

---

# **BASES DE DATOS**

---

Practica 01

**DIAZ MEDINA JESÚS KAIMORTS**

**Viernes, 10 de Febrero del 2017.**

---

# INDICE

---

MARCO TEÓRICO.....	2
DESARROLLO DE LA PRÁCTICA.....	8
INSTRUCCIONES Y CODIFICACIÓN.....	8
ENTORNO DE DESARROLLO DE LA PRÁCTICA (SCREENSHOTS).....	10
DIAGRAMA ENTIDAD-RELACION.....	13
CONCLUSIONES.....	14
BIBLIOGRAFÍA .....	14

# BASES DE DATOS

## Practica 01

### MARCO TEÓRICO

En el desarrollo de ésta práctica se creó una base de datos cuyo conjunto de entidades creadas, formaron un producto terminal constituido por tablas como: tt, profesor, departamento y presentación. Asimismo, se hizo una asociación llamada dirige, entre profesor y tt, la cual posteriormente se adaptó como una entidad sin qué se viera afectada.

Para poder comprender el desarrollo y los términos utilizados dentro de esta práctica es necesario conocer la conceptualización que apoyaron a la creación de la base de datos, así como la de sus entidades utilizadas.

Comenzaremos por definir una **base de datos**, la cual es una estructura que aloja un conjunto de datos relacionados entre sí. Dicha estructura está basada en uno (o más) modelado de datos los cuales apoyan a las colecciones de registros para ser dirigidas a **medios de almacenamientos**: ópticos, magnéticos o de estado sólido, *con el fin de poder ser manipulados* para cualquier operación de escritura (  $e(x)$  ) o lectura (  $l(x)$  ) que se necesite.

Para realizar dichas operaciones, los sistemas de base de datos están compuestas de:

a) Aplicaciones.

Es la **interfaz** entre la *base de datos* y *el usuario*; estas pueden ser desarrolladas por un lenguaje de alto nivel y/o aplicaciones desarrolladas en un lenguaje semiestructurado.



Figura 1.1

### Datos

a) Conjunto de hechos relevantes que pueden ser registrados de algún modo, y que cuentan con un significado implícito.

b) Reflejan situaciones del mundo real y cambios en esas situaciones.

### Relacionados

Debe existir homogeneidad en la colección de datos que conforma una BD.

No se trata de un conjunto seleccionado de forma aleatoria.

Los datos se recopilan y registran con una finalidad.

Los datos deben ser relevantes con respecto a esa finalidad.

b) Conectores (drivers).

Son los componentes que permiten el enlace entre el SGBD y las interfaces desarrolladas en un lenguaje de programación. Éstas contienen las clases y/o funciones necesarias para llevar a cabo la comunicación entre las aplicaciones con el Sistema Gestor de Base de Datos.<sup>1</sup>



Imagen 1.2

c) Sistema Gestor de Base de Datos.

Es un software especializado que permite a los usuarios definir, crear y mantener la base de datos y proporciona acceso controlado a la misma. Éste conjunto de programas de propósito general, proporcionan funcionalidades horizontales para facilitar la gestión de la información contenida en una base de datos.

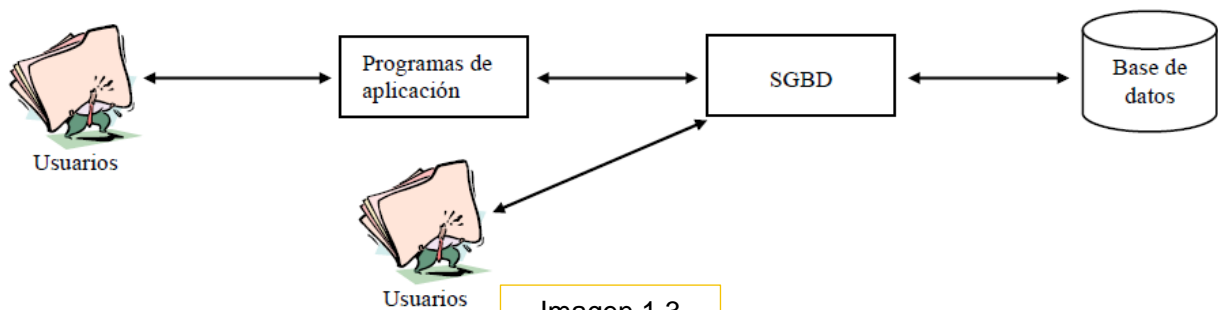


Imagen 1.3

Éstos actúan de intermediarios entre los datos y los programas de aplicación (y sus usuarios) que los procesan y utilizan.<sup>2</sup>

Cada uno de estos componentes se relacionan entre sí para componer un todo llamado repositorio, el cual aloja **datos** (y **metadatos** [atributos]), los cuales se leen a partir de **índices**. Estos se encargan de acelerar la búsqueda de los datos y otros objetos. Dichas consultas generan **log's** lo cuales crean bitácoras de: acceso, recuperación y transacciones; escritura (  $e(x)$  ) y lectura (  $l(x)$  ).

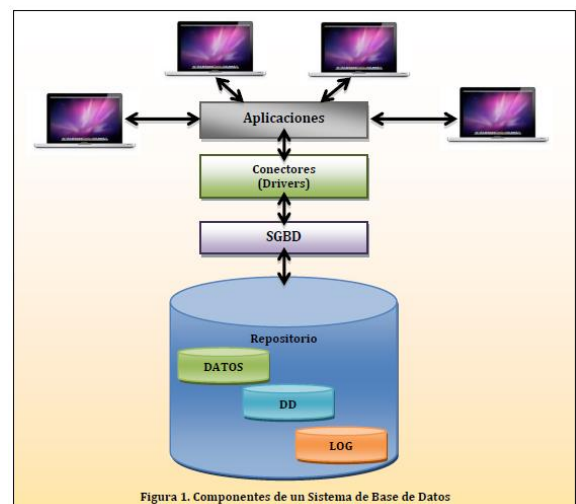


Figura 1. Componentes de un Sistema de Base de Datos

Imagen 1.4

<sup>1</sup> (Contreras, 2017)

<sup>2</sup> (Rodríguez., 2011-2012)

En esta práctica se utilizó la consola de MySQL para la codificación requerida de la base de datos en apoyo de un editor de textos usando sentencias del **lenguaje de definición de datos (DDL)** para construir dicha base de datos.

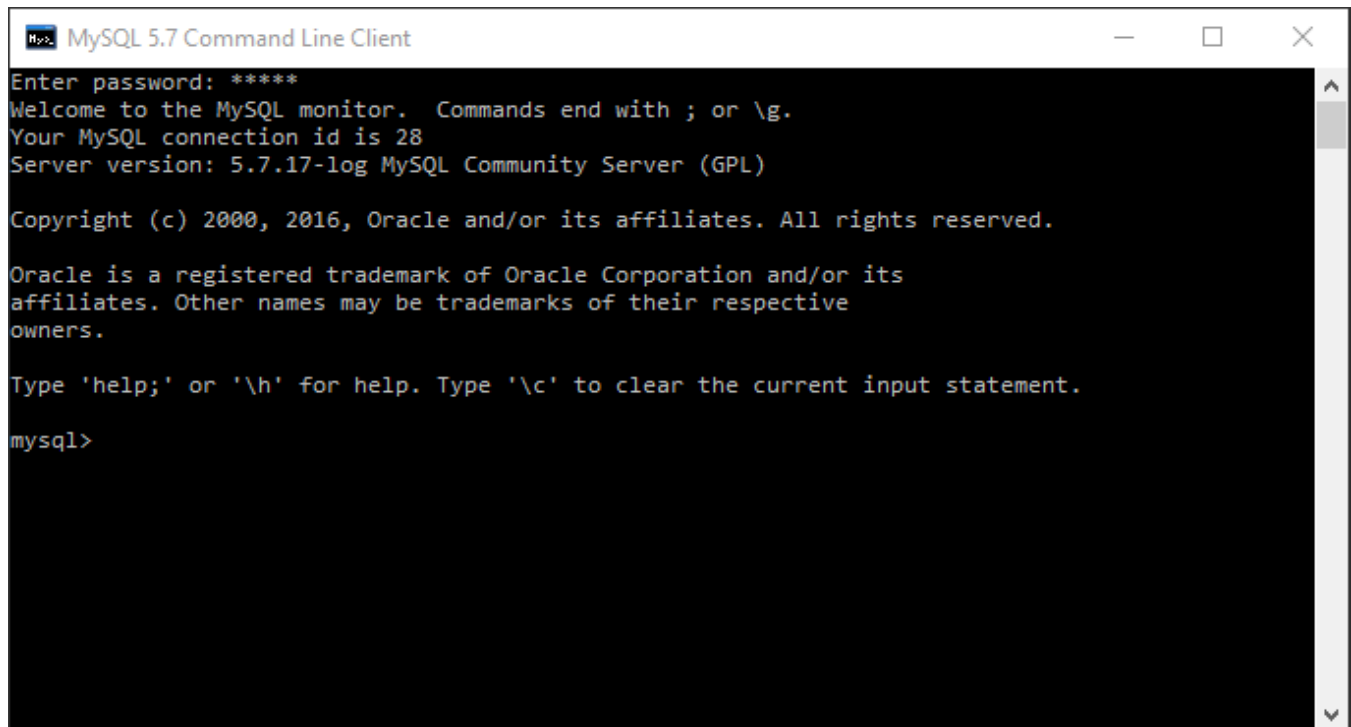
Abordaremos el DDL, así como sus comandos, definiendo algunos conceptos importantes para la comprensión de las sentencias que se presentarán.

- **CREACIÓN DE UNA BASE DE DATOS.**

La sentencia para crear una base de datos en MySQL es: `CREATE DATABASE nombreBD;`

Al crear una base de datos no se selecciona ésta de manera automática; debemos hacerlo de manera explícita, por ello usamos el comando **USE**.

La base de datos se crea sólo una vez, pero nosotros debemos seleccionarla cada vez que iniciamos una sesión con MySQL. Por ello es recomendable que se indique la base de datos sobre la que vamos a trabajar al momento de invocar al monitor de MySQL.



```
MySQL 5.7 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 28
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>
```

Imagen 2.1

- **CREACIÓN DE UNA TABLA.**

Las **tablas** son la estructura de datos básica en cualquier base de datos relacional. Una tabla es una *colección organizada de registros* (o filas), todas ellas *con los mismos atributos* (columnas o campos). Las columnas de la tabla describen la estructura de la misma y las restricciones de integridad de la tabla describen los datos que son válidos dentro de la misma.

La sentencia para la creación de tablas en MySQL es:

**CREATE TABLE nameBD (definición de columnas...);**

En definición de columnas, hay que especificar el nombre de la columna seguida de su tipo de datos.

#### • TIPOS DE DATOS.

- Numéricos:** Como su nombre lo indica representan a los campos que contendrán valores numéricos, como por ejemplo la edad, precios, etc. (Figura 2.2)
- Datos de fecha y hora:** Como su nombre lo indica son para almacenar datos de tipo temporales (fecha y hora), como por ejemplo la fecha de nacimiento, el horario de una clase, etc. (Figura 2.3)
- Datos de Tipo String o Cadenas de Texto:** Este tipo de datos lo utilizamos para almacenar las cadenas de texto que necesitemos dentro de nuestra base de datos, como por ejemplo un nombre, apellido etc. (Figura 2.4).<sup>3</sup>

TIPO DE CAMPO	TAMAÑO DE ALMACENAMIENTO
CHAR(n)	n bytes
VARCHAR(n)	n +1 bytes
TINYBLOB, TINYTEXT	Longitud+1 bytes
BLOB, TEXT	Longitud +2 bytes
MEDIUMBLOB, MEDIUMTEXT	Longitud +3 bytes
LOBLOB, LONGTEXT	Longitud +4 bytes
ENUM('value1','value2',...)	1 ó dos bytes dependiendo del número de valores
SET('value1','value2',...)	1, 2, 3, 4 ó 8 bytes, dependiendo del número de valores

Figura 2.4

TIPO DE CAMPO	TAMAÑO DE ALMACENAMIENTO
TINYINT	1 byte
SMALLINT	2 bytes
MEDIUMINT	3 bytes
INT	4 bytes
INTEGER	4 bytes
BIGINT	8 bytes
FLOAT(X)	4 ó 8 bytes
FLOAT	4 bytes
DOUBLE	8 bytes
DOUBLE PRECISION	8 bytes
REAL	8 bytes
DECIMAL(M,D)	M+2 bytes si D > 0, M+1 bytes si D = 0
NUMERIC(M,D)	M+2 bytes if D > 0, M+1 bytes if D = 0

Figura 2.2

TAMAÑO	FORMATO
14	AñoMesDiaHoraMinutoSegundo aaaammddhhmmss
12	AñoMesDiaHoraMinutoSegundo aammddhhmmss
8	ñoMesDia aaaammdd
6	AñoMesDia aammdd
4	AñoMes aamm
2	Año aa

Figura 2.3

<sup>3</sup> (Belisario, 2011) y (Flórez, 2014).

- **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.<sup>4</sup>

---

<sup>4</sup> (Atienza, 2008-2015)



# DESARROLLO DE LA PRÁCTICA.

## INSTRUCCIONES Y CODIFICACIÓN

1. Crear una base de datos

```
create database tt;
```

2. Usar el dbSPACE

```
use tt;
```

```
select database();
```

```
create table tt(  
nott int not null primary key,  
titulo varchar(100)  
);
```

```
create table profesor(  
idProf int not null primary key,  
nombre varchar(20),  
ap varchar(30),  
am varchar(30),  
academia varchar(50),  
salario double,  
idDepto int,  
foreign key(idDepto) references Depto(idDepto)  
on delete cascade on update cascade  
);
```

```
create table depto(  
idDepto int not null primary key,  
nombre varchar(50)  
);
```

```
create table presentacion(  
idPresentacion int not null primary key,  
fecha date,  
califRevisor float,  
califSinodales float,  
tipo varchar(30)  
);
```

```
create table dirige(  
idProf int not null,  
nott int not null,  
primary key(idProf, nott),  
foreign key(idprof) references Profesor(idprof)  
on delete cascade on update cascade,  
foreign key(nott) references tt(nott)  
on delete cascade on update cascade  
);
```

1. Renombrar la relación profesor y llamarle catedratico

```
alter table profesor RENAME AS catedratico;
```

2. Agregar un campo en la relación presentación que almacene el dictamen

```
alter table presentacion ADD COLUMN dictamen varchar(15);
```

3. Renombrar el campo nombre de Depto y llamarle depto

```
alter table depto CHANGE COLUMN nombre depto varchar(50);
```

4. Agregar la FK en presentacion

```
Presentacion(idPresentacion, ....., nott(FK))
```

a) Agregar el campo foraneo...  
agregar.. nott

```
alter table presentacion add column nott int;
```

b) Agregar el **constraint**.... FK

```
alter table presentacion add  
FOREIGN KEY(nott) references tt(nott)  
on delete cascade on update cascade;
```

5. Agregar un campo en la relación catedrático para almacenar el tel.

```
alter table catedratico add column tel int;
```

6. cambiar la definición de la PK en la relación presentación

```
primary key(idpresentacion, fecha)
```

a) Eliminar la PK.....

```
alter table presentacion  
drop primary key;
```

b) Hacer el **constraint** PK compuesta...

```
alter table presentacion  
add primary key(idpresentacion, fecha);
```

Eliminar la FK de la relación catedratico....

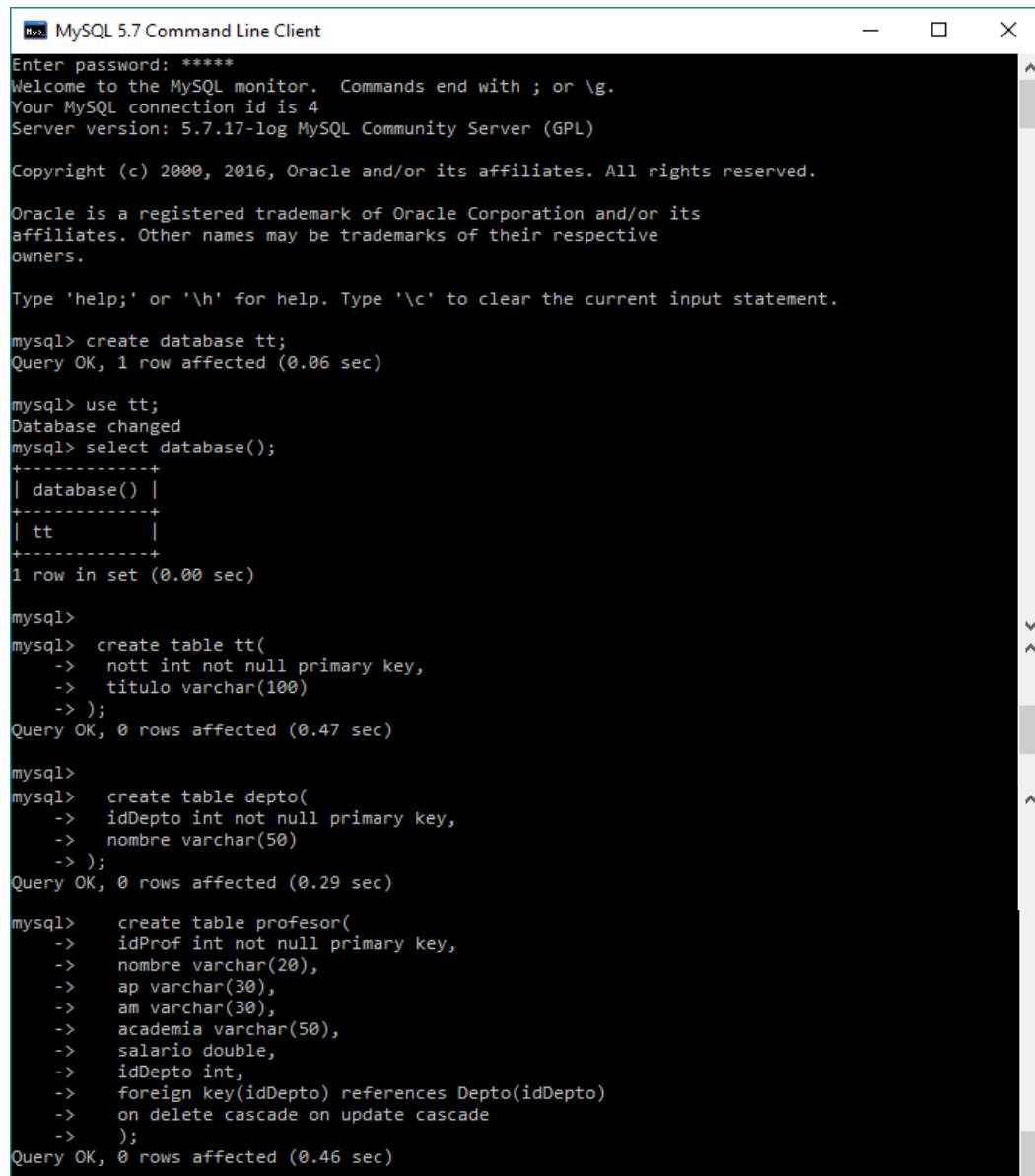
1. conocer el identificador del **constraint**...

```
show create table catedratico
```

2. Eliminar la FK

```
alter table catedratico  
drop foreign key catedratico_ibfk_1;
```

## ENTORNO DE DESARROLLO DE LA PRÁCTICA (SCREENSHOTS)



```
MySQL 5.7 Command Line Client
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> create database tt;
Query OK, 1 row affected (0.06 sec)

mysql> use tt;
Database changed
mysql> select database();
+-----+
| database() |
+-----+
| tt          |
+-----+
1 row in set (0.00 sec)

mysql>
mysql> create table tt(
  ->   nott int not null primary key,
  ->   titulo varchar(100)
  -> );
Query OK, 0 rows affected (0.47 sec)

mysql>
mysql> create table depto(
  ->   idDepto int not null primary key,
  ->   nombre varchar(50)
  -> );
Query OK, 0 rows affected (0.29 sec)

mysql>
mysql> create table profesor(
  ->   idProf int not null primary key,
  ->   nombre varchar(20),
  ->   ap varchar(30),
  ->   am varchar(30),
  ->   academia varchar(50),
  ->   salario double,
  ->   idDepto int,
  ->   foreign key(idDepto) references Depto(idDepto)
  ->   on delete cascade on update cascade
  -> );
Query OK, 0 rows affected (0.46 sec)
```

```

mysql> create table presentacion(
-> idPresentacion int not null primary key,
-> fecha date,
-> califRevisor float,
-> califSinodales float,
-> tipo varchar(30)
-> );
Query OK, 0 rows affected (0.44 sec)

mysql> create table dirige(
-> idProf int not null,
-> nott int not null,
-> primary key(idProf, nott),
-> foreign key(idProf) references Profesor(idprof)
-> on delete cascade on update cascade,
-> foreign key(nott) references tt(nott)
-> on delete cascade on update cascade
-> );
Query OK, 0 rows affected (0.45 sec)

mysql> alter table profesor RENAME AS catedratico;
Query OK, 0 rows affected (0.31 sec)

mysql> alter table presentacion ADD COLUMN dictamen varchar(15);
Query OK, 0 rows affected (0.73 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table depto CHANGE COLUMN nombre depto varchar(50);
Query OK, 0 rows affected (0.15 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> alter table profesor RENAME AS catedratico;
Query OK, 0 rows affected (0.31 sec)

mysql> alter table presentacion ADD COLUMN dictamen varchar(15);
Query OK, 0 rows affected (0.73 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table depto CHANGE COLUMN nombre depto varchar(50);
Query OK, 0 rows affected (0.15 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table presentacion add column nott int;
Query OK, 0 rows affected (0.50 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table presentacion add
-> FOREIGN KEY(nott) references tt(nott)
-> on delete cascade on update cascade;
Query OK, 0 rows affected (0.90 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table presentacion
-> drop primary key;
Query OK, 0 rows affected (1.02 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table presentacion
-> add primary key(idpresentacion, fecha);
Query OK, 0 rows affected (0.50 sec)
Records: 0 Duplicates: 0 Warnings: 0

```

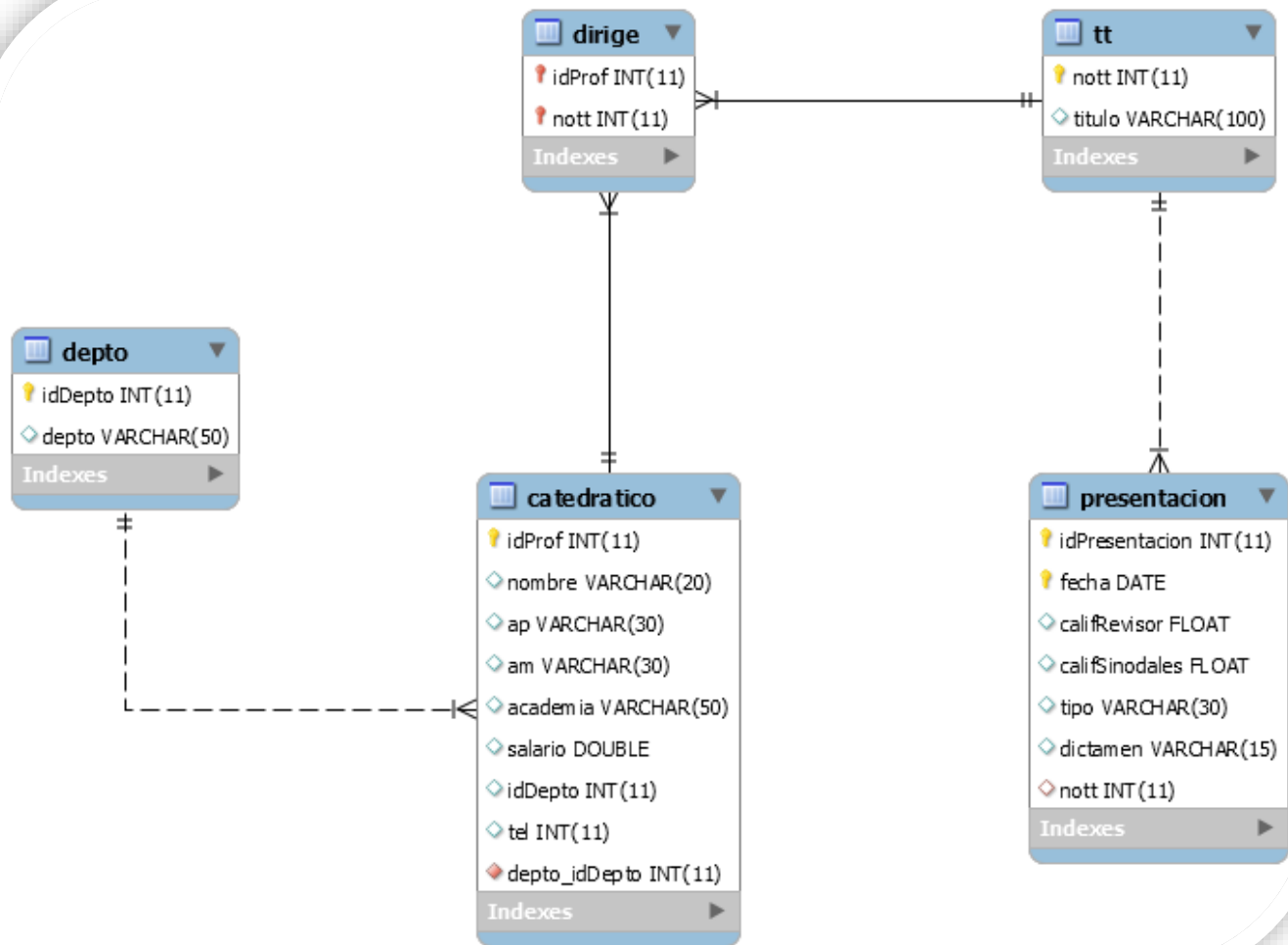
```

mysql> show create table catedratico
-> ;
+-----+-----+
| Table          | Create Table
+-----+-----+
| catedratico | CREATE TABLE `catedratico` (
  `idProf` int(11) NOT NULL,
  `nombre` varchar(20) DEFAULT NULL,
  `ap` varchar(30) DEFAULT NULL,
  `am` varchar(30) DEFAULT NULL,
  `academia` varchar(50) DEFAULT NULL,
  `salario` double DEFAULT NULL,
  `idDepto` int(11) DEFAULT NULL,
  `tel` int(11) DEFAULT NULL,
  PRIMARY KEY (`idProf`),
  KEY `idDepto` (`idDepto`),
  CONSTRAINT `catedratico_ibfk_1` FOREIGN KEY (`idDepto`) REFERENCES `depto` (`idDepto`) ON DE
LETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-----+-----+
1 row in set (0.08 sec)

mysql> alter table catedratico
-> drop foreign key catedratico_ibfk_1;
Query OK, 0 rows affected (0.14 sec)
Records: 0 Duplicates: 0 Warnings: 0

```

## DIAGRAMA ENTIDAD-RELACION



# CONCLUSIONES

En el desarrollo de esta práctica tuve un primer acercamiento con las bases de datos, así como con el lenguaje de MySQL y DDL (Lenguaje de Definición de Datos). De manera que cada creación y modificación de tablas y metadatos, la vinculaba con lo visto en las sesiones de teoría. Ya que pude comprender de manera gráfica lo que es un sistema de base de datos con un único repositorio, en donde todas las tablas se relacionaban de manera directa o indirectamente. Además, de ver cómo se realizan los **constraints** (restricciones) con las asociaciones necesarias y ver el proceso de operación al establecer la base de datos en un estado vacío, donde solamente cree el modelado de la base de datos, pero sin datos.

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