Introducción a la base de datos MySQL®

FAVA - Formación en Ambientes Virtuales de Aprendizaje

SENA - Servicio Nacional de Aprendizaje.

ESTRUCTURA DE CONTENIDOS

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).

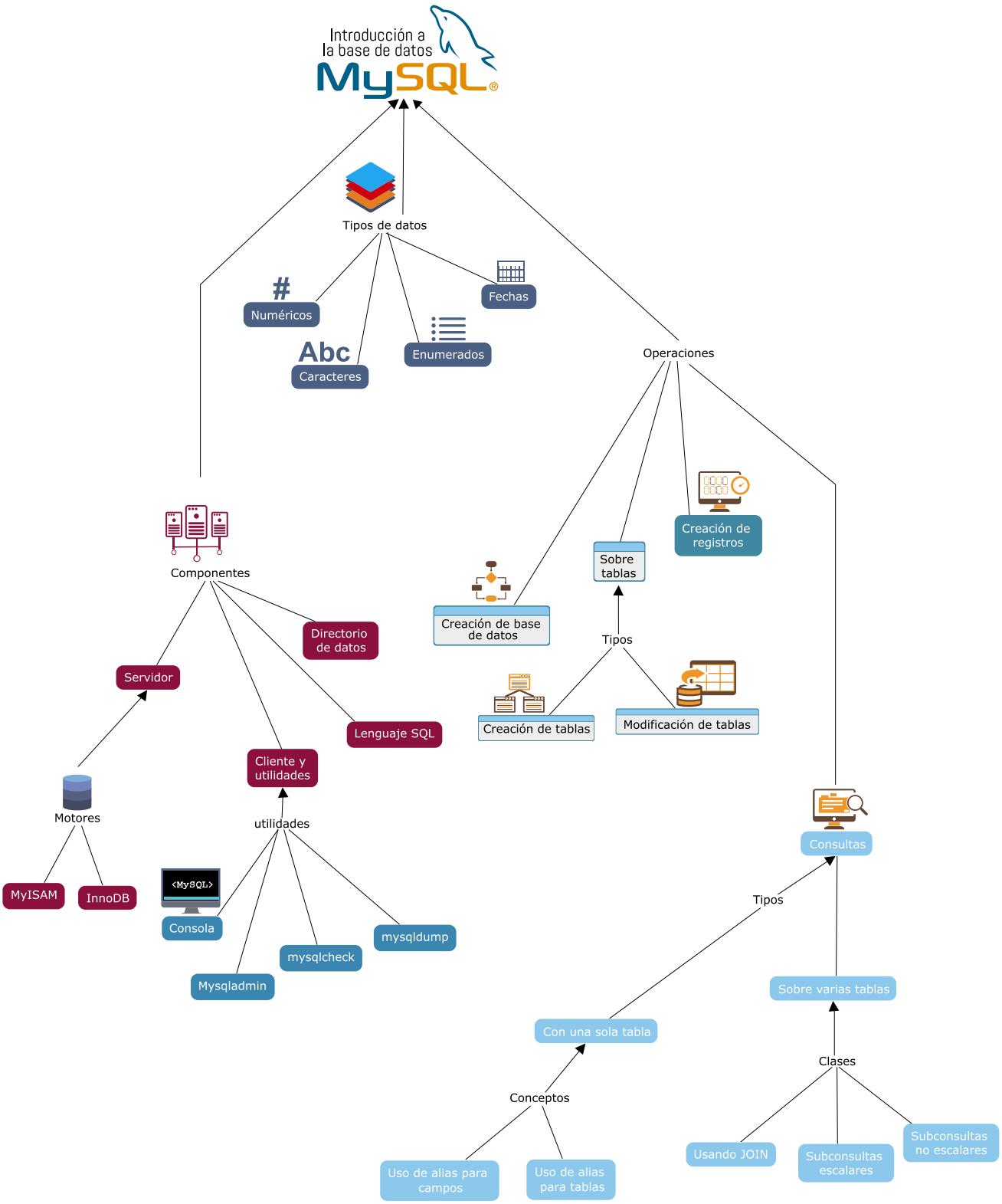
Su popularidad se debe principalmente a su licencia libre y a su facilidad de uso y administración. Por otra parte ha sido integrada con otras herramientas libres como son Linux, Apache, PHP, entre otras. Esta combinación e integración de tecnologías dio nombre a la plataforma de desarrollo conocida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

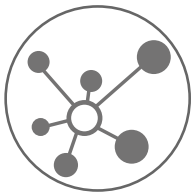
Velocidad
MySQL es veloz comparado con la mayoría de las bases libres.

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

Soporta el lenguaje SQL
MySQL soporta el lenguaje estructurado de consultas (SQL).

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

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

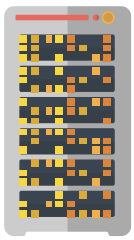


Conectividad
MySQL soporta distintos esquemas de conectividad y las bases de datos pueden ser accedidas desde cualquier sitio de Internet.

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

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:

a. **Un archivo .frm:** cada tabla de InnoDB es representada por un archivo .frm que contiene la definición de la estructura.
b. **Tablespace compartido:** consiste en uno o más archivos que contienen todos los datos de las tables de manera contigua. Por defecto InnoDB almacena los datos de manera contigua.
c. **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:

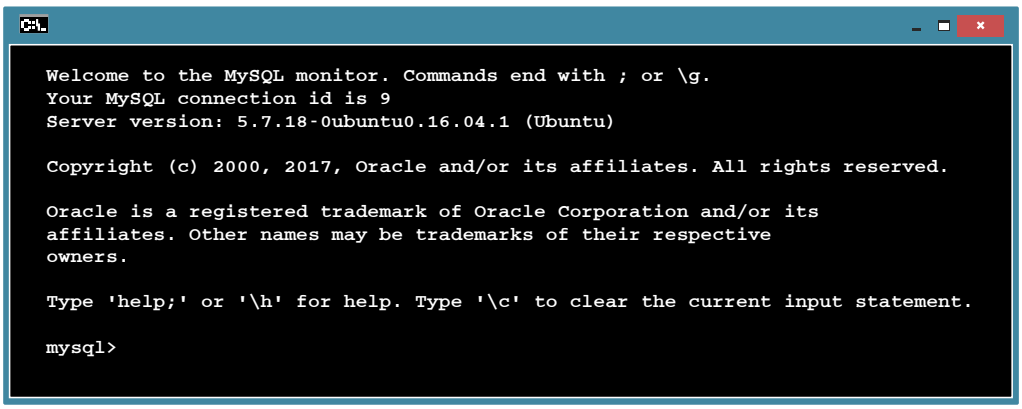


Figura 1.1. Consola de MySQL.

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	
LONGBLOB 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.

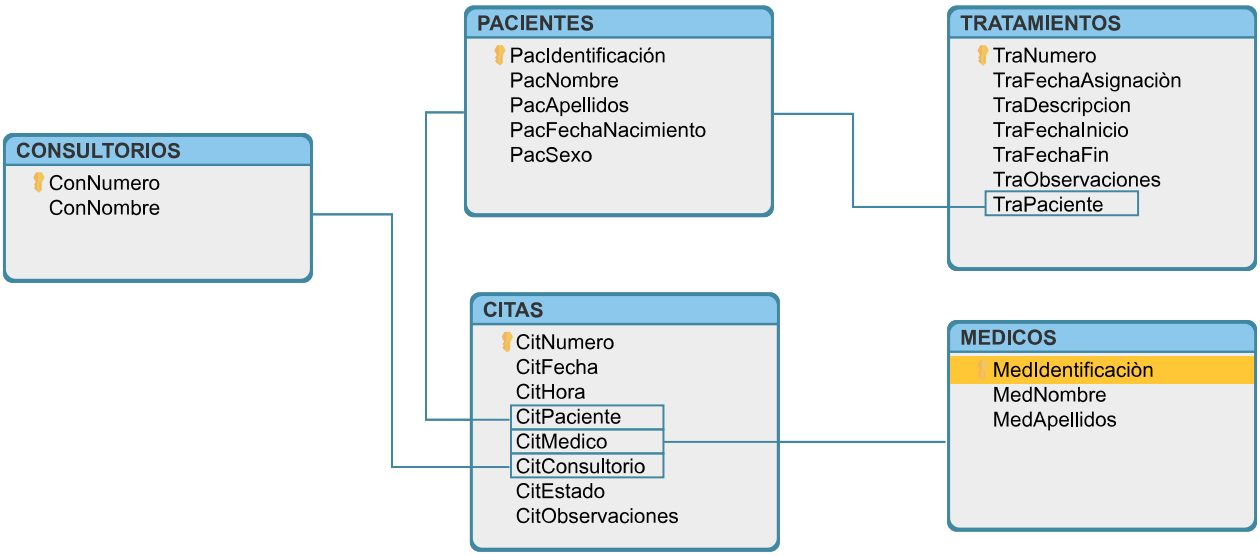


Figura 2.1. Diagrama relacional de la base de datos "Citas".

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
PacIdentificacion	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 (PacientIdentificacion)

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 (PacientIdentificacion)
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.

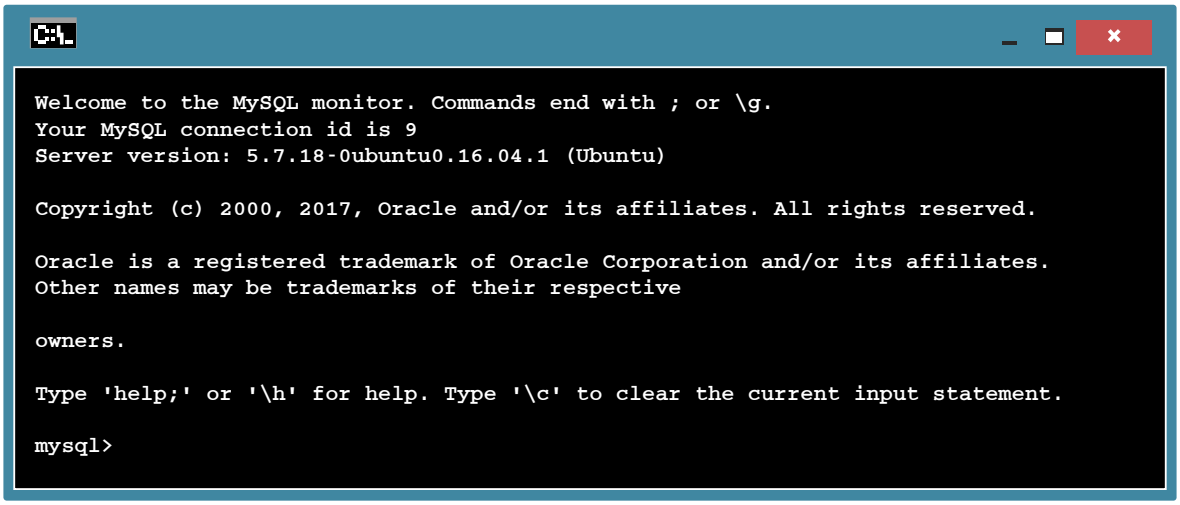


Figura 3.1.Consola MySQL lista para recibir instrucciones.

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:

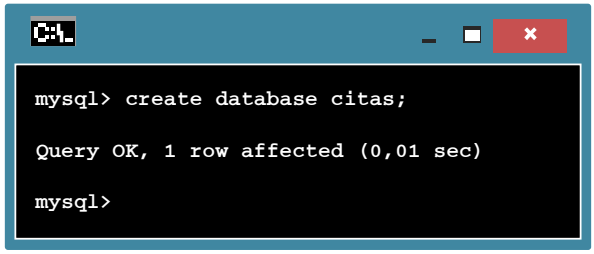


Figura 3.2. SQL para la creación de la base de datos.

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:

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 ;
```

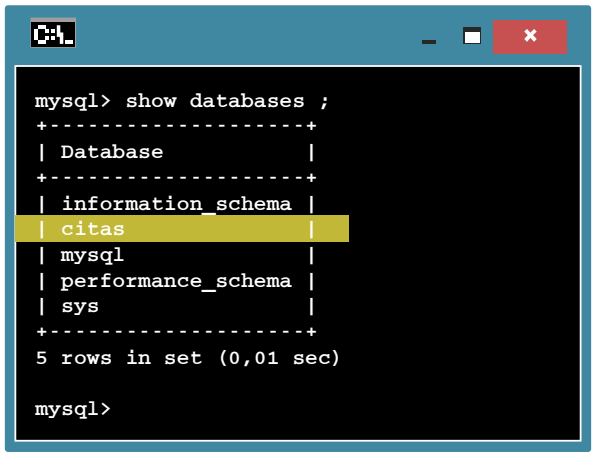


Figura 3.4. SQL para mostrar las bases de datos creadas.

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:

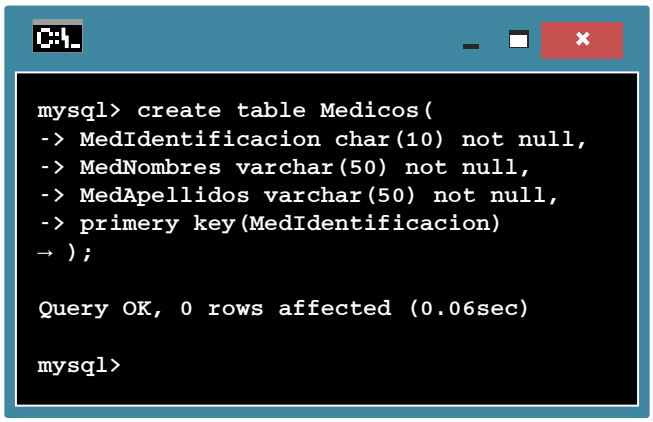


Figura 3.5. SQL para crear una tabla.

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:

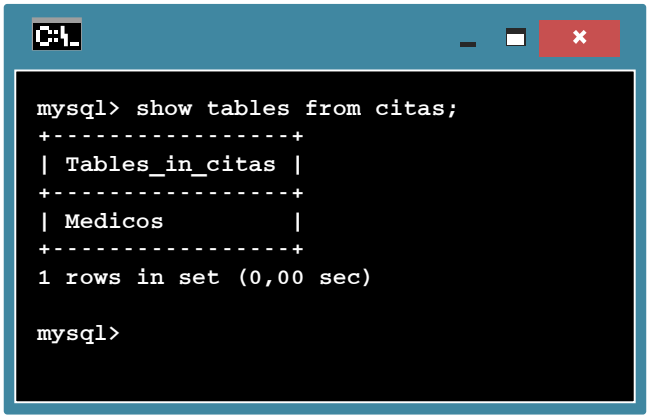


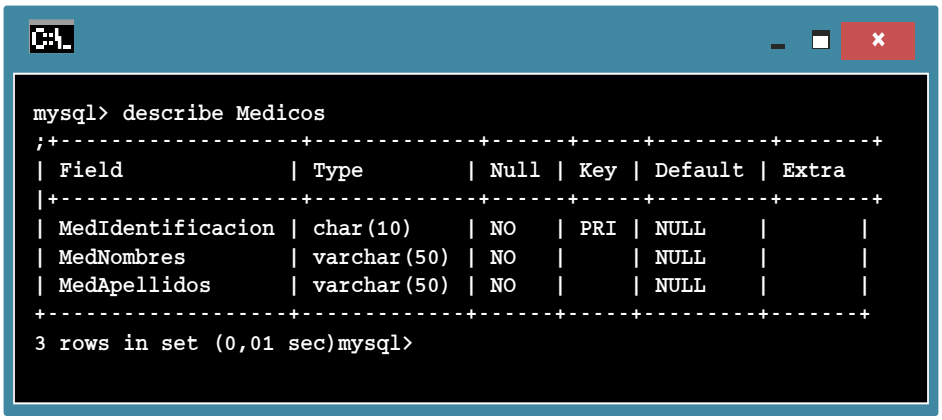
Figura 3.6. SQL para mostrar las tablas creadas en una base de datos.

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:

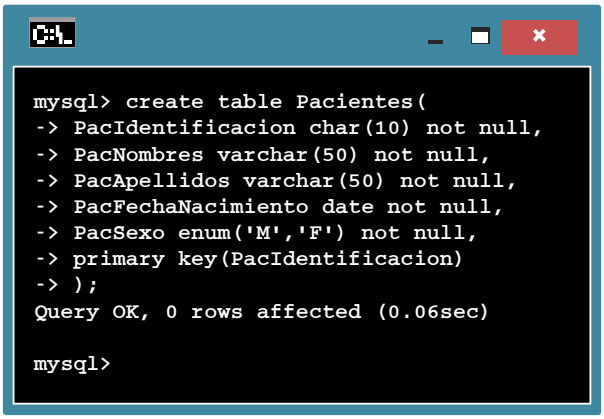


```
mysql> describe Medicos
;+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra
|+------------------+-------------+------+-----+---------+-------+
| MedIdentificacion | char(10)    | NO   | PRI | NULL    |       |
| MedNombres        | 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".

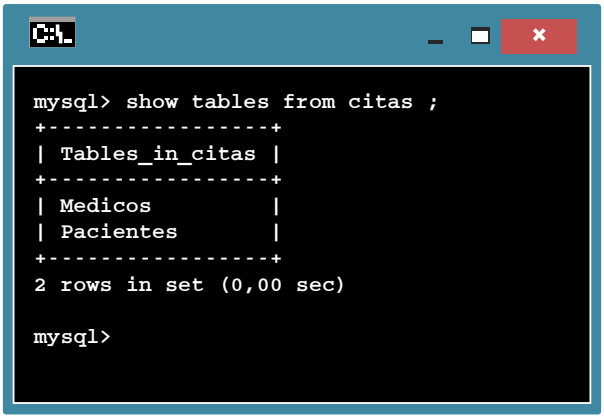
Para crear la tabla "Pacientes" se procede con el siguiente comando:



```
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)
```

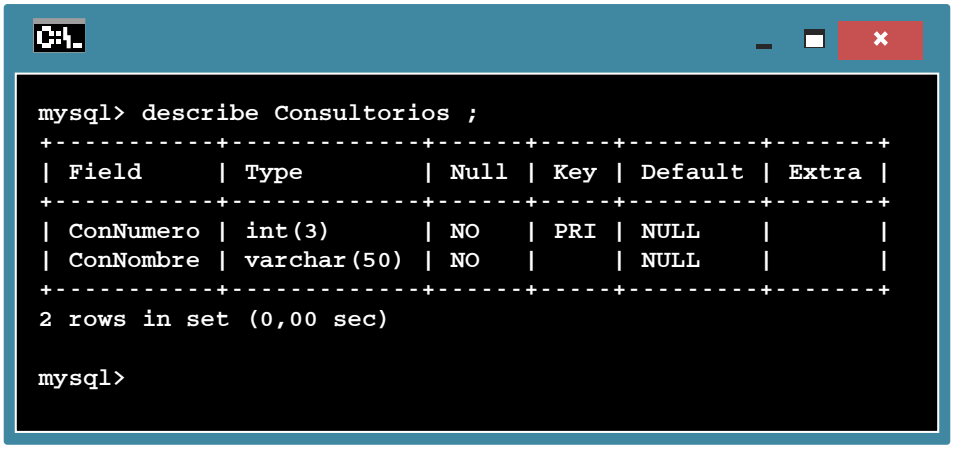
Figura 3.11. SQL para crear una tabla.

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.

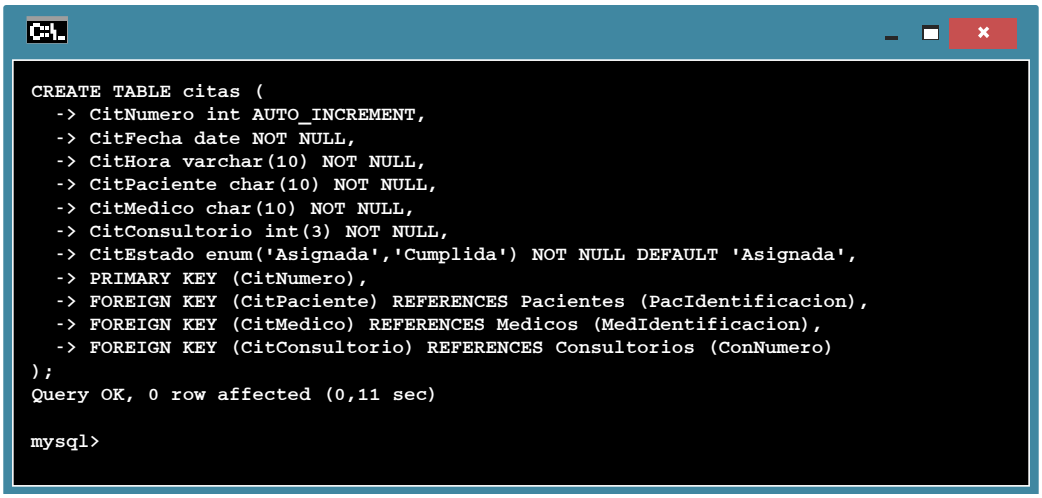
3.2.4. Creación tabla "Citas".

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.

4. Modificación de la estructura de las tablas.

Para modificar la estructura de una tabla se usa el siguiente comando:

```
mysql> alter table <nombre tabla> <atributo> <opciones> ;
```

La lista de los atributos o comandos es la siguiente:

ATRIBUTOS	FUNCIÓN	SINTAXIS
MODIFY	Modifica un campo, esto puede ser el tipo de dato y/o el ancho del campo. También se utiliza para agregar y/o eliminar modificaciones.	alter table nombre_tabla MODIFY COLUMN nombre_columna modificador; Para realizar este cambio se deben incluir los modificadores que tiene el campo y los que se van a agregar.
CHANGE	Modifica el nombre de un campo.	alter table nombre_tabla CHAGE COLUMN nombre_columna_actual nuevo_nombre_columna tipo_dato ancho_dato modificador(es);
RENAME	Cambia el nombre de una tabla.	alter table personas rename clientes;
DROP	Borra Columnas.	alter table tabla DROP COLUMN ncolumnaABorrar;
ADD	Adiciona Columnas a una Tabla, es importante informar la ubicación de la nueva columna. Para ello se utiliza AFTER.	alter table nombre_tabla ADD nueva_Columna Tipo_dato Modificador AFTER Nombre_columna;

Figura 4.1. Atributos para modificar la estructura de una tabla.

Se debe tener cuidado al momento de cambiar la estructura de una tabla ya que estos cambios pueden afectar toda la base de datos. El MySQL verifica cada cambio antes de aplicar. Si los cambios rompen las reglas de integridad el motor de base de datos no dejará aplicarlos y en su lugar emitirá un código y mensaje de error.

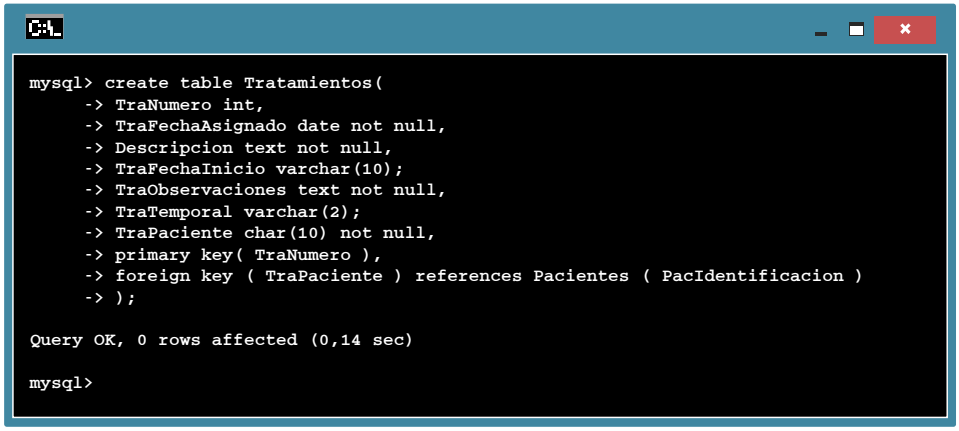
Para practicar esta instrucción se va a crear la tabla “Tratamientos” con una estructura inicial a la cual posteriormente se le introducirán unos cambios para llegar a una estructura deseada.

La estructura inicial de la tabla es la siguiente:

Campo	Tipo de campo	Longitud	Modificador
Tranumero	Int		Primary Key
TraFechaAsignado	Date		Not null
Descripción	Text		Not null
TraFechaInicio	Varchar	10	Not null
TraObservaciones	Text		Not null
TraTemporal	Varchar	2	
TraPaciente	Char	10	Pacientes (PacIdentificacion)

Figura 4.2. Estructura de la tabla Tratamientos.

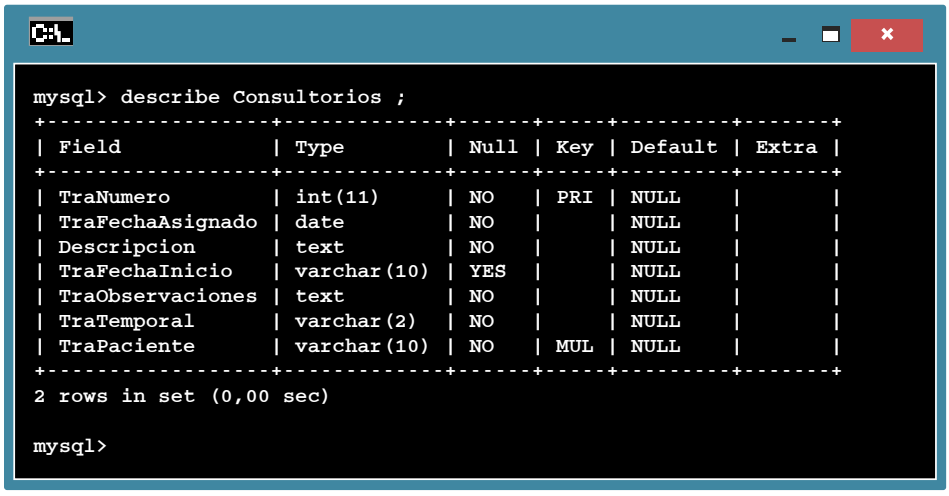
Para crearla se usa el siguiente comando:



```
mysql> create table Tratamientos(
    -> TraNumero int,
    -> TraFechaAsignado date not null,
    -> Descripcion text not null,
    -> TraFechaInicio varchar(10);
    -> TraObservaciones text not null,
    -> TraTemporal varchar(2);
    -> TraPaciente char(10) not null,
    -> primary key( TraNumero ),
    -> foreign key ( TraPaciente ) references Pacientes ( PacIdentificacion )
    -> );

Query OK, 0 rows affected (0,14 sec)

mysql>
```

Figura 4.3. SQL para crear una tabla.

Para verificar su estructura se usa el comando “describe” así:



```
mysql> describe Consultorios ;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| TraNumero        | int(11)     | NO   | PRI | NULL    |       |
| TraFechaAsignado | date        | NO   |     | NULL    |       |
| Descripcion      | text        | NO   |     | NULL    |       |
| TraFechaInicio   | varchar(10) | YES  |     | NULL    |       |
| TraObservaciones | text        | NO   |     | NULL    |       |
| TraTemporal      | varchar(2)  | NO   |     | NULL    |       |
| TraPaciente      | varchar(10) | NO   | MUL | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
2 rows in set (0,00 sec)

mysql>
```

Figura 4.4. SQL para mostrar la estructura de una tabla.

Los cambios que se necesitan aplicar para llegar a la estructura requerida son los siguientes:

No	Descripción
1	Se requiere que el campo “TraNumero” sea auto-incremental.
2	Se requiere que el campo “Descripcion” sea renombrado a “TraDescripcion” para seguir el estándar de nombres de campos.
3	Cambia el tipo de datos del campo “TraFechaInicio” al tipo “date”.
4	Agregar el campo “TraFechaFin” de tipo “date” después de la fecha de inicio.

Figura 4.5. Lista de las modificaciones a realizar a la tabla Tratamientos.

1) Para aplicar el primer cambio: “Se requiere que el campo TraNumero sea autoincremental” se utiliza el siguiente comando:

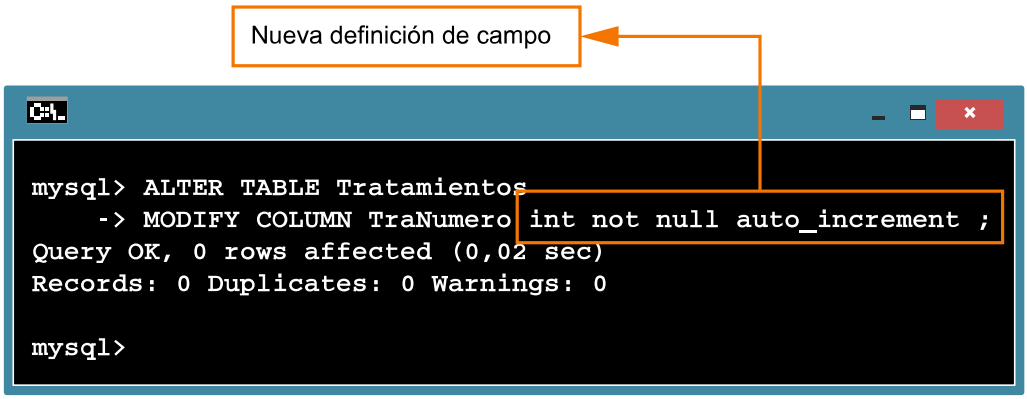


Figura 4.6. Descripción de la instrucción Alter table modify column.

2) Para aplicar el segundo cambio: se requiere que el campo *Descripcion* sea renombrado a *TraDescripcion* se utiliza el comando “change” como se describe a continuación:

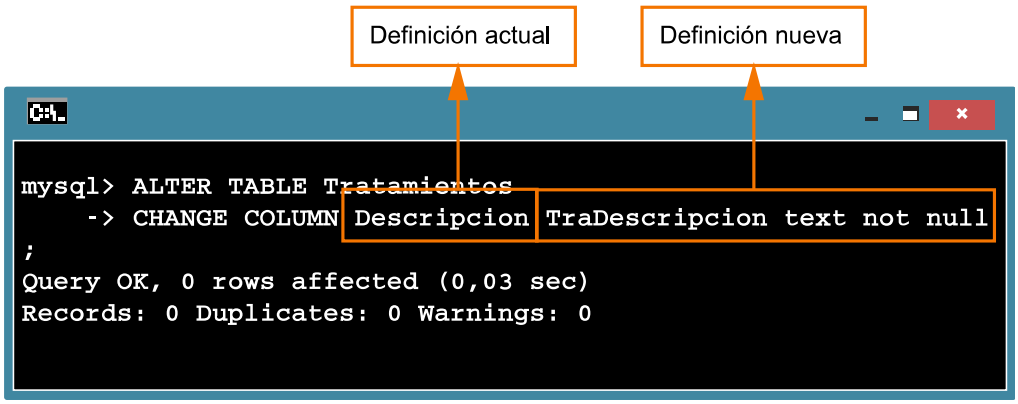


Figura 4.7. Descripción de la instrucción Alter table change column.

3) Para aplicar el tercer cambio: “Cambiar el tipo de dato del campo “TraFechaInicio” al tipo date “ se utiliza el comando “modify” como se describe a continuación:

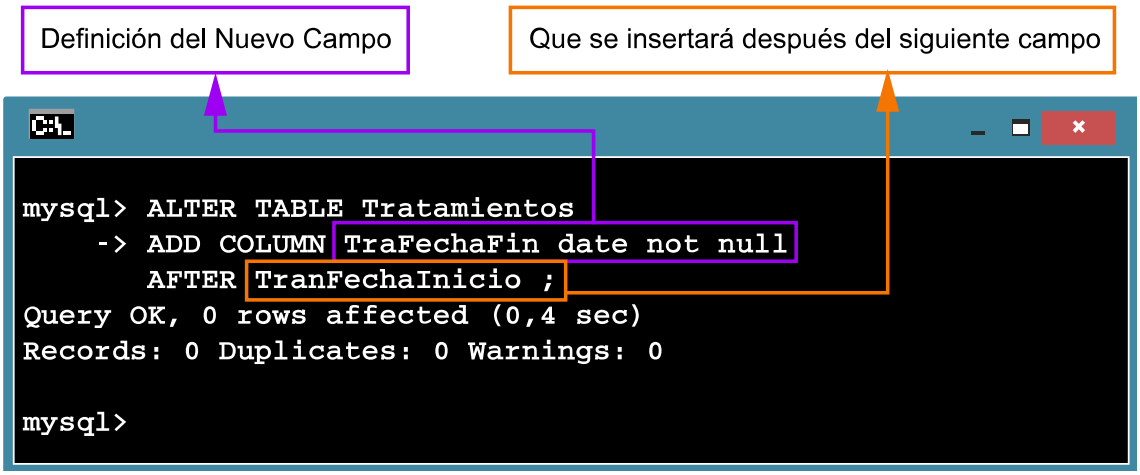


Figura 4.8. Descripción de la instrucción Alter table modify column.

4) Para aplicar el cuarto cambio: “Agregar el campo TraFechaFin de tipo date después de la fecha de inicio.” se aplica el siguiente comando:

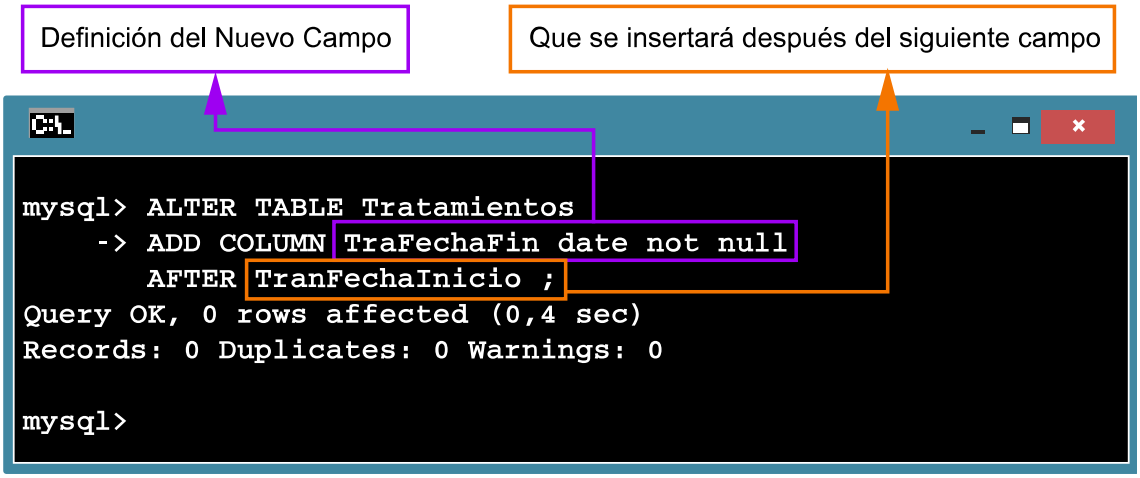


Figura 4.9. Descripción de la instrucción Alter table add column.

5) Para aplicar el quinto cambio: “Eliminar el campo TraTemporal” se procede con el comando “drop column” como se describe a continuación:

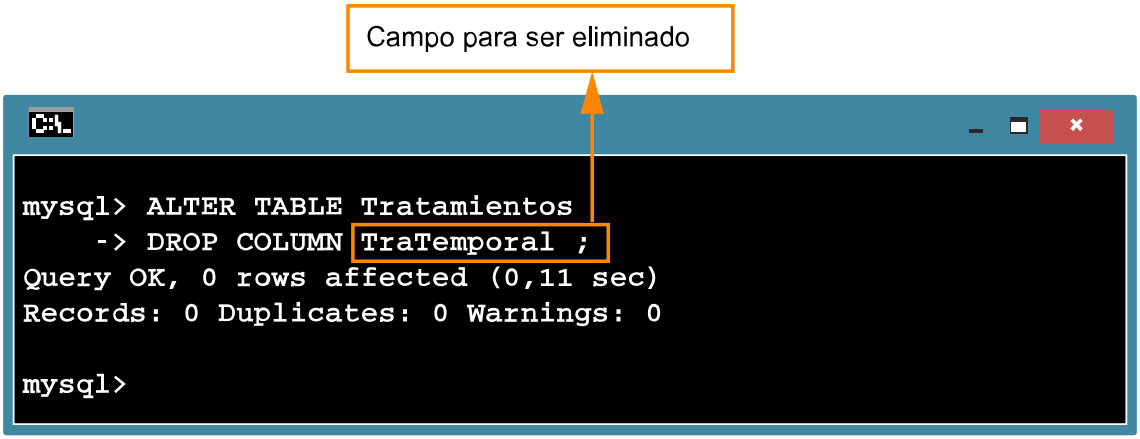


Figura 4.10. Descripción de la instrucción Alter table drop column.

Por último se verifica la nueva estructura de la tabla usando el comando “describe” como se muestra a continuación:

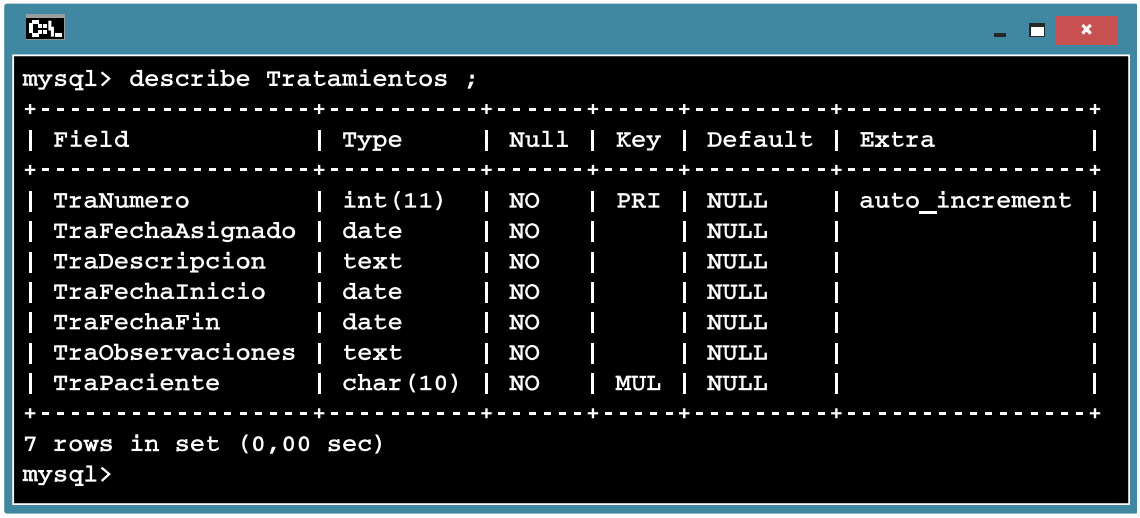


Figura 4.11. Descripción de la tabla Tratamientos después de los cambios.

5. Actualización e inserción de registros.

5.1 Inserción de registros.

Para realizar las operaciones de Inserción de Registros se utiliza la sentencia “INSERT INTO” del lenguaje de manipulación de datos (D.M.L.).

El comando “INSERT INTO” permite incluir los datos en cada uno de los campos que se tienen en las tablas de la base de datos creada. Su sintaxis es:

```
mysql> INSERT INTO <nombre tabla> (<campo1>, <campo2>, …, <campoN>)
        VALUES (<valor1>,<valor2>, … , <valorN>) ;
```

Para los establecer lo valores que tomarán los campos se debe tener en cuenta lo siguiente:

a) Los campos tipo texto, char o varchar deben estar encerrados entre comillas. Ejemplo: ‘Gomez Rodriguez’, ‘Carrera 42 45 -11 Apto 1102’.

b) Los campos tipo entero debe ir sin ningún cambio.

c) Los campos tipo decimal o flotante: la parte entera debe estar separada de la parte decimal por un punto “.”. Lo anterior ya que la coma es usada como separador de campos.

d) Los campos tipo fecha: pueden tomar varios tipos de formatos. El formato predefinido es ‘AAAA-MM-DD’ donde AAAA es el año , MM el mes y DD el día. El valor debe además ir encerrado entre comillas. Por ejemplo el 13 de Julio de 2017 quedaría expresado como ‘2017-07-13’ .

5.1.1 Inserción de registros sin llaves foráneas.

Para insertar el siguiente registro :

Identificación	91.222.333
Nombres	Carlos Jesús
Apellidos	Rodríguez Cala
Fecha Nacimiento	23 de Enero de 1970
Sexo	M

Figura 5.1. Registro ejemplo a insertar.

Se utiliza el siguiente comando:

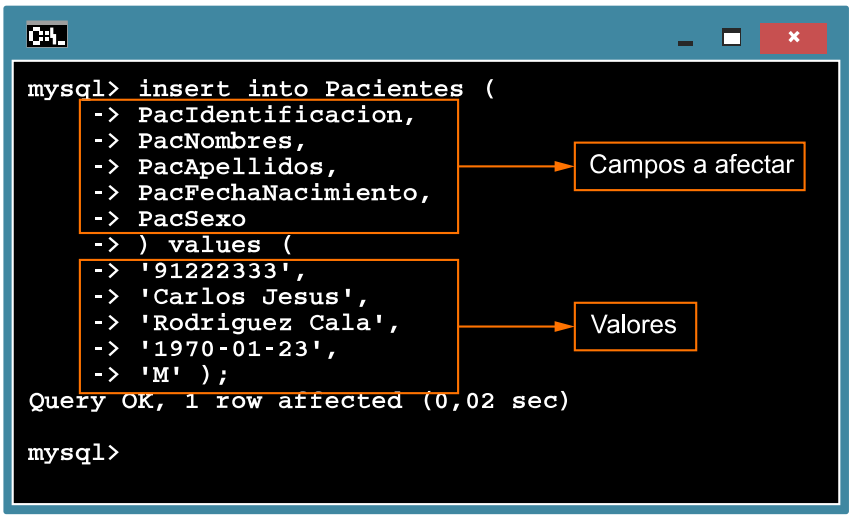


Figura 5.2. Ejemplo del comando Insert into.

5.1.2 Inserción de registros con llaves foráneas.

Las llaves foráneas son reglas de integridad que se deben cumplir para que la base de datos sea consistente.

Para insertar registros en tablas que tienen llaves foráneas se debe identificar el valor de esas llaves para incluirlos en la sentencia de inserción.

Por ejemplo, para insertar un registro en la tabla “Tratamientos” se debe determinar la llave primaria del paciente en la tabla “Pacientes”.

Para insertar el siguiente registro:

TraFechaAsignado	“2017-07-13”
TraDescripcion	Tratamiento de Conductos
TraFechainicio	“2017-08-01”
TraFechaFin	“2017-08-03”
TraObservaciones	Paciente con hipertensión
TraPaciente	91.222.333

Figura 5.3. Ejemplo de registro con llaves foráneas a insertar.

Se procede así:

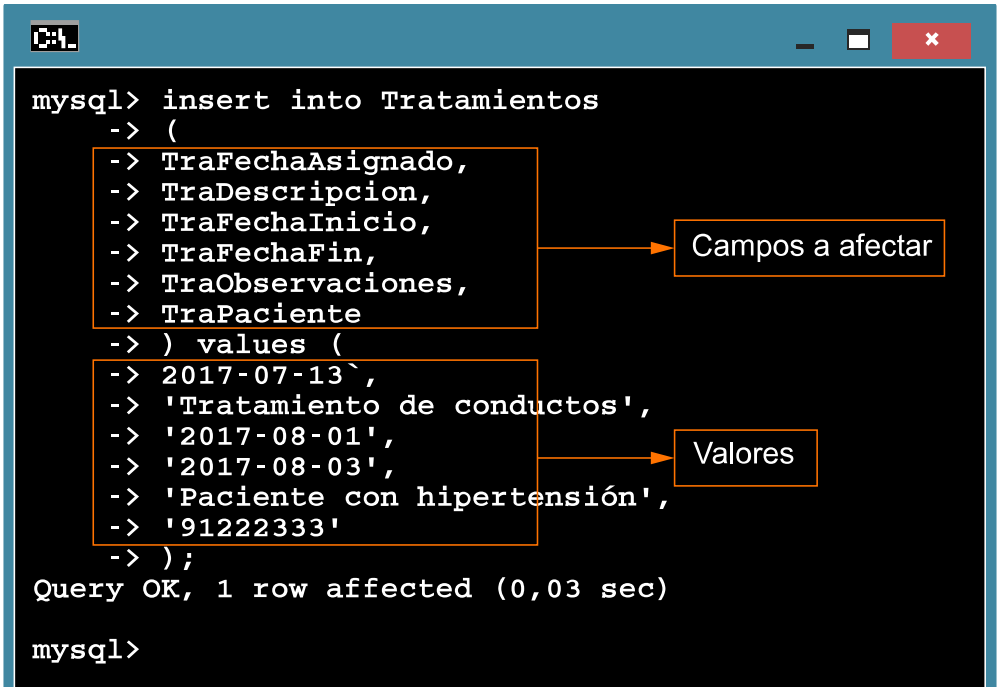


Figura 5.4. Ejemplo de registro con llaves foráneas a insertar.

Para la inserción del anterior registro no se tuvo que definir el valor del campo “TraNumero” ya que fue definido como “auto_increment”, es decir, que se auto incrementa con cada registro nuevo. Para verificar que el registro fue insertado se ejecuta una consulta como se muestra en la figura 5.5.

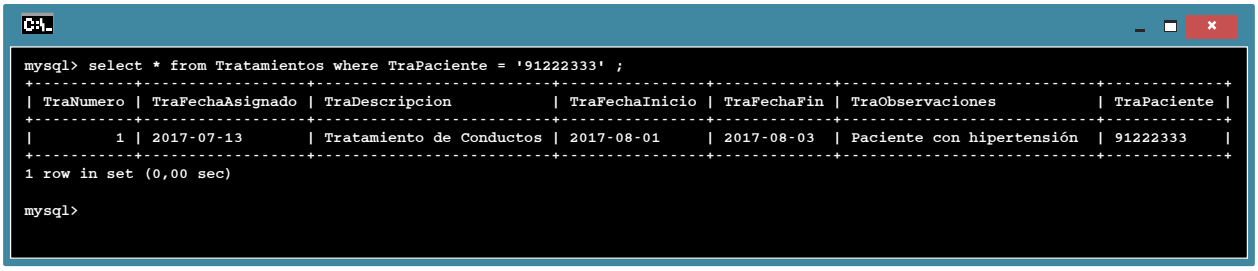


Figura 5.5. Consulta para verificar registro creado.

6. Consultas de registros.

Para realizar las operaciones de Consulta de Registros se utiliza la sentencia SELECT del lenguaje de manipulación de datos (D.M.L.).

La sentencia SELECT permite visualizar la información de la Base de Datos, los datos que se presentan corresponden a una o más filas de una tabla o requiere a una o más filas de una o más tablas.

La sintaxis básica es:

```
SELECT <lista de campos >
FROM <tablas >
WHERE <condiciones>
GROUP BY <campos de agrupamiento>
HAVING <condiciones sobre los grupos >
ORDER BY <ordenamientos>
LIMIT <cantidad de registros a traer >
OFFSET <número de página >
```

6.1 Datos de prueba.

Para el ejemplo se procederá con los siguientes registros de prueba:

IDENTIFICACIÓN	NOMBRES	APELLIDOS	FECHA NACIMIENTO	SEXO
29.234.333	Alejandra Marcela	Díaz Granados	1980-03-24	F
1098.343.678	Juan Antonio	Pérez Pereira	1978-08-09	M
37.456.298	Diana Marcela	Estévez	1985-09-06	F
51.890.654	Adriana María	Pataquiva	1990-12-24	F
91.222.333	Carlos Jesús	Rodríguez Cala	1970-01-23	M

Figura 6.1. Conjunto de registros de ejemplo.

TraFechaAsignado	TraDescripción	TraFechaInicio	TraFechaFin	TraObservaciones	TraPaciente
2017-07-13	Tratamiento de Conductos	2017-08-01	2017-08-03	Paciente con hipertensión	91.222.333
2017-07-02	Profilaxis	2017-07-02	2017-07-02	Sin novedad	51.890.654
2017-06-05	Resina	2017-06-05	2017-06-05	Paciente con sensibilidad	37.456.298
2017-05-23	Profilaxis	2017-05-23	2017-05-23	Sin novedad	29.234.333

Figura 6.2. Conjunto de registros de ejemplo.

6.2 Consultas básicas.

Para hacer un ejemplo de consultas básicas se tomará la tabla pacientes. Esta consulta puede generarse de diferentes maneras dependiendo de los datos que se necesiten visualizar.

Si se quiere visualizar todos los registros y campos de la tabla se usa el asterisco (*) el cual indica al intérprete de comandos que extraiga todos los campos.

La instrucción es:

```
foreign key(nombre_de_campo) references tabla(campo_tabla)
mysql> SELECT * FROM Pacientes ;
```

Al ejecutar el comando en la consola de MySQL aparece los siguiente:
Figura 6.3. SQL para hacer consultas a la base de datos.

Para limitar el número de registros se usa el comando “LIMIT x”. Ejemplo:

```
mysql> select * from Pacientes ;
+------------------+------------------+----------------+--------------------+---------+
| PacIdentificacion | PacNombres       | PacApellidos   | PacFechaNacimiento | PacSexo |
+------------------+------------------+----------------+--------------------+---------+
| 1098343678       | Juan Antonio     | Perez Pereira  | 1978-08-09         | M       |
| 29234333         | Alejandra Marcela | Diaz Granados | 1980-03-24         | F       |
| 37456298         | Diana Marcela    | Estevez        | 1985-09-06         | F       |
| 51890654         | Adriana María    | Pataquiva      | 1990-12-24         | F       |
| 91222333         | Carlos Jesus     | Rodriguez Cala | 1970-01-23         | M       |
+------------------+------------------+----------------+--------------------+---------+
5 rows in set (0,00 sec)mysql>
```

Figura 6.4. SQL para hacer consultas a la base de datos con límite.

La consulta anterior limita el resultado a máximo tres registros.

El asterisco indica al intérprete de comandos seleccionar todos los campos. Esto no es adecuado ya que puede generar consultas pesadas que consumen valiosos recursos del servidor. Para especificar los campos se procede así:

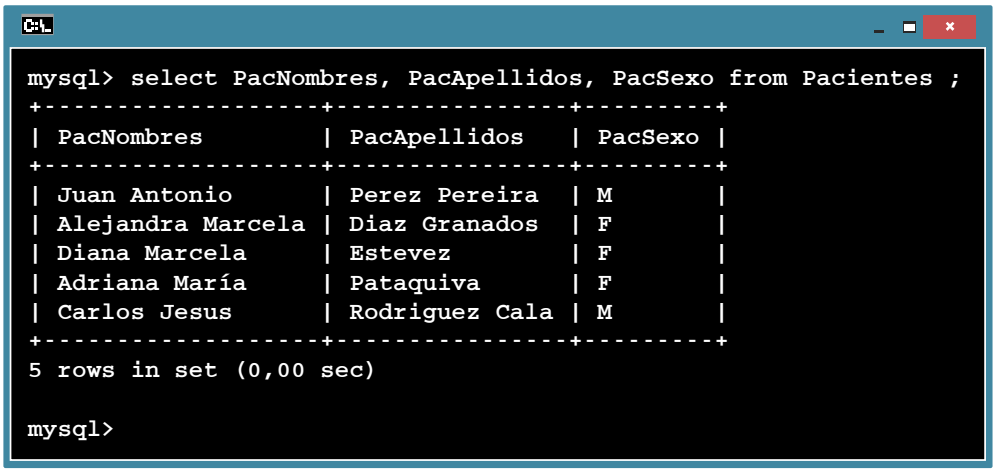


Figura 6.5. SQL para hacer consultas a la base de datos con límite de campos.

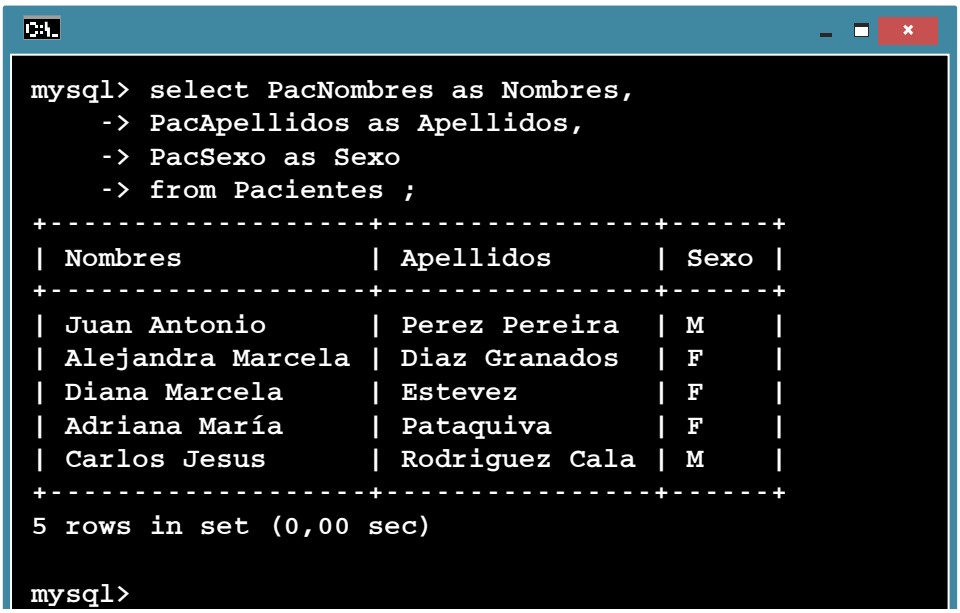
En la anterior consulta solo se trajeron los campos especificados.

Se pueden cambiar los nombres de los campos para que sean más entendibles para el usuario final a través de un “alias”.

La sintaxis para cuando se usan alias es la siguiente:

```
mysql> SELECT <nombre_campo> AS nombre_alterno_de_campo FROM <tabla>;
```

Para el ejemplo anterior se puede mejorar los nombres de los campos así:



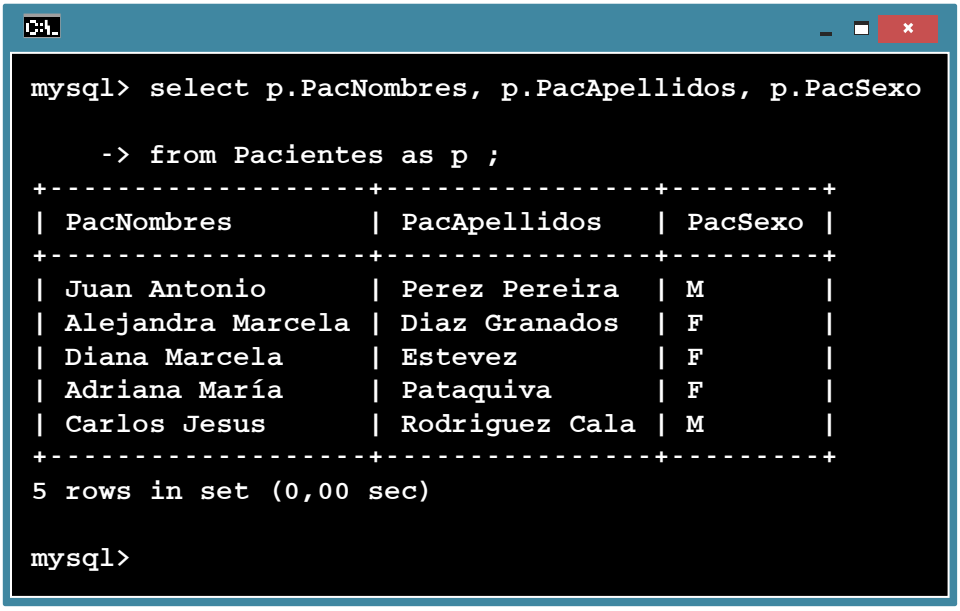
```
mysql> select PacNombres as Nombres,
    -> PacApellidos as Apellidos,
    -> PacSexo as Sexo
    -> from Pacientes ;
+-------------------+----------------+------+
| Nombres           | Apellidos      | Sexo |
+-------------------+----------------+------+
| Juan Antonio      | Perez Pereira  | M    |
| Alejandra Marcela | Diaz Granados  | F    |
| Diana Marcela     | Estevez        | F    |
| Adriana María     | Pataquiva      | F    |
| Carlos Jesus      | Rodriguez Cala | M    |
+-------------------+----------------+------+
5 rows in set (0,00 sec)

mysql>
```

Figura 6.6. SQL para hacer consultas a la base de datos con alias de campos.

Las tablas tambien pueden tener un alias que será de gran utilidad cuando se realicen los JOIN o enlaces. La sintaxis es la siguiente:

```
mysql> SELECT <nombre_campo> FROM <tabla> [AS] nombre_alterno_de_tabla;
```

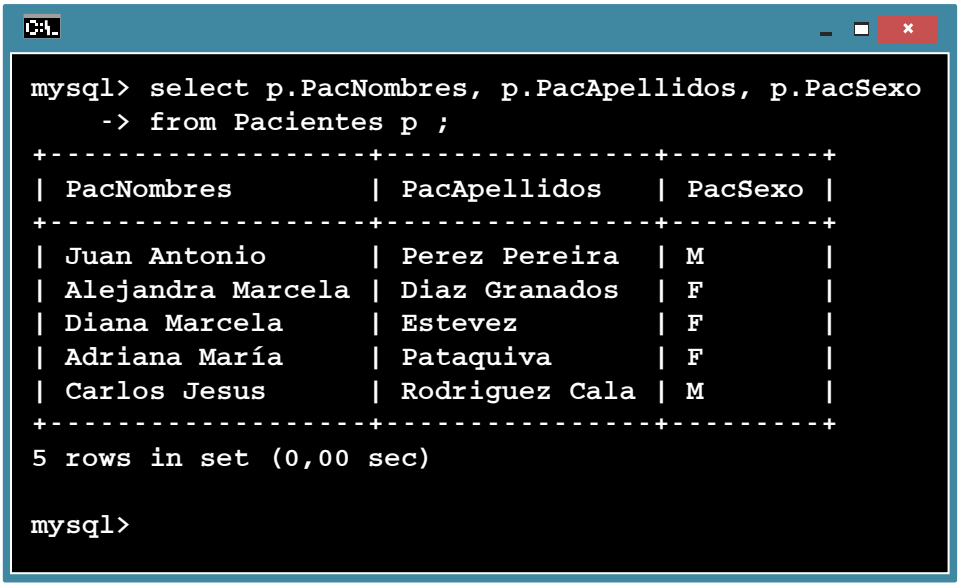
Para el ejemplo anterior se puede aplicar un alias a la tabla así:



```
mysql> select p.PacNombres, p.PacApellidos, p.PacSexo

    -> from Pacientes as p ;
+-------------------+----------------+---------+
| PacNombres        | PacApellidos   | PacSexo |
+-------------------+----------------+---------+
| Juan Antonio      | Perez Pereira  | M       |
| Alejandra Marcela | Diaz Granados  | F       |
| Diana Marcela     | Estevez        | F       |
| Adriana María     | Pataquiva      | F       |
| Carlos Jesus      | Rodriguez Cala | M       |
+-------------------+----------------+---------+
5 rows in set (0,00 sec)

mysql>
```

Figura 6.7. SQL para hacer consultas a la base de datos con alias de campos y tablas.

El prefijo “AS” se puede omitir. Para el ejemplo requiere funciona lo siguiente:



```
mysql> select p.PacNombres, p.PacApellidos, p.PacSexo
    -> from Pacientes p ;
+------------------+---------------+---------+
| PacNombres       | PacApellidos  | PacSexo |
+------------------+---------------+---------+
| Juan Antonio     | Perez Pereira | M       |
| Alejandra Marcela| Diaz Granados | F       |
| Diana Marcela    | Estevez       | F       |
| Adriana María    | Pataquiva     | F       |
| Carlos Jesus     | Rodriguez Cala| M       |
+------------------+---------------+---------+
5 rows in set (0,00 sec)

mysql>
```

Figura 6.8. SQL para hacer consultas con alias de campos y tablas.

6.3 Aplicación de filtros a las consultas.

El filtro como su nombre lo indica limita el número de registros a mostrar a través de la incorporación de una o varias condiciones lógicas.

En el lenguaje SQL los filtros se implementan con la cláusula “WHERE” seguido de una o más preguntas lógicas. Estas preguntas se evalúan una a una y se aplican operaciones de matemática booleana para determinar el resultado lógico que siempre deberá ser “verdadero” o “falso”

La sintaxis es:

```
SELECT <campos>
FROM <tablas>
WHERE <bloque de condiciones>
```

Ejemplo, para seleccionar los pacientes de sexo femenino se procede con la siguiente consulta.

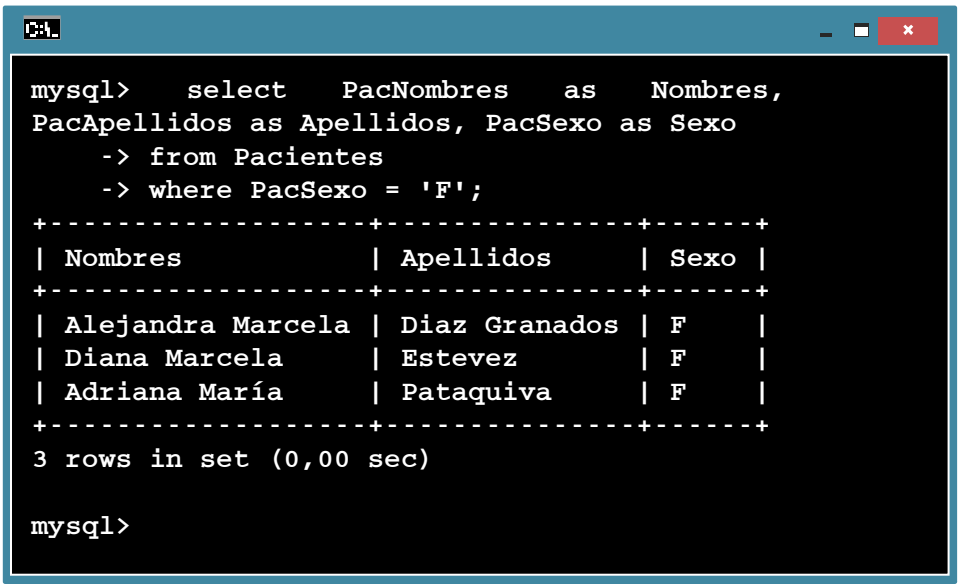


Figura 6.9. SQL para hacer consultas con alias de campos y tablas.

En el ejemplo anterior la pregunta " PacSexo = 'F' " se hace a cada registro de la tabla "Pacientes". Aquellos registros en los cuales la pregunta es verdadera son extraidos.

Se pueden hacer múltiples preguntas apoyados en los operadores lógicos AND y OR.

Las preguntas se pueden agrupar con paréntesis. De la misma forma como se evalúan las expresiones aritméticas las expresiones booleanas encerradas en paréntesis se evalúan primero.

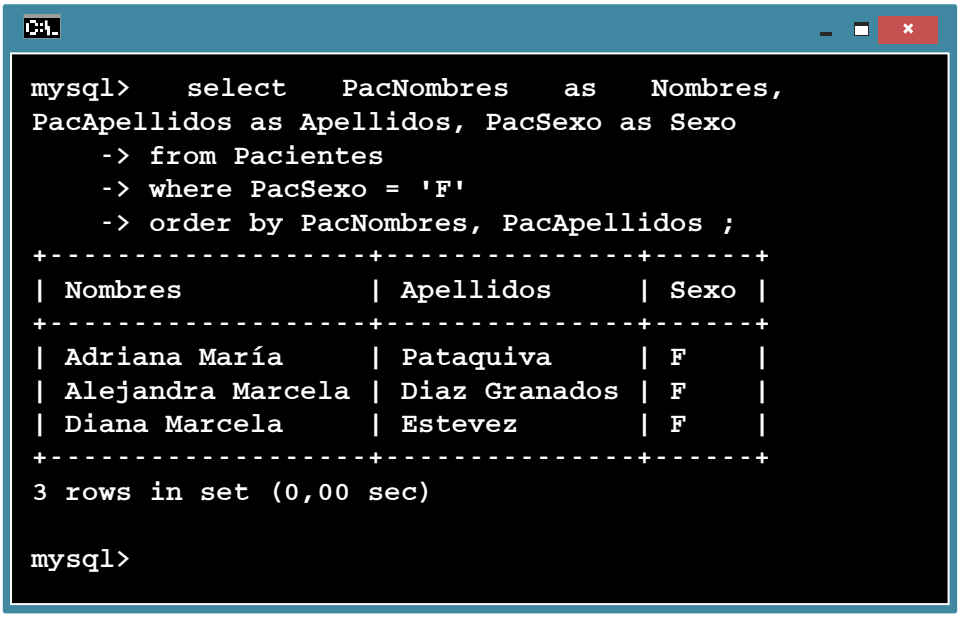
6.4 Ordenamiento de las consultas.

Los resultados de las consultas se pueden ordenar en un orden especificado con el comando "ORDER BY".

La sintaxis es la siguiente:

```
SELECT <campos>
FROM <tablas>
WHERE <bloque de condiciones>
ORDER BY <lista de los campos separados por coma> ASC | DESC
```

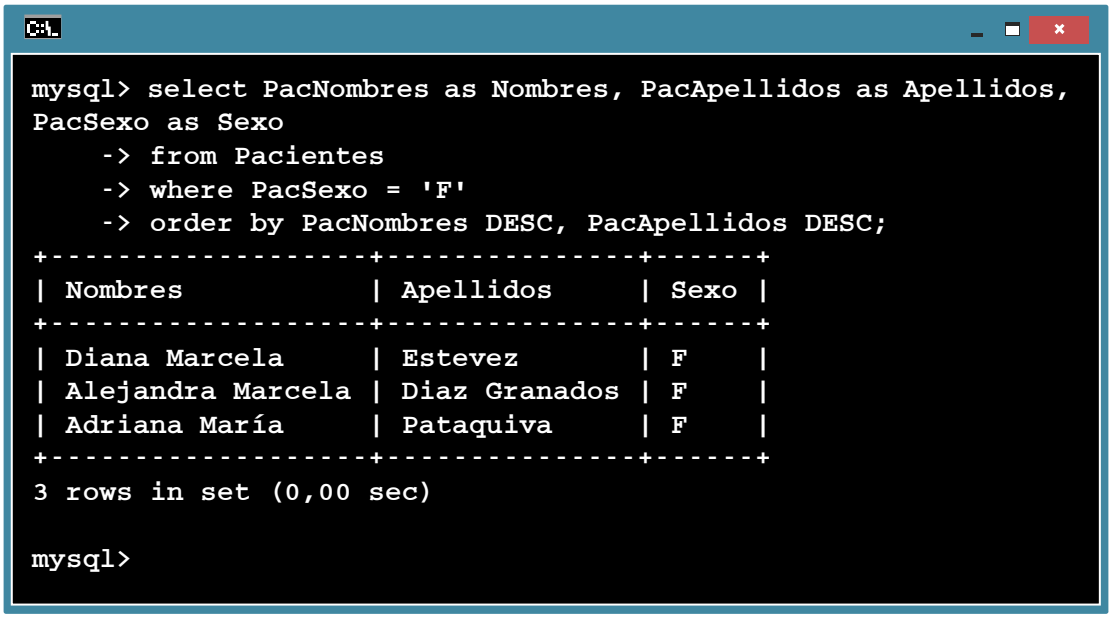
En el siguiente ejemplo la consulta es ordenada por dos campos: PacNombres y PacApellidos.



```
mysql>   select   PacNombres   as   Nombres,
PacApellidos as Apellidos, PacSexo as Sexo
    -> from Pacientes
    -> where PacSexo = 'F'
    -> order by PacNombres, PacApellidos ;
+------------------+--------------+------+
| Nombres          | Apellidos    | Sexo |
+------------------+--------------+------+
| Adriana María    | Pataquiva    | F    |
| Alejandra Marcela | Diaz Granados | F    |
| Diana Marcela    | Estevez      | F    |
+------------------+--------------+------+
3 rows in set (0,00 sec)

mysql>
```

Figura 6.10. SQL query con filtro y ordenamiento.

Por defecto el ordenamiento es ascendente. Si se requiere en orden descendente se procede aplicando el modificador “DESC” así:



```
mysql> select PacNombres as Nombres, PacApellidos as Apellidos,
PacSexo as Sexo
    -> from Pacientes
    -> where PacSexo = 'F'
    -> order by PacNombres DESC, PacApellidos DESC;
+------------------+--------------+------+
| Nombres          | Apellidos    | Sexo |
+------------------+--------------+------+
| Diana Marcela    | Estevez      | F    |
| Alejandra Marcela | Diaz Granados | F    |
| Adriana María    | Pataquiva    | F    |
+------------------+--------------+------+
3 rows in set (0,00 sec)

mysql>
```

Figura 6.11. SQL query con filtro y ordenamiento descendente.

6.5 Introducción a las consultas multi-tablas

En muchos casos se requiere que la consulta extraiga la información de varias tablas al

mismo tiempo.

En estos casos MySQL y en general el lenguaje SQL suministra la opción “JOIN”. La sintaxis es la siguiente:

```
SELECT <campos>
FROM <tabla1 alias t1 > JOIN <tabla2 alias t2 > ON <condición de unión>
WHERE <bloque de condiciones>
```

Ejemplo: se requiere listado con la cédula, nombres, apellidos y la descripción del tratamiento realizado por cada paciente.

Identificación	Nombres	Apellidos	Tratamiento	Observaciones
91.222..333	Carlos Jesús	Rodríguez Cala	Tratamiento de conductos	Paciente con hipertensión

Figura 6.12. Ejemplo para consultas multi-tablas.

Los campos nombres y apellidos están contenidos en la tabla “Pacientes”.
Los campos descripción y observaciones del tratamiento está contenido en la tabla “Tratamientos”.
La cédula es la llave primaria en la tabla “Pacientes” y es una llave foránea en la tabla “Tratamientos”.

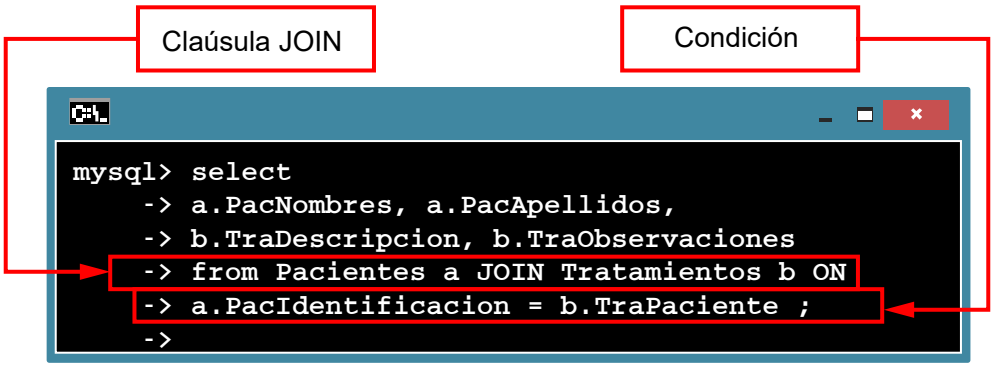


Figura 6.13. Ejemplo de una consulta con join.

Para hacer el enlace o “join” de las tablas se procede así:

El resultado es el siguiente:

```
+------------------+---------------+-------------------------+----------------------------+
| PacNombres       | PacApellidos  | TraDescripcion          | TraObservaciones           |
+------------------+---------------+-------------------------+----------------------------+
| Carlos Jesus     | Rodriguez Cala | Tratamiento de Conductos | Paciente con hipertensión |
| Adriana María    | Pataquiva     | Profilaxis              | Sin novedad                |
| Diana Marcela    | Estevez       | Resina                  | Paciente con sensibilidad  |
| Alejandra Marcela | Diaz Granados | Profilaxis             | Sin novedad                |
+------------------+---------------+-------------------------+----------------------------+
4 rows in set (0,01 sec)

mysql>
```

Figura 6.14. Resultado de la consulta con join.

El enlace o "join" genera una tabla a la cual se le pueden aplicar otros filtros. Para el ejemplo anterior se puede restringir la consulta sólo a los pacientes femeninos. En este caso el comando quedaría así:

```
mysql> select
    -> a.PacNombres, a.PacApellidos,
    -> b.TraDescripcion, b.TraObservaciones
    -> from Pacientes a JOIN Tratamientos b ON
    -> a.PacIdentificacion = b.TraPaciente
    → where a.PacSexo = 'F` ;
+------------------+---------------+-------------------------+----------------------------+
| PacNombres       | PacApellidos  | TraDescripcion          | TraObservaciones           |
+------------------+---------------+-------------------------+----------------------------+
| Alejandra Marcela | Diaz Granados | Profilaxis             | Sin novedad                |
| Diana Marcela    | Estevez       | Resina                  | Paciente con sensibilidad  |
| Adriana María    | Pataquiva     | Profilaxis              | Sin novedad                |
+------------------+---------------+-------------------------+----------------------------+
3 rows in set (0,01 sec)

mysql>
```

Figura 6.15. Resultado de la consulta con join.

6.6. Subconsultas.

Las subconsultas son como su nombre lo indica una consulta que se lleva a cabo dentro de otra consulta. Esto se presenta cuando se requiere consultar un dato o varios datos que están en distintas tablas pero no se quiere o no se puede realizar un JOIN que sería la forma más organizada de realizar esta labor.

6.6.1. Subconsultas con resultados escalares.

MySQL y en general el lenguaje SQL permite realizar subconsultas con la siguiente sintaxis:

```
SELECT * FROM tabla1 WHERE campo1 = (SELECT campo1 FROM t2);
```

Como se puede observar la subconsulta debe estar contenida entre un paréntesis. El lenguaje SQL primero resuelve la subconsulta y luego compara los resultados.

En este caso la subconsulta debe arrojar un único valor de lo contrario no se podrá realizar la comparación y el MySQL generará un error.

Ejemplo. Se puede recorrer la tabla Tratamientos y traer el nombre del paciente sin necesidad de hacer JOIN de la siguiente manera:

```
mysql> select
    -> ( SELECT a.PacNombres FROM Pacientes a WHERE a.PacIdentificacion = b.TraPaciente ) as
nombre,
   -> ( SELECT a.PacApellidos FROM Pacientes a WHERE a.PacIdentificacion = b.TraPaciente) as ape-
llidos,
    -> b.TraDescripcion,
    -> b.TraObservaciones
    -> from Tratamientos b
    -> ;
+------------------+----------------+--------------------------+----------------------------+
| nombre           | apellidos      | TraDescripcion           | TraObservaciones           |
+------------------+----------------+--------------------------+----------------------------+
| Carlos Jesus     | Rodriguez Cala | Tratamiento de Conductos | Paciente con hipertensión  |
| Adriana María    | Pataquiva      | Profilaxis               | Sin novedad                |
| Diana Marcela    | Estevez        | Resina                   | Paciente con sensibilidad  |
| Alejandra Marcela | Diaz Granados | Profilaxis               | Sin novedad                |
+------------------+----------------+--------------------------+----------------------------+
4 rows in set (0,00 sec)

mysql>
```

Figura 6.16. Ejemplo de una consulta con subconsultas.

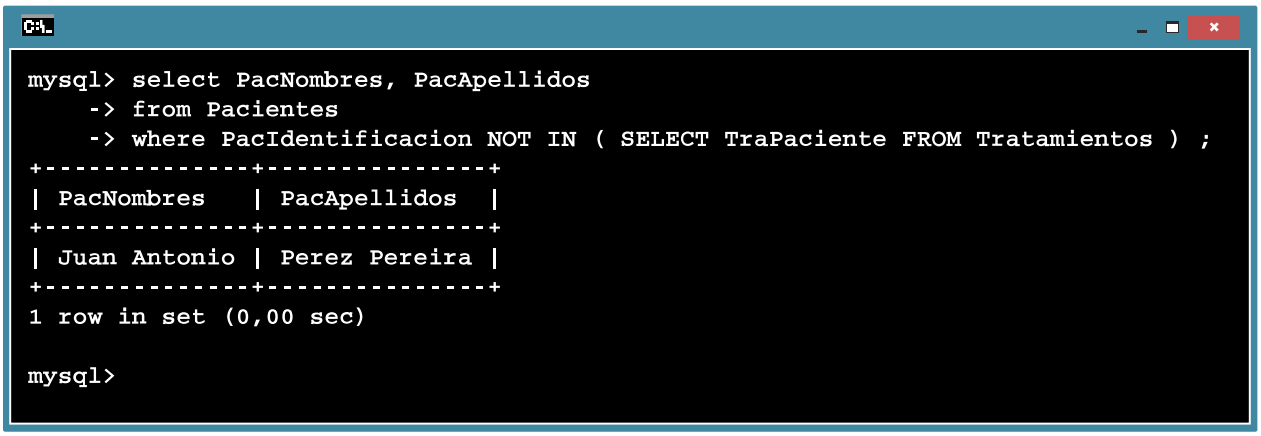
6.6.2. Subconsultas que retornan conjuntos de registros.

Las subconsultas pueden arrojar múltiples datos. En estos casos ya no se pueden comparar con los comparadores escalares como “=” , “>” , “<”. Se deben comparar con operadores de conjuntos. La sintaxis es la siguiente:

```
SELECT * FROM tabla1 WHERE campo1 [NOT] IN (SELECT campo1 FROM t2);
```

Esta sintaxis indica que la cláusula WHERE es verdadera si el campo1 de la tabla1 está contenido en el resultado de la sub consulta.

Ejemplo. Se puede determinar cuales pacientes aún no tienen un tratamiento así:



```
mysql> select PacNombres, PacApellidos
    -> from Pacientes
    -> where PacIdentificacion NOT IN ( SELECT TraPaciente FROM Tratamientos ) ;
+--------------+---------------+
| PacNombres   | PacApellidos  |
+--------------+---------------+
| Juan Antonio | Perez Pereira |
+--------------+---------------+
1 row in set (0,00 sec)

mysql>
```

Figura 6.17. Ejemplo de una subconsulta que retorna un conjunto de registros.

Además del comparador IN existen otros comparadores como ANY, ALL, SOME.

GLOSARIO

Alias: nombre alterno que pueden tomar los campos de una tabla o las tablas mismas para efectos de simplificar o mejorar su descripción.

Condición: partes que en su conjunto forman un filtro.

Consulta: conjunto de instrucciones en lenguaje SQL que realizan una consulta a la base de datos.

D.D.L.: acrónimo de Data Definition Language. Subconjunto del lenguaje SQL utilizado para crear, modificar o borrar los componentes de las bases de datos.

D.M.L: acrónimo de Data Manipulation Language. Subconjunto del lenguaje SQL que trata la forma con los registros son creados, actualizados, consultados o borrados.

Filtro: condición que limita la cantidad de registros en una consulta.

GPL: acrónimo de General Public License. Tipo de licenciamiento abierto para software.

Join: unión entre una o más tablas.

Llave primaria: campo de una tabla cuyo valor identifica inequívocamente un registro.

Llave foránea: campo de una tabla cuyo valor apunta a la llave primaria de otra tabla.

SQL: acrónimo de Structured Query Language. Lenguaje para el manejo de bases de datos.

BIBLIOGRAFÍA

Dubois, P. (2009). *Mysql Fourth Edition.* Madrid: Addison-Wesley.

Mysql. (2017). *Mysql 5.6. Reference Manual*. Recuperado de https://dev.mysql.com/doc/refman/5.6/en/

CONTROL DEL DOCUMENTO

CONSTRUCCIÓN OBJETO DE APRENDIZAJE

INTRODUCCIÓN A LA BASE DE DATOS MYSQL

Centro Industrial de Mantenimiento Integral - CIMI
Regional Santander

Líder línea de producción:	Santiago Lozada Garcés.
Asesores pedagógicos:	Rosa Elvia Quintero Guasca. Claudia Milena Hernández Naranjo.
Líder expertos temáticos:	Rita Rubiela Rincón Badillo.
Expertos temáticos:	Magda Milena García Gamboa.(V1) Nelson Mauricio Silva Maldonado.(V2)
Diseño multimedia:	Tirso Fernán Tabares Carreño
Programador:	Francisco José Lizcano Reyes
Producción de audio:	Víctor Hugo Tabares Carreño