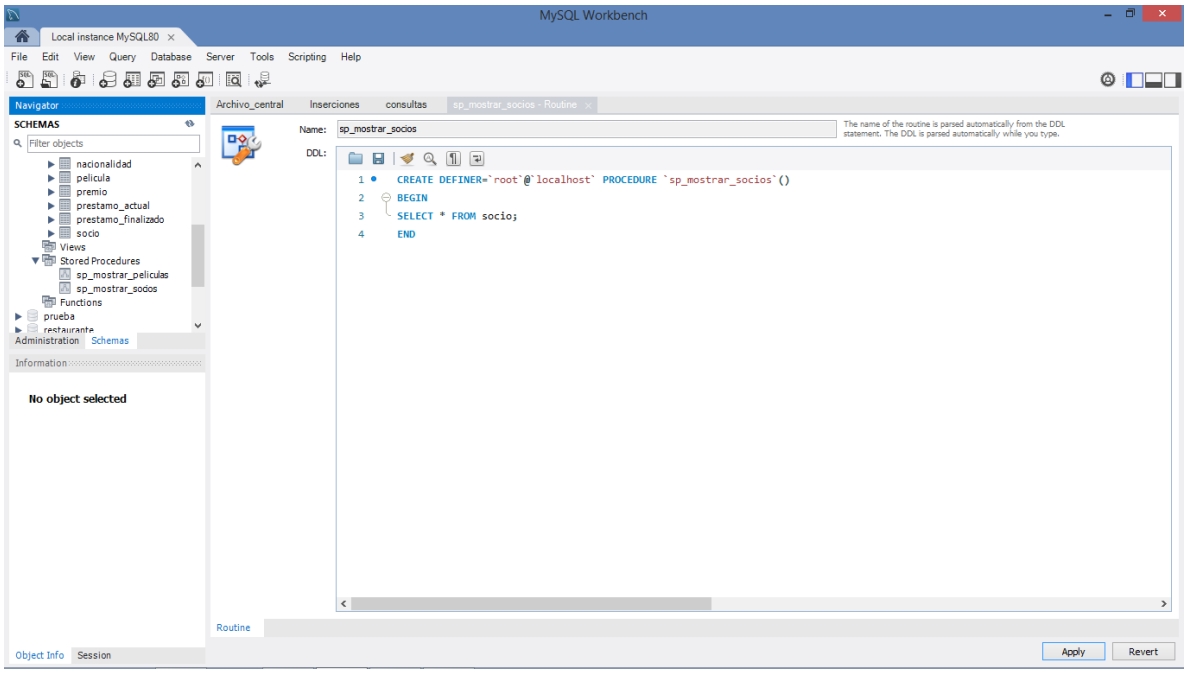
Gracias a esto, más adelante se podrán encontrar cada una de las operaciones que corresponden a CRUD.

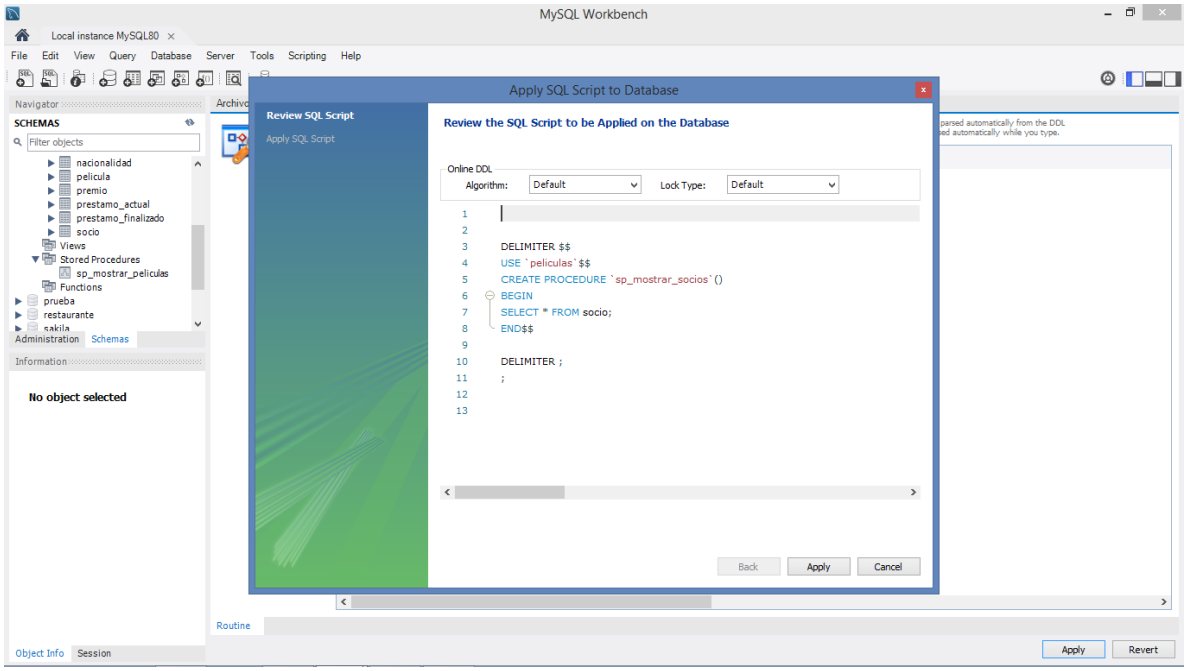
Películas



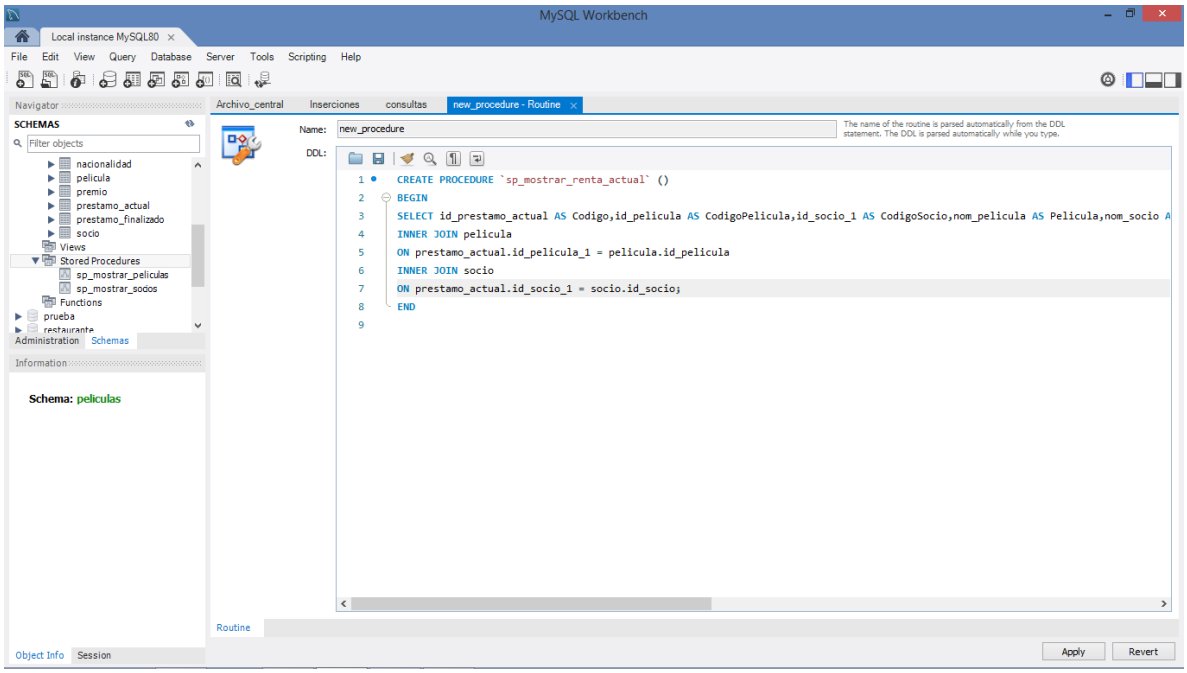


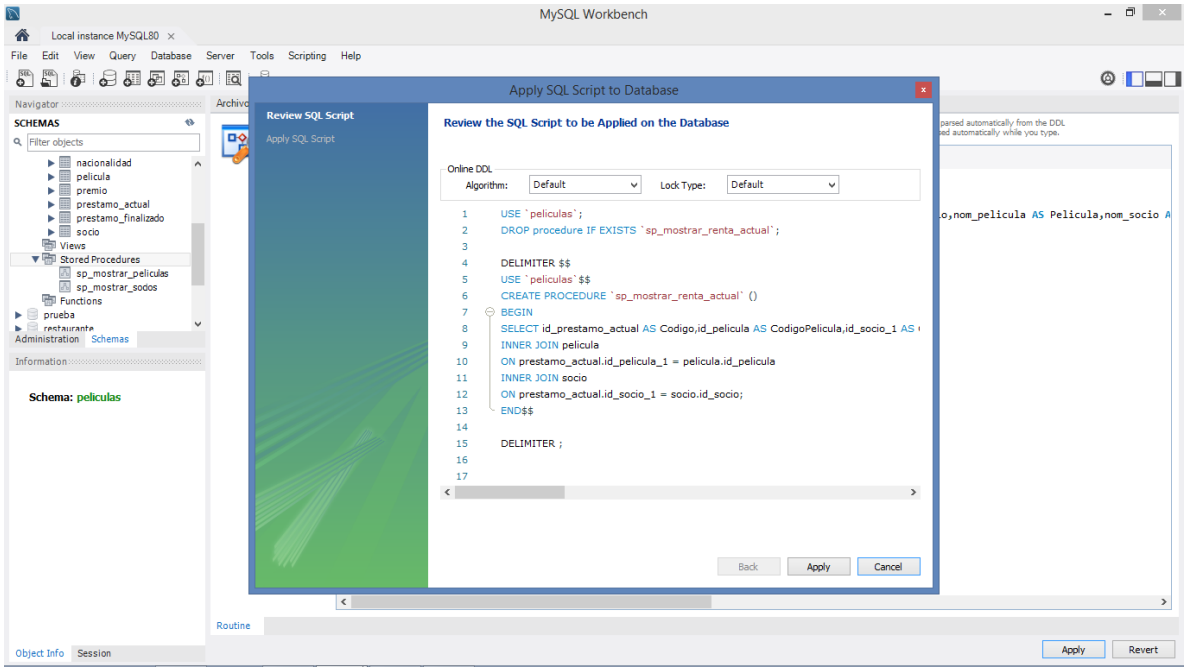
Socios



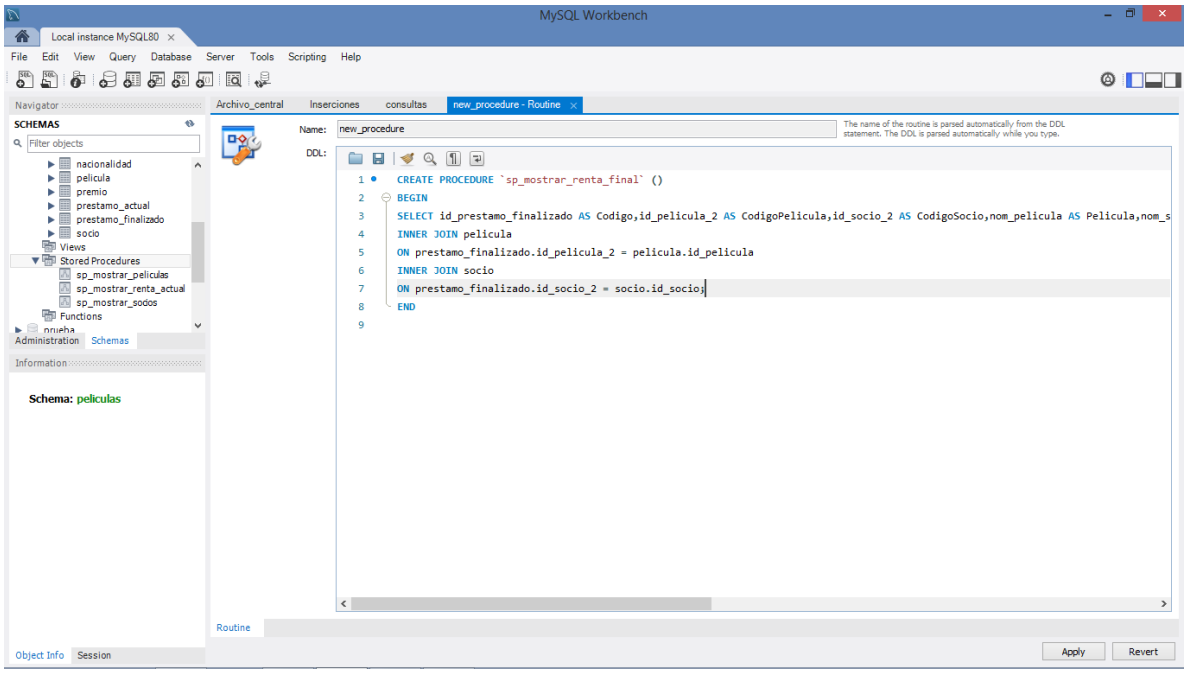


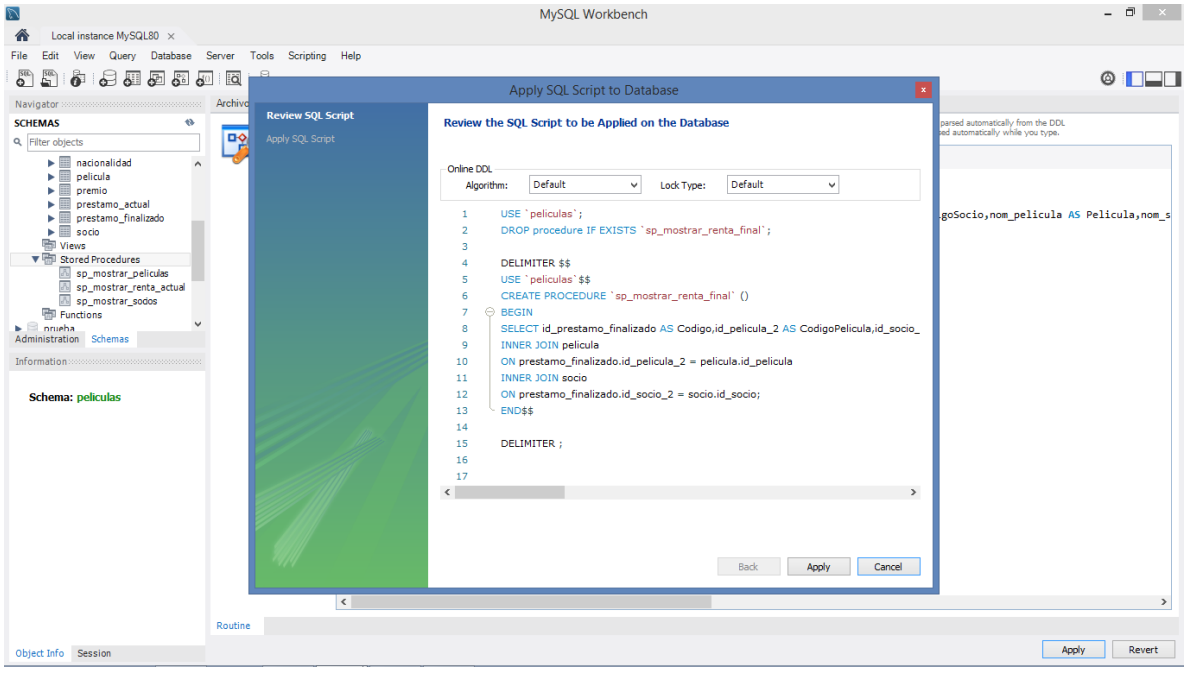
Préstamo actual





Préstamo finalizado:


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
nacionalidad
pelicula
premio
prestamo_actual
prestamo_finalizado
socio
Views
Stored Procedures
sp_mostrar_peliculas
sp_mostrar_renta_actual
sp_mostrar_sodos
Functions
prueba
Administration Schemas
Information
Schema: peliculas
Object Info Session
Archivo_central Inserciones consultas new_procedure - Routine
Name: new_procedure
The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.
DDL:
1 CREATE PROCEDURE `sp_mostrar_renta_final` ()
2 BEGIN
3 SELECT id_prestamo_finalizado AS Codigo,id_pelicula_2 AS CodigoPelicula,id_socio_2 AS CodigoSocio,nom_pelicula AS Pelicula,nom_s
4 INNER JOIN pelicula
5 ON prestamo_finalizado.id_pelicula_2 = pelicula.id_pelicula
6 INNER JOIN socio
7 ON prestamo_finalizado.id_socio_2 = socio.id_socio;
8 END
9
Routine
Apply
Revert

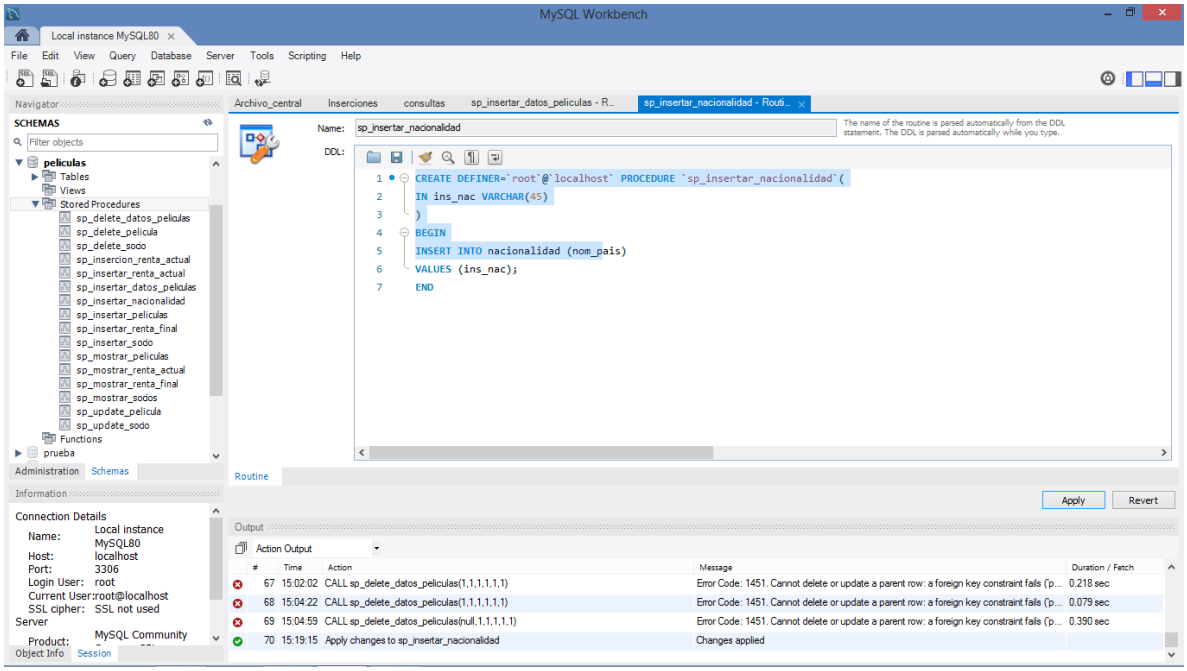

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Apply SQL Script to Database
Review SQL Script
Apply SQL Script
Review the SQL Script to be Applied on the Database
Online DDL
Algorithm: Default
Lock Type: Default
1 USE `peliculas`;
2 DROP procedure IF EXISTS `sp_mostrar_renta_final`;
3
4 DELIMITER $$
5 USE `peliculas`$$
6 CREATE PROCEDURE `sp_mostrar_renta_final` ()
7 BEGIN
8 SELECT id_prestamo_finalizado AS Codigo,id_pelicula_2 AS CodigoPelicula,id_socio_
9 INNER JOIN pelicula
10 ON prestamo_finalizado.id_pelicula_2 = pelicula.id_pelicula
11 INNER JOIN socio
12 ON prestamo_finalizado.id_socio_2 = socio.id_socio;
13 END$$
14
15 DELIMITER ;
16
17
Back
Apply
Cancel
Routine
Apply
Revert

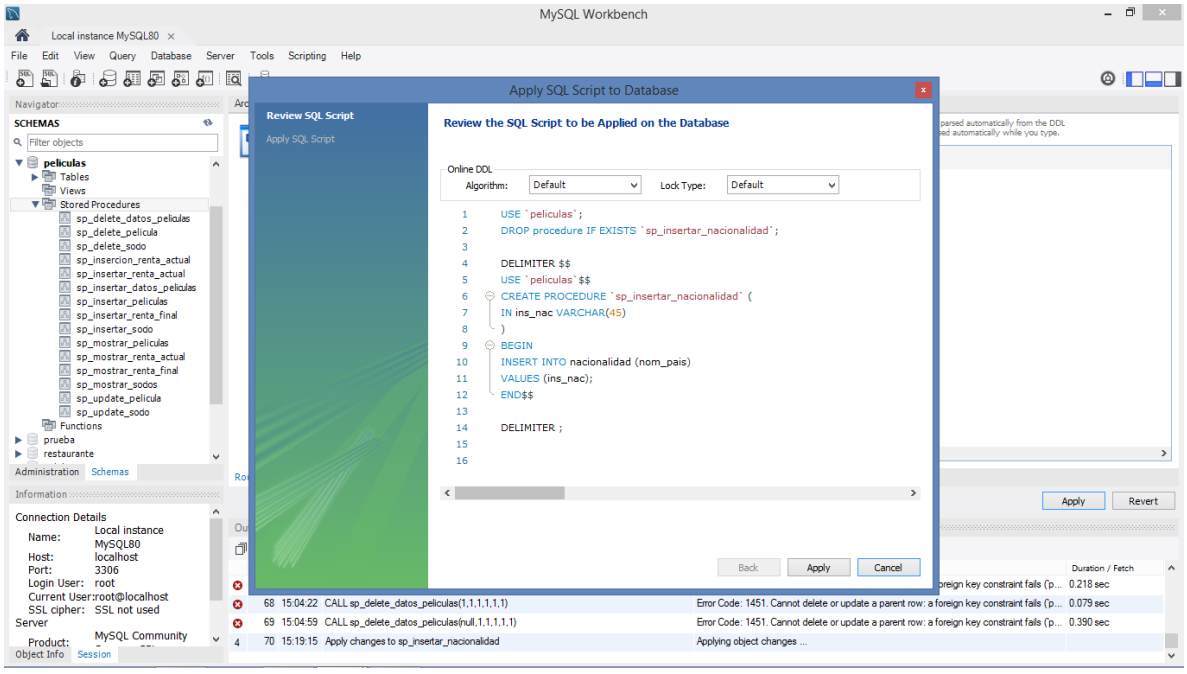
INSERT

Con este comando, se crean registros en una tabla.

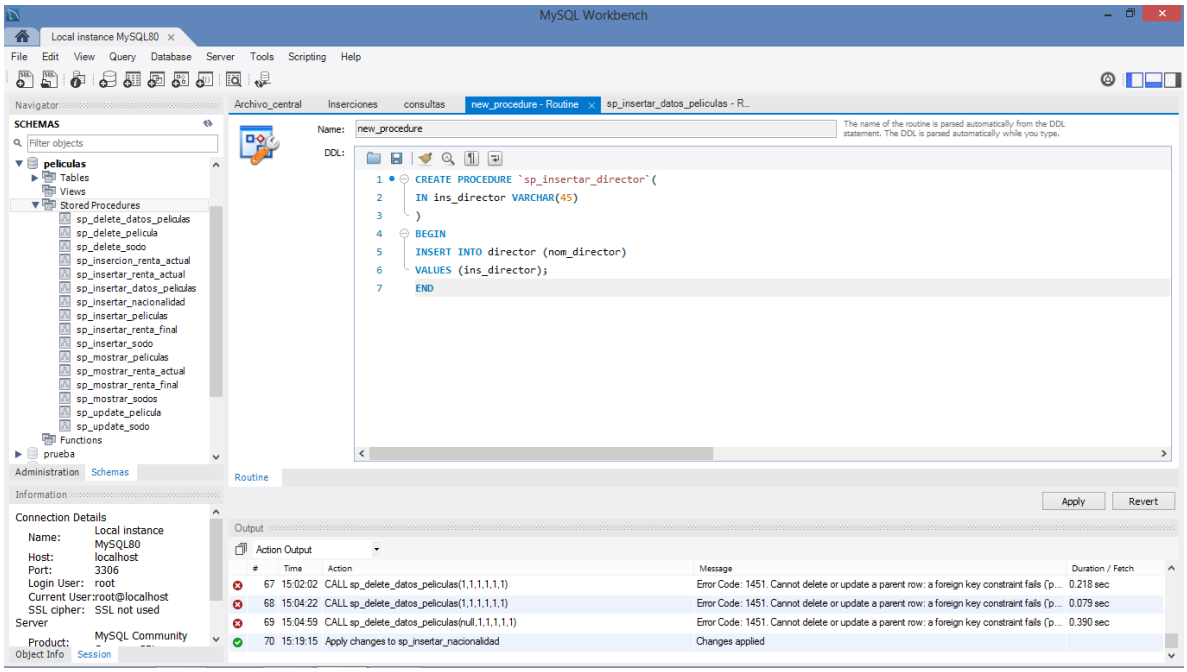
Se hará una inserción Individual de los datos de las películas:

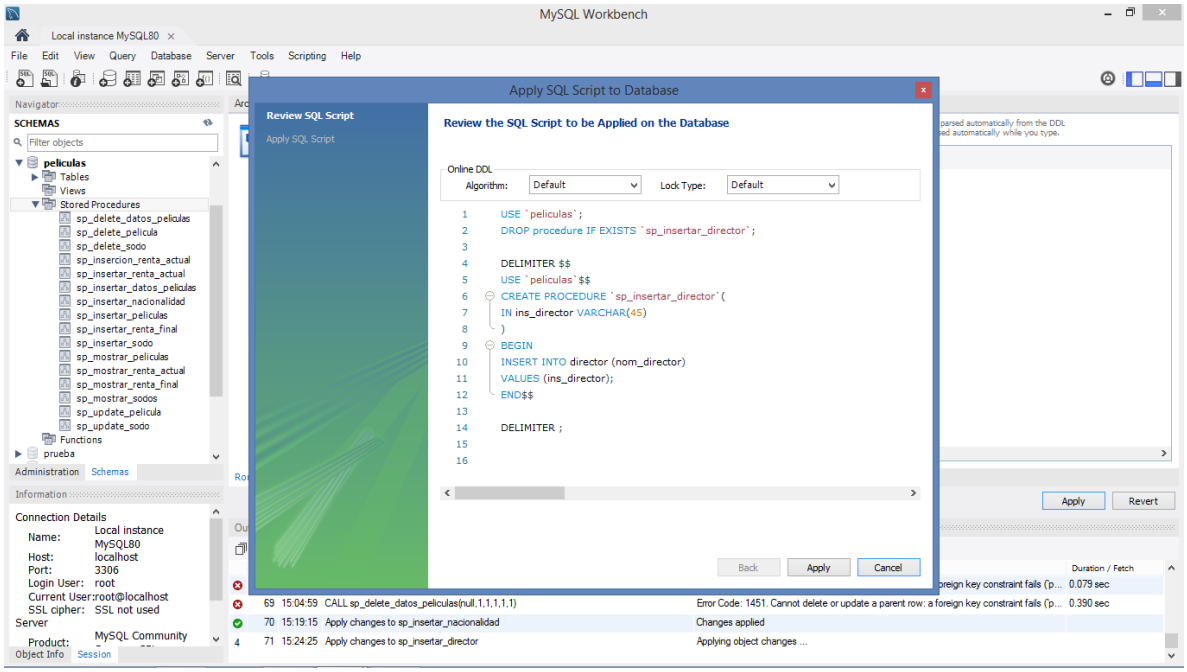
Nacionalidad de la película:



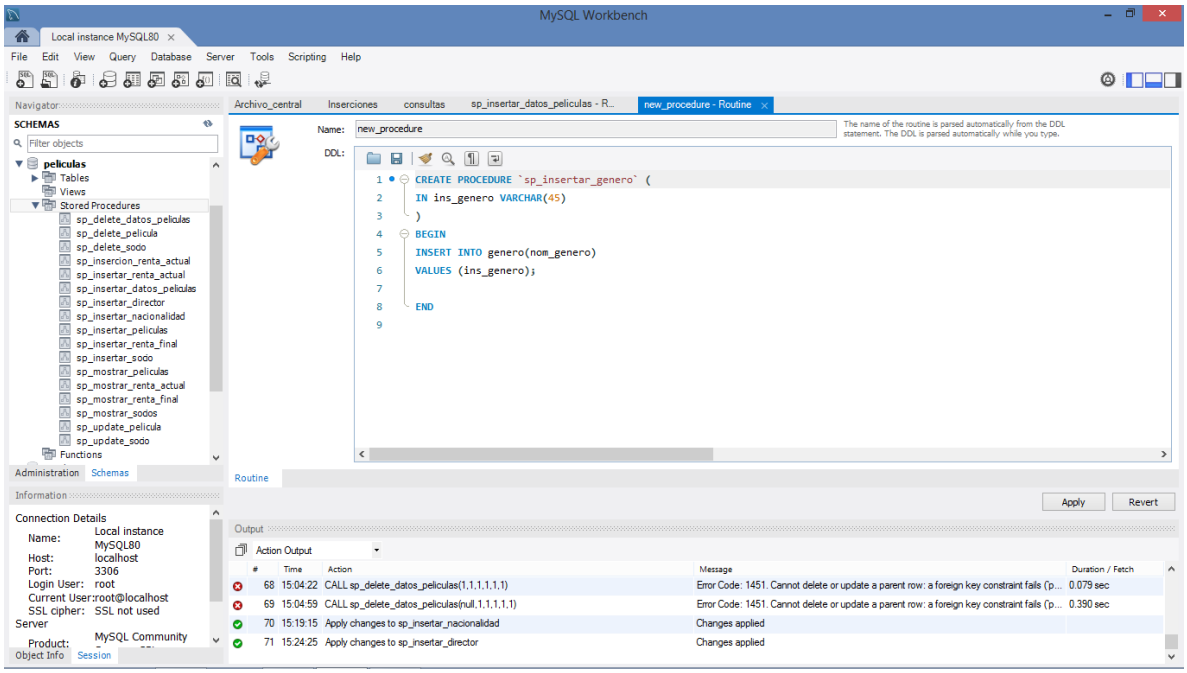


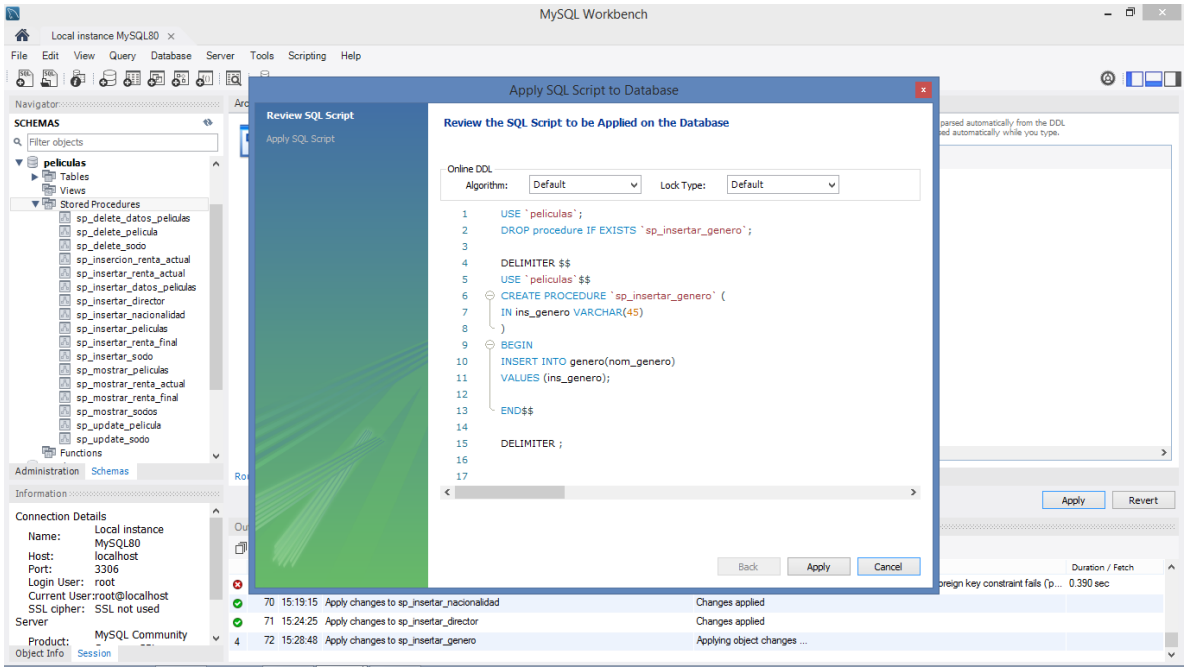
Director de la película:



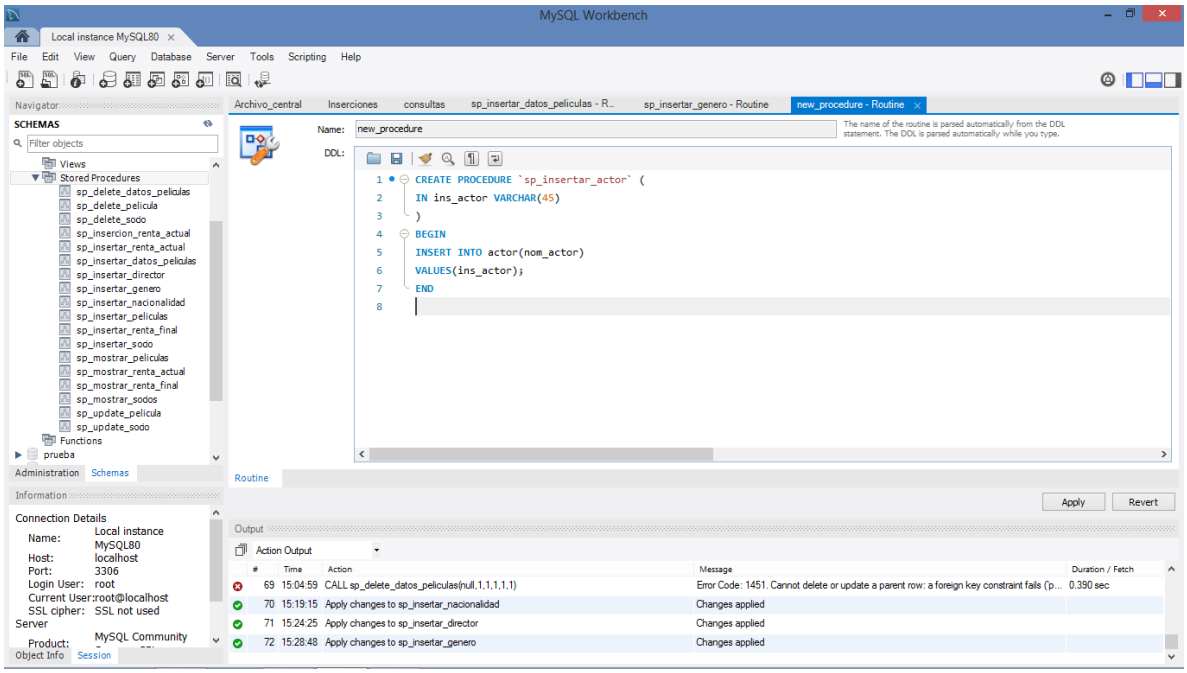


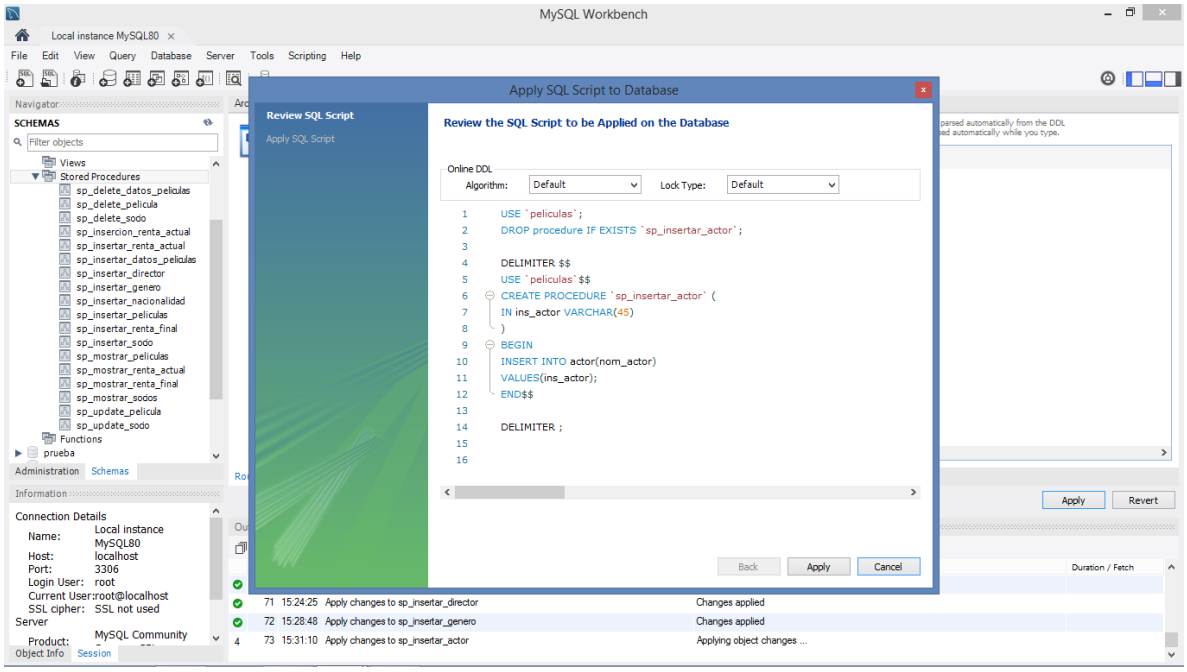
Genero película:


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
peliculas
Tables
Views
Stored Procedures
sp_delete_datos_peliculas
sp_delete_pelicula
sp_delete_sodo
sp_insercion_renta_actual
sp_insertar_renta_actual
sp_insertar_datos_peliculas
sp_insertar_director
sp_insertar_nacionalidad
sp_insertar_peliculas
sp_insertar_renta_final
sp_insertar_sodo
sp_mostrar_peliculas
sp_mostrar_renta_actual
sp_mostrar_renta_final
sp_mostrar_sodos
sp_update_pelicula
sp_update_sodo
Functions
Administration Schemas
Information
Connection Details
Name: Local instance MySQL80
Host: localhost
Port: 3306
Login User: root
Current User: root@localhost
SSL cipher: SSL not used
Server
Product: MySQL Community
Object Info Session
Archivo_central Inserciones consultas sp_insertar_datos_peliculas - R... new_procedure - Routine
Name: new_procedure
The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.
DDL:
1 CREATE PROCEDURE `sp_insertar_genero` (
2 IN ins_genero VARCHAR(45)
3)
4 BEGIN
5 INSERT INTO genero(nom_genero)
6 VALUES (ins_genero);
7
8 END
9
Routine
Apply Revert
Output
Action Output
Time Action Message Duration / Fetch
68 15:04:22 CALL sp_delete_datos_peliculas(1,1,1,1,1,1) Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`p... 0.079 sec
69 15:04:59 CALL sp_delete_datos_peliculas(null,1,1,1,1,1) Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`p... 0.390 sec
70 15:19:15 Apply changes to sp_insertar_nacionalidad Changes applied
71 15:24:25 Apply changes to sp_insertar_director Changes applied

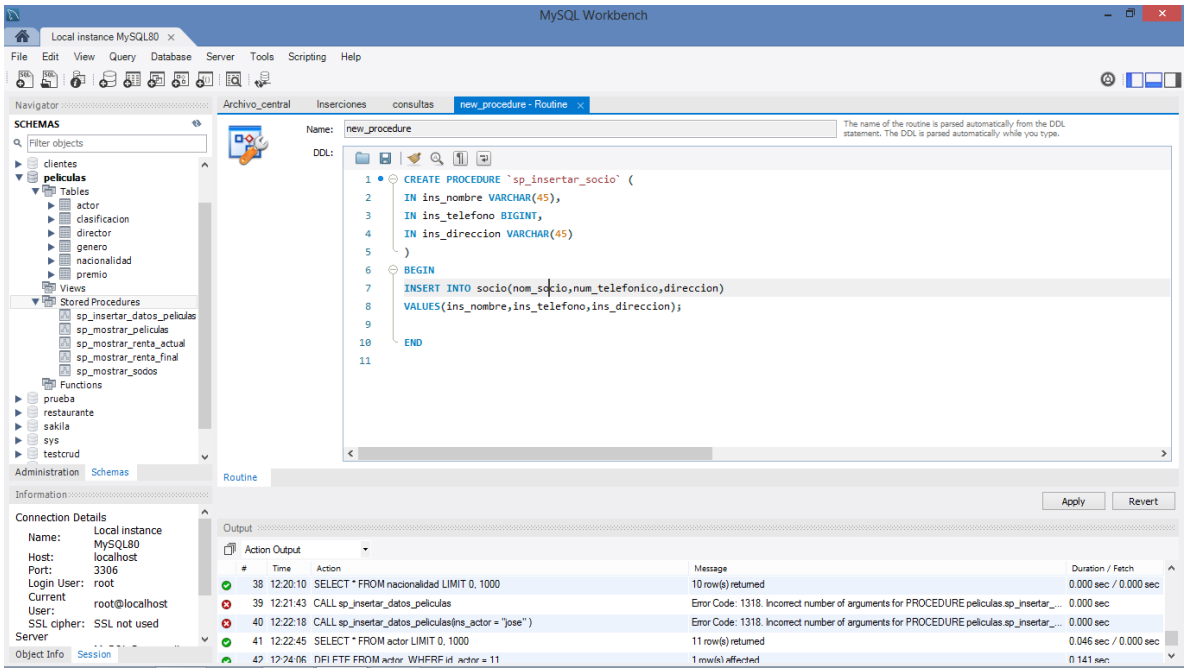

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Apply SQL Script to Database
Review SQL Script
Apply SQL Script
Review the SQL Script to be Applied on the Database
Online DDL
Algorithm: Default
Lock Type: Default
1 USE `peliculas`;
2 DROP procedure IF EXISTS `sp_insertar_genero`;
3
4 DELIMITER $$
5 USE `peliculas`$$
6 CREATE PROCEDURE `sp_insertar_genero` (
7 IN ins_genero VARCHAR(45)
8)
9 BEGIN
10 INSERT INTO genero(nom_genero)
11 VALUES (ins_genero);
12
13 END$$
14
15 DELIMITER ;
16
17
Back Apply Cancel
70 15:19:15 Apply changes to sp_insertar_nacionalidad Changes applied
71 15:24:25 Apply changes to sp_insertar_director Changes applied
4 72 15:28:48 Apply changes to sp_insertar_genero Applying object changes ...

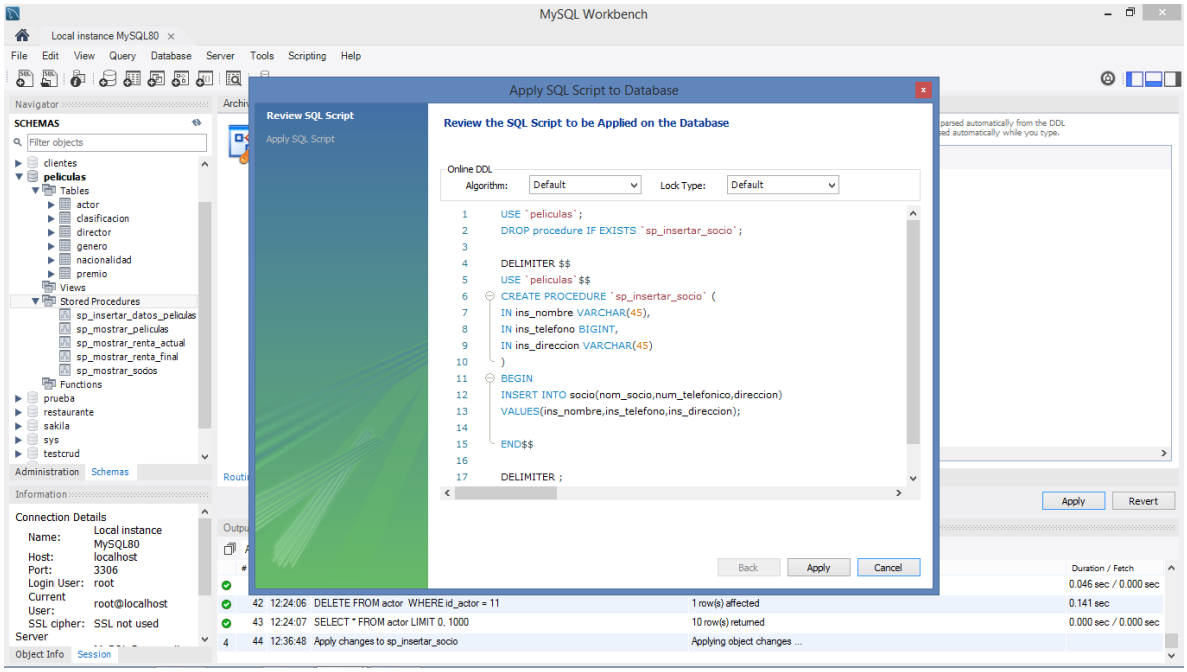
Actor:


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Archivo_central Inserciones consultas sp_insertar_datos_peliculas - R... sp_insertar_genero - Routine new_procedure - Routine
Name: new_procedure
DDL:
1 CREATE PROCEDURE `sp_insertar_actor` (
2 IN ins_actor VARCHAR(45)
3)
4 BEGIN
5 INSERT INTO actor(nom_actor)
6 VALUES(ins_actor);
7 END
8
Routine
Apply Revert
Output
Action Output
Time Action Message Duration / Fetch
69 15:04:59 CALL sp_delete_datos_peliculas(null,1,1,1,1,1) Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`p... 0.390 sec
70 15:19:15 Apply changes to sp_insertar_nacionalidad Changes applied
71 15:24:25 Apply changes to sp_insertar_director Changes applied
72 15:28:48 Apply changes to sp_insertar_genero Changes applied
SCHEMAS
Filter objects
Views
Stored Procedures
sp_delete_datos_peliculas
sp_delete_pelicula
sp_delete_sodo
sp_insercion_renta_actual
sp_insertar_renta_actual
sp_insertar_datos_peliculas
sp_insertar_director
sp_insertar_genero
sp_insertar_nacionalidad
sp_insertar_peliculas
sp_insertar_renta_final
sp_insertar_sodo
sp_mostrar_peliculas
sp_mostrar_renta_actual
sp_mostrar_renta_final
sp_mostrar_sodos
sp_update_pelicula
sp_update_sodo
Functions
prueba
Administration Schemas
Information
Connection Details
Name: Local instance MySQL80
Host: localhost
Port: 3306
Login User: root
Current User: root@localhost
SSL cipher: SSL not used
Server
Product: MySQL Community
Object Info Session

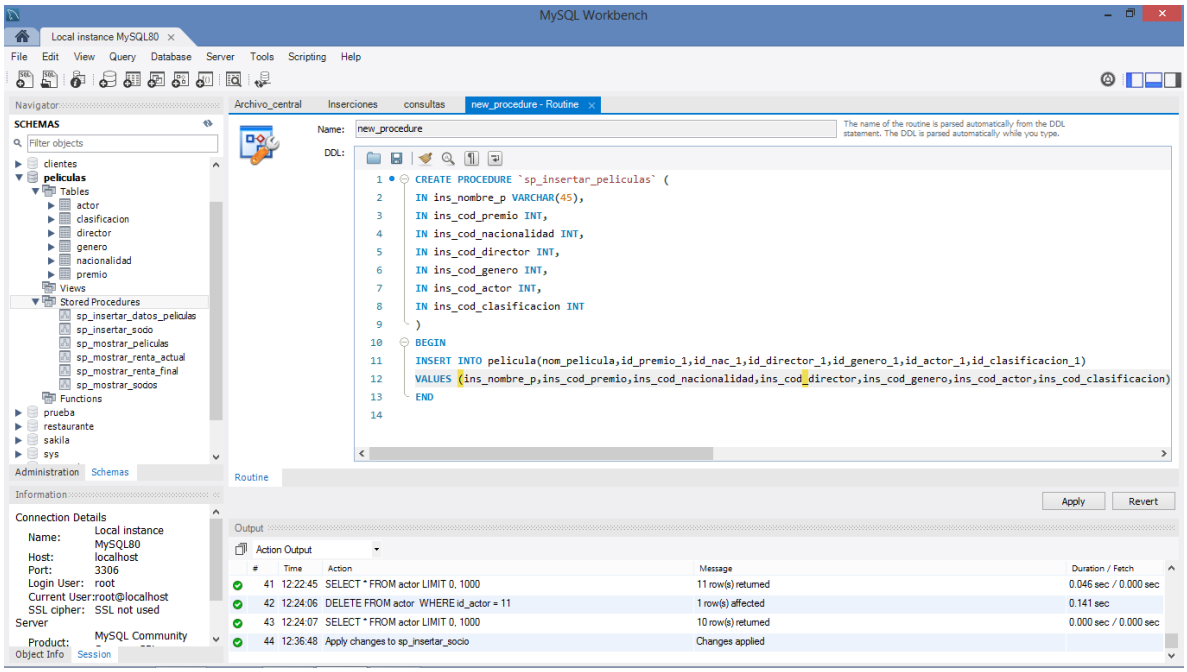

MySQL Workbench
Local instance MySQL80
Apply SQL Script to Database
Review SQL Script
Apply SQL Script
Review the SQL Script to be Applied on the Database
Online DDL
Algorithm: Default
Lock Type: Default
1 USE `peliculas`;
2 DROP procedure IF EXISTS `sp_insertar_actor`;
3
4 DELIMITER $$
5 USE `peliculas`$$
6 CREATE PROCEDURE `sp_insertar_actor` (
7 IN ins_actor VARCHAR(45)
8)
9 BEGIN
10 INSERT INTO actor(nom_actor)
11 VALUES(ins_actor);
12 END$$
13
14 DELIMITER ;
15
16
Back Apply Cancel
71 15:24:25 Apply changes to sp_insertar_director Changes applied
72 15:28:48 Apply changes to sp_insertar_genero Changes applied
4 73 15:31:10 Apply changes to sp_insertar_actor Applying object changes ...

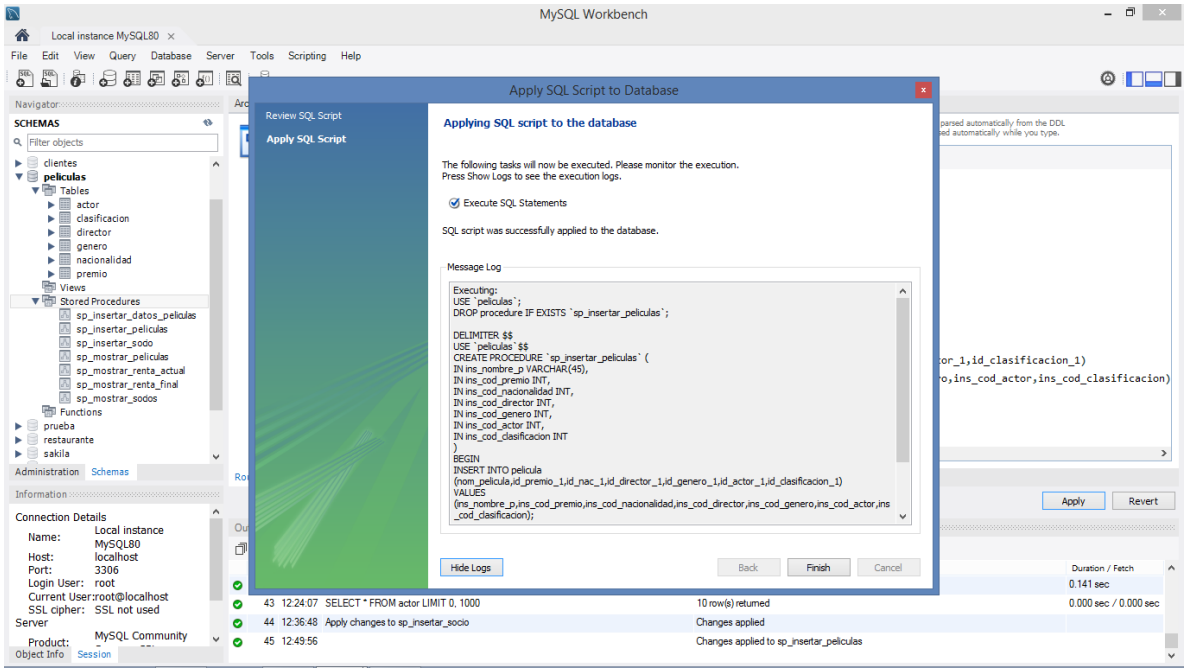
Inserción de socio:



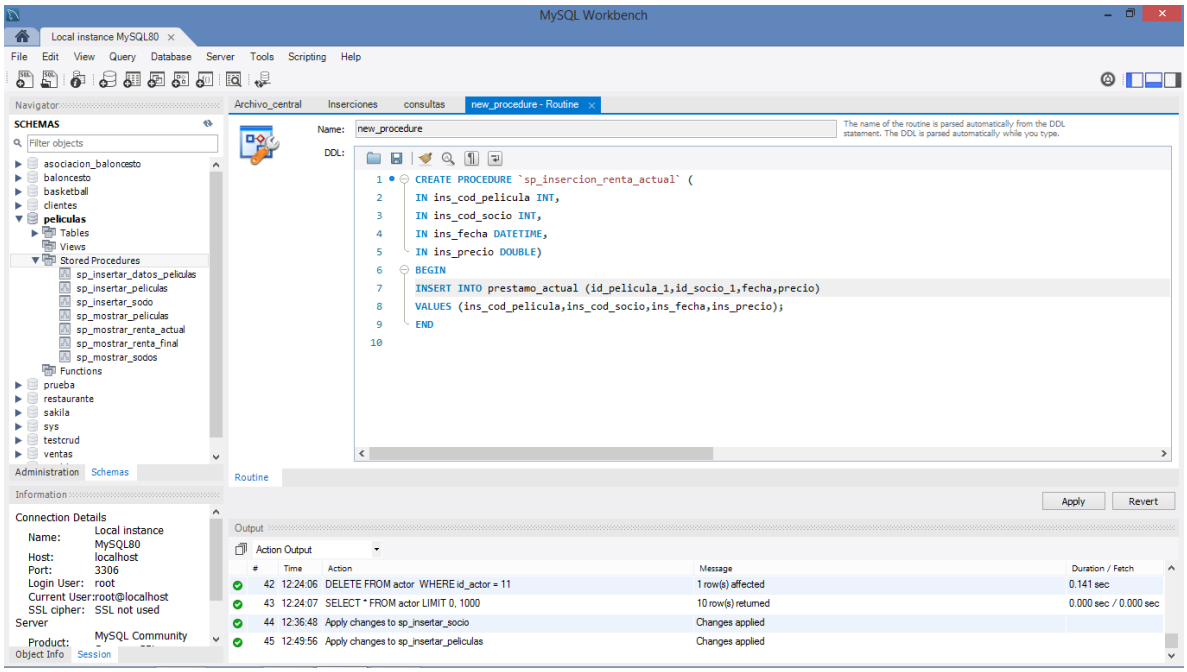


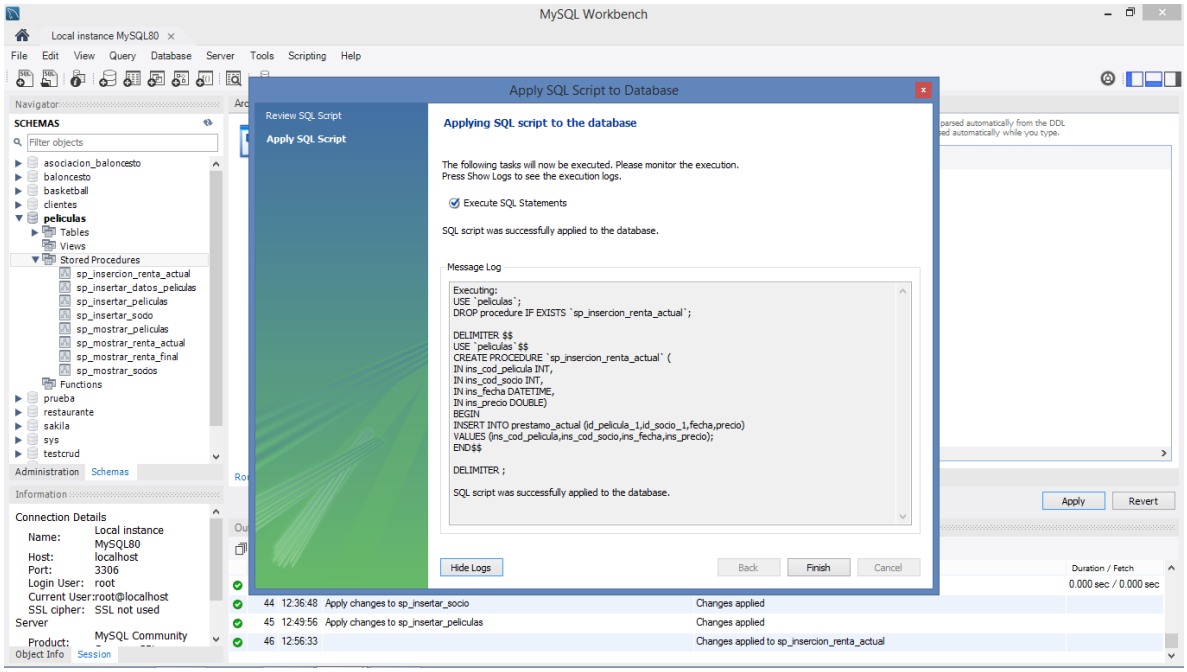
Inserción películas:



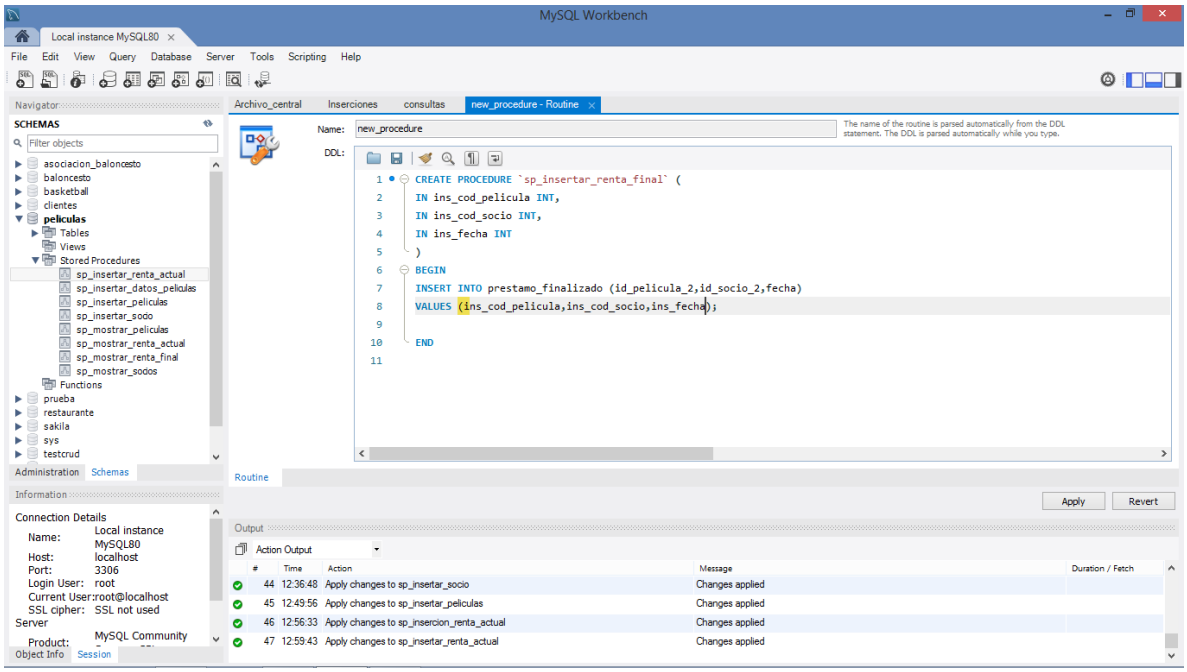


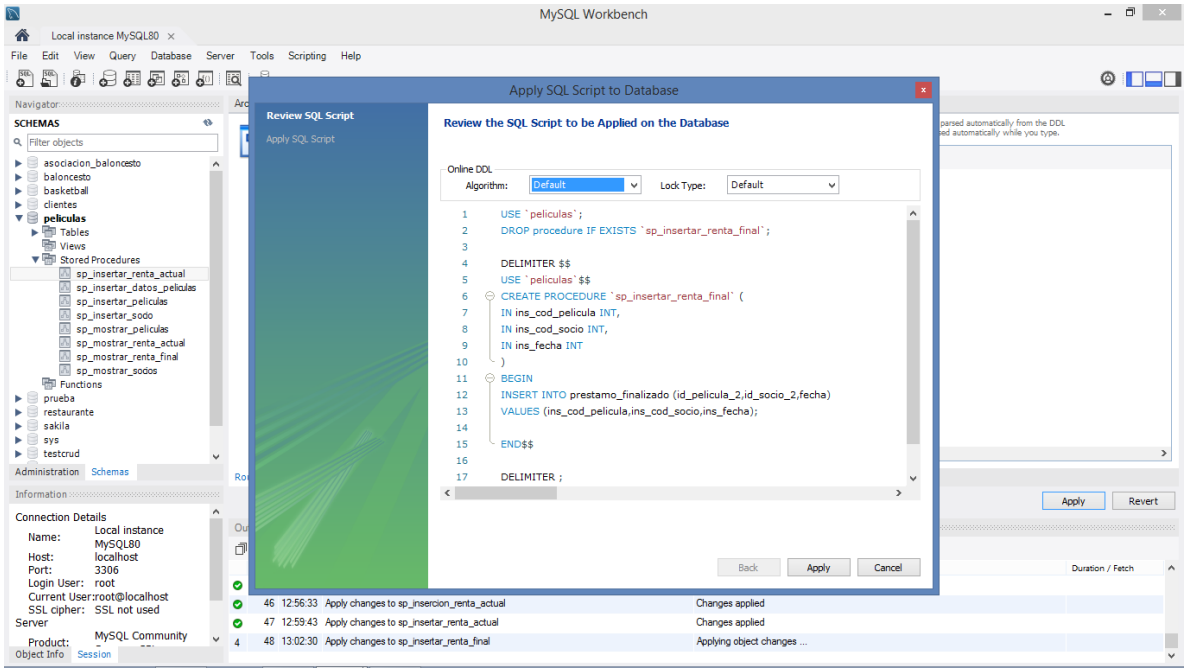
Préstamo actual:





Préstamo finalizado:

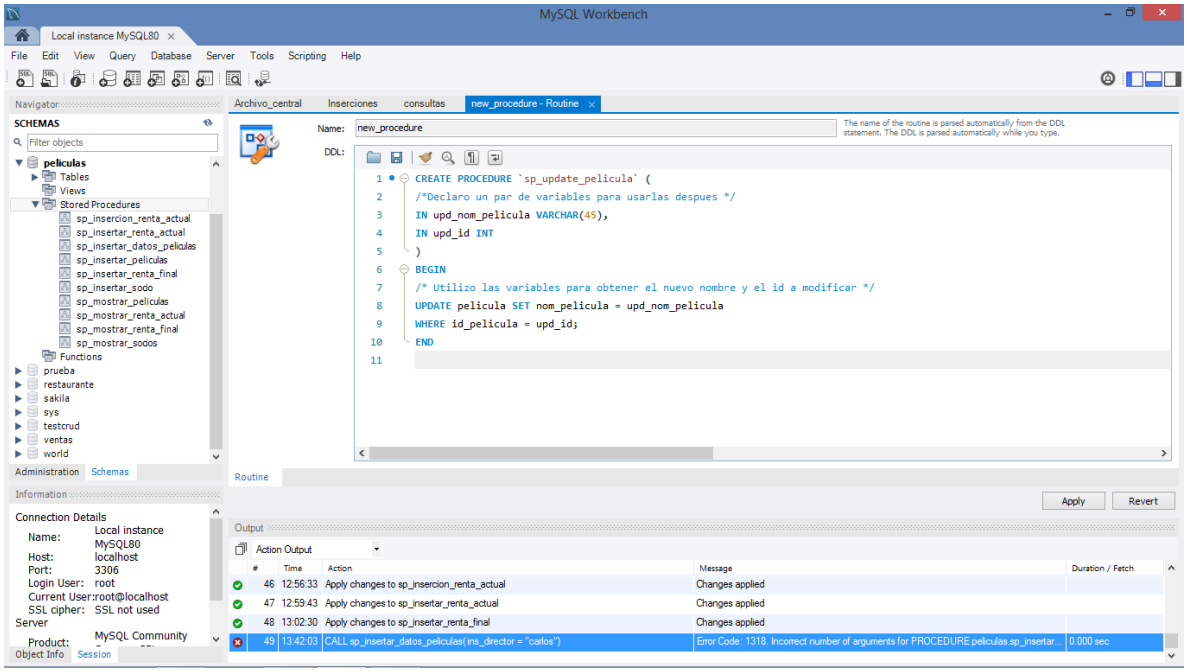


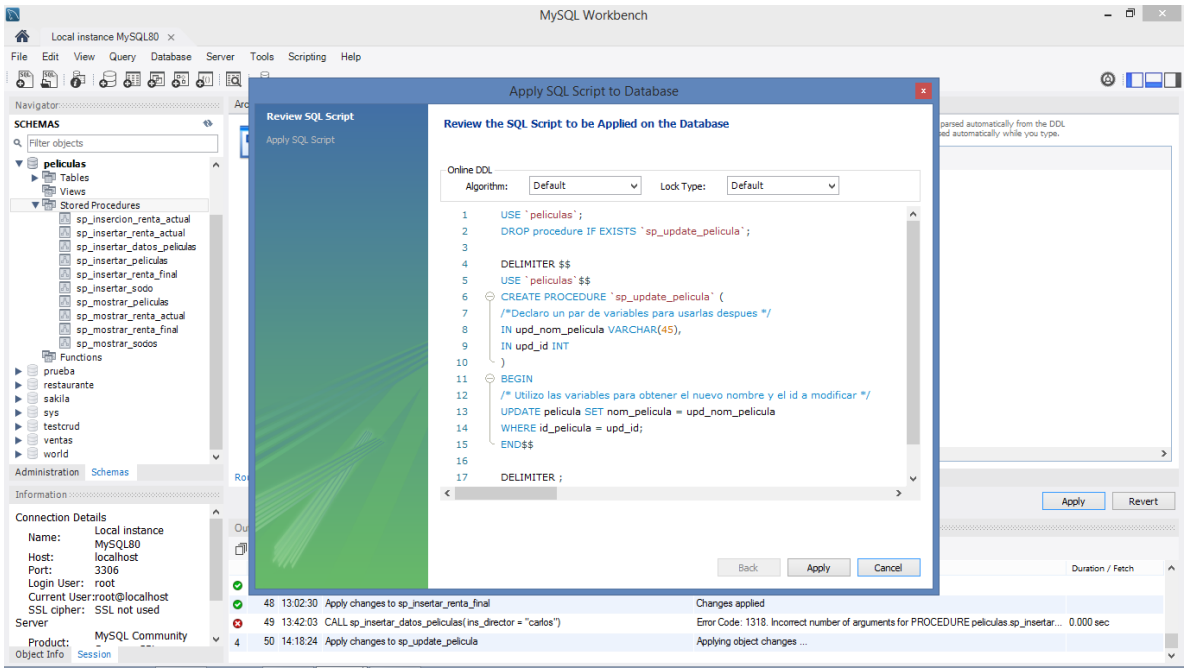


UPDATES:

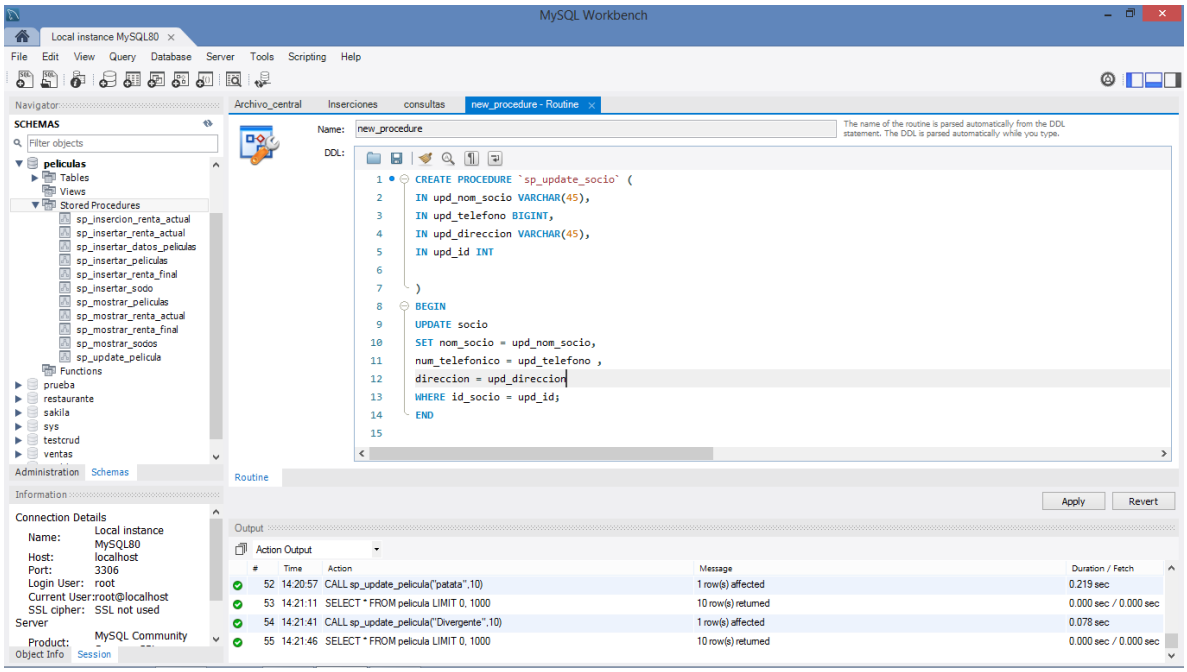
Con este se podrá actualizar los datos que se presentarán a continuación.

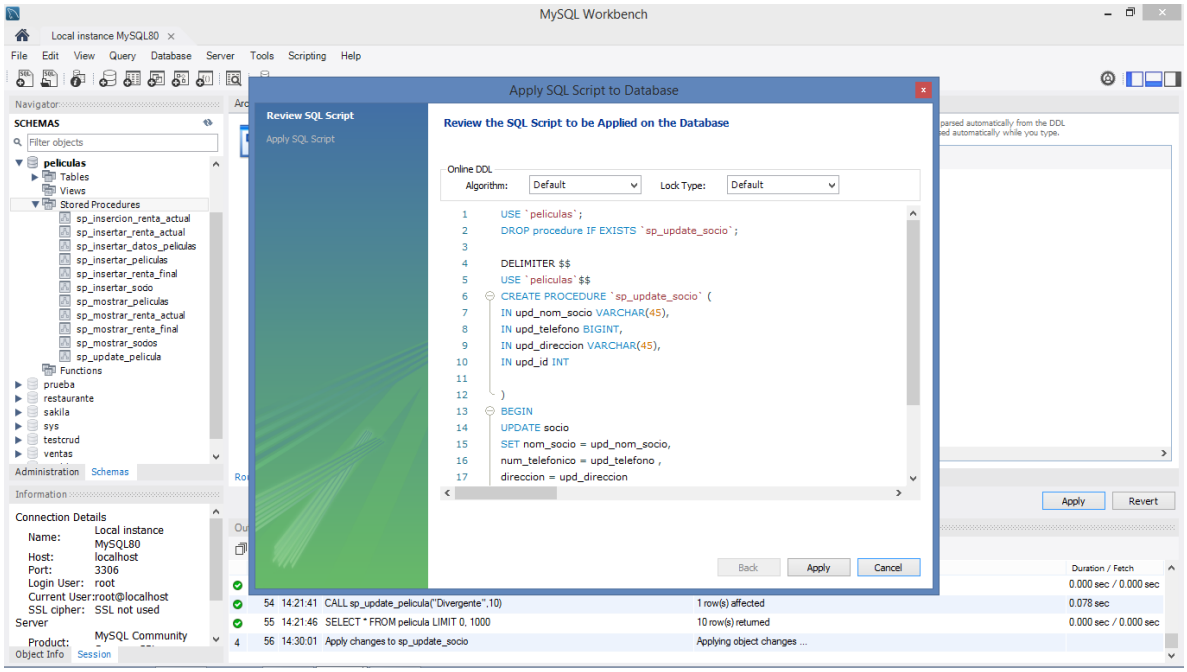
Películas





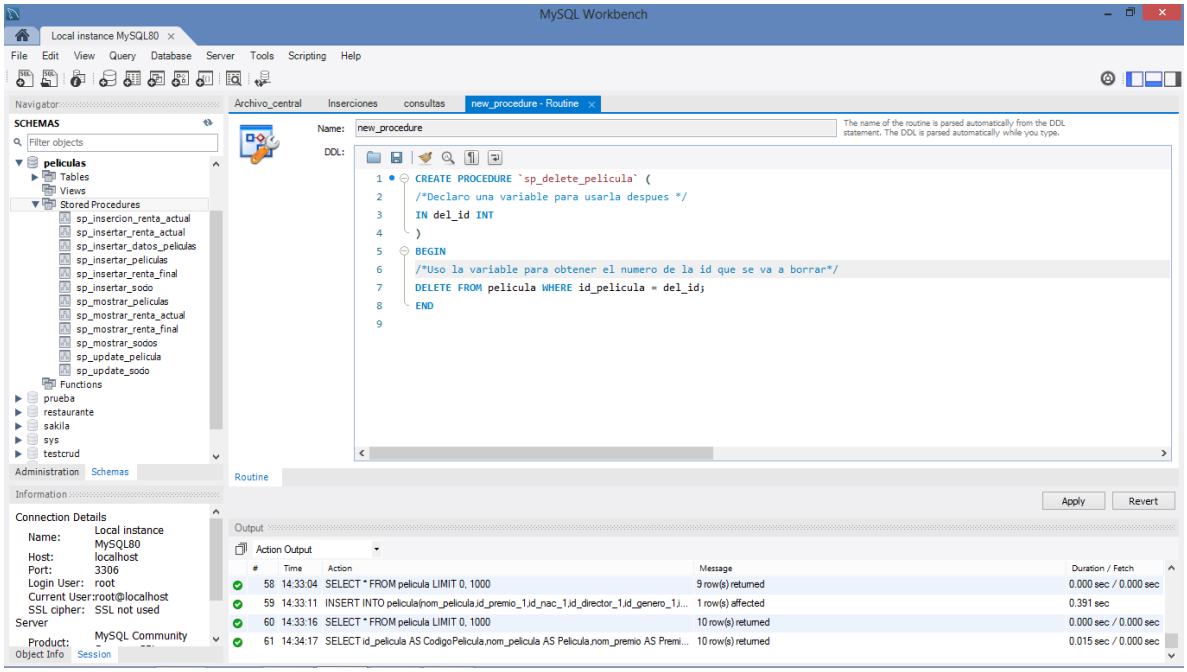
Socios

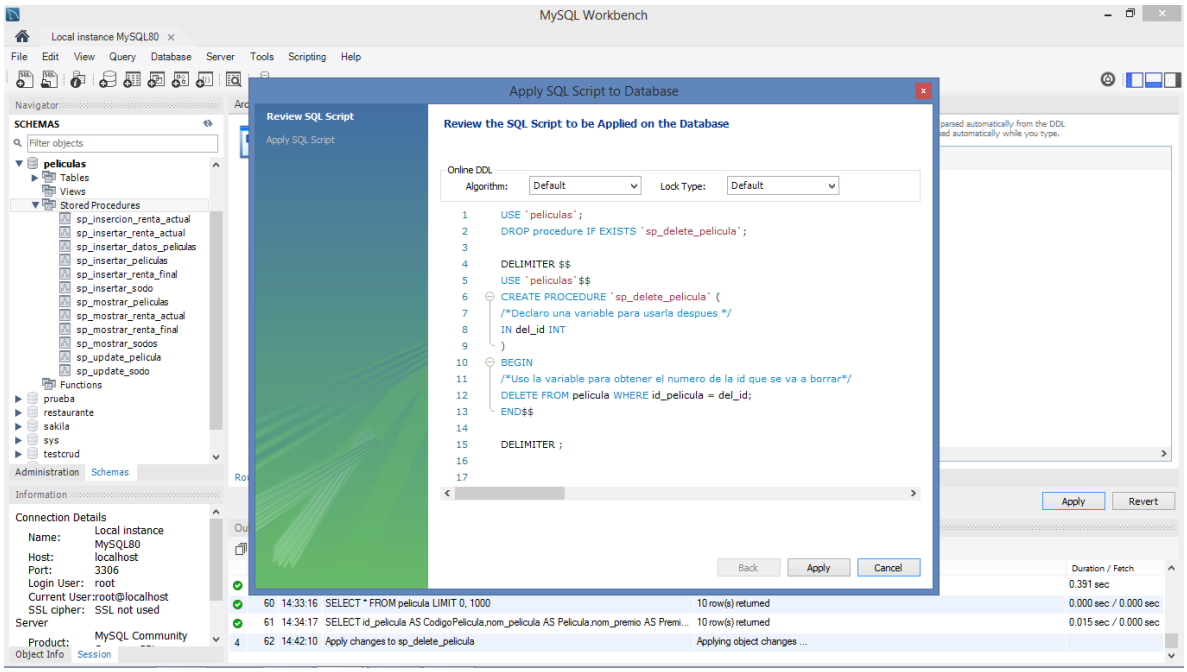

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
peliculas
Tables
Views
Stored Procedures
sp_insercion_renta_actual
sp_insertar_renta_actual
sp_insertar_datos_peliculas
sp_insertar_peliculas
sp_insertar_renta_final
sp_insertar_sodo
sp_mostrar_peliculas
sp_mostrar_renta_actual
sp_mostrar_renta_final
sp_mostrar_sodos
sp_update_pelicula
Functions
prueba
restaurante
sakila
sys
testcrud
ventas
Administration Schemas
Information
Connection Details
Name: Local instance MySQL80
Host: localhost
Port: 3306
Login User: root
Current User:root@localhost
SSL cipher: SSL not used
Server
Product: MySQL Community
Object Info Session
Archivo_central Inserciones consultas new_procedure - Routine
Name: new_procedure
The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.
DDL:
1 CREATE PROCEDURE `sp_update_socio` (
2 IN upd_nom_socio VARCHAR(45),
3 IN upd_telefono BIGINT,
4 IN upd_direccion VARCHAR(45),
5 IN upd_id INT
6
7)
8 BEGIN
9 UPDATE socio
10 SET nom_socio = upd_nom_socio,
11 num_telefonico = upd_telefono ,
12 direccion = upd_direccion
13 WHERE id_socio = upd_id;
14 END
15
Routine
Apply Revert
Output
Action Output
Time Action Message Duration / Fetch
52 14:20:57 CALL sp_update_pelicula("patata",10) 1 row(s) affected 0.219 sec
53 14:21:11 SELECT * FROM pelicula LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
54 14:21:41 CALL sp_update_pelicula("Divergente",10) 1 row(s) affected 0.078 sec
55 14:21:46 SELECT * FROM pelicula LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Apply SQL Script to Database
Review SQL Script
Apply SQL Script
Review the SQL Script to be Applied on the Database
Online DDL
Algorithm: Default
Lock Type: Default
1 USE `peliculas`;
2 DROP procedure IF EXISTS `sp_update_socio`;
3
4 DELIMITER $$
5 USE `peliculas`$$
6 CREATE PROCEDURE `sp_update_socio` (
7 IN upd_nom_socio VARCHAR(45),
8 IN upd_telefono BIGINT,
9 IN upd_direccion VARCHAR(45),
10 IN upd_id INT
11
12)
13 BEGIN
14 UPDATE socio
15 SET nom_socio = upd_nom_socio,
16 num_telefonico = upd_telefono ,
17 direccion = upd_direccion
Back Apply Cancel
54 14:21:41 CALL sp_update_pelicula("Divergente",10) 1 row(s) affected 0.078 sec
55 14:21:46 SELECT * FROM pelicula LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
4 56 14:30:01 Apply changes to sp_update_socio Applying object changes ...

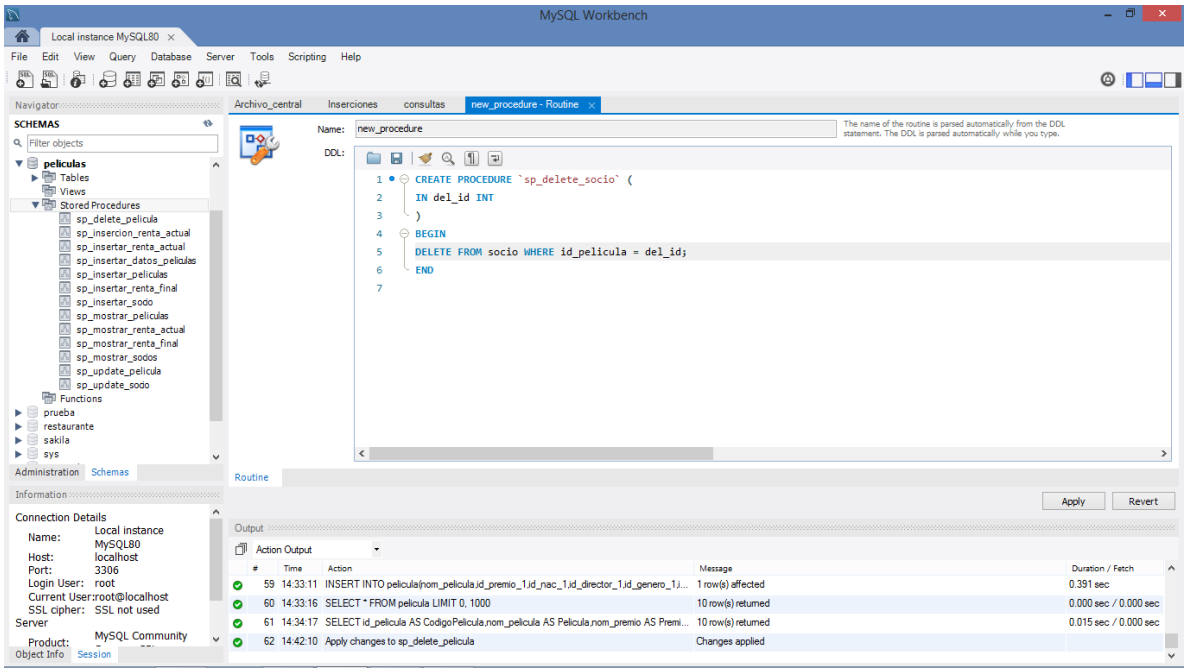
DELETES:

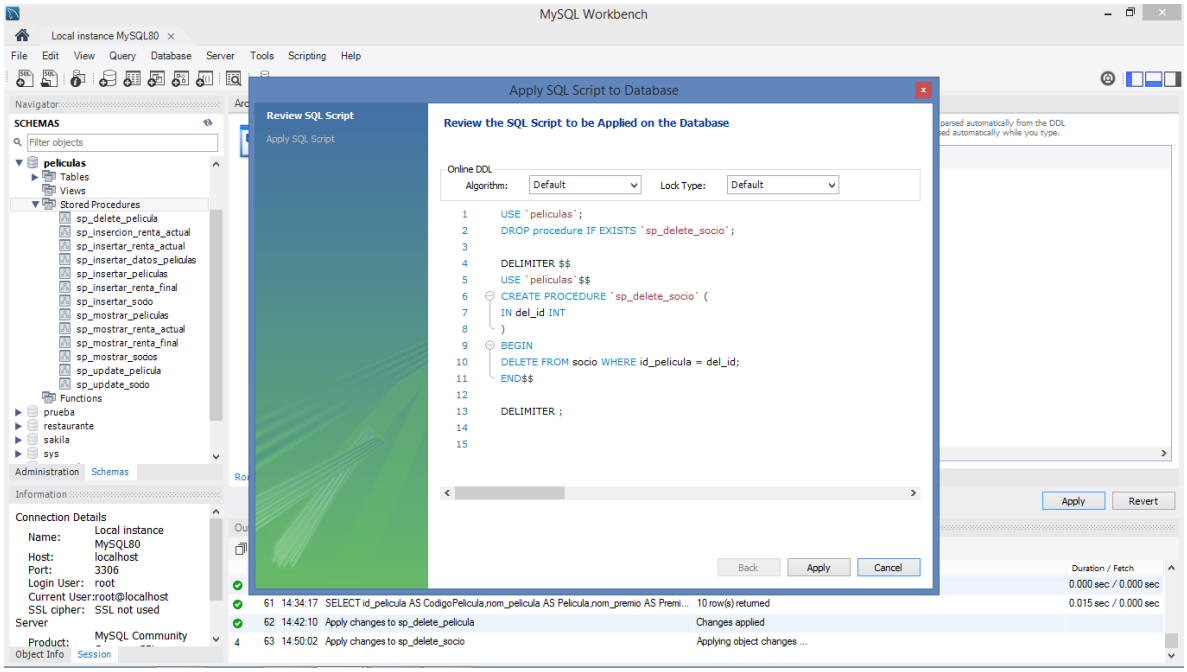
Finalmente, crearemos un nuevo procedimiento almacenado para eliminar socios y datos de película.



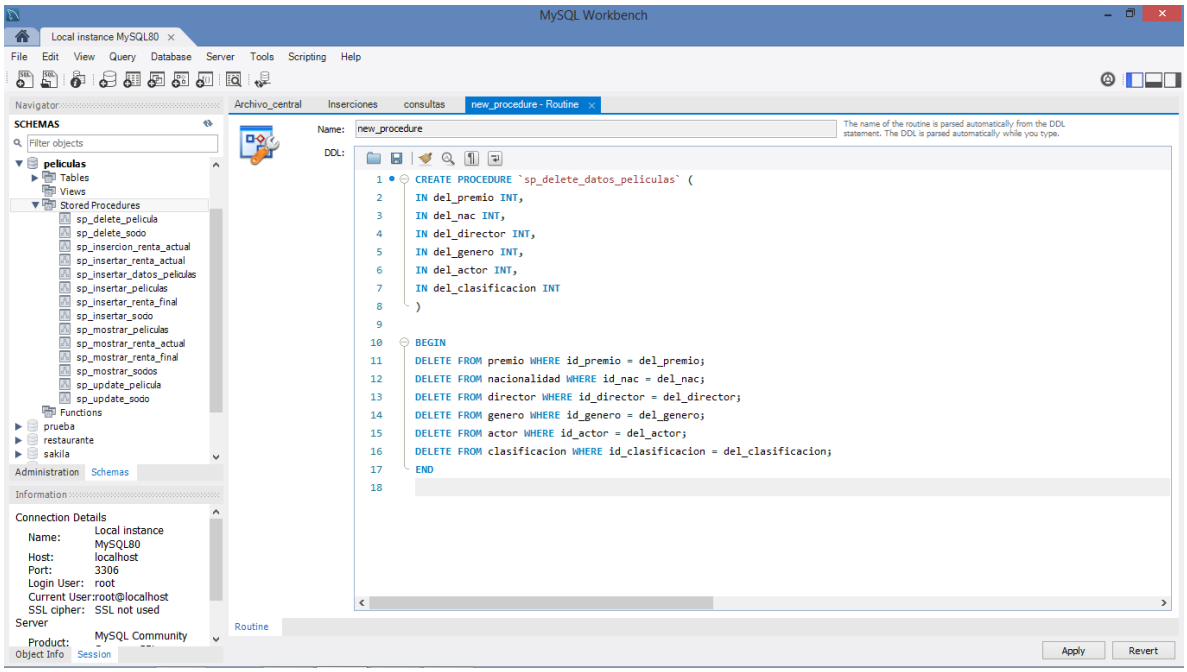


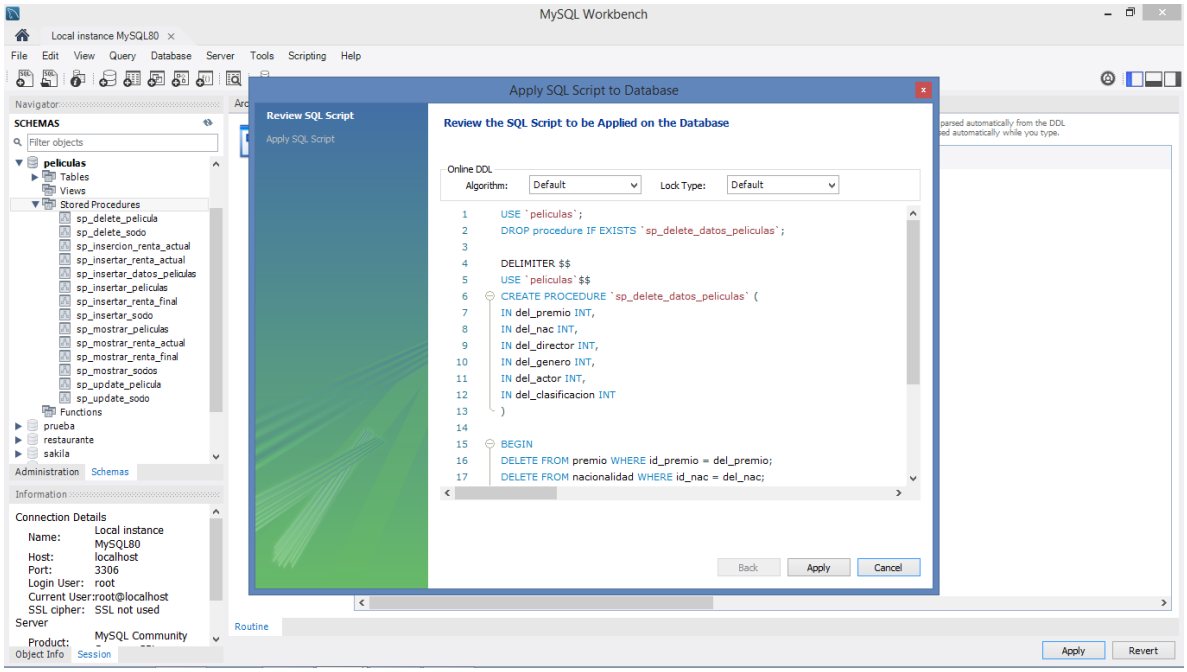
Socio:


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
peliculas
Tables
Views
Stored Procedures
sp_delete_pelicula
sp_insercion_renta_actual
sp_insertar_renta_actual
sp_insertar_datos_peliculas
sp_insertar_peliculas
sp_insertar_renta_final
sp_insertar_sodo
sp_mostrar_peliculas
sp_mostrar_renta_actual
sp_mostrar_renta_final
sp_mostrar_sodos
sp_update_pelicula
sp_update_sodo
Functions
prueba
restaurante
sakila
sys
Administration Schemas
Information
Connection Details
Name: Local instance MySQL80
Host: localhost
Port: 3306
Login User: root
Current User:root@localhost
SSL cipher: SSL not used
Server
Product: MySQL Community
Object Info Session
Archivo_central Inserciones consultas new_procedure - Routine
Name: new_procedure
The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.
DDL:
1 CREATE PROCEDURE `sp_delete_socio` (
2 IN del_id INT
3)
4 BEGIN
5 DELETE FROM socio WHERE id_pelicula = del_id;
6 END
7
Routine
Apply Revert
Output
Action Output
Time Action Message Duration / Fetch
59 14:33:11 INSERT INTO pelicula(nom_pelicula,id_premio_1,id_nac_1,id_director_1,id_genero_1,i... 1 row(s) affected 0.391 sec
60 14:33:16 SELECT * FROM pelicula LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
61 14:34:17 SELECT id_pelicula AS CodigoPelicula,nom_pelicula AS Pelicula,nom_premio AS Premi... 10 row(s) returned 0.015 sec / 0.000 sec
62 14:42:10 Apply changes to sp_delete_pelicula Changes applied


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Apply SQL Script to Database
Review SQL Script
Apply SQL Script
Review the SQL Script to be Applied on the Database
Online DDL
Algorithm: Default
Lock Type: Default
1 USE `peliculas`;
2 DROP procedure IF EXISTS `sp_delete_socio`;
3
4 DELIMITER $$
5 USE `peliculas`$$
6 CREATE PROCEDURE `sp_delete_socio` (
7 IN del_id INT
8)
9 BEGIN
10 DELETE FROM socio WHERE id_pelicula = del_id;
11 END$$
12
13 DELIMITER ;
14
15
Back Apply Cancel
61 14:34:17 SELECT id_pelicula AS CodigoPelicula,nom_pelicula AS Pelicula,nom_premio AS Premi... 10 row(s) returned 0.015 sec / 0.000 sec
62 14:42:10 Apply changes to sp_delete_pelicula Changes applied
63 14:50:02 Apply changes to sp_delete_socio Applying object changes ...

Datos de las películas:





Luego se invocan los procesos almacenados según se requiera:

En el caso de los de consulta:

```
3      /*De Consulta:*/
4 •    CALL sp_mostrar_peliculas;
5 •    CALL sp_mostrar_socios;
6 •    CALL sp_mostrar_renta_actual;
7 •    CALL sp_mostrar_renta_final;
```

Lo cual genera los siguientes resultados:

Para películas:

CodigoPelicula	Pelicula	Premio	Pais	Genero	ActorPrincipal	Clasificacion
1	IT	Oscar	Estados Unidos	Terror	Ricardo	mayores de dieciocho
2	Yo antes de ti	Directors Guild of America Award	Autralia	Romance	Angelina	Para todo publico
3	Tormenta	Oscar	Inglaterra	Terror	Daniel	mayores de dieciocho
4	Monsters	Directors Guild of America Award	Autralia	Infantil	Marlon	Para todo publico
5	Fuerza Elite	Directors Guild of America Award	Peru	Accion	Angelina	Mayores de quince años
6	Amelie	Globo De Oro	Mexico	Ciencia ficcion	Tom	Mayores de quince años
7	Sueños de libertad	New York Film Critics Circle Awards	Estados Unidos	Drama	Richard	Mayores de quince años
8	Tomb rider	Oscar	Estados Unidos	Aventura	Kim	Para todo publico
9	Venganza	Directors Guild of America Award	Brasil	Ciencia ficcion	Jack	mayores de dieciocho
11	Divergente	Screen Actors Guild Awards	Mexico	Accion	Kim	Mayores de quince años

Para socios:

id_socio	nom_socio	num_telefonico	direccion
1	Andres	3025632012	dl 1 No. 20-56
2	Carlos	3052105689	Av 5 No. 30-11
3	Sandra	2728058956	cra 30 No. 50-150
4	Diego	4561238256	dl 81 No. 121-50
5	Zuleima	3694125874	Av 4 No. 21-28
6	Hernan	3259874563	cra 32 No. 7-15
7	German	3255236589	cra 15 No. 25-70
8	Ariel	3597895646	dl 5 No. 10-8
9	Bianca	3598962356	Av 8 No. 6-11
10	Carla	3598962356	cra 9 No. 56-120

Para préstamo actual:

Codigo	CodigoPelicula	CodigoSocio	Pelicula	Socio	Telefono	Direccion	precio	fecha
1	4	1	Monsters	Andres	3025632012	dl 1 No. 20-56	10000	2019-12-28 00:00:00
2	1	2	IT	Carlos	3052105689	Av 5 No. 30-11	10000	2020-01-05 00:00:00
3	5	8	Fuerza Elite	Ariel	3597895646	dl 5 No. 10-8	10000	2020-01-07 00:00:00
4	7	5	Sueños de libertad	Zuleima	3694125874	Av 4 No. 21-28		01-10 00:00:00
5	3	9	Tormenta	Bianca	3598962356	Av 8 No. 6-11	10000	2020-01-13 00:00:00

Av 4 No. 21-28

Para préstamo final:

Codigo	CodigoPelicula	CodigoSocio	Pelicula	Socio	Telefono	Direccion	fecha
1	4	1	Monsters	Andres	3025632012	cll 1 No. 20-56	2020-01-08 00:00:00
2	1	2	IT	Carlos	3052105689	Av 5 No. 30-11	2020-01-15 00:00:00

Ya, en los procesos de almacenado para insertar, los cuales permiten ingresar nueva información a la base de datos.

```
9     /*De Insercion*/
10
11    /* datos de peliculas */
12    CALL sp_insertar_datos_peliculas;
13    /*datos de peliculas individuales*/
14    CALL sp_insertar_nacionalidad();
15    CALL sp_insertar_director();
16    CALL sp_insertar_genero();
17    CALL sp_insertar_actor();
18    /*socios*/
19    CALL sp_insertar_datos_socio( );
20    /*peliculas*/
21    CALL sp_insertar_peliculas();
22    /*renta actual */
23    CALL sp_insertar_renta_actual();
24    /*renta final*/
25    CALL sp_insertar_renta_final();
26
```

Gracias al proceso CRUD toda la información puede ser alternada, eliminada y/o cambiada una a una con el fin de que sea la manera más sencilla para la tienda en la cual pueda hacer los cambios pertinentes no solo con una base sino con diversas.

EJEMPLO:

Al insertar un nuevo préstamo escribiendo los argumentos que se enviarán al proceso de almacenado se obtiene lo siguiente:

CALL sp_insertar_renta_actual(5,3,"2020-01-14 00:00:00",10000);

Antes

Codigo	CodigoPelicula	CodigoSocio	Pelicula	Socio	Telefono	Direccion	precio	fecha
1	4	1	Monsters	Andres	3025632012	dl 1 No. 20-56	10000	2019-12-28 00:00:00
2	1	2	IT	Carlos	3052105689	Av 5 No. 30-11	10000	2020-01-05 00:00:00
3	5	8	Fuerza Elite	Ariel	3597895646	dl 5 No. 10-8	10000	2020-01-07 00:00:00
4	7	5	Sueños de libertad	Zuleima	3694125874	Av 4 No. 21-28	10000	2020-01-10 00:00:00
5	3	9	Tormenta	Bianca	3598962356	Av 8 No. 6-11	10000	2020-01-13 00:00:00

Av 4 No. 21-28

Después

Codigo	CodigoPelicula	CodigoSocio	Pelicula	Socio	Telefono	Direccion	precio	fecha
1	4	1	Monsters	Andres	3025632012	dl 1 No. 20-56	10000	2019-12-28 00:00:00
2	1	2	IT	Carlos	3052105689	Av 5 No. 30-11	10000	2020-01-05 00:00:00
3	5	8	Fuerza Elite	Ariel	3597895646	dl 5 No. 10-8	10000	2020-01-07 00:00:00
4	7	5	Sueños de libertad	Zuleima	3694125874	Av 4 No. 21-28	10000	2020-01-10 00:00:00
5	3	9	Tormenta	Bianca	3598962356	Av 8 No. 6-11	10000	2020-01-13 00:00:00
7	5	3	Fuerza Elite	Sandra	2728058956	cra 30 No. 50-150	10000	2020-01-14 00:00:00

Procesos almacenados de actualizar:

Con estos procesos almacenados se actualizar la información de los socios y de las películas, en dado caso que haya una modificación como la dirección o número de teléfono del socio.

```
30    /*UPDATES*/
31    /*pelicula*/
32 •  CALL sp_update_pelicula();
33    /*socio*/
34 •  CALL sp_update_socio;
```

EJEMPLO

Se va a actualizar el número de teléfono y dirección del socio Carla con id 10

CALL sp_update_socios("Carla","563214789","Av 80 No 50-120",10);

Antes:

id_socio	nom_socio	num_telefonico	direccion
1	Andres	3025632012	dl 1 No. 20-56
2	Carlos	3052105689	Av 5 No. 30-11
3	Sandra	2728058956	cra 30 No. 50-150
4	Diego	4561238256	dl 81 No. 121-50
5	Zuleima	3694125874	Av 4 No. 21-28
6	Hernan	3259874563	cra 32 No. 7-15
7	German	3255236589	cra 15 No. 25-70
8	Ariel	3597895646	dl 5 No. 10-8
9	Bianca	3598962356	Av 8 No. 6-11
10	Carla	3598962356	cra 9 No. 56-120

Después:

Se hizo el cambio pertinente para el socio Carla en la actualización de sus datos

id_socio	nom_socio	num_telefonico	direccion
1	Andres	3025632012	dl 1 No. 20-56
2	Carlos	3052105689	Av 5 No. 30-11
3	Sandra	2728058956	cra 30 No. 50-150
4	Diego	4561238256	dl 81 No. 121-50
5	Zuleima	3694125874	Av 4 No. 21-28
6	Hernan	3259874563	cra 32 No. 7-15
7	German	3255236589	cra 15 No. 25-70
8	Ariel	3597895646	dl 5 No. 10-8
9	Bianca	3598962356	Av 8 No. 6-11
10	Carla	563214789	Av 80 No 50-120

Proceso de almacenado de eliminar:

Con ellos se pueden eliminar registros en las tablas, pero, todo esto condicionado a las id para que no borren todos los registros de la tabla.

```
39    /*DELETE*/
40 •  CALL sp_delete_pelicula();
41 •  CALL sp_delete_socio();
42 •  CALL sp_delete_datos_peliculas();
```

EJEMPLO

Se va a eliminar una película que ya no está disponible, en este caso será Divergente con id 11:

Enviamos la id a eliminar como argumento:

CALL sp_delete_pelicula(11);

Antes:

CodigoPelicula	Pelicula	Premio	Pais	Genero	ActorPrincipal	Clasificacion
1	IT	Oscar	Estados Unidos	Terror	Ricardo	mayores de dieciocho
2	Yo antes de ti	Directors Guild of America Award	Autralia	Romance	Angelina	Para todo publico
3	Tormenta	Oscar	Inglaterra	Terror	Daniel	mayores de dieciocho
4	Monsters	Directors Guild of America Award	Autralia	Infantil	Marlon	Para todo publico
5	Fuerza Elite	Directors Guild of America Award	Peru	Accion	Angelina	Mayores de quince años
6	Amelie	Globo De Oro	Mexico	Ciencia ficcion	Tom	Mayores de quince años
7	Sueños de libertad	New York Film Critics Circle Awards	Estados Unidos	Drama	Richard	Mayores de quince años
8	Tomb rider	Oscar	Estados Unidos	Aventura	Kim	Para todo publico
9	Venganza	Directors Guild of America Award	Brasil	Ciencia ficcion	Jack	mayores de dieciocho
11	Divergente	Screen Actors Guild Awards	Mexico	Accion	Kim	Mayores de quince años

Después:

Como se puede observar la película Divergente con el id 11 ha desaparecido.

CodigoPelicula	Pelicula	Premio	Pais	Genero	ActorPrincipal	Clasificacion
1	IT	Oscar	Estados Unidos	Terror	Ricardo	mayores de dieciocho
2	Yo antes de ti	Directors Guild of America Award	Autralia	Romance	Angelina	Para todo publico
3	Tormenta	Oscar	Inglaterra	Terror	Daniel	mayores de dieciocho
4	Monsters	Directors Guild of America Award	Autralia	Infantil	Marlon	Para todo publico
5	Fuerza Elite	Directors Guild of America Award	Peru	Accion	Angelina	Mayores de quince años
6	Amelie	Globo De Oro	Mexico	Ciencia ficcion	Tom	Mayores de quince años
7	Sueños de libertad	New York Film Critics Circle Awards	Estados Unidos	Drama	Richard	Mayores de quince años
8	Tomb rider	Oscar	Estados Unidos	Aventura	Kim	Para todo publico
9	Venganza	Directors Guild of America Award	Brasil	Ciencia ficcion	Jack	mayores de dieciocho

CONCLUSIÓN

Se buscaba generar una organización al momento de prestar una película, porque existe una base de datos enfocada netamente a los socios donde ellos son los más interesados en adquirirla por un periodo determinado de tiempo. Cada cliente tiene una necesidad especifica por eso es por lo que la base de datos realizada anteriormente tiene consultas diferentes que permiten agilizar la búsqueda y generar filtros que permitan la obtención de información.

Hay que tener en cuenta que al ser un local dirigido al alquiler de películas todo puede irse modificando, los clientes pueden perder el interés de seguir alquilando o simplemente se retiran por falta de interés, así como las películas que fueron en un momento el boom y perdieron el interés. Hay variables externas que tener en cuenta al momento de modificar las bases de datos, pero sin duda al tener todo organizado y poder modificarlos cuantas veces se requiere es lo que se busca en un momento determinado.