

BASES DE DATOS

Practica 02

DIAZ MEDINA JESÚS KAIMORTS

Viernes, 17 de Febrero del 2017.

INDICE

MARCO TEÓRICO.....	2
DESARROLLO DE LA PRÁCTICA.....	8
INSTRUCCIONES Y CODIFICACIÓN.....	8
ENTORNO DE DESARROLLO DE LA PRÁCTICA (SCREENSHOTS).....	12
DIAGRAMA ENTIDAD-RELACION.....	16
CONCLUSIONES.....	17
BIBLIOGRAFÍA	17

BASES DE DATOS

Practica 02

MARCO TEÓRICO

- **CAMPOS CLAVES EN MYSQL.**

Los campos claves son campos que indicamos para hacer más fáciles nuestras búsquedas ya que se crea una indexación de nuestros datos, dentro de este tipo de campo tenemos **PRIMARY KEY, FOREIGN KEY Y UNIQUE.**

- a) **PRIMARY KEY:** Como su nombre lo indica son claves primarias, en palabras sencillas lo que quiere decir esto es que *en la indexación de nuestros datos MySQL le da prioridad a este tipo de clave seguido de las claves únicas.* Las claves primarias no permiten que haya datos duplicados dentro de este tipo de campo.
- b) **FOREIGN KEY:** Las claves foráneas son *aquellas que hacen referencia a la clave primaria de otra tabla, se utilizan para crear relaciones entre ellas,* este tipo de clave solo se implementan en MySQL **para el motor de almacenamiento InnoDB**, se habla de que MyISAM posteriormente soportara este tipo de claves.
- c) **UNIQUE:** Este tipo de clave como su nombre lo indica *no permite que exista duplicidad de datos.*

Ej.

```
CREATE TABLE personas (id INT(11) PRIMARY KEY AUTO_INCREMENT NOT NULL, nombre VARCHAR(40) NOT NULL, cedula varchar(20) UNIQUE NOT NULL);
```

- **MODIFICACIÓN DE ESTRUCTURA DE UNA TABLA**

La modificación y administración de tablas de una base de datos se realizan básicamente a partir de las sentencias CREATE TABLE (vista en el punto anterior), **ALTER TABLE Y DROP TABLE**.

Después de crear una tabla, es posible cambiar muchas de las opciones que fueron definidas cuando se creó originalmente, por ejemplo, es posible:

- a) Agregar, modificar o eliminar columnas. Así se puede cambiar el nombre, la longitud, el tipo de datos, la precisión, la escala y la aceptación de valores NULL de la columna, aunque hay algunas restricciones.
- b) Agregar o eliminar restricciones PRIMARY KEY y FOREIGN KEY.
- c) Agregar o eliminar restricciones UNIQUE y CHECK.

Inicialmente la sintaxis de **ALTER TABLE** podría ser la siguiente

```
ALTER TABLE nombreTabla  
| CHANGE nombreColumna nombreColumna_nuevo  
| ADD COLUMN declaracionColumna  
| DROP COLUMN nombreColumna  
| MODIFY COLUMN declaracionColumna  
| RENAME nombreTabla_nueva
```

En la sintaxis de **ALTER TABLE** ya hemos visto que **CHANGE**, **MODIFY**, **RENAME** y **ADD** cláusulas cuya finalidad es modificar propiedades de las columnas de las tablas, renombrar las tablas y añadir elementos a las mismas como columnas, índices y restricciones de integridad.

Las cláusulas **CHANGE** y **MODIFY** se utilizan para cambiar el tipo de las columnas de una tabla, pero **CHANGE** puede además renombrar columnas. La cláusula **RENAME** permite renombrar tablas. Por su parte, la cláusula **ADD** permite añadir columnas, índices y restricciones de integridad a las tablas.

- **COMANDOS PARA EL BORRADO**

- a) **Borrado de tablas**

La sentencia **DROP TABLE** permite borrar tablas completas de la base de datos. Al suprimir una tabla también se suprimen los índices y las concesiones asociadas a ella. Los sinónimos contruidos sobre tablas suprimidas se marcan como inválidas y dejan de funcionar.

Sintaxis: **DROP TABLE nombreTabla;**

Nota: No se puede utilizar DROP TABLE para quitar una tabla a la que se haga referencia con una restricción FOREIGN KEY, ya que primero se debe quitar la restricción FOREIGN KEY o la tabla de referencia.

Tampoco se puede utilizar la instrucción DROP TABLE sobre las tablas del sistema. Si elimina todas las filas de una tabla (DELETE tabla), la tabla existe hasta que se quite. Los permisos para utilizar DROP TABLE pertenecen de manera predeterminada al propietario de la tabla y no se pueden transferir.

b) Borrado de bases de datos.

En MySQL es muy sencillo borrar una base de datos completa con todos sus objetos mediante la sentencia DROP DATABASE.

Sintaxis: DROP DATABASE nombreBD;

Nota: Si se intenta eliminar una base de datos que no existe, MySQL ofrece un mensaje de error. Después de eliminar una base de datos es necesario tener precaución. Ya sabemos que la sentencia falla si no existe la base de datos que se quiere borrar, pero también falla en caso de no disponer de los permisos necesarios.¹

• MOTORES DE ALMACENAMIENTO

Un motor de almacenamiento es una parte esencial de un SGDB puesto que se encarga de crear, recuperar, actualizar y borrar los datos de una base de datos.²

Para conocer que motores están disponibles, basta con escribir en una consola de MySQL la siguiente consulta:

show engines;

Y obtendremos algo como lo siguiente:

Engine	Support	Comment
MyISAM	DEFAULT	Default engine as of MySQL 3.23 with great performance
MEMORY	YES	Hash based, stored in memory, useful for temporary tables
InnoDB	YES	Supports transactions, row-level locking, and foreign keys
BerkeleyDB	NO	Supports transactions and page-level locking
BLACKHOLE	YES	/dev/null storage engine (anything you write to it disappears)
EXAMPLE	NO	Example storage engine
ARCHIVE	YES	Archive storage engine
CSV	YES	CSV storage engine
ndbcluster	DISABLED	Clustered, fault-tolerant, memory-based tables
FEDERATED	YES	Federated MySQL storage engine
MRG_MYISAM	YES	Collection of identical MyISAM tables
ISAM	NO	Obsolete storage engine

¹ (Atienza, 2008-2015)

² (Mario López, 2008)

- TIPOS DE MOTORES DE ALMACENAMIENTO

Durante esta práctica solamente nos centraremos en dos motores de almacenamientos de los previamente mostrados, los cuales son: **MyISAM**, **MERGE** y **InnoDB**.

MyISAM.

Se basa en el antiguo ISAM, al que añade muchas mejoras, es el motor que usa MySQL por defecto. Es una buena combinación entre funcionalidad y rendimiento, aunque carece de algunas características interesantes.

Características

Límite de 2 ³² registros.	BLOB ³ y TEXT pueden ser índices.
Máximo de 64 índices por tabla.	Permite un gran tamaño en las tablas (hasta 256TB)
Máximo de 16 columnas por índice.	No soporta transacciones.
Los datos son independientes de la máquina y el sistema operativo.	Bloquea los datos a nivel de tabla.
Permite campos índices como NULL	No permite claves ajenas.

Este motor pone especial empeño en la rapidez de las operaciones de lectura (predominio de SELECT), es una de las razones por las que MySQL es tan popular en la web, ya que la mayoría de las operaciones que se realizan son de este tipo. Que no tenga que hacer comprobaciones de integridad referencial también influye en su velocidad.

Ejemplo

```
CREATE TABLE pruebaMyISAM (
    codigo varchar(5) default NOT NULL,
    descripcion varchar(255) default NULL,
    PRIMARY KEY (codigo)
) ENGINE=MyISAM;
```

En general no hará falta indicar el uso de este motor pues es el que se usa por defecto.

MERGE.

Permite combinar varias tablas de igual estructura en una única tabla, pudiendo así realizar consultas sobre una tabla que nos devuelve datos de varias.

Límite de 2 ³² registros.	No permite REPLACE
Las tablas “base” deben ser MyISAM	No soporta transacciones
Bloqueo a nivel de tabla	En su creación no comprueba que las tablas que usa existan y tengan una estructura idéntica.
No tiene índices, usa los de las tablas “base” (salvo FULLTEXT)	La lectura es más lenta al tener que ir consultando la clava en cada una de las tablas subyacentes.

³ Binary Large Object (Generalmente son imágenes, archivos de sonido y otros objetos multimedia)

Una de sus funcionalidades puede ser partir una tabla muy grande en otras más pequeñas y, al unirlos con MERGE, permitirnos trabajar con ellas como si fuesen una sola.

```
mysql> CREATE TABLE t1 (
->   a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
->   message CHAR(20));
mysql> CREATE TABLE t2 (
->   a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
->   message CHAR(20));
mysql> INSERT INTO t1 (message) VALUES ('Testing'),('table'),('t1');
mysql> INSERT INTO t2 (message) VALUES ('Testing'),('table'),('t2');
mysql> CREATE TABLE total (
->   a INT NOT NULL AUTO_INCREMENT,
->   message CHAR(20), INDEX(a))
->   TYPE=MERGE UNION=(t1,t2) INSERT_METHOD=LAST;
```

InnoDB.

- Está considerado como uno de los motores más avanzados para el almacenamiento de datos en MySQL. Provee un motor sólido con soporte completo de transacciones (es ACID compliant), permite el bloqueo de datos a nivel de registro permitiendo gran flexibilidad a la hora de utilizar las tablas, controla la integridad referencial, permite claves ajenas y tiene un sistema de recuperación de caídas. No obstante, la piedra de toque de InnoDB es su mecanismo de indexación y cache de los registros pues mantiene una caché de índices y datos en memoria y en disco proporcionando un muy alto rendimiento.

ACID Compliant	Permite claves ajenas y transacciones, soporte de integridad referencial.
El tamaño máximo para una tabla es de 64TB.	No permite índices de FULLTEXT
No mantiene un contador interno de registros (selec count (*) from tabla lento al tener que recorrer todo el índice)	Sistema de recuperación de caídas.
Cambiar la ubicación de la base de datos/tabla es complicado.	Los tamaños de sus logs deben ser inferior a 4GB.
Una tabla no puede tener más de 1000 columnas.	Bloqueo de datos a nivel de registro y no bloquea la lectura durante los selects (mejora la concurrencia)

Ejemplo

```
mysql> CREATE TABLE CUSTOMER (A INT, B CHAR (20), INDEX (A))
-> ENGINE=InnoDB;
Query OK, 0 rows affected (0.00 sec)
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO CUSTOMER VALUES (10, 'Heikki');
Query OK, 1 row affected (0.00 sec)
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
mysql> SET AUTOCOMMIT=0;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO CUSTOMER VALUES (15, 'John');
Query OK, 1 row affected (0.00 sec)
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)
mysql> SELECT * FROM CUSTOMER;
+-----+-----+
| A    | B      |
+-----+-----+
| 10   | Heikki |
+-----+-----+
1 row in set (0.00 sec)
```

RESTRICCIONES DE INTEGRIDAD

Durante esta práctica solamente abarcaremos restricciones de integridad CASCADE, ya que es la restricción que hemos utilizado en las prácticas desarrolladas.

Las restricciones de integridad referencial en cascada permiten definir las acciones que MySQL lleva a cabo cuando un usuario intenta eliminar o actualizar una clave a la que apuntan las claves externas existentes.

Las cláusulas REFERENCES de las instrucciones CREATE TABLE y ALTER TABLE admiten las cláusulas ON DELETE y ON UPDATE. Las acciones en cascada también se pueden definir mediante el cuadro de diálogo Relaciones de clave externa.

- [ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }]
- [ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }]

NO ACTION es el valor predeterminado si no se especifica ON DELETE u ON UPDATE.

ON DELETE NO ACTION

Especifica que si se intenta eliminar una fila con una clave a la que hacen referencia las claves externas de las filas existentes en otras tablas, se produce un error y se revierte la instrucción DELETE.

ON UPDATE NO ACTION

Especifica que si se intenta actualizar un valor de clave en una fila a cuya clave hacen referencia las claves externas de filas existentes en otras tablas, se produce un error y se revierte la instrucción UPDATE.

ON DELETE CASCADE

Especifica que si se intenta eliminar una fila con una clave a la que hacen referencia claves externas de filas existentes en otras tablas, todas las filas que contienen dichas claves externas también se eliminan.

ON UPDATE CASCADE

Especifica que si se intenta actualizar un valor de clave de una fila a cuyo valor de clave hacen referencia claves externas de filas existentes en otras tablas, también se actualizan todos los valores que conforman la clave externa al nuevo valor especificado para la clave.⁴

⁴ (Q., 2013)

DESARROLLO DE LA PRÁCTICA.

INSTRUCCIONES Y CODIFICACIÓN

1. Crear una base de Datos

```
create database cine;
```

2. Crear las relaciones propietarias.....miembro

```
create table empleado(  
    idEmpleado int not null primary key,  
    nombre varchar(30),  
    dir varchar(100),  
    tel int,  
    genero varchar(10)  
);  
  
create table cinemex(  
    idCinemex int not null primary key,  
    nombre varchar(50),  
    dir varchar(100),  
    tel int,  
    email varchar(60)  
);
```

Crear la relacion miembro

```
create table ec(  
    idEmpleado int not null,  
    idCinemex int not null,  
    primary key(idEmpleado,idCinemex),  
  
    foreign key (idEmpleado) references Empleado(idEmpleado)  
    on delete cascade on update cascade,  
    foreign key(idCinemex) references Cinemex(idCinemex)  
    on delete cascade on update cascade  
);
```

PARA CAMBIAR LA OPCION DE LA TABLA
ENGINE = InnoDB

1. alter table _____
2. ENGINE = InnoDB;

No hacer.

3. Agregar dos campos que permitan almacenar el salario y el correo electronico en los empleados.

```
alter table empleado add column salario double;  
alter table empleado add column email varchar(60);
```

Verificar: desc empleado;

4. Crear la relacion gerente

```
Gerente( idGerente, nombre, turno, nocel, salario)
```

Y ademas vamos a tener una llave forantea FK(idCinemex)

```
create table gerente(  
idGerente int not null primary key,  
nombre varchar(50),  
turno varchar(15),  
nocel int,  
salario double,  
  
idCinemex int,  
foreign key(idCinemex) references cinemex(idCinemex)  
on delete cascade on update cascade  
);
```

```
Verificar: desc gerente;  
show create table gerente;
```

5. Cambiar el tipo de dato del nocel gerente a **varchar**

```
alter table gerente MODIFY COLUMN nocel varchar(15);
```

```
Verificar: desc gerente;
```

6. Renombar la relación empleado y llamarle Asociado

```
alter table empleado rename as Asociado;
```

```
Verificar: desc gerente;
```

7. Aumentar el tamaño de tipo de dato de la dir de asociado

```
alter table Asociado modify column dir varchar(200);
```

```
Verificar: desc gerente;
```

8.

1. Eliminar llave **primary** pero primero correspondientes a las intermedias, las **foreign**

2. REDEFINIR las llaves primarias

3. Asociarlo con gerente

8.1 Eliminar PK

```
alter table Cinemex drop primary key; (marcará error).
```

8.2 Eliminar FK con las que se relacionan

```
show create table gerente;
```

(idCinemex es la FK con la que se hace referencia)

```
alter table gerente drop foreign key gerente_ibfk_1;
```

Verificar : show **create table** gerente;

desc ec;

```
show create table ec;
```

//escogemos la FK que hace referencia a idCinemex

```
alter table ec drop foreign key ec_ibfk_2;
```

Verificar : show **create table** ec;

8.3 Eliminar la PK ahora si

```
alter table Cinemex drop primary key;
```

8.4 Redefinir la PK.

```
alter table cinemex add primary key(idCinemex, nombre);
```

8.5 Asociarlo con el gerente.

8.5.1 agregar la sucursal

```
alter table gerente add column nomCine varchar(50);
```

8.6 Asociarlo con ec.

8.6.1 agregar la sucursal

```
alter table ec add column nomCine varchar(50);
```

//La **variable** nomCine debe tener el mismo tipo de dato con la misma cantidad para que **no** haya complicaciones en las consultas.

8.7 Haremos una llave foranea compuesta entre gerente y ec;

8.7.1 Agregar la llave foranea sobre cinemex de gerente y ec.

```
alter table gerente add foreign key(idCinemex,nomCine)  
references cinemex(idCinemex, nombre)  
on delete cascade on update cascade;
```

La llave primaria es a quien se hace referencia, en donde debe de ser el mismo tipo de dato.

Verificar: **desc** gerente;
show **create table**;

```
alter table ec add foreign key(idCinemex,nomCine)  
references cinemex(idCinemex, nombre)  
on delete cascade on update cascade;
```

8.8 Crear la relacion cartelera

```
cartelera(  
    idCartelera,nombre, fechaInicio,fechaFin, clasificacion)
```

```
create table cartelera(  
    idCartelera int not null primary key,  
    nombre varchar(50),  
    fechaInicio date,  
    fechaFin date,  
    clasificacion varchar(4)  
);
```

Verificar: **desc** Cartelera;

8.9 Asociarla con Cinemex

```
cinemex(idCinemex, ....., FK(idCartelera));
```

desc Cinemex;

```
alter table cinemex add column idCartelera int;
```

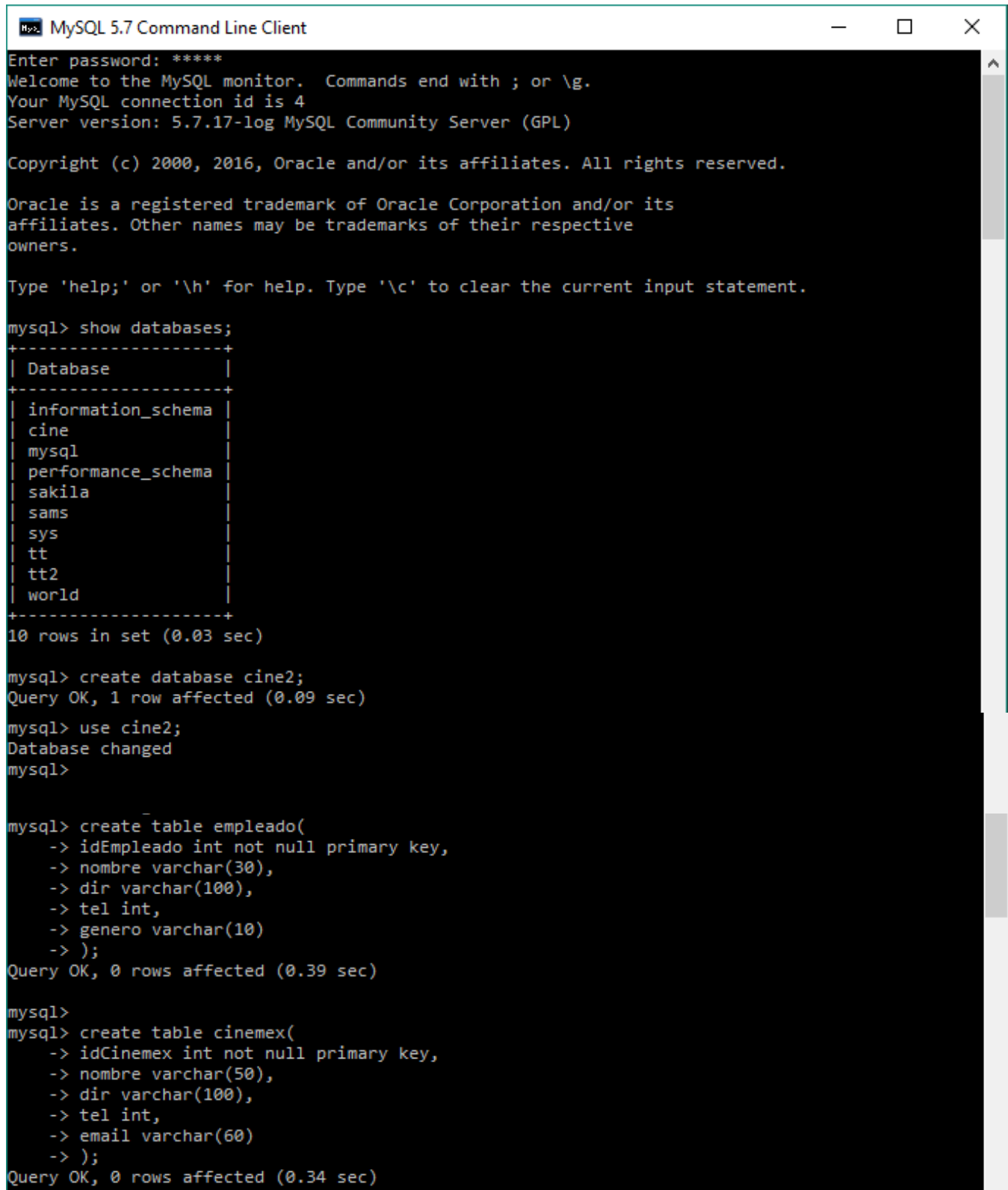
```
alter table cinemex add foreign key(idCartelera) references  
cartelera(idCartelera)on delete cascade on update cascade;
```

RESPALDO PARA UNA BD:

Entrar a Archivos de programa/mysql/mysqlServer 5.7/bin

mysqldump -v root -p nombreBD > path:\archivo.sql -> Cuando
una maquina **no** puede conectarse

ENTORNO DE DESARROLLO DE LA PRÁCTICA (SCREENSHOTS)



The screenshot shows a MySQL 5.7 Command Line Client window. The terminal displays the following text:

```
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.17-log MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

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> show databases;
+-----+
| Database |
+-----+
| information_schema |
| cine      |
| mysql     |
| performance_schema |
| sakila    |
| sams      |
| sys       |
| tt        |
| tt2       |
| world     |
+-----+
10 rows in set (0.03 sec)

mysql> create database cine2;
Query OK, 1 row affected (0.09 sec)

mysql> use cine2;
Database changed
mysql>

mysql> create table empleado(
  -> idEmpleado int not null primary key,
  -> nombre varchar(30),
  -> dir varchar(100),
  -> tel int,
  -> genero varchar(10)
  -> );
Query OK, 0 rows affected (0.39 sec)

mysql>
mysql> create table cinemex(
  -> idCinemex int not null primary key,
  -> nombre varchar(50),
  -> dir varchar(100),
  -> tel int,
  -> email varchar(60)
  -> );
Query OK, 0 rows affected (0.34 sec)
```

```

mysql> create table ec(
-> idEmpleado int not null,
-> idCinemex int not null,
-> primary key(idEmpleado,idCinemex),
->
-> foreign key (idEmpleado) references Empleado(idEmpleado)
-> on delete cascade on update cascade,
-> foreign key(idCinemex) references Cinemex(idCinemex)
-> on delete cascade on update cascade
-> );
Query OK, 0 rows affected (0.56 sec)

mysql> alter table empleado add column salario double;
Query OK, 0 rows affected (0.72 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table empleado add column email varchar(60);
Query OK, 0 rows affected (0.54 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> create table gerente(
-> idGerente int not null primary key,
-> nombre varchar(50),
-> turno varchar(15),
-> nocel int,
-> salario double,
->
-> idCinemex int,
-> foreign key(idCinemex) references cinemex(idCinemex)
-> on delete cascade on update cascade
-> );
Query OK, 0 rows affected (0.43 sec)

mysql> desc gerente;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idGerente  | int(11)       | NO   | PRI | NULL    |       |
| nombre     | varchar(50)   | YES  |     | NULL    |       |
| turno      | varchar(15)   | YES  |     | NULL    |       |
| nocel      | int(11)       | YES  |     | NULL    |       |
| salario    | double        | YES  |     | NULL    |       |
| idCinemex  | int(11)       | YES  | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> alter table gerente MODIFY COLUMN nocel varchar(15);
Query OK, 0 rows affected (0.93 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table empleado rename as Asociado;
Query OK, 0 rows affected (0.15 sec)

mysql> alter table Asociado modify column dir varchar(200);
Query OK, 0 rows affected (0.25 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table Cinemex drop primary key;
ERROR 1025 (HY000): Error on rename of '.\cine2\#sql-95c_4' to '.\cine2\cinemex' (errno: 150 -
Foreign key constraint is incorrectly formed)
mysql>

```

```

mysql> alter table gerente drop foreign key gerente_ibfk_1;
Query OK, 0 rows affected (0.14 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> show create table gerente;

mysql> desc ec;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idEmpleado | int(11)   | NO   | PRI | NULL    |       |
| idCinemex  | int(11)   | NO   | PRI | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> alter table ec drop foreign key ec_ibfk_2;
Query OK, 0 rows affected (0.15 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> show create table ec;

mysql> alter table Cinemex drop primary key;
Query OK, 0 rows affected (0.74 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table cinemex add primary key(idCinemex, nombre);
Query OK, 0 rows affected (0.46 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table gerente add column nomCine varchar(50);
Query OK, 0 rows affected (0.55 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table ec add column nomCine varchar(50);
Query OK, 0 rows affected (0.57 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table gerente add foreign key(idCinemex,nomCine)
-> references cinemex(idCinemex, nombre)
-> on delete cascade on update cascade;
Query OK, 0 rows affected (0.74 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> desc gerente;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idGerente  | int(11)   | NO   | PRI | NULL    |       |
| nombre     | varchar(50) | YES  |     | NULL    |       |
| turno     | varchar(15) | YES  |     | NULL    |       |
| nocel      | varchar(15) | YES  |     | NULL    |       |
| salario    | double     | YES  |     | NULL    |       |
| idCinemex  | int(11)   | YES  | MUL | NULL    |       |
| nomCine    | varchar(50) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

```

```
mysql> alter table ec add foreign key(idCinemex,nomCine)
-> references cinemex(idCinemex, nombre)
-> on delete cascade on update cascade;
Query OK, 0 rows affected (0.85 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> create table cartelera(
-> idCartelera int not null primary key,
-> nombre varchar(50),
-> fechaInicio date,
-> fechaFin date,
-> clasificacion varchar(4)
-> );
Query OK, 0 rows affected (0.43 sec)
```

```
mysql> desc Cinemex;
```

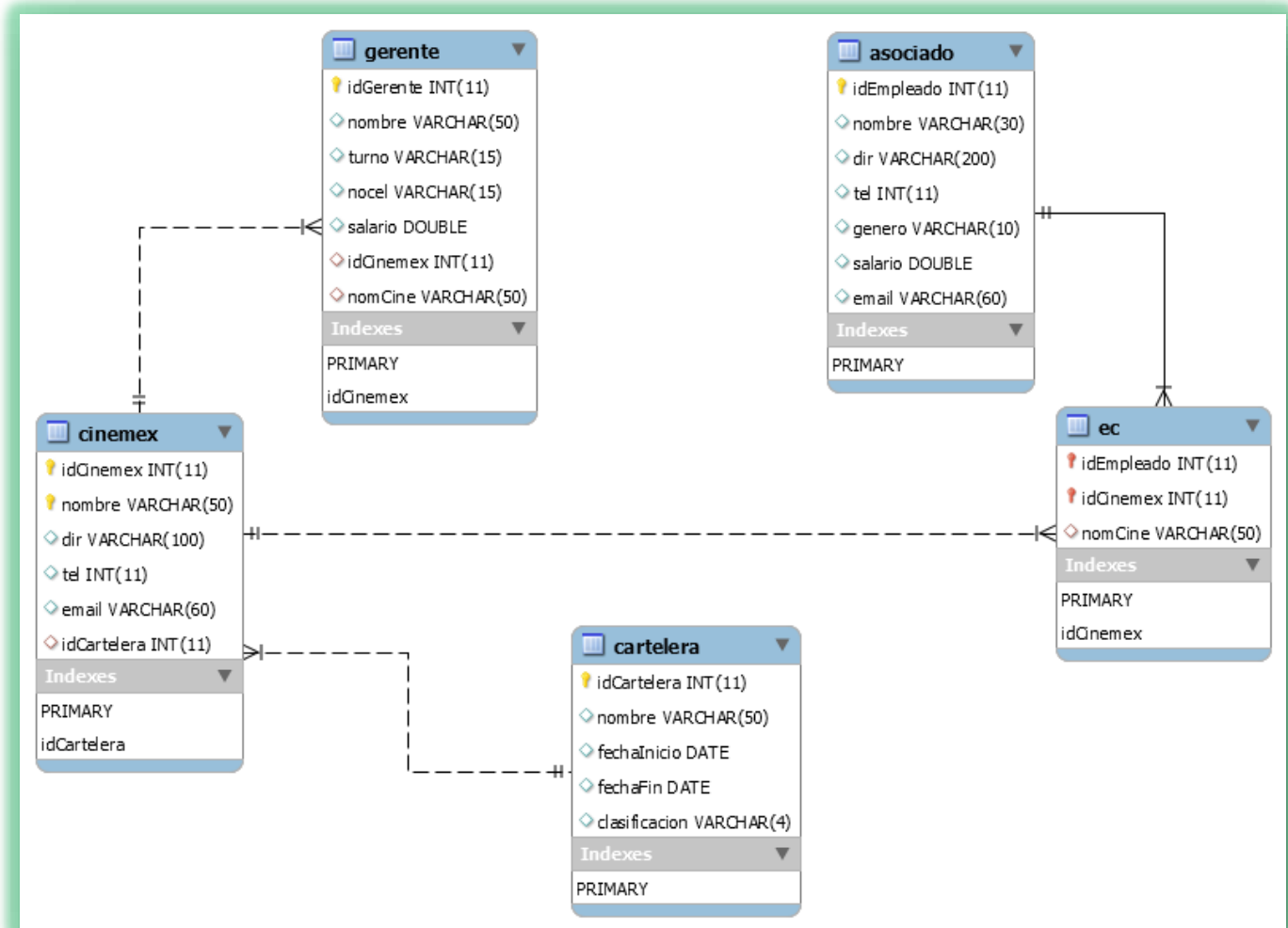
Field	Type	Null	Key	Default	Extra
idCinemex	int(11)	NO	PRI	NULL	
nombre	varchar(50)	NO	PRI	NULL	
dir	varchar(100)	YES		NULL	
tel	int(11)	YES		NULL	
email	varchar(60)	YES		NULL	

```
5 rows in set (0.00 sec)
```

```
mysql>
mysql> alter table cinemex add column idCartelera int;
Query OK, 0 rows affected (0.59 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql>
mysql> alter table cinemex add foreign key(idCartelera) references cartelera(idCartelera)
-> on delete cascade on update cascade;
Query OK, 0 rows affected (0.85 sec)
Records: 0 Duplicates: 0 Warnings: 0
```


DIAGRAMA ENTIDAD-RELACION



CONCLUSIONES

Durante esta práctica se implementó el Lenguaje de Definición de Datos (DML) para la creación de la base de datos, el cual ayudó a comprender de manera gráfica lo visto en clase con respecto a varios temas. De los más sobresalientes, son los niveles de aislamiento con relación a los objetos de una base de datos, de los cuales se establecieron sus singulares restricciones (llaves primarias y foráneas) gracias a un modelo de entidad-relación expuesto durante la práctica (por el profesor).

Asimismo, ayudó para la comprensión del modelo lógico de una base de datos, así como su funcionamiento. Y aunque no se viera netamente la arquitectura de tres capas, realice una analogía con el esquema conceptual de dicha edificación, ya que establecimos un contexto definido para el uso de nuestra base de datos: un cine, con sus respectivos metadatos dentro de cada entidad. Cabe mencionar que cuando creábamos cada tabla de la base de datos se me facilitaba ya que relacionaba cada dato creado con su respectiva independencia lógica. Además, pude vincular los conceptos del evaluador de operadores al ir creando, por consola y editor de textos, el repositorio siguiendo el análisis léxico y sintáctico de MySQL.

Para así concluir de manera exitosa la práctica número 2.

BIBLIOGRAFÍA

Atienza, A. L. (2008-2015). *lopezatienda.com*. Obtenido de <http://www.lopezatienda.com/mysql/mysql-sentencias-ddl-en-mysql/>

Belisario, C. (07 de Enero de 2011). *Desarrollo PHP para todos*. Obtenido de <http://desphpparatodos.blogspot.mx: http://desphpparatodos.blogspot.mx/2011/01/introduccion-mysql-createalterdrop-ddl.html>

Contreras, M. e. (Febrero de 2017). Unidad I. Introducción a las Bases de Datos. Ciudad de México, México.

Flórez, W. P. (2014). Descripción de los distintos tipos de datos de MySQL. Obtenido de <http://slideplayer.es/slide/1023079/>

Mario López, J. A. (15 de 01 de 2008). Obtenido de Motores de Almacenamiento en MySQL 5.0: <https://iessanvicente.com/colaboraciones/motoresMySQL.pdf>

Q., E. (20 de 12 de 2013). *Microsoft*. Obtenido de Microsoft TechNet: [https://technet.microsoft.com/es-es/library/ms186973\(v=sql.105\).aspx](https://technet.microsoft.com/es-es/library/ms186973(v=sql.105).aspx)

Rodríguez., J. R. (2011-2012). Introducción a las Bases de Datos. .