



Introducción a
la base de datos



ESTRUCTURA DE CONTENIDOS

	Pág.
Estructura de contenidos.....	2
Mapa de contenido.....	4
Desarrollo de contenidos.....	5
1. Generalidades de MySQL.	5
1.1. Componentes de MySQL.	5
1.2 Motores de almacenamiento de datos.	6
1.3. La consola de MySQL.	7
1.4 Tipos de datos.	8
3. Creación de la estructura de almacenamiento.	14
3.1. Creación de la base de datos.....	15
3.2. Creación de las tablas.	16
4. Modificación de la estructura de las tablas.....	21
5. Actualización e inserción de registros.	25
5.1. Inserción de registros.	25
5.1.2. Inserción de registros con llaves foráneas.	27
6. Consultas de registros.....	29
6.1. Datos de prueba.	29
6.2. Consultas básicas.	30
6.3. Aplicación de filtros a las consultas.	33
6.4. Ordenamiento de las consultas.....	34
6.5. Introducción a las consultas multi-tablas.....	35
6.6. Subconsultas.	37
Glosario	40
Bibliografía.....	41
Control del documento	42

INTRODUCCIÓN A LA BASE DE DATOS MySQL

INTRODUCCIÓN

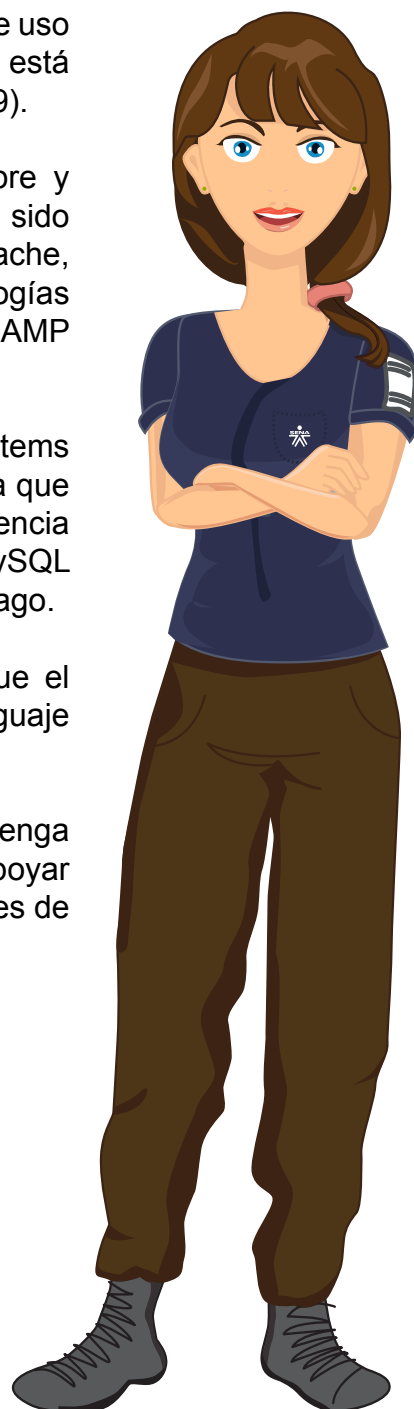
MySQL es un sistema manejador de bases de datos de libre uso y distribución bajo licencia GPL de los más utilizados y que está disponible para varios sistemas operativos (DUBOIS, 2009).

Su popularidad se debe principalmente a su licencia libre y a su facilidad de uso y administración. Por otra parte ha sido integrada con otras herramientas libres como son Linux, Apache, PHP, entre otras. Esta combinación e integración de tecnologías dio nombre a la plataforma de desarrollo conocida como LAMP (Linux, Apache, MySQL y PHP).

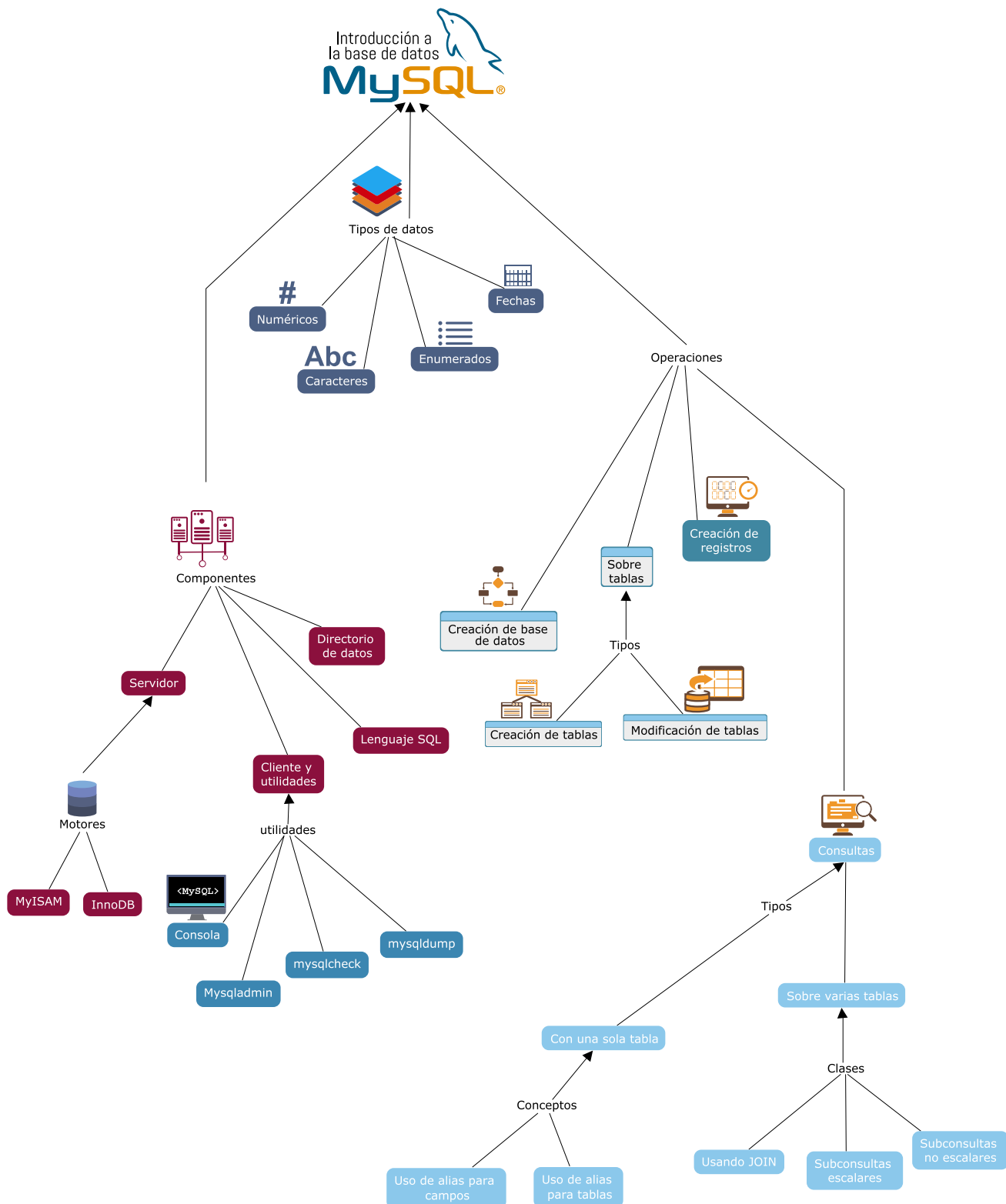
En años recientes MySQL fue adquirida por Sun Microsystems que luego fue adquirida por Oracle. Lo anterior no significa que Mysql deje de ser libre ya que su desarrollo está bajo la licencia GPL. Sin embargo, Oracle ofrece otras versiones de MySQL orientadas a empresas con modelo de licenciamiento de pago.

Para una mejor comprensión del recurso es necesario que el aprendiz haya estudiado los recursos de introducción al lenguaje SQL de la actividad de proyecto 6.

Para el desarrollo de este recurso se requiere que se tenga instalado el MySQL en el computador del aprendiz. Para apoyar la instalación existe un video tutorial en el área de materiales de la actividad de proyecto 6.



MAPA DE CONTENIDO



DESARROLLO DE CONTENIDOS

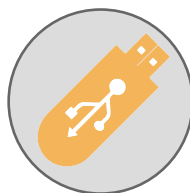
1. Generalidades de MySQL.

MySQL como manejador de bases de datos (SGBD) presenta las siguientes características (DUBOIS, 2009):



Velocidad

MySQL es veloz comparado con la mayoría de las bases libres.



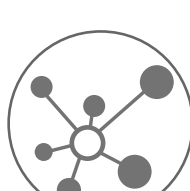
Portabilidad

MySQL corre en muchos sistemas operativos entre ellos windows, linux, unix.



Facilidad de uso

MySQL es de alto desempeño pero a la vez fácil de usar.



Conectividad

MySQL soporta distintos esquemas de conectividad y las bases de datos pueden ser accedidas desde cualquier sitio de Internet.



Soporta el lenguaje SQL

MySQL soporta el lenguaje estructurado de consultas (SQL).



Seguridad

MySQL maneja esquemas de seguridad que permiten asignar permisos a nivel de usuario.

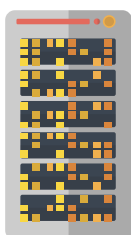


Robustez

El servidor MySQL es multi-hilo y puede atender varios usuarios de manera simultánea.

1.1. Componentes de MySQL.

Los componentes de MySQL se pueden dividir en cuatro (DUBOIS, 2009):



El servidor.

El programa principal que corre como un servicio tanto en plataformas Linux como windows y es quien coordina todas las operaciones del manejador.



El cliente y las utilidades.

MySQL suministra varias utilidades a saber:

- **MySQL:** Un programa interactivo que permite enviar consultas SQL al servidor. Requiere se puede llamar la consola.
- **mysqladmin:** Un programa administrativo para realizar tareas como subir y bajar el servidor, cambiar la configuración del servidor, monitorear el estado del mismo, entre otras.
- **mysqldump:** herramienta para hacer copias de seguridad o copiar datos a otros servidores.
- **mysqlcheck:** herramienta para realizar el chequeo, análisis y optimización de la base de datos.



Lenguaje SQL del servidor.

Implementación del lenguaje estándar SQL dentro del servidor.



El directorio de datos.

Es la ubicación donde se almacenan las bases de datos gestionadas por MySQL.

1.2 Motores de almacenamiento de datos.

Muchas de las características de las bases de datos están provistas por el motor de almacenamiento. MySQL trae varios motores pero los más usados son MyISAM e InnoDB (DUBOIS,2009).

1.2.1. Motor MyISAM.

Este es el motor por defecto usado por MySQL y representa cada tabla por medio de tres archivos en el sistema de archivos. Cada archivo tiene un nombre base que es el mismo que el de la tabla y una extensión que indica su función. Por ejemplo para una tabla llamada “clientes” existen los siguientes archivos:

- a. **clientes.frm:** es el archivo de formato que contiene la definición de la estructura de la tabla.
- b. **clientes.MYD:** es el archivo que contiene los datos de la tabla.
- c. **clientes.MYI:** es el archivo que contiene los índices de la tabla.

1.2.2. Motor InnoDB.

Este motor de almacenamiento está pensado para bases de datos transaccionales, es decir, que realizan operaciones que requieren ser tratadas como transacciones tipo CRUD (en inglés ACID).

Almacena las tablas en el sistema operativo de la siguiente manera:

- Un archivo .frm:** cada tabla de InnoDB es representada por un archivo .frm que contiene la definición de la estructura.
- Tablespace compartido:** consiste en uno o más archivos que contienen todos los datos de las tablas de manera contigua. Por defecto InnoDB almacena los datos de manera contigua.
- Tablespace individual:** se puede configurar InnoDB para que una tabla tenga un tablespace separado.

1.3. La consola de MySQL.

La consola es la utilidad que permite transmitir las instrucciones al servidor MySQL.

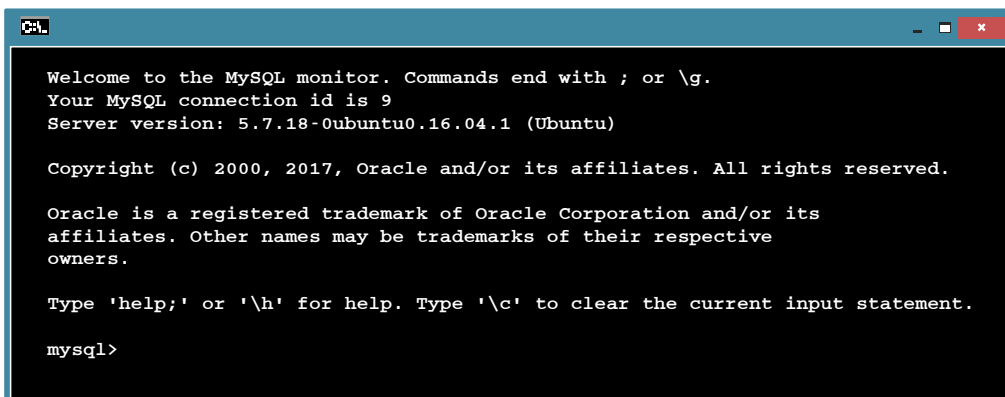
Para invocarla se usa la siguiente sintaxis:

```
$ mysql -h <nombre_del_host> -p -u <nombre_de_usuario>
```

Donde:

- h <nombre del host>: indica a cual servidor o host conectarse.
- u <nombre_de_usuario>: indica el nombre de usuario a conectar.
- p: indica al MySQL que pida el password del usuario.

Una vez ejecutado el comando aparece lo siguiente en pantalla:



```
mysql> Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.7.18-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, 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>
```

Figura 1.1. Consola de MySQL.

1.4 Tipos de datos.

1.4.1. Tipos Numéricos.

MySQL soporta todos los tipos de datos SQL numéricos estándar: los tipos de datos enteros y los tipos de datos en coma flotante.

Tipos de datos enteros.

TIPO	Bytes	Valor Mínimo (Con signo/Sin signo)	Valor Máximo (Con signo/Sin signo)
TINYINT	1	-128	127
		0	255
BIT (BOOL,BOOLEAN)	Número entero con valor 0 o 1. Sinónimo de TINYINT(1)		
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

Tabla 1.1. Tipos de datos enteros.

Tipos de datos en coma flotante.

Tipo	Tamaño
FLOAT (m,d)	Contiene un número en coma flotante de precisión sencilla. El valor M es la anchura a mostrar y D es el número de decimales.
DOUBLE (m,d)	Contiene un número en coma flotante de precisión doble. Igual que FLOAT la diferencia es el rango de valores posibles.
DECIMAL (m [,d])	Se usan para guardar valores para los que es importante preservar una precisión exacta, por ejemplo con datos monetarios. Ejemplo: salario DECIMAL(5,2) Si se omite D el valor por defecto es 0, los valores no tendrán punto decimal ni decimales.

Tabla 1.2. Tipos de datos en coma flotante.

Ejemplo:

Si en la base de Datos se requiere almacenar las edades de las personas, cuyo valor máximo será 100, la opción más adecuada para el tipo de dato sería “TINYINT”.

Si se requiere sistematizar las notas de un colegio y la nota definitiva se debe dar en decimales y el valor máximo es 10.00, la opción más adecuada es “FLOAT”.

1.4.2. Tipos de cadenas de caracteres.

Listado de cada uno de los tipos de dato con formato string en MySQL, su ocupación en disco y valores.

Tipo	Tamaño	Sintaxis
CHAR (M)	Los valores válidos para M son de 0 a 255 caracteres. Contiene una cadena de longitud constante. Para mantener la longitud de la cadena, se rellena a la derecha con espacios. Estos espacios se eliminan al recuperar el valor.	PacIdentificacion CHAR(10)
VARCHAR (M)	Los valores válidos para M son de 0 a 255 caracteres. Contiene una cadena de longitud variable. Los espacios al final se eliminan.	PacNombres VARCHAR(50)
BLOB	Una longitud máxima de 65.535 caracteres. Válido para objetos binarios como imágenes, ficheros de texto, audio o video.	PacImagenFoto BLOB
TEXT	Una longitud máxima de 65.535 caracteres. Sirve para almacenar texto plano sin formato. Distingue entre minúsculas y mayúsculas.	PacDescripcion TEXT
TINYBLOB TINYTEXT	Longitud máxima de 255 caracteres.	
MEDIUMBLOB MEDIUMTEXT	Longitud máxima de 16777215 caracteres	
LOB LONGTEXT	Longitud máxima de 4294967298 caracteres.	

SET	Contiene un conjunto. Un objeto de tipo cadena que puede tener cero o más valores, cada uno de los cuales debe estar entre una lista de valores 'valor1', 'valor2', ...	SET('valor1','valor2',...)
ENUM	Contiene un enumerado. Un objeto de tipo cadena que puede tener un único valor, entre una lista de valores 'valor1', 'valor2', ...,	ENUM('valor1','valor2',...)

Tabla 1.3. Tipos de datos de caracteres.

En tabla se observa que el campo PacIdentificacion se declaró como un “CHAR” de 10 caracteres porque los documentos de identidad tienen entre 1 y 10 caracteres.

Para almacenar nombres, direcciones e información con máximo de 100 caracteres se recomienda el tipo “VARCHAR”. En la tabla anterior el campo PacNombres se declaró varchar(50) porque los nombres tienen una cantidad de caracteres variable.

Para almacenar grandes cantidades de caracteres como descripciones, observaciones, comentarios en este caso se tomaría TEXT o BLOB.

1.4.3. Tipos de fecha y hora.

Los tipos de fecha y hora para representar valores temporales son:

TIPO	RANGO	FORMATO
DATE	Válido para almacenar una fecha con año, mes y día. Su rango oscila entre: '1000-01-01' y '9999-12-31'.	AAAA-MM-DD
DATETIME	Almacena una fecha y una hora. Su rango oscila entre '1000-01-01 00:00:00' y '9999-12-31 23:59:59'.	AAAA-MM-DD HH:MM:SS
TIME	Una hora. El rango está entre '-838:59:59' y '838:59:59'.	HH:MM:SS
TIMESTAMP	Almacena una fecha y hora UTC. El rango está entre '1970-01-01 00:00:00' y algún momento del año 2037.	AAAA-MM-DD HH:MM:SS

YEAR (2 4)	Almacena un año dado con 2 ó 4 dígitos de longitud (por defecto son 4). El rango de valores oscila entre 1901 y 2155 con 4 dígitos. Mientras que con 2 dígitos el rango es desde 1970 a 2069 (70-69).	AA ó AAAA
------------	--	-----------

Tabla 1.4. Tipos de datos fecha.

Ejemplo:

Para llevar el control de acceso (con horas, minutos y segundos) de los usuarios a un sistema de Información lo recomendable es tener un campo de tipo “DATETIME”.

1.4. Modificadores.

Además de los tipos de datos requiere es necesario conocer algunos modificadores que se utilizan para el manejo de los campos. Estos modificadores se presentan a continuación:

MODIFICADOR	USO	TIPO DE CAMPO QUE APLICA
AUTO_INCREMENT	El valor se va incrementando automáticamente en cada registro (1,2,3,etc).	Enteros
DEFAULT	Coloca un valor por defecto (el valor se coloca justo detrás de esta palabra).	Todos excepto TEXT y BLOB
NOT NULL	Impide que un campo sea nulo.	Todos
PRIMARY KEY	Hace que el campo se considere llave primaria.	Todos
UNIQUE	65565 bytes. Evita la repetición de valores.	Todos

Tabla 1.5. Modificadores.

2. Base de datos didáctica.

En este recurso se presenta cómo hacer uso del motor de Base de Datos MySQL para crear una base de datos. Se utilizará para esto la base de datos “Citas” compuesta por cinco tablas: Pacientes, Medicos, Consultorios, Citas y Tratamientos, como se presenta en la figura 2.1.

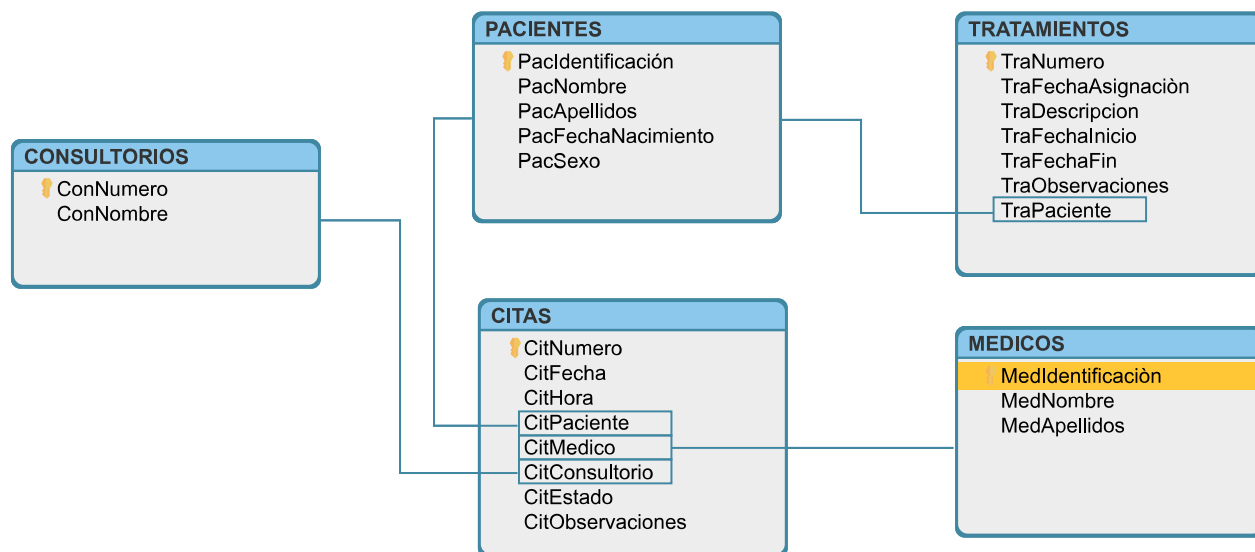


Figura 2.1. Diagrama relacional de la base de datos “Citas”.

Se considera importante presentar la estructura de la base de datos detallando las tablas, tipos de datos de los campos y modificadores a utilizar. Con el fin de que pueda proceder a su creación usando SQL como Lenguaje de Definición de Datos (DDL).

2.1. Tabla “Pacientes”.

Tabla “Pacientes”				
Atributo - Campo	Tipo de Dato	Long.	Modificador	Tabla y Campo Foráneo
PacIdentificación	Char	10	Primary key, not null	
PacNombres	Varchar	50	not null	
PacApellidos	Varchar	50	not null	
PacFechaNacimiento	Date		not null	
PacSexo	Por tener el Modificador enum, no se declara		(ENUM('M','F'))	

Figura 2.2. Definición de la tabla Pacientes.

2.2. Tabla “Medicos”.

Tabla “Médicos”				
Atributo - Campo	Tipo de Dato	Long.	Modificador	Tabla y Campo
MedIdentificacion	Char	10	Primary key, not null	
MedNombres	Varchar	50	not null	
MedApellidos	Varchar	50	not null	

2.3. Tabla “Consultorios”.

Tabla “Consultorios”				
Atributo - Campo	Tipo de Dato	Long.	Modificador	Tabla y Campo Foráneo
ConNumero	Int	3	Primary key, not null	
ConNombre	Varchar	50	not null	

Figura 2.4. Definición de la tabla Consultorios.

2.4. Tabla “Tratamientos”.

Tabla “Tratamientos”				
Atributo - Campo	Tipo de Dato	Long.	Modificador	Tabla y Campo Foráneo
TraNumero	Int		Primary key, not null auto_increment	
TraFechaAsignado	Date		not null	
TraDescripcion	Text		not null	
TraFechaInicio	Date		Primer key, not null	
TraFechaFin	Text		not null	
TraPacientes	Char	10	not null	Pacientes (PatientIdentificacion)

Figura 2.5. Definición de la tabla Tratamientos.

2.5. Tabla “Citas”.

Tabla “Citas”				
Atributo - Campo	Tipo de Dato	Long.	Modificador	Tabla y Campo Foráneo
CitNumero	Int		Primary key, auto_increment	
CitFecha	Date		not null	
CitHora	Varchar	10	not null	
CitPaciente	Char	10	not null	Pacientes (PacienteIdentificacion)
CitMedico	Char	10	not null	Medicos(MedIdentificacion)
CitConsultorio	Int		not null	Consultorio (ConNumero)
CitEstado	Por tener el Modificador enum, no se declara		(ENUM('Asignada','Cumplida')) DEFAULT "Asignada"	
CitObservaciones	Text		not null	

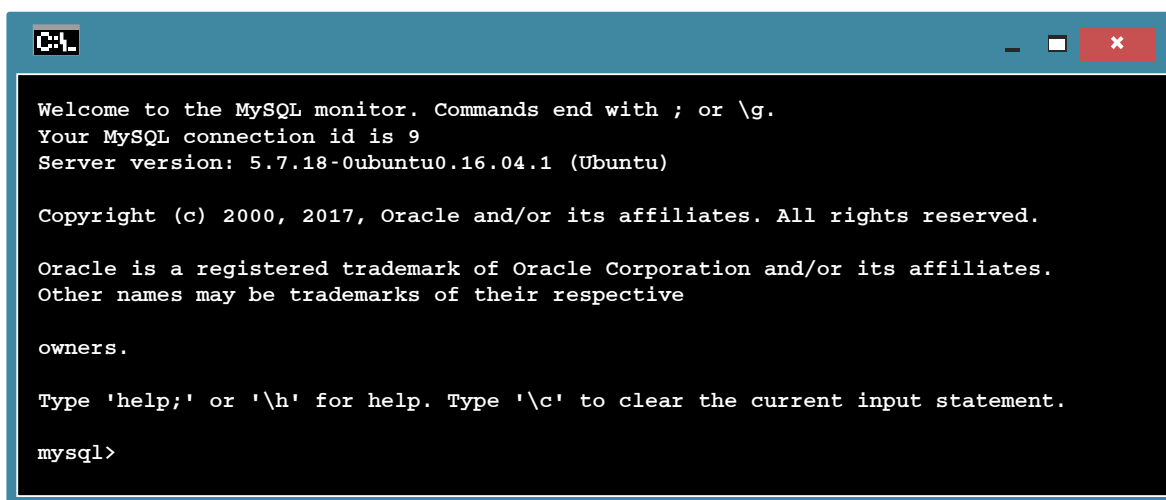
Figura 2.6. Definición de la tabla Citas.

3. Creación de la estructura de almacenamiento.

Para iniciar el proceso de definición de datos en MySQL se debe realizar el siguiente procedimiento:

a) Abrir el bloc de notas del equipo para digitar cada una de las instrucciones del Lenguaje de Definición de Datos, esto con el fin de ir construyendo el script completo de creación de la base de datos.

b) Iniciar la consola de MySQL como se explica en videotutorial sobre instalación de MySQL.



```

C:\
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.7.18-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, 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>

```

Figura 3.1. Consola MySQL lista para recibir instrucciones.

c) Escribir o pegar las instrucciones en la consola. Las instrucciones deben terminar con punto y coma para que se ejecuten.

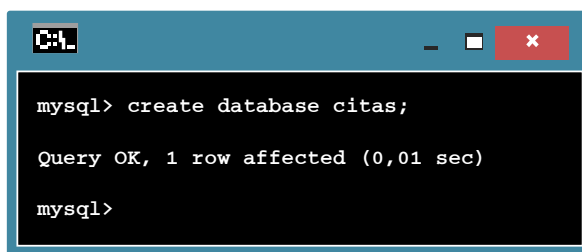
3.1 Creación de la base de datos.

Como el servicio ya está inicializado, se crea la Base de Datos, para nuestro ejemplo, CITAS.

La sintaxis de la instrucción es:

```
mysql> create database nombre_basedatos;
```

Para crear la base de datos “citas” se procede así:



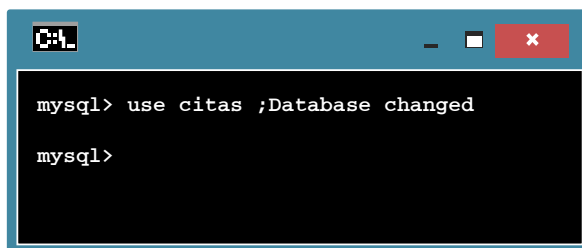
```
C:\nmysql> create database citas;  
Query OK, 1 row affected (0,01 sec)  
mysql>
```

Figura 3.2. SQL para la creación de la base de datos.

Para crear las tablas debemos seleccionar “citas” como base de datos predefinida. La instrucción es:

```
mysql> use nombre_base_de_datos ;
```

Para este ejemplo se tiene:

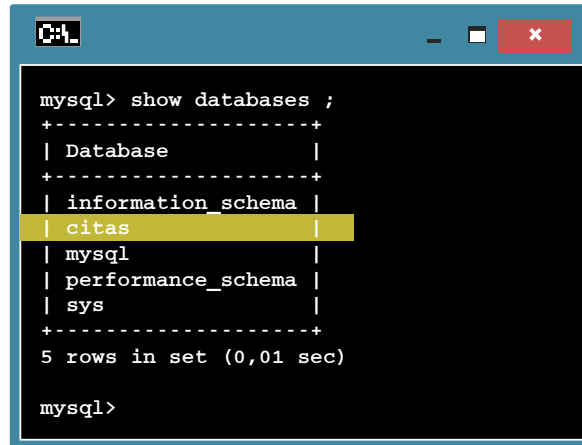


```
C:\nmysql> use citas ;Database changed  
mysql>
```

Figura 3.3. SQL para seleccionar una base de datos.

Para verificar que la base de datos fue creada se usa el siguiente comando:

```
mysql> show databases ;
```



```
mysql> show databases ;
+-----+
| Database |
+-----+
| information_schema |
| citas |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0,01 sec)

mysql>
```

Figura 3.4. SQL para mostrar las bases de datos creadas.

En la pantalla anterior se puede observar que aparece la base de datos “citas”.

3.2 Creación de las tablas.

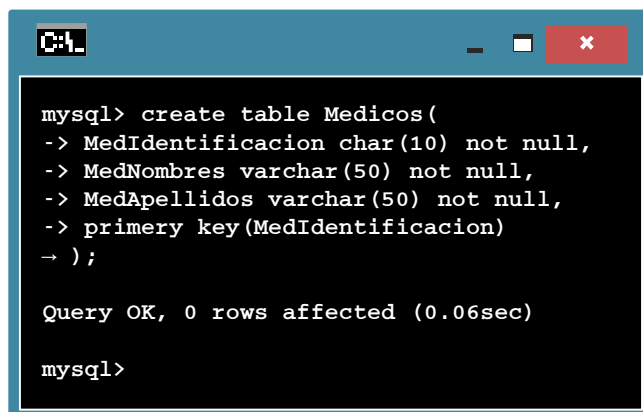
Una vez definida “citas” como base de datos predeterminada, se procede a crear las respectivas tablas.

La sintaxis para la creación de tablas es la siguiente:

```
mysql> create table nombreTabla (
  nombrecampo1 tipodatos(tamaño) modificador,
  nombrecampo2 tipodatos(tamaño) modificador,
  ...
  primary key (nombrecampo1)
);
```

3.2.1 Creación de la tabla “Medicos”.

Para la creación de la tabla “Medicos” se procede con el siguiente comando:



```
mysql> create table Medicos(
-> MedIdentificacion char(10) not null,
-> MedNombres varchar(50) not null,
-> MedApellidos varchar(50) not null,
-> primary key(MedIdentificacion)
-> );

Query OK, 0 rows affected (0.06sec)

mysql>
```

Figura 3.5. SQL para crear una tabla.

Una vez creada se verifica que la tabla aparezca en la base de datos con el siguiente comando:

```
mysql> show tables from base_de_datos ;
```

Al ejecutar el comando en la consola de MySQL muestra los siguiente:



```
mysql> show tables from citas;
+-----+
| Tables_in_citas |
+-----+
| Medicos          |
+-----+
1 rows in set (0,00 sec)

mysql>
```

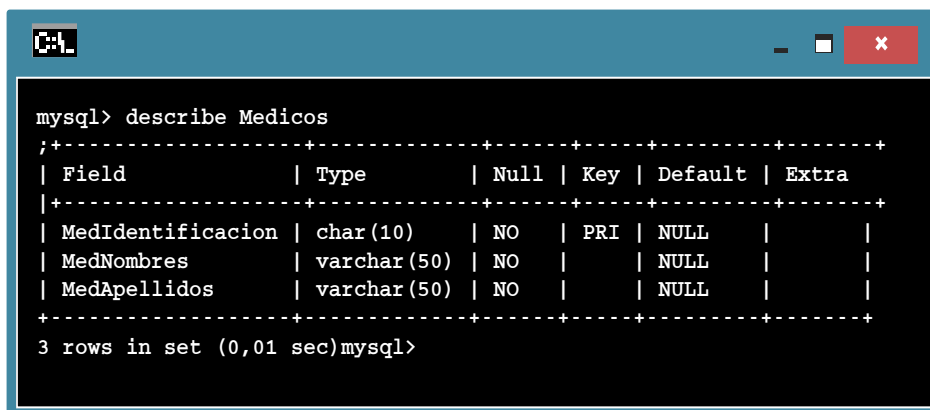
Figura 3.6. SQL para mostrar las tablas creadas en una base de datos.

Se puede observar que aparece la table “Medicos” en el listado.

Para verificar que la estructura de la tabla se usa el comando:

```
mysql> describe nombre_de_tabla ;
```

Para verificar la tabla “Medicos” se procede como se muestra a continuación:

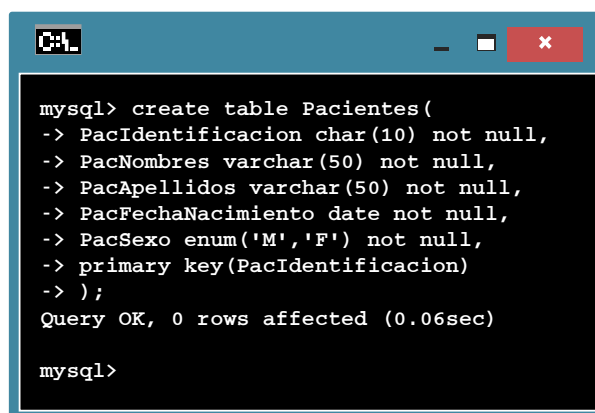


```
mysql> describe Medicos
;+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| MedIdentificacion | char(10)      | NO   | PRI | NULL    |       |
| MedNombres       | varchar(50)   | NO   |     | NULL    |       |
| MedApellidos     | varchar(50)   | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0,01 sec)mysql>
```

Figura 3.7. SQL para mostrar la estructura de una tabla.

3.2.2 Creación de la tabla “Pacientes”.

Para crear la tabla “Pacientes” se procede con el siguiente comando:

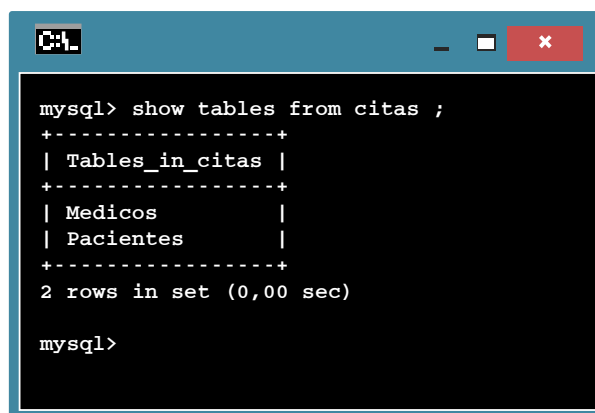


```
mysql> create table Pacientes(
-> PacIdentificacion char(10) not null,
-> PacNombres varchar(50) not null,
-> PacApellidos varchar(50) not null,
-> PacFechaNacimiento date not null,
-> PacSexo enum('M','F') not null,
-> primary key(PacIdentificacion)
-> );
Query OK, 0 rows affected (0.06sec)

mysql>
```

Figura 3.8. SQL para crear una tabla.

Se puede verificar que la tabla quedó creada con el siguiente comando:

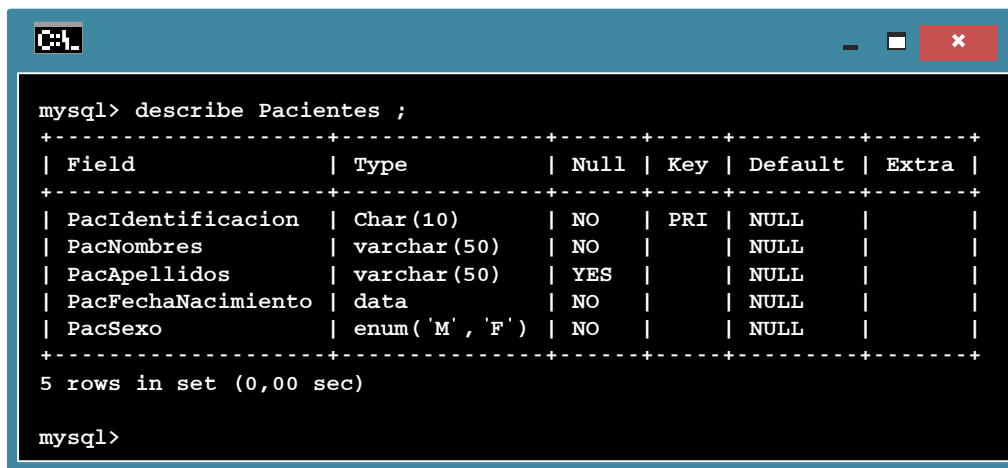


```
mysql> show tables from citas ;
+-----+
| Tables_in_citas |
+-----+
| Medicos          |
| Pacientes        |
+-----+
2 rows in set (0,00 sec)

mysql>
```

Figura 3.9. SQL para mostrar las tablas de una base de datos.

Para verificar la estructura de la tabla se procede con el siguiente comando:



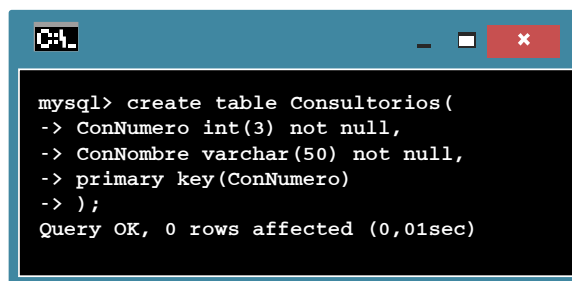
```
mysql> describe Pacientes ;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| PacIdentificacion | Char(10)      | NO   | PRI | NULL    |       |
| PacNombres       | varchar(50)   | NO   |     | NULL    |       |
| PacApellidos     | varchar(50)   | YES  |     | NULL    |       |
| PacFechaNacimiento | data         | NO   |     | NULL    |       |
| PacSexo          | enum('M', 'F') | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0,00 sec)

mysql>
```

Figura 3.10. SQL para conocer la estructura de una tabla.

3.2.3. Creación tabla “Consultorios”.

Para crear la tabla “Consultorios” se usa el siguiente comando:



```
mysql> create table Consultorios(
-> ConNumero int(3) not null,
-> ConNombre varchar(50) not null,
-> primary key(ConNumero)
-> );
Query OK, 0 rows affected (0,01sec)

mysql>
```

Figura 3.11. SQL para crear una tabla.

Para verificar que la tabla haya sido creada se procede así:

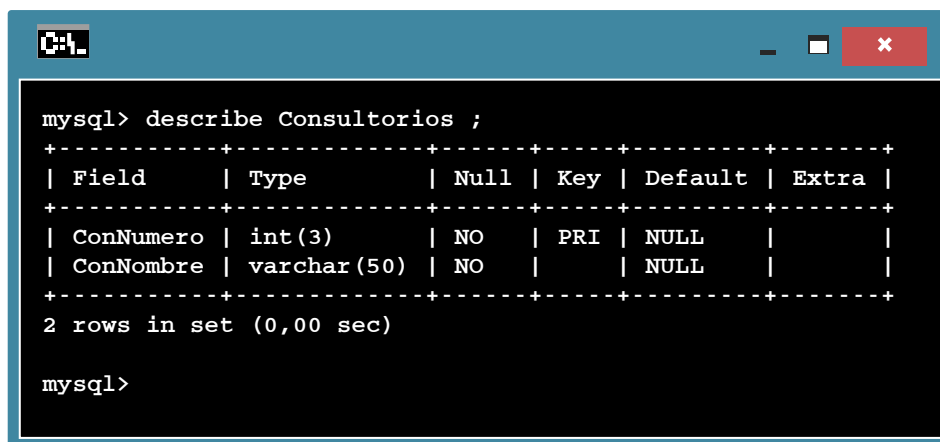


```
mysql> show tables from citas ;
+-----+
| Tables_in_citas |
+-----+
| Consultorios     |
| Medicos          |
| Pacientes        |
+-----+
3 rows in set (0,00 sec)

mysql>
```

Figura 3.12. SQL para mostrar las tablas de una base de datos.

Para verificar la estructura de la tabla se usa el siguiente comando:



```
mysql> describe Consultorios ;
+-----+-----+-----+-----+-----+-----+
| Field      | Type        | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| ConNumero  | int(3)       | NO   | PRI | NULL    |       |
| ConNombre  | varchar(50)  | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0,00 sec)

mysql>
```

Figura 3.13. SQL para mostrar la estructura de una tabla.

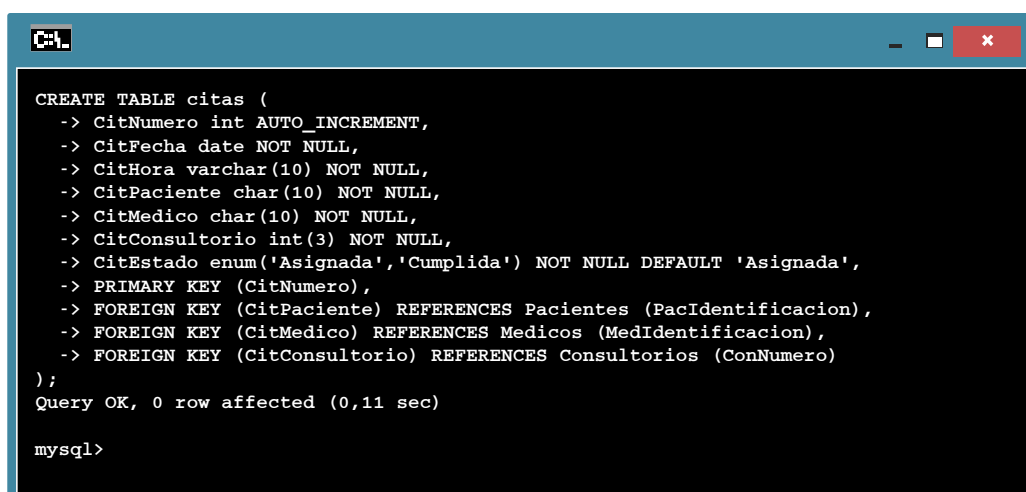
3.2.4. Creación tabla “Citas”.

Al revisar la estructura de la tabla “Citas” definida en un numeral anterior se puede observar que esta tabla tiene unos modificadores nuevos. Además esta tabla contiene llaves foráneas, es decir, está relacionada con otras tablas.

La sintaxis para la creación de una llave foránea es la siguiente:

```
foreign key(nombre_de_campo) references tabla(campo_tabla)
```

Al aplicar la sintaxis anterior a la creación de la tabla “Citas” se tiene lo siguiente:



```
CREATE TABLE citas (
-> CitNumero int AUTO_INCREMENT,
-> CitFecha date NOT NULL,
-> CitHora varchar(10) NOT NULL,
-> CitPaciente char(10) NOT NULL,
-> CitMedico char(10) NOT NULL,
-> CitConsultorio int(3) NOT NULL,
-> CitEstado enum('Asignada','Cumplida') NOT NULL DEFAULT 'Asignada',
-> PRIMARY KEY (CitNumero),
-> FOREIGN KEY (CitPaciente) REFERENCES Pacientes (PacIdentificacion),
-> FOREIGN KEY (CitMedico) REFERENCES Medicos (MedIdentificacion),
-> FOREIGN KEY (CitConsultorio) REFERENCES Consultorios (ConNumero)
);
Query OK, 0 row affected (0,11 sec)

mysql>
```

Figura 3.14. SQL para crear una tabla.