

BASE DE DATOS

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	
LOBLOB LOBTEXT	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.

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.

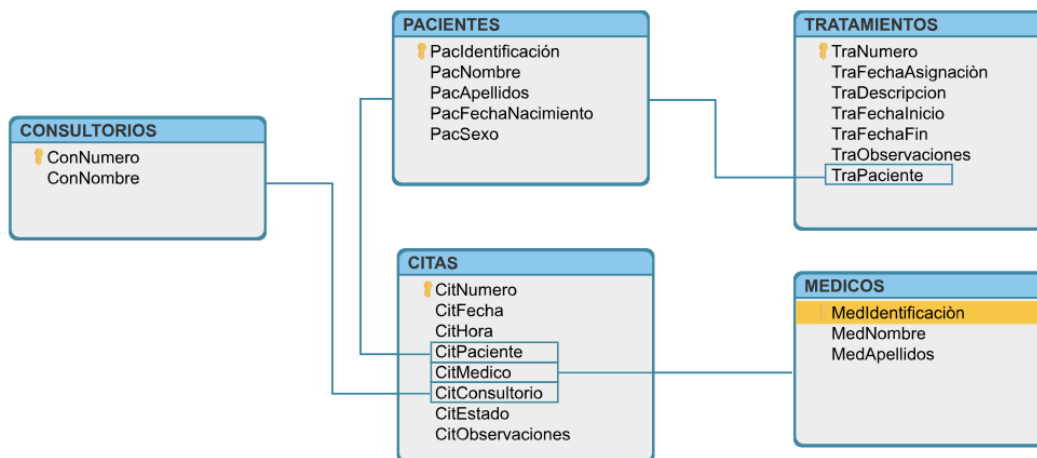


Figura 2.1. Diagrama relacional de la base de datos "Citas".

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 "Médicos".

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.

CREACION DE LA BASE DE DATOS:

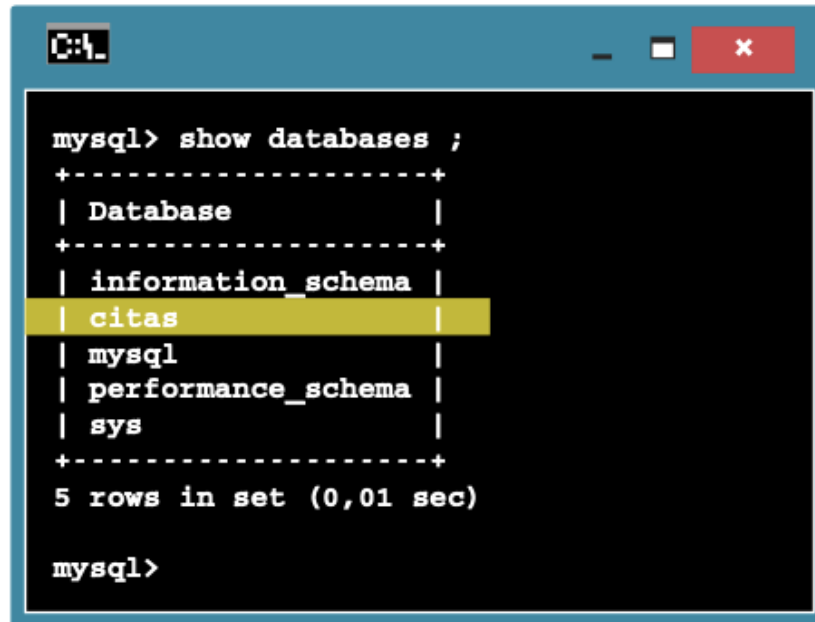
```
create database nombre_basedatos;
```

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

```
use nombre_base_de_datos ;
```

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

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.

3.2 Creación de las tablas.

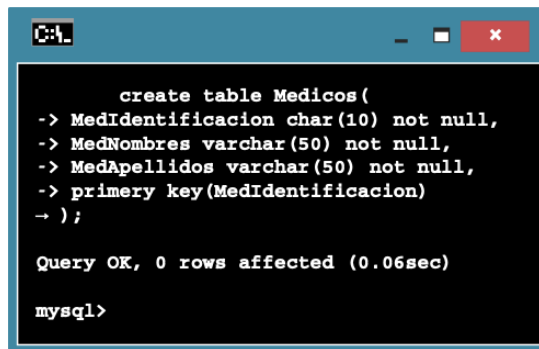
Una vez definida “citas” como base de datos predeterminada, se procede a crear las respectivas tablas.

La sintaxis para la creación de tablas es la siguiente:

```
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:



```
C:\>

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)

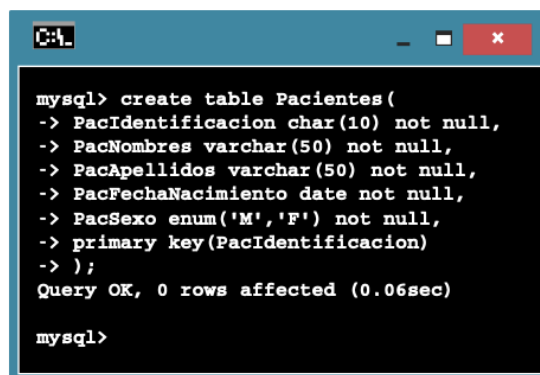
mysql>
```

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:

3.2.2 Creación de la tabla “Pacientes”.

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



```
C:\>

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)
-> );

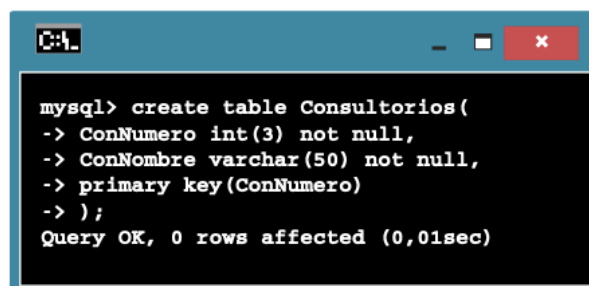
Query OK, 0 rows affected (0.06sec)

mysql>
```

Figura 3.8. SQL para crear una tabla.

3.2.3. Creación tabla “Consultorios”.

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



```
C:\>

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.

3.2.4. Creación tabla "Citas".

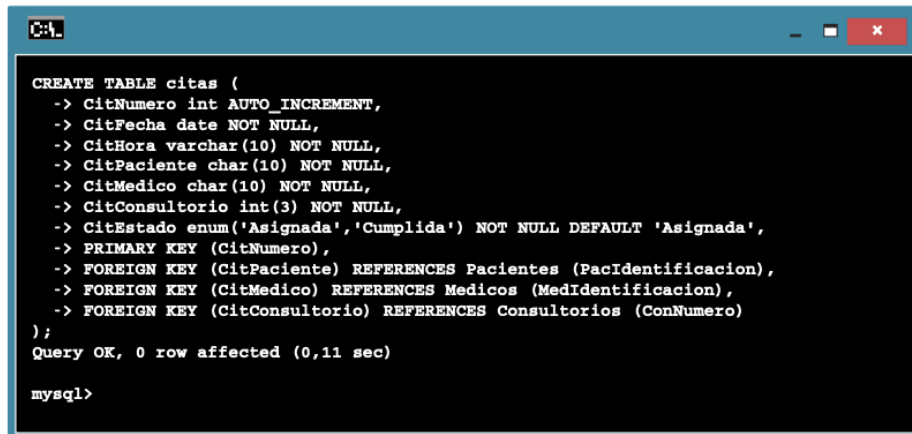
Al revisar la estructura de la tabla "Citas" definida en un numeral anterior se puede observar que esta tabla tiene unos modificadores nuevos. Además esta tabla contiene llaves foráneas, es decir, está relacionada con otras tablas.

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:

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.

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.

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.

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:

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>
```

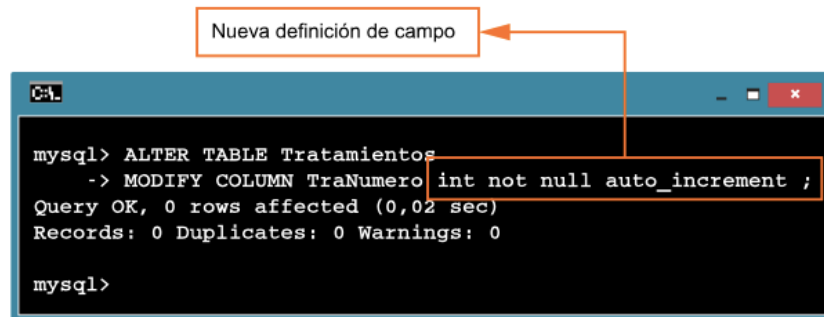
Figura 4.3. SQL para crear una tabla.

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:



```
mysql> ALTER TABLE Tratamientos
-> MODIFY COLUMN TraNumero int not null auto_increment ;
Query OK, 0 rows affected (0,02 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql>
```

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

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>);
```

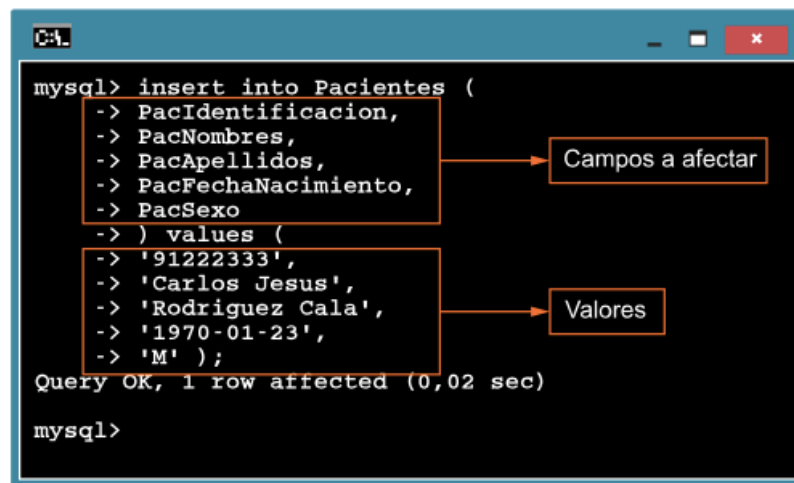
5.1.1 Inserción de registros sin llaves foráneas.

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:



```
mysql> insert into Pacientes (  
-> PacIdentificacion,  
-> PacNombres,  
-> PacApellidos,  
-> PacFechaNacimiento,  
-> PacSexo  
-> ) values (  
-> '91222333',  
-> 'Carlos Jesus',  
-> 'Rodriguez Cala',  
-> '1970-01-23',  
-> 'M' );  
Query OK, 1 row affected (0,02 sec)  
mysql>
```

The screenshot shows a MySQL command prompt window with a black background and white text. The command being entered is an INSERT statement into the 'Pacientes' table. The fields to be inserted are PacIdentificacion, PacNombres, PacApellidos, PacFechaNacimiento, and PacSexo. The values are '91222333', 'Carlos Jesus', 'Rodriguez Cala', '1970-01-23', and 'M'. The command is executed successfully, returning 'Query OK, 1 row affected (0,02 sec)'. There are two orange boxes with arrows pointing to specific parts of the command: one box labeled 'Campos a afectar' points to the list of field names in parentheses, and another box labeled 'Valores' points to the list of values in parentheses.

Figura 5.2. Ejemplo del comando Insert into.

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.

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.

```
SELECT * FROM Pacientes ;
```



```
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   | Pataquiwa     | 1990-12-24         | F       |
| 91222333         | Carlos Jesus    | Rodríguez 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.

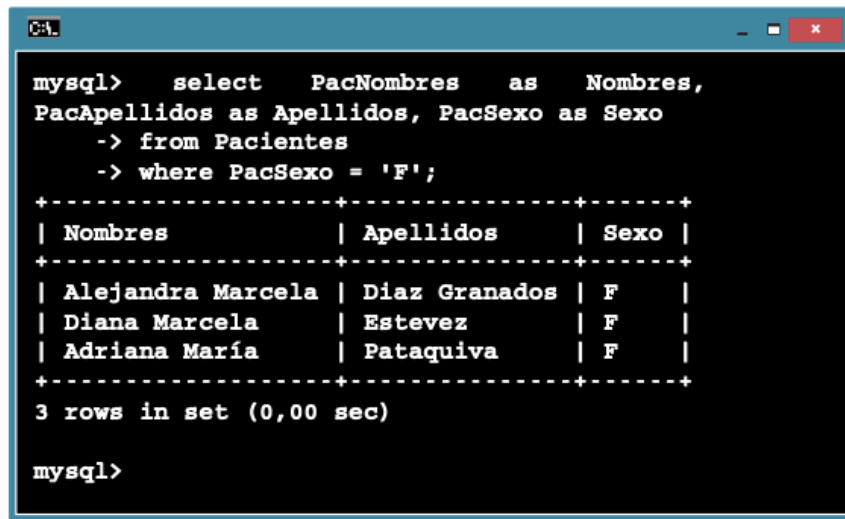
6.3 Aplicación de filtros a las consultas.

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>
```



```
mysql> select PacNombres as Nombres,
PacApellidos as Apellidos, PacSexo as Sexo
-> from Pacientes
-> where PacSexo = 'F';

+-----+-----+-----+
| Nombres      | Apellidos    | Sexo |
+-----+-----+-----+
| Alejandra Marcela | Diaz Granados | F    |
| Diana Marcela    | Estevez      | F    |
| Adriana María    | Pataquiwa    | F    |
+-----+-----+-----+
3 rows in set (0,00 sec)

mysql>
```

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 extraídos.

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 | Díaz 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 | Díaz Granados  | F    |
| Adriana María     | Pataquiva      | F    |
+-----+-----+-----+
3 rows in set (0,00 sec)

mysql>
```

Figura 6.11. SQL query con filtro y ordenamiento descendente.

GLOSARIO

Alias: nombre alternativo 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.