

Introducción a MySQL

Por Francisco Javier Arce Anguiano

Introducción a MySQL

Introducción a MySQL	4
Términos básicos: Tabla, campo y registro.....	4
Diseño de una base de datos.....	4
Tipo de datos	5
Valores numéricos.....	5
Valores (caracteres) de cadena.....	6
Valores de fecha y hora.....	7
Valor NULL	7
Tipo de columnas MySQL.....	7
Tipo de columna entera.....	8
Atributos de campos numéricos	9
Tipo de columna decimal.....	9
Tipo de columna cadena.....	10
Tipo de columna de fecha y tiempo	12
Cómo elegir entre los tipos de las columnas.....	14
Algo más sobre los atributos	14
Evaluación de la expresión y conversión del tipo	15
El SQL de MySQL.....	17
Como entrar a MySQL.....	19
Desde el sistema operativo con comandos de línea:	19
Desde PHPmyAdmin	20
Desde un sistema.....	21
Crear una base de datos.....	21
Seleccionar una base de datos	22
Mostrar las bases de datos.....	22
Crear una tabla	22
Mostrar las tablas en una base de datos.....	24
Eliminación de bases de datos y de tablas.....	24
Insertando datos	25
Insertar datos por medio de un archivo SQL.....	26
Insertar datos desde un archivo externo	26
Consulta de datos	29
FROM	33
WHERE.....	33
ORDER BY	36
GROUP BY.....	39
HAVING	40
LIMIT	41
ALIAS	42
USO DE OPERADORES DE COMPARACIÓN ESPECIALES	43
IN.....	44
BETWEEN x AND y	44
LIKE.....	45
IS NULL.....	46
FUNCIONES DE AGREGADO.....	46

Introducción a MySQL

Actualización de datos	49
Borrando datos	50
Modificación de tablas con ALTER TABLE	51
Cambiar una columna	52
Añadir una columna	52
Añadir un índice	52
Añadir una llave principal	53
Cambio del valor predeterminado	53
Eliminar una columna	54
Borrar un índice	54
Borrar una llave principal	54
Modificar la declaración de una columna	55
Renombrar una tabla	55
Ver la estructura de una tabla	55
Optimización y mantenimiento de una base de datos	57
Optimizar el espacio de una tabla	57
Cambiar de contraseña a un usuario	57
Bloquear el acceso a una tabla	57
Desbloquear una tabla	57

Introducción a MySQL

Introducción a MySQL

MySQL fue desarrollado en 1996 por una empresa sueca llamada TCX. Es una base de datos relacional de código abierto que se encuentra estrechamente relacionada con productos como PHP, Apache y Linux, que también son de código abierto.

MySQL es un sistema administrador de bases de datos relacionales que permite el manejo, acceso y almacenamiento seguro y rápido de información, especializado en Internet (sistemas cliente-servidor). Usted puede bajar la base de datos en forma gratuita de la página <http://www.mysql.org>. Recomendamos ampliamente la instalación de la paquetería gratuita de Easyphp.org, que instala en un mismo paquete el sistema operativo web Apache y PHP con MySQL.

Antes de abordar el manejo de MySQL definiremos términos bases.

Términos básicos: Tabla, campo y registro

Una TABLA es un conjunto de información que tiene relación entre sí. Una BASE DE DATOS es el conjunto de tablas que describen una entidad de información mayor. Las tablas pueden tener relación entre ellas y ser complementarias.

Las tablas están formadas por REGISTROS o RENGLONES. Estos son la unidad básica que describe a un objeto o a una transacción, por ejemplo, los datos de un artículo de venta, la información de una persona para enviarle un correo, etc. A su vez el REGISTRO está formado por CAMPOS o COLUMNAS, que son la unidad básica de información, y son una propiedad específica de un objeto o transacción, por ejemplo, la fecha de compra, el número de ISBN de un libro o el correo electrónico de una persona.

Dependiendo de la COLUMNA, esta tendrá un TIPO DE DATO específico, que indicará la forma de almacenar, las características y limitantes del CAMPO.

Diseño de una base de datos

Para diseñar una base de datos podemos seguir los siguientes pasos:

Introducción a MySQL

1. Cada tabla deberá tener un nombre único y específico.
2. Cada una de las tablas deberá tener al menos un campo
3. Cada tabla puede tener cero o más filas. estas pueden estar desordenadas.
4. Cada valor en una COLUMNA tendrá el mismo tipo de dato.
5. A su vez, el campo o conjunto de campos único (que no se repite su valor entre los diferentes REGISTROS) con el cual se accesa la información de las tablas, y cuyo valor identifica unívocamente al REGISTRO se le conoce como LLAVE PRIMARIA.
6. Solo existirá una llave primaria por tabla.
7. Una LLAVE SECUNDARIA (externa o foránea) es un campo o conjunto de campos, que es una LLAVE PRIMARIA en otra tabla.
8. La relación entre LLAVES PRIMARIAS - LLAVES FORANEAS crea una relación de padre-hijo entre las TABLAS de una base de datos.

Por ejemplo:

Supongamos que tenemos un sistema escolar. Tendremos una tabla con los datos del alumno (llamada ALUMNOS), y en ella almacenaremos una llave para el número de salón. Esta será una LLAVE SECUNDARIA.

En otra tabla (la llamaremos SALONES) almacenaremos los datos de los salones, como su ubicación en el edificio, su número de asientos y su equipamiento (pizarrones, proyectores, mapas, etc). La LLAVE PRIMARIA de esta tabla es la LLAVE SECUNDARIA de la tabla ALUMOS, por lo que se tiene una relación de PADRE (Tabla ALUMNOS) e hijo (tabla SALONES).

Antes de pasar a la creación de tablas y bases de datos, estudiaremos el elemento básico, que son los tipos de datos y los tipos de columnas para formar las tablas.

Tipo de datos

MySQL reconoce varios tipos de datos, o lo que es lo mismo, categorías generales (abstracciones) cuyos valores se pueden representar:

Valores numéricos

Introducción a MySQL

Los números son valores como 48 o 193.62. MySQL entiende los números especificados como enteros (sin parte fraccional) o valores de coma flotante (que si lo tienen). Los enteros se pueden especificar de forma decimal o hexadecimal.

Un entero consiste en una secuencia de dígitos. Un entero hexadecimal está compuesto por "0x" seguido de uno o mas números hexadecimal.

MySQL reconoce las notaciones científicas. Van indicadas inmediatamente después de un número entero o de coma flotante con una "e" o "E", un signo "+" o "-" y un exponente entero. 1.34E+12 y 42.27e-1 son números con notación científica correcta.

Los decimales se conocerán como DOBLES o de punto FLOTANTE, ya que se visualizan como un número con dos partes enteras divididas por un punto, por ejemplo 100.30.

Valores (caracteres) de cadena

Las cadenas son valores como "MySQL es una base de datos" o "en el mar, la vida es más sabrosa". Puede utilizar comillas dobles o sencillas para agrupar a una cadena. Para representar caracteres de escape existen varias secuencias de escape. Cada secuencia comienza con una barra invertida (\), que significa un escape temporal de las reglas para la interpretación de los caracteres.

Secuencia	Significado
\0	ASCII 0
\'	Comilla simple
\"	Comilla doble
\b	Retroceso
\n	Nueva línea
\r	Retorno
\t	Tabulador
\\	Barra invertida

Por ejemplo:

```
'I can \t'
```

```
"Marco dijo, \"Hasta Luego\""
```

Introducción a MySQL

Valores de fecha y hora

Fechas y horas son valores del tipo "1999-06-17" o "12:30:43". También reconoce las combinaciones de ambos valores como "1999-06-17 12:30:43". Como se puede observar, MySQL mantiene el formato YYYYMMDDHHMMSS, sin embargo se puede cambiar el mismo con la función DATE_FORMAT().

Valor NULL

NULL se puede considerar un valor "sin tipo". Normalmente quiere decir "sin valor", "valor desconocido", etc- Puede insertar en la tabla valores NULL y hacer selecciones de campos si tienen en valor NULL, etc. No se puede realizar ninguna operación con el valor NULL, pues su resultado siempre será NULL.

Tipo de columnas MySQL

Como ya se había dicho, una tabla es un conjunto de uno o más columnas. Cuando se crea una tabla con la sentencia CREATE TABLE, se especifica el tipo deseado para cada columna. Un TIPO DE COLUMNA es más específico que un TIPO DE DATO. El tipo de dato es una categoría general como "número" o "cadena". A un tipo de cadena le caracteriza precisamente la clase de valores que puede contener, como SMALLINT o VARCHAR(32).

Los tipos de columna MySQL son los responsables de describir qué clase de valores quiere que contenga una columna, lo que determina a su vez la manera en que MySQL tratará dichos valores. Cada tipo de columna tiene varias características:

- Qué clase de valores quiere almacenar.
- Cuánto espacio ocupan los valores, si son de longitud fija o de longitud variable.
- Cómo se clasifican o comparan los valores del tipo.
- Si el tipo permite o no valores NULL
- Si el tipo se puede indexar o no

Comprender los tipos de columna es muy importante para reducir el tamaño de la base de datos y aumentar su velocidad de los procesos.

MySQL proporciona tipos de columna para todas las categorías de valores de datos en general,

Introducción a MySQL

excepto para el valor NULL.

Tipo de columna entera

Para números enteros (que no tienen parte fraccional), como 43, -1, 3, 0 o -789345, los formatos que MySQL puede manejar son:

Nombre del tipo	Significado	Rango
TINYINT	Entera muy pequeña Requiere 1 byte	Con signo -128 a 127 Sin signo 0 a 255
SMALLINT	Entera pequeña Requiere 2 bytes	Con signo -32768 a 32767 Sin signo 0 a 65535
MEDIUMINT	Entera mediana Requiere 3 bytes	Con signo -8388608 a 8388607 Sin signo 0 a 1677215
INT	Entera estándar Requiere 4 bytes	Con signo -2147683648 a 2147483647 Sin signo 0 a 4,294,967,295
BIGINT	Entero muy grande Requiere 8 bytes	Con signo -9223372036854775808 a 9223372036854775807 Sin signo 0 a 18,446,744,073,709,551,615

Observe que el rango de los cinco tipos de enteros depende de si tiene o no signo (UNSIGNED), que es uno de los atributos. Un ATRIBUTO es una característica adicional a la columna. Para seleccionar algún tipo de columna necesitamos analizar el rango de nuestra información y seleccionar el tipo más pequeño que satisfaga ese rango, de lo contrario estaremos desperdiciando recursos de nuestra base de datos.

Por ejemplo, si tenemos un campo que almacenará la edad de una persona (entre 0 y 120 años) nos funciona perfectamente la columna tipo TINYINT. Si se utilizara para este campo un tipo de columna como INT (con 4 bytes), en realidad está desperdiciando 3 bytes. Si su base de datos es grande, por ejemplo, la población de un municipio con 100,000 habitantes, usted está desperdiciando en un solo campo 300,000 bytes.

Introducción a MySQL

Si usted intenta almacenar un campo mayor al rango del tipo de columna, el proceso marcará un error y solo cargará hasta el rango máximo, por ejemplo: Tiene un campo tipo TINYINT con signo, y almacena un valor como 500, solo se almacenará 127.

Atributos de campos numéricos

ZEROFILL: Tanto en Enteros como en punto flotante, se tiene el atributo ZEROFILL que sirve para rellenar la parte izquierda de la columna en cero.

AUTO_INCREMENT: El sistema numerará desde el número uno a cada uno de los renglones que sean creados en la tabla. Solo se puede tener un campo de este tipo y debe ser nombrado como una llave (PRIMARY KEY). o como clave (UNIQUE). No acepta valores nulos (NOT NULL).

UNSIGNED: Este comportamiento rechaza los valores negativos de una columna y se ajusta el rango aceptado como se muestra en la tabla de los números enteros.

Tipo de columna decimal

Nombre del tipo	Significado	Rango
FLOAT	Número único de coma flotante Requiere 4 bytes	+ -1.175494351E-38
DOUBLE	Número doble de coma flotante Requiere 8 bytes	+ -2.2250738585072014E-308
DECIMAL(M,D)	Número coma flotante representado como una cadena Requiere M+2 bytes	Depende del valor de M y D

Los número de coma flotante siempre son positivos y negativos. No tenemos atributos UNSIGNED. El tipo DECIMAL difiere del FLOAT en que el primero se maneja como cadena desde la versión 3.23 de MySQL. Los rangos para una columna DECIMAL son:

Introducción a MySQL

Campo	Rango
DECIMAL(4,1)	-999.9 a 9999.9
DECIMAL(5,1)	-9999.9 a 99999.9
DECIMAL(6,1)	-99999.9 a 999999.9
DECIMAL(6,2)	-9999.99 a 99999.99
DECIMAL(6,3)	-999.999 a 9999.999

La longitud de una columna DECIMAL equivale a DECIMAL(M+2,D), donde D se adapta al tamaño de M+2.

Tipo de columna cadena

MySQL proporciona varios tipos de cadena para mantener los datos de un carácter. Para todos los tipos de cadena, los valores demasiado largos se cortan para ajustarlos. Los tipos de cadena que maneja MySQL son (donde L es la longitud del valor):

Nombre del tipo	Significado
CHAR	Cadena de caracteres de longitud fija
VARCHAR	Cadena de caracteres de longitud variable
TINYBLOB	Binary Large Object (BLOB) muy pequeño
BLOB	BLOB pequeño
MEDIUMBLOB	BLOB Mediano
LOB	BLOB Largo
TINYTEXT	Cadena de texto muy pequeña
TEXT	Cadena de texto pequeña
MEDIUMTEXT	Cadena de texto mediana
LONGTEXT	Cadena de texto larga
ENUM	Una enumeración: columnas a las que se pueden asignar un miembro de enumeración
SET	Un conjunto: columnas a las que se pueden asignar múltiples conjuntos de miembros.

Introducción a MySQL

Especificación tipo	Tamaño máximo	Almacenamiento
CHAR(M)	M bytes	M bytes
VARCHAR(M)	M bytes	L+1 bytes
TINYBLOB, TINYTEXT	255 bytes	L+1 bytes
BLOB, TEXT	65535 bytes	L+2 bytes
MEDIUMBLOB MEDIUMTEXT	16,777,216 bytes (16 megas)	L+3 bytes
LOBLOB LOBTEXT	32 megas	

Los tipos de columnas más utilizados son CHAR y VARCHAR. Difieren que el primero es de una longitud fija y que el segundo es de una longitud variable. Si los campos que se van a almacenar no varían de longitud drásticamente (por ejemplo, un número telefónico) no conviene utilizar el campo VARCHAR, pues tiene un byte adicional de control, y los campos de longitud fija se procesan más rápido. Si existe una columna de longitud variable, todos los registros serán de longitud variable. Solo hay que utilizar los campos de longitud variable cuando se tenga un problema de espacio en disco.

Si usted combina columnas CHAR y VARCHAR, MySQL convertirá las columnas CHAR (con una longitud mayor a 4) a tipo VARCHAR.

MySQL tiene un bloque de campos TEXT (TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT) con longitud variable de los textos. Por su tamaño, es fácil que estos tipos de tablas se fragmenten (queden almacenadas en diferentes áreas del disco duro), por lo que es necesario continuamente optimizarlo (OPTIMIZE TABLE).

El tipo ENUM y SET permiten que los campos tengan un valor entre una enumeración (ENUM) o conjunto (SET). La diferencia es que un campo del tipo ENUM solo puede tomar un valores dentro de los valores determinados, por ejemplo:

```
CREATE TABLE e_table (e ENUM("a","b","c"));
```

Introducción a MySQL

Tipo de columna de fecha y tiempo

MySQL proporciona varios tipos de columna para valores temporales:

Nombre de tipo	Significado
DATE	Valor de fecha en formato YYYY-MM-DD "1000-01-01" a "9999-12-31" Requiere 3 bytes
TIME	Valor de hora en formato HH:MM:SS "-838:59:59" a "838:59:59" Requiere 3 bytes
DATETIME	Valor de fecha y hora en formato YYYY-MM-DD hh:mm:ss "1000-01-01 00:00:00" a "9999-12-31 23:59:59" Requiere 8 bytes
TIMESTAMP	Valor de un lapso de tiempo en formato YYYYMMDDhhmmss Requiere 4 bytes
YEAR	Año en formato YYYY. Requiere 1 byte

Si insertamos un valor ilegal, en los campos aparecerá el siguiente valor equivalente a "cero".

Especificación del tipo	Valor cero
DATE	0000-00-00
TIME	00:00:00
DATETIME	0000-00-00 00:00:00
TIMESTAMP	00000000000000
YEAR	0000

MySQL siempre representa las fechas con el año en primer lugar, según la especificación de ANSI.

Los tipos DATE, TIME y DATETIME contienen la fecha, la hora y los valores combinados de

Introducción a MySQL

ambos. Los formatos son YYYY-MM-DD, HH:MM:SS y YYYY-MM-DD HH:MM:SS respectivamente. Para DATETIME se necesita la parte de la Fecha y la Hora. Si falta alguna de ellas automáticamente MySQL las pone en cero.

El valor representado en TIME es el transcurso de tiempo, en cambio en DATETIME represente una hora del día. Si usted almacena información en una columna TIME (formato HH:DD:SS) un valor como 12:30 VERIFIQUE que el dato se almacene correctamente, pues en ocasiones se puede almacenar como 00:12:30, en lugar de 12:30:00.

Las columnas TIMESTAMP representan valores en formato YYYYMMDDHHMMSS, con un rango que puede oscilar entre el primero de enero de 1997 (la hora cero para UNIX) o sea 19700101000000 y cualquier fecha del antes del año 2037 (esta fecha puede variar según el sistema operativo). El tipo de columna TIMESTAMP podemos indicarle un "largo de pantalla" (caracteres que se desplegarán, de un número entre 2 y 14). Por ejemplo:

Especificación	Formato
TIMESTAMP(14)	YYYYMMDDHHMMSS
TIMESTAMP(8)	YYYYMMDD
TIMESTAMP(4)	YYMM

Obviamente este tipo de "pantalla" no afecta nada al dato almacenado en la base de datos, solo funciona para mostrar el dato al momento de desplegarse. Es posible que este largo de pantalla NO funcione cuando utiliza phpMyAdmin.

El tipo YEAR tiene un rango entre 1901 y 2115. Es útil cuando solo necesitamos la fecha de un año, pues se almacena en un byte, al igual que TINYINT, pero cambian los rangos (0 a 255).

En general puede asignar libremente valores entre los tipos DATE, DATETIME y TIMESTAMP, aunque hay ciertas restricciones:

- Si asigna un valor DATETIME o TIMESTAMP a una columna tipo DATE, se descartará la parte de la hora.
- Si asigna un valor DATE a un DATETIME o TIMESTAMP, la parte constitutiva de la hora se iguala a cero.
- Aunque los tipos DATETIME y TIMESTAMP tienen el mismo formato, tienen diferentes

Introducción a MySQL

rangos. TIMESTAMP tiene un rango entre 1970 y 2037. Si el valor de DATETIME no entra en estos años, la transferencia puede causar error.

Cómo elegir entre los tipos de las columnas

Para poder escoger entre todas las variaciones de tipo de columna, no está demás hacerse las siguientes preguntas (aunque algunas veces parecerán obvias):

¿Qué tipo de valores, números, cadenas y fechas albergará la columna?

¿Sus valores se corresponden con algún rango?

¿Qué tipos de columnas son los más eficientes?

Operaciones de cadena vs. numéricas.

Tipos más pequeños vs. tipos más grandes.

Tipo de longitud fija vs. longitud variable.

¿Como hay que comparar sus valores?

¿Bajo qué campos será indexada la tabla?

Algo más sobre los atributos

AUTO_INCREMENT es un atributo de columna que debería utilizar sólo con tipos enteros. Esto limita las posibilidades desde TINYINT a BIGINT.

Las columnas AUTO_INCREMENT debería estar indexadas de forma que se pueda averiguar rápidamente el número máximo de secuencia sin necesidad de revisar toda la tabla. Además, para evitar que los números se vuelvan a utilizar, el índice debe ser único, lo que significa que deberá declarar la columna como PRIMARY KEY o índice UNIQUE.

En este punto ya sabemos cuales son los tipos de datos y los tipos de columnas en que podemos almacenar la información. Antes de pasar a la operación de las bases de datos, explicaremos la evaluación de las expresiones regulares y de conversiones de tipo de datos.

Introducción a MySQL

Evaluación de la expresión y conversión del tipo

MySQL le permitirá escribir expresiones regulares que incluyen constantes, llamadas a funciones y referencias a las columnas de la tabla. Estos valores pueden combinarse usando diferentes tipos de operadores, como los aritméticos o los de comparación, y los términos de una expresión deberán aparecer entre paréntesis. Ejemplos de expresiones:

```
SELECT      CONCAT(nombre," ",apellidoPaterno),  
              (TO_DAYS(fechaFallecimiento)-TO_DAYS(fechaNacimiento)/365)  
FROM        alumnos  
WHERE       fechaNacimiento > "1990-01-01" AND genero = "M";
```

Las expresiones pueden utilizar llamadas a funciones. Algunas de estas pueden llevar argumentos y otras no. No debe haber espacios entre el nombre de la función y su paréntesis inicial:

```
COUNT(*)      //Bien definida  
COUNT (*)    //Mal definida
```

Si en una selección se utilizan más de una tabla, los nombres de las columnas pueden ir anteceditos del nombre de la tabla o de un ALIAS, por ejemplo:

```
SELECT       alumno.nombre, alumno.apellido, profesor.nombre, profesor.apellido  
FROM        alumno, profesor  
WHERE       alumno.apellido = profesor.apellido
```

Se pueden combinar todos los tipos de valores para formar expresiones más complejas.

Dentro de los operadores de comparación encontramos algunos que NO se encuentran en otros lenguajes:

Operador	Sintaxis	Significado
IN	a IN (b1,...,bn)	verdadero si está en la lista de valores
BETWEEN	a BETWEEN b AND c	Verdadero si se encuentra entre los valores de b y c
LIKE	a LIKE b	Verdadero si a es igual a b

Introducción a MySQL

NOT LIKE	a NOT LIKE b	Verdadero si a no es igual a b
IS NULL	a IS NULL	Verdadero si a es igual a NULL
IS NOT NULL	a IS NOT NULL	Verdadero si el operador no es nulo

Los patrones utilizados con el operador LIKE pueden incluir caracteres comodín tales como % y _, por ejemplo:

"Franklin" LIKE "Frank%"	resulta verdadero
"Frankfurter" LIKE "Frank%"	resulta verdadero

Introducción a MySQL

El SQL de MySQL

La forma que crearemos y manipularemos la información es por medio de un lenguaje llamado SQL (Structured Query Language), diseñado por IBM en los años setenta. El lenguaje se divide, en términos generales, en los siguientes módulos:

Selección de información (Data Query Language o DQL)

SELECT

Modificación de información de tablas

(Data Modification Language o DML)

DELETE

INSERT

LOAD DATA

OPTIMIZE TABLE

REPLACE

UPDATE

(Data definition language DDL)

Creación, vaciado y selección de bases de datos

CREATE DATABASE

DROP DATABASE

USE

Creación, alteración y vaciado de tablas e índices

ALTER TABLE

CREATE INDEX

CREATE TABLE

DROP INDEX

DROP TABLE

(Data control language DCL)

Sentencia de administración

FLUSH

GRANT

KILL

Introducción a MySQL

REVOKE

Obtención de información sobre base de datos y consultas

DESCRIBE

EXPLAIN

SHOW

Misceláneas de sentencias

CREATE FUNCTION

DROP FUNCTION

LOCK TABLES

SET

UNLOCK TABLES

Es una convención que las palabras clave y sentencias de SQL sean escritas en mayúscula y de ser posible en renglones separados, por ejemplo:

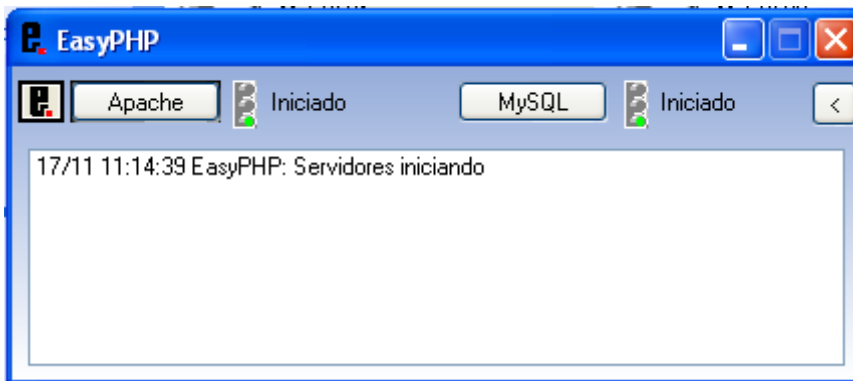
```
SELECT      CONCAT(nombre," ",apellidoPaterno),  
              (TO_DAYS(fechaFallecimiento)-TO_DAYS(fechaNacimiento)/365)  
FROM        alumnos  
WHERE       fechaNacimiento > "1990-01-01" AND genero = "M";
```

Aquí seguiremos en lo que sea posible esta convención.

Introducción a MySQL

Como entrar a MySQL

Para poder utilizar MySQL es necesario que sus servicios estén levantados, ya sea que trabaje en su propia computadora o en el servidor. Por ejemplo, si utiliza el módulo de easyPhp deberá ejecutarlo primero en su computadora:

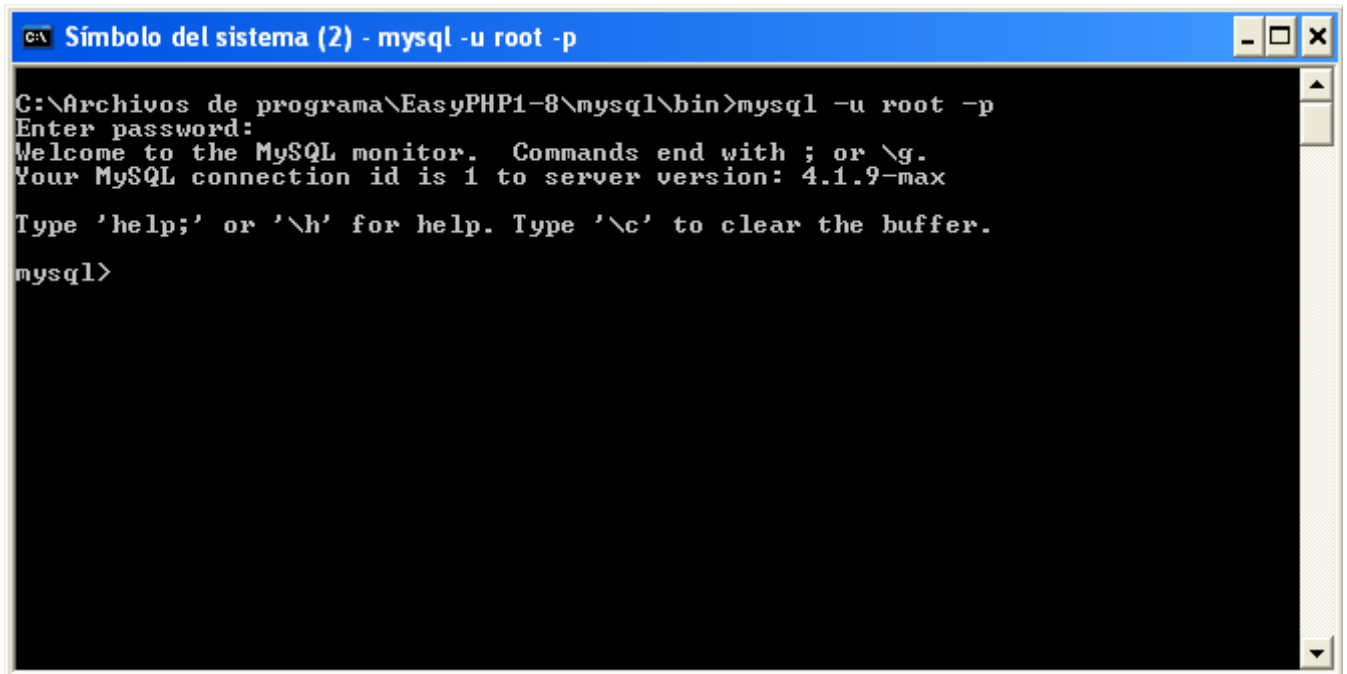


Una vez iniciado el servicio podemos acceder a MySQL de varias formas:

Desde el sistema operativo con comandos de línea:

Para ello deberá abrir una ventana del sistema operativo (tipo MSDOS) y llamar desde el directorio de su instalación al comando **mysql -u root -p**, por ejemplo:

Introducción a MySQL



```
C:\ Archivos de programa\EasyPHP1-8\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 4.1.9-max

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

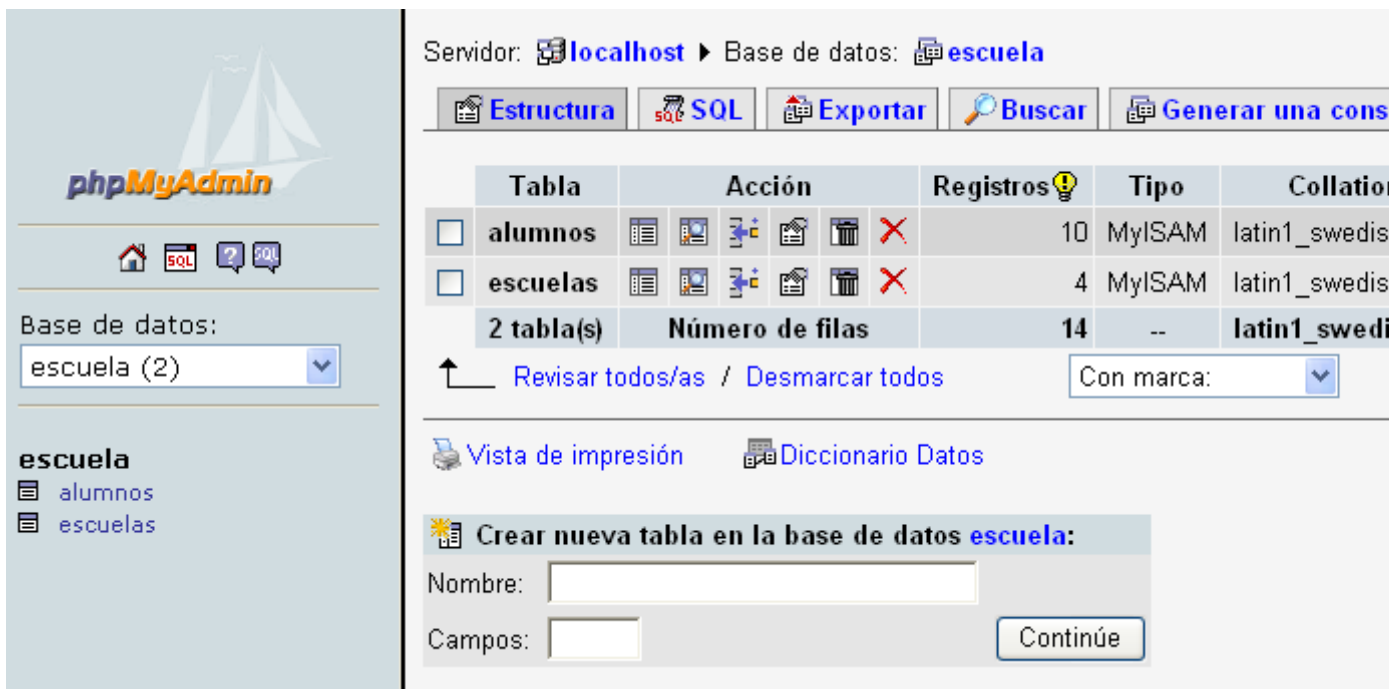
mysql>
```

Cuando se encuentre dentro del sistema, tendrá que teclear las sentencias y finalizarlas con un símbolo de punto y coma (;) para ejecutar la sentencia. Para salir de esta pantalla teclee el comando "quit".

Desde PHPmyAdmin

Todas las instalaciones de la base de datos tienen una interfase desarrollada en PHP que simplifica mucho las operaciones de la base de datos. Su formato general es:

Introducción a MySQL



Desde un sistema

Podemos llamar a la base de datos desde un guión en PHP, CGI o C, y a su vez llamar a estos desde una aplicación en HTML o Flash. Más adelante veremos algunos ejemplos de estos guiones y aplicaciones.

Primero abordaremos el manejo del lenguaje desde el prompt de MySQL, y en capítulos posteriores abordaremos el uso de la herramienta phpMyAdmin y de guines hechos con PHP.

Crear una base de datos

En MySQL se pueden crear varias bases de datos, y dentro de estas varias tablas. Para crear una base de datos se utiliza la sentencia

`CREATE DATABASE nombre`

Por ejemplo, entre a MySQL y cree una base de datos que se llame "ESCUELA".

Introducción a MySQL

Seleccionar una base de datos

Una vez creada una base de datos, es necesario seleccionarla para trabajar con ella, para ello tenemos el comando:

USE nombre

Por ejemplo, seleccione la base de datos que acaba de crear, ESCUELA.

MySQL le respondera por la selección o le marcará un mensaje de error:

```
C:\Archivos de programa\EasyPHP1-8\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8 to server version: 4.1.9-max

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> USE escuela;
Database changed
mysql> _
```

Ahora ya puede manipular los datos de la base de datos.

Mostrar las bases de datos

Para mostrar las bases de datos que tiene sobre su manejador MySQL puede teclear el comando:

SHOW databases;

Ejemplo: En la línea de comando teclee la sentencia anterior.

Crear una tabla

La sentencia para crear una tabla es:

CREATE TABLE nombre (definición de columnas)

Los tipo de las columnas ya se vieron en el capítulo anterior, pero los principales son:

Introducción a MySQL

1. *Integer*: Número entero con rango de -2,147,483,648 y 2,147,483,647
2. *VarChar(n)*: Es una cadena con longitud variable, con un máximo de 255 caracteres. El parámetro *n* marca el máximo de la longitud esperada.
3. *Text*: Campo de texto que puede almacenar hasta 65,535 caracteres.
4. *MediumText*: Campo de texto que permite almacenar hasta 16,777,215 caracteres.
5. *Datetime*: Guarda una fecha en el formato YYYYDDMMHHMMSS
6. *Timestamp*: Tipo de campo que almacena la fecha actual en formato YYYYDDMMHHMMSS

Cada uno de los campos o columnas podrá tener alguno de los siguientes comportamientos:

1. *NOT NULL* | *NULL*: Permite que el campo tenga o no valor vacíos o nulos.
2. *DEFAULT*: Valor que tomará el campo por default.
3. *AUTO_INCREMENT*: Se le asignará al campo un número a partir de l valor más alto de esa columna más uno.
4. *PRIMARY_KEY*: Es la llave principal de la tabla, por lo que no puede haber valores duplicados o nulos. Es una buena práctica tener un campo con llave principal en cada tabla.

Por ejemplo: con la base de datos ESCUELA seleccionada, cree una tabla para almacenar a los alumnos:

```
CREATE TABLE alumnos(  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(30) NOT NULL,  
    apellidos VARCHAR(30) NOT NULL,  
    nacimiento DATE,  
    promedio TINYINT,  
    sexo CHAR(1),  
    idEscuela TINYINT,  
    idSalon TINYINT,  
    idCurso TINYINT
```

Introducción a MySQL

);

Cree las tablas:

```
CREATE TABLE escuelas(  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    descripcion VARCHAR(60);  
);
```

```
CREATE TABLE salon(  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    descripcion VARCHAR(60);  
);
```

```
CREATE TABLE cursos(  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    descripcion VARCHAR(60);  
);
```

Mostrar las tablas en una base de datos

Para que usted pueda ver las tablas existentes, necesita ejecutar el siguiente comando:

SHOW tables;

Ejecute el comando con punto y coma al final.

Eliminación de bases de datos y de tablas

Para borrar una tabla o una base de datos basta con la sentencia

DROP TABLE nombreTabla;

DROP DATABASE base de datos;

Tenga cuidado al ejecutar esta sentencia, ya que una vez ejecutada no habrá manera de recuperar

Introducción a MySQL

la información.

Ejemplo: Borre la tabla "salon", que por error pusimos su nombre en singular, cuando debería ser "salones", y vuelva a crearla con el nombre correcto.

Insertando datos

Para insertar datos a una tabla, necesitamos la siguiente instrucción:

INSERT INTO tabla (columnas) VALUES (valores);

Por ejemplo, para insertar un registro en nuestra tabla de contactos utilizaríamos la siguiente sentencia:

**INSERT INTO alumnos (nombre, apellido, nacimiento, promedio, sexo, idEscuela, idSalon, idCurso)
VALUES ("Paco", "Arce", "1966-02-10", 8, "M", 1, 1,1);**

Observe que los campos autonuméricos no es necesario incluirlos en el insert. MySQL calculará el valor y efectuará la modificación.

Desde la línea de comandos inserte tres alumnos a la tabla respectiva.

También se pueden omitir los nombres de los campos, pero MySQL los insertará en orden en que aparecen los campos en la tabla. Hay que tener cuidado al respecto:

**INSERT INTO alumnos
VALUES (0, "Paco", "Arce", "1966-02-10", 8, "M", 1, 1,1);**

Observe que pusimos un cero en la primera columna que es un campo AUTO_INCREMENT, de lo contrario el sistema enviará un error.

Inserte tres alumnos de esta manera.

Introducción a MySQL

Insertar datos por medio de un archivo SQL

Otra forma de insertar datos es por medio de un archivo de tipo texto que contiene las instrucciones SQL.

Por ejemplo:

En un procesador de palabra que no maneje caracteres especiales teclee las siguientes instrucciones:

```
INSERT INTO escuelas VALUES(0,"Escuela 1");  
INSERT INTO escuelas VALUES(0,"Libre de Derecho");  
INSERT INTO escuelas VALUES(0,"Escuela 2");  
INSERT INTO escuelas VALUES(0,"Abierta de economía");
```

Grabe el archivo como insertEscuelas.sql (la extensión es importante) en la carpeta BIN (donde ejecuta MySQL).

Desde el sistema operativo tecle:

```
mysql -u usuario -p password ESCUELA < insertEscuelas.sql
```

Obviamente deberá de cambiar el usuario y el password al que le corresponde dentro del sistema.

Ahora en la tabla escuelas habrá insertado 4 filas.

Insertar datos desde un archivo externo

Podemos insertar una gran cantidad de información desde un archivo externo por medio de la instrucción LOAD DATA. Su sintaxis general es:

```
LOAD DATA [LOW_PRIORITY] [LOCAL] INFILE "nombre_archivo.txt"  
[IGNORE | REPLACE]  
INTO TABLE nombre_tabla  
[FIELDS  
    [TERMINATED BY 'cadena']  
    [OPTIONALLY] ENCLOSED BY 'caracter']
```

Introducción a MySQL

[ESCAPED BY 'caracter']]
[LINES TERMINATED BY 'cadena']
[IGNORE n LINES]
[(lista_columnas)]

LOW_PRIORITY retrasa la sentencia hasta que ningún cliente esté leyendo de la tabla.

LOCAL hará que el archivo se lea de la computadora del cliente y se envíe al servidor.

IGNORE No carga las columnas que tengan la llave principal duplicada

REPLACE Reemplaza los registros con llaves duplicadas

Si se usa FIELDS hay que especificar TERMINATED BY (describe el carácter o caracteres que delimitan valores dentro de una fila)

ENCLOSED BY especifica un carácter entrecomillado que se quita al final del campo de valores si se usa OPTIONALLY.

ESCAPED BY especifica el carácter de escape de los caracteres especiales.

FIELDS siempre debe preceder a LINES.

LINES TERMINATED BY especifica un carácter o caracteres que significan el final de las líneas.

Si no se especifica ni FIELDS ni LINES, los valores por default serán:

FIELDS

TERMINATED BY '\t'

ENCLOSED BY ''

ESCAPED BY '\\'

LINES TERMINATED BY '\n'

La sentencia IGNORE n LINES indica que se descartan las primeras n líneas.

Supongamos que tenemos un archivo con campos separados por tabuladores y cuyas líneas terminan con una Línea nueva y un retorno de carro.

```
1      ActionScript
2      PHP
3      MySQL
4      Photoshop
5      Dreamweaver
6      Flash
```

Introducción a MySQL

La sentencia deberá ser escrita como sigue:

```
LOAD DATA LOCAL INFILE "archivo.txt" INTO TABLE cursos LINES TERMINATED BY "\r\n";
```

MySQL le indicará cuantas lineas han sido añadidas:

Introducción a MySQL

Consulta de datos

Para consultar los datos utilizaremos la sentencia **SELECT** como se muestra a continuación en su versión simplificada:

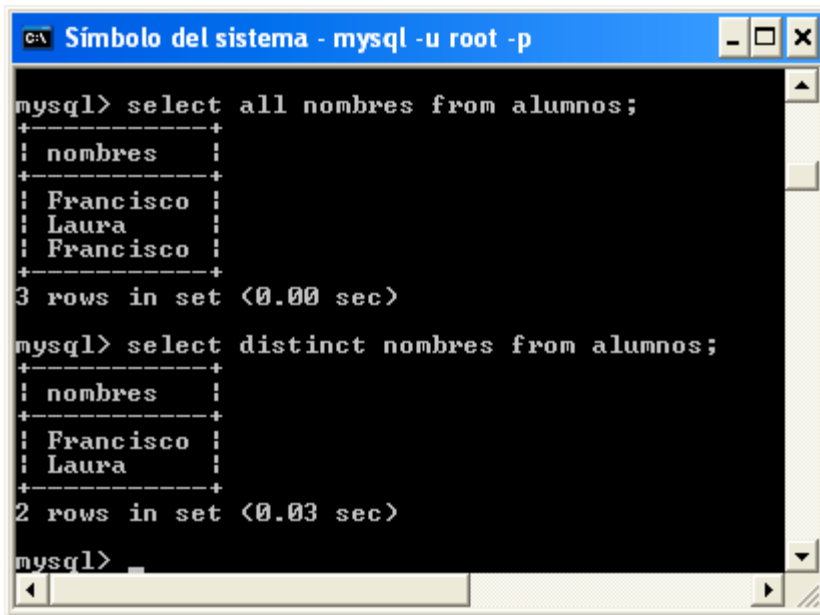
```
SELECT [DISTINCT | ALL | otras_opciones] lista_selección  
[INTO OUTFILE 'nombre_archivo' opciones_exportación]  
FROM tabla_origen [, tabla_origen...]  
[WHERE condición]  
[LIMIT m,n];  
[GROUP BY expresión_de_agrupación]  
[HAVING condición_de_búsqueda]  
[ORDER BY expresión_order_by [ASC | DESC]]
```

En resumen de la sentencia anterior podemos decir que SELECT recupera columnas de la *lista_selección*

de una o más tablas seleccionadas en la sentencia FROM,
que satisfagan las condiciones de la sentencia WHERE,
agrupada según las reglas de GROUP BY
cumpliendo las reglas de búsqueda de HAVING

La cláusula SELECT especifica las columnas que va a devolver la consulta. La opción ALL indica que pueden aparecer registros duplicados (valor por omisión) y DISTINCT indica lo contrario.

Introducción a MySQL



```
C:\> Símbolo del sistema - mysql -u root -p

mysql> select all nombres from alumnos;
+-----+
| nombres |
+-----+
| Francisco |
| Laura    |
| Francisco |
+-----+
3 rows in set (0.00 sec)

mysql> select distinct nombres from alumnos;
+-----+
| nombres |
+-----+
| Francisco |
| Laura    |
+-----+
2 rows in set (0.03 sec)

mysql>
```

Los valores NULL se consideran iguales para la sentencia DISTINCT.

La "lista de selección" puede ser una serie de expresiones separadas por comas donde cada expresión puede ser:

un nombre de columna,

una constante,

una función o

una combinación de todas las anteriores conectadas mediante operadores.

Por ejemplo,

1) Tomemos el caso de nuestra tabla *Alumnos*. Si deseáramos consultar **todas** las columnas y **todos** los registros podríamos utilizar la siguiente sentencia:

```
SELECT * FROM alumnos;
```

Donde el asterisco nos indica que queremos todas las columnas de la tabla.

Introducción a MySQL



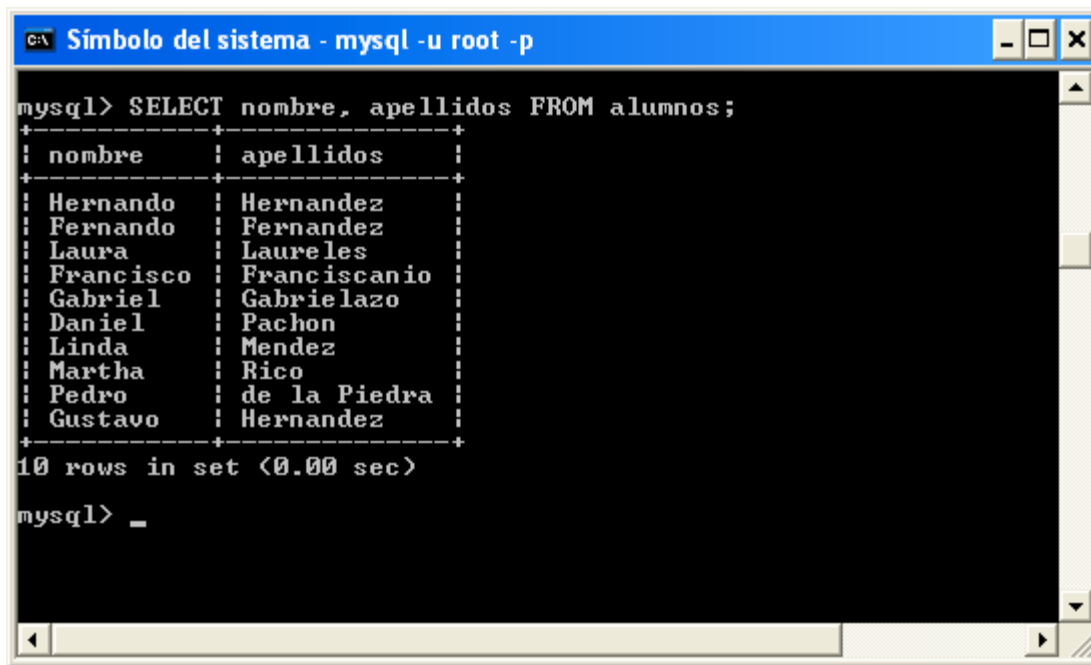
```
C:\> Símbolo del sistema - mysql -u root -p

mysql> SELECT * FROM alumnos;
+----+-----+-----+-----+-----+-----+
| id | Nombre | Apellidos | Nacimiento | Promedio | Sexo |
+----+-----+-----+-----+-----+-----+
| 1  | Hernando | Hernández | 1990-01-01 | 9        | M    |
| 2  | Fernando | Fernández | 1985-12-28 | 8        | M    |
| 3  | Laura    | Laureles  | 1970-10-03 | 10       | F    |
| 4  | Francisco | Franciscanio | 1966-02-10 | 8        | M    |
| 5  | Gabriel  | Gabrielazo | 1985-06-10 | 7        | M    |
| 6  | Daniel   | Pachón    | 1996-04-07 | 5        | M    |
| 7  | Linda    | Méndez    | 1990-01-01 | 8        | F    |
| 8  | Martha   | Rico      | 1986-08-20 | 7        | F    |
| 9  | Pedro    | de la Piedra | 1984-04-04 | 6        | M    |
| 10 | Gustavo  | Hernández | 1980-03-07 | 5        | M    |
+----+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

2) Supongamos que queremos **solo** sustraer el nombre de los alumnos. La sentencia sería:

SELECT nombre, apellidos FROM alumnos;



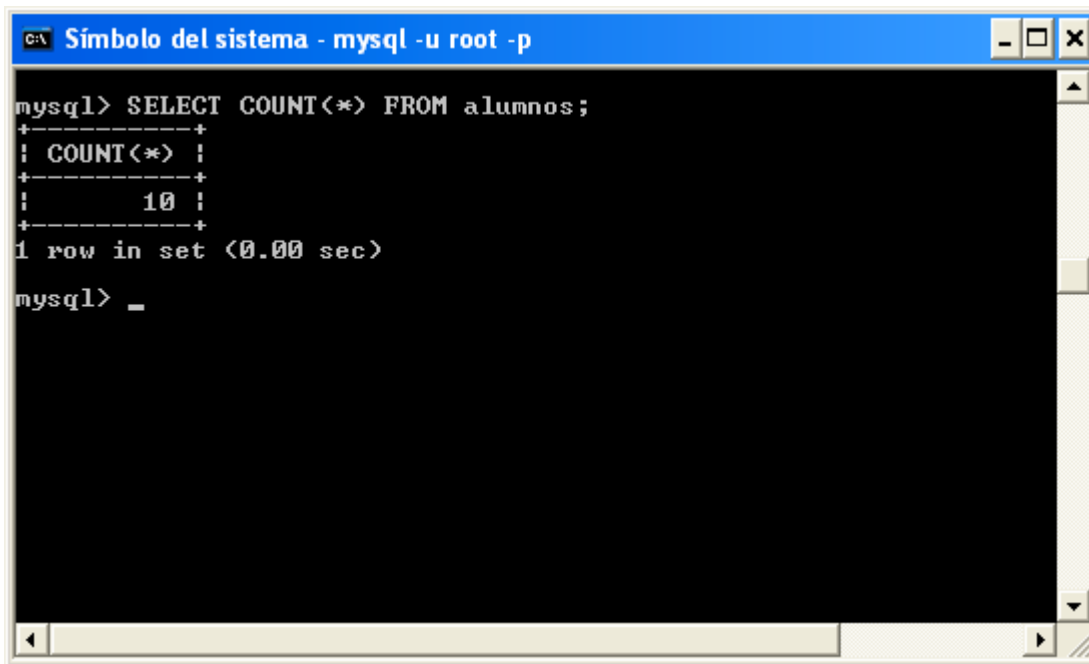
```
C:\> Símbolo del sistema - mysql -u root -p

mysql> SELECT nombre, apellidos FROM alumnos;
+-----+-----+
| nombre | apellidos |
+-----+-----+
| Hernando | Hernandez |
| Fernando | Fernandez |
| Laura    | Laureles  |
| Francisco | Franciscanio |
| Gabriel  | Gabrielazo |
| Daniel   | Pachon    |
| Linda    | Mendez    |
| Martha   | Rico      |
| Pedro    | de la Piedra |
| Gustavo  | Hernandez |
+-----+-----+
10 rows in set (0.00 sec)

mysql> _
```

Introducción a MySQL

3) Podemos incluir en la selección una fórmula, por ejemplo contar al número de alumnos con la función COUNT(*):

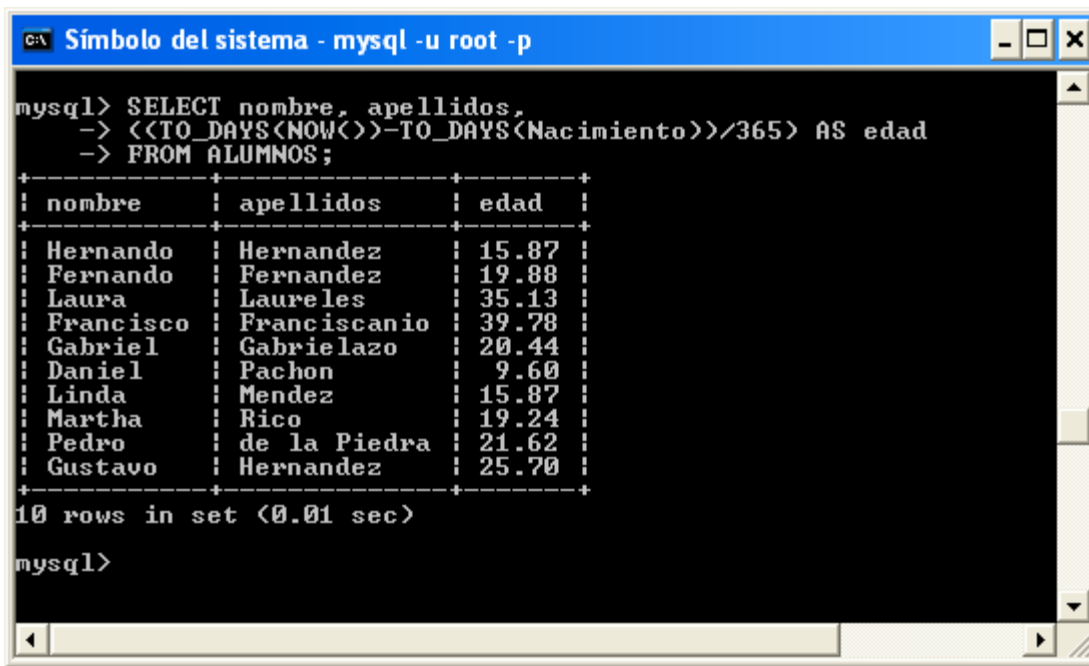


```
Símbolo del sistema - mysql -u root -p

mysql> SELECT COUNT(*) FROM alumnos;
+-----+
| COUNT(*) |
+-----+
|        10 |
+-----+
1 row in set (0.00 sec)

mysql> _
```

4) Podemos utilizar una serie de columnas y operaciones. Por ejemplo, enlistar los nombres de los alumnos y sus edades:



```
Símbolo del sistema - mysql -u root -p

mysql> SELECT nombre, apellidos,
-> <<TO_DAYS(NOW())-TO_DAYS(Nacimiento)>>/365> AS edad
-> FROM ALUMNOS;
+-----+-----+-----+
| nombre | apellidos | edad |
+-----+-----+-----+
| Hernando | Hernandez | 15.87 |
| Fernando | Fernandez | 19.88 |
| Laura | Laureles | 35.13 |
| Francisco | Franciscanio | 39.78 |
| Gabriel | Gabrielazo | 20.44 |
| Daniel | Pachon | 9.60 |
| Linda | Mendez | 15.87 |
| Martha | Rico | 19.24 |
| Pedro | de la Piedra | 21.62 |
| Gustavo | Hernandez | 25.70 |
+-----+-----+-----+
10 rows in set (0.01 sec)

mysql>
```


Introducción a MySQL

FROM

Esta cláusula especifica las tablas de donde se van a obtener las filas. La cláusula FROM es necesaria siempre. Su sintaxis básica es:

FROM lista_tablas

Por o general se tendrán una o mas nombre de tabla separadas por comas. Sin embargo también se pueden tener **cláusulas de unión**, como JOIN, CROSS JOIN, INNER JOIN, STRAIGHT_JOIN y LEFT JOIN (que se verán más adelante).

Las tablas también pueden utilizar la sentencia **AS** para el uso de "Alias". Se verá más adelante un ejemplo cuando analicemos la sentencia WHERE.

WHERE

Especifica una condición de búsqueda para restringir las filas que se van a devolver. Su sintaxis es la siguiente:

[WHERE condición_de_búsqueda | nombre de columna {= | =*} nombre de columna]*

El primer argumento "condición de búsqueda" limita las filas devueltas en el conjunto de resultados mediante el uso de predicados.

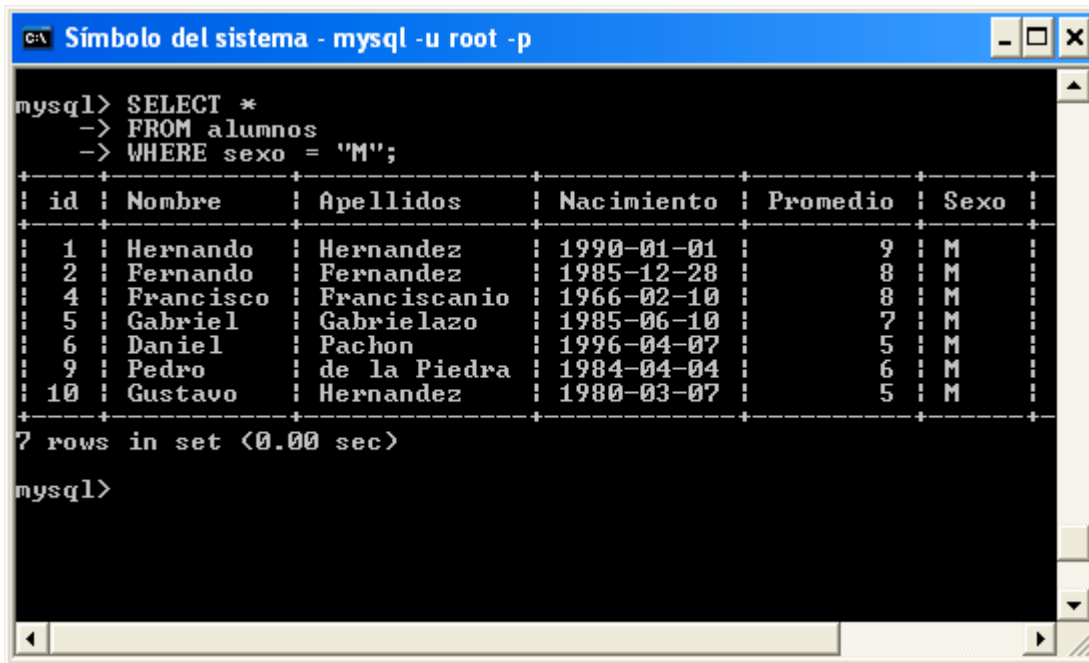
El segundo argumento "*nombre de columna {*= | =*} nombre de columna*" funcionará para las combinaciones, las cuales se verán más adelante.

El conjunto de resultados puede estar limitado por HAVING y LIMIT, como también se verá más adelante.

Introducción a MySQL

Ejemplo:

1) De la tabla de alumnos podemos seleccionar aquellos que sean hombres:



```
Símbolo del sistema - mysql -u root -p

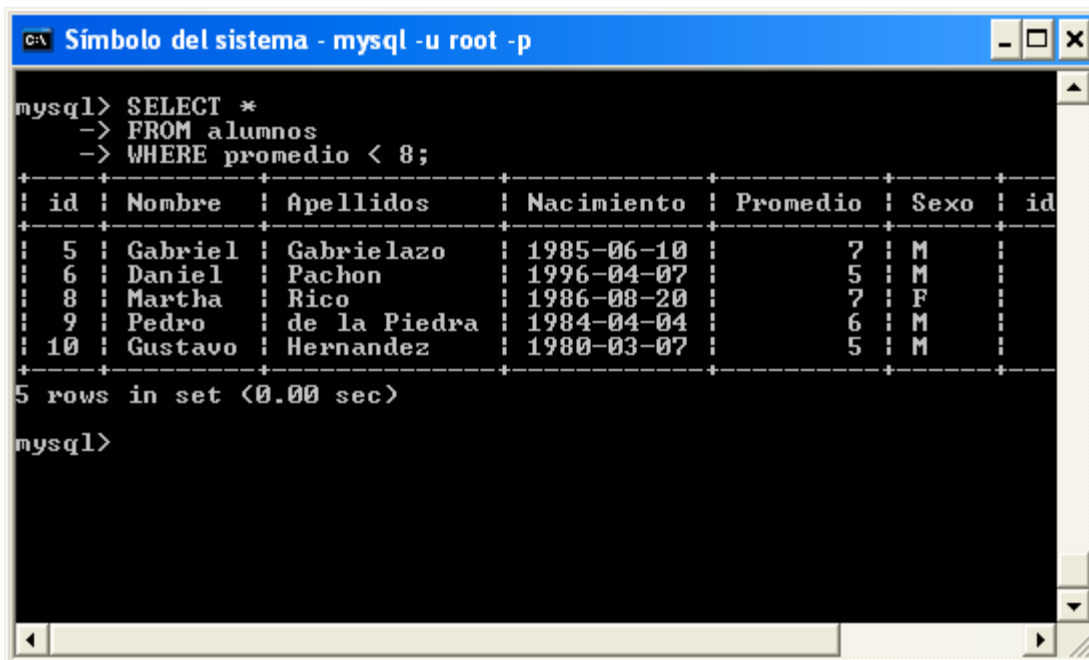
mysql> SELECT *
-> FROM alumnos
-> WHERE sexo = 'M';

+----+-----+-----+-----+-----+-----+
| id | Nombre | Apellidos | Nacimiento | Promedio | Sexo |
+----+-----+-----+-----+-----+-----+
| 1  | Hernando | Hernandez | 1990-01-01 | 9        | M    |
| 2  | Fernando | Fernandez | 1985-12-28 | 8        | M    |
| 4  | Francisco | Franciscanio | 1966-02-10 | 8        | M    |
| 5  | Gabriel | Gabrielazo | 1985-06-10 | 7        | M    |
| 6  | Daniel | Pachon | 1996-04-07 | 5        | M    |
| 9  | Pedro | de la Piedra | 1984-04-04 | 6        | M    |
| 10 | Gustavo | Hernandez | 1980-03-07 | 5        | M    |
+----+-----+-----+-----+-----+-----+

7 rows in set (0.00 sec)

mysql>
```

2) Podemos seleccionar aquellos que tengan promedio **menor** a 8:



```
Símbolo del sistema - mysql -u root -p

mysql> SELECT *
-> FROM alumnos
-> WHERE promedio < 8;

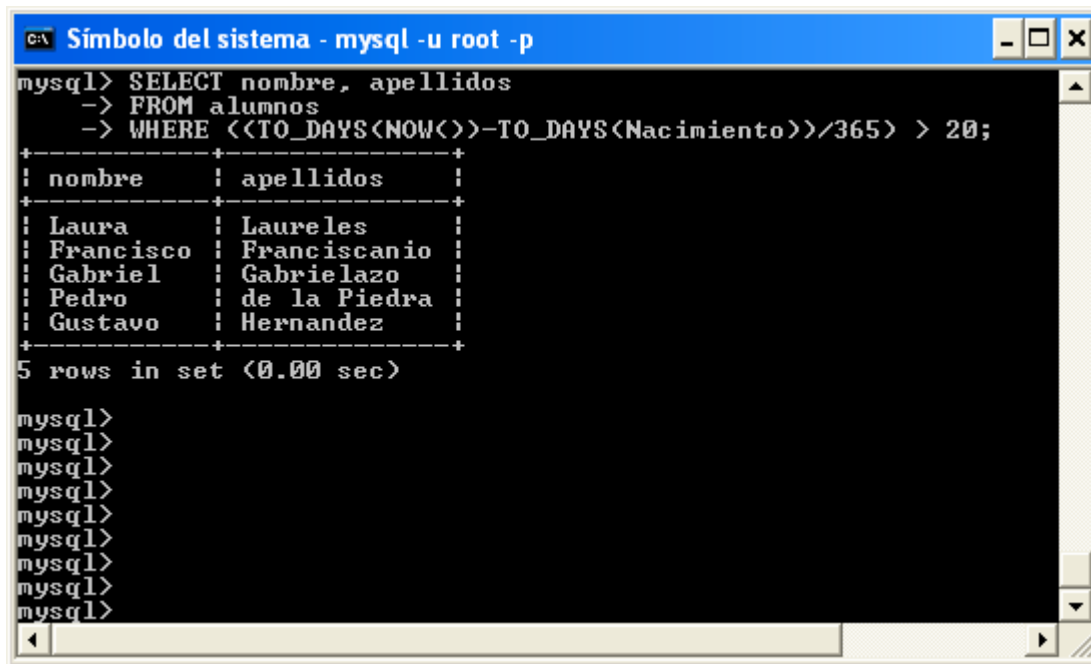
+----+-----+-----+-----+-----+-----+----+
| id | Nombre | Apellidos | Nacimiento | Promedio | Sexo | id |
+----+-----+-----+-----+-----+-----+----+
| 5  | Gabriel | Gabrielazo | 1985-06-10 | 7        | M    | 5  |
| 6  | Daniel | Pachon | 1996-04-07 | 5        | M    | 6  |
| 8  | Martha | Rico | 1986-08-20 | 7        | F    | 8  |
| 9  | Pedro | de la Piedra | 1984-04-04 | 6        | M    | 9  |
| 10 | Gustavo | Hernandez | 1980-03-07 | 5        | M    | 10 |
+----+-----+-----+-----+-----+-----+----+

5 rows in set (0.00 sec)

mysql>
```

Introducción a MySQL

3) Podemos seleccionar por una fórmula, por ejemplo los que sean mayores de 20 años:



```
C:\> Símbolo del sistema - mysql -u root -p
mysql> SELECT nombre, apellidos
-> FROM alumnos
-> WHERE <(TO_DAYS(NOW())-TO_DAYS(Nacimiento))/365> > 20;
+-----+-----+
| nombre | apellidos |
+-----+-----+
| Laura  | Laureles  |
| Francisco | Franciscanio |
| Gabriel | Gabrielazo |
| Pedro  | de la Piedra |
| Gustavo | Hernandez  |
+-----+-----+
5 rows in set (0.00 sec)

mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
```

nombre	apellidos
Laura	Laureles
Francisco	Franciscanio
Gabriel	Gabrielazo
Pedro	de la Piedra
Gustavo	Hernandez

4) Por medio de la sentencia WHERE podemos consultar simultáneamente más de una tabla. Por ejemplo, en la base de datos el campo "idEscuela" de la tabla "alumnos" es la llave para unir la descripción de la escuela en la tabla "escuelas". Para unir las debemos igualar en la sentencia WHERE idEscuela=id

Introducción a MySQL



```
C:\> Símbolo del sistema - mysql -u root -p
mysql> SELECT alumnos.nombre,
-> alumnos.apellidos,
-> escuelas.descripcion
-> FROM alumnos, escuelas
-> WHERE alumnos.idEscuela = escuelas.id;
+-----+-----+-----+
| nombre | apellidos | descripcion |
+-----+-----+-----+
| Hernando | Hernandez | Panamericana |
| Laura | Laureles | Panamericana |
| Linda | Mendez | Panamericana |
| Fernando | Fernandez | Alemana |
| Daniel | Pachon | Alemana |
| Francisco | Franciscanio | Periodistas |
| Martha | Rico | Periodistas |
| Gustavo | Hernandez | Periodistas |
| Gabriel | Gabrielazo | Abierta |
| Pedro | de la Piedra | Abierta |
+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
mysql>
```

Observe que ahora a las columnas les antepone el nombre de la tabla. Cuando traemos los valores de catálogos se le llama "armar las llaves".

ORDER BY

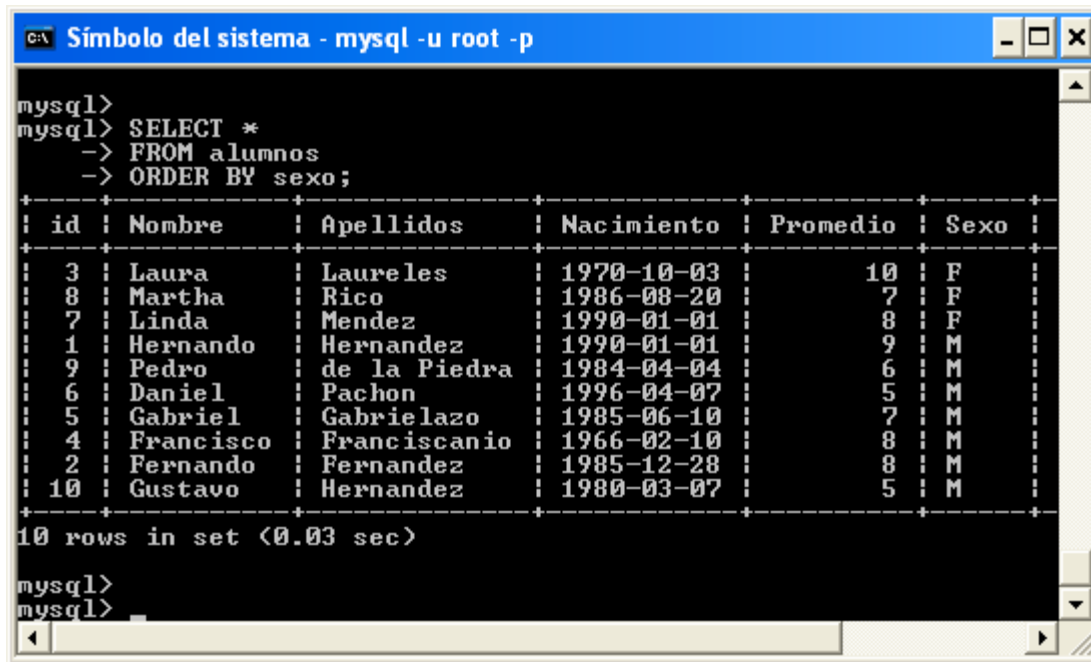
Muchas veces necesitamos ordenar la información bajo un criterio. La sentencia ORDER BY nos servirá para eso. Su sintaxis es:

[ORDER BY expresión_de_orden [AS | DESC]]

Donde la "expresión_de_orden" son las columnas con las que ordenaremos, ya sea en forma ascendente (ASC) o en forma descendente (DESC).

1) Ordenar por una columna. Ordenemos a la lista de alumno por su sexo:

Introducción a MySQL



The screenshot shows a Windows command prompt window titled "Símbolo del sistema - mysql -u root -p". The prompt is "mysql>". The user enters the SQL query: "SELECT * FROM alumnos ORDER BY sexo;". The results are displayed in a table format with columns: id, Nombre, Apellidos, Nacimiento, Promedio, and Sexo. There are 10 rows of data. Below the table, it says "10 rows in set (0.03 sec)". The prompt returns to "mysql>".

```
mysql>
mysql> SELECT *
  -> FROM alumnos
  -> ORDER BY sexo;
+----+-----+-----+-----+-----+-----+
| id | Nombre | Apellidos | Nacimiento | Promedio | Sexo |
+----+-----+-----+-----+-----+-----+
| 3  | Laura  | Laureles  | 1970-10-03 | 10       | F    |
| 8  | Martha | Rico      | 1986-08-20 | 7        | F    |
| 7  | Linda  | Mendez    | 1990-01-01 | 8        | F    |
| 1  | Hernando | Hernandez | 1990-01-01 | 9        | M    |
| 9  | Pedro  | de la Piedra | 1984-04-04 | 6        | M    |
| 6  | Daniel | Pachon    | 1996-04-07 | 5        | M    |
| 5  | Gabriel | Gabrielazo | 1985-06-10 | 7        | M    |
| 4  | Francisco | Franciscanio | 1966-02-10 | 8        | M    |
| 2  | Fernando | Fernandez | 1985-12-28 | 8        | M    |
| 10 | Gustavo | Hernandez  | 1980-03-07 | 5        | M    |
+----+-----+-----+-----+-----+-----+
10 rows in set (0.03 sec)

mysql>
mysql>
```

Introducción a MySQL

2) Ordenar por en forma descendente. Hagamos el mismo ordenamiento, pero en forma descente:



```
C:\> Símbolo del sistema - mysql -u root -p

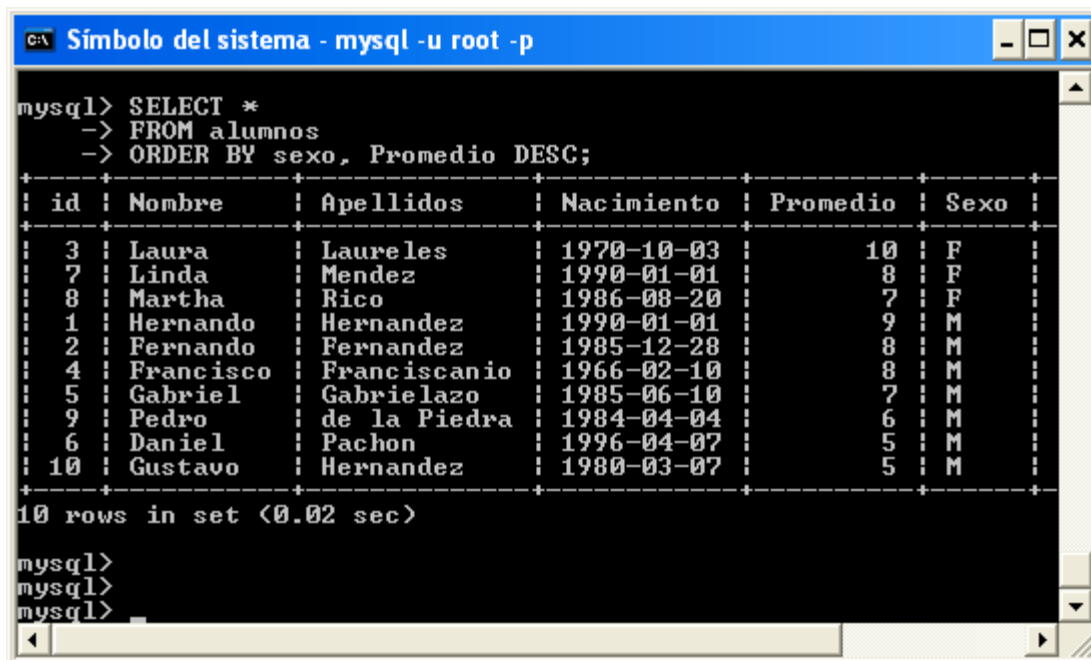
+-----+
10 rows in set (0.00 sec)

mysql> SELECT *
-> FROM alumnos
-> ORDER BY sexo DESC;
+----+-----+-----+-----+-----+-----+
| id | Nombre | Apellidos | Nacimiento | Promedio | Sexo |
+----+-----+-----+-----+-----+-----+
| 1  | Hernando | Hernandez | 1990-01-01 | 9        | M    |
| 9  | Pedro   | de la Piedra | 1984-04-04 | 6        | M    |
| 6  | Daniel  | Pachon      | 1996-04-07 | 5        | M    |
| 5  | Gabriel | Gabrielazo  | 1985-06-10 | 7        | M    |
| 4  | Francisco | Franciscanio | 1966-02-10 | 8        | M    |
| 2  | Fernando | Fernandez   | 1985-12-28 | 8        | M    |
| 10 | Gustavo | Hernandez   | 1980-03-07 | 5        | M    |
| 3  | Laura   | Laureles    | 1970-10-03 | 10       | F    |
| 7  | Linda   | Mendez      | 1990-01-01 | 8        | F    |
| 8  | Martha  | Rico        | 1986-08-20 | 7        | F    |
+----+-----+-----+-----+-----+-----+

10 rows in set (0.00 sec)

mysql>
```

3) Podemos ordenar por varias columnas: Por ejemplo, queremos la lista ordenada por sexo en forma ascendente y por calificación en forma descentente:



```
C:\> Símbolo del sistema - mysql -u root -p

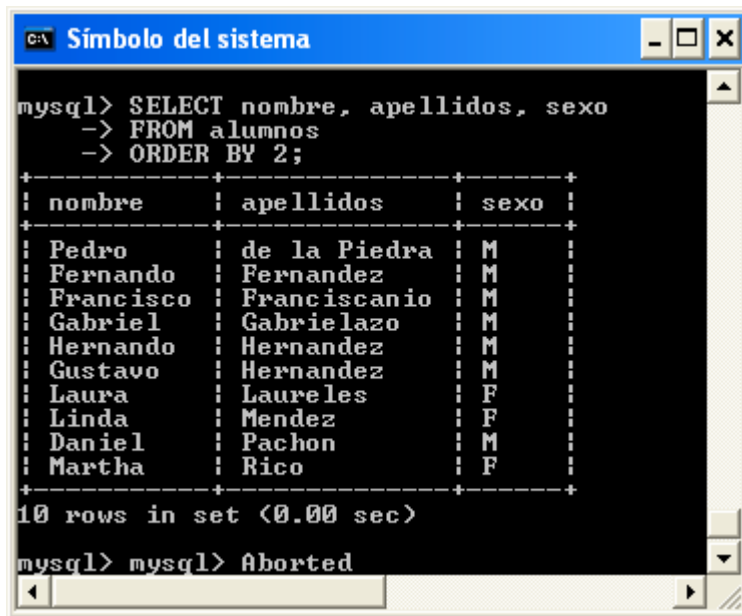
mysql> SELECT *
-> FROM alumnos
-> ORDER BY sexo, Promedio DESC;
+----+-----+-----+-----+-----+-----+
| id | Nombre | Apellidos | Nacimiento | Promedio | Sexo |
+----+-----+-----+-----+-----+-----+
| 3  | Laura   | Laureles    | 1970-10-03 | 10       | F    |
| 7  | Linda   | Mendez      | 1990-01-01 | 8        | F    |
| 8  | Martha  | Rico        | 1986-08-20 | 7        | F    |
| 1  | Hernando | Hernandez   | 1990-01-01 | 9        | M    |
| 2  | Fernando | Fernandez   | 1985-12-28 | 8        | M    |
| 4  | Francisco | Franciscanio | 1966-02-10 | 8        | M    |
| 5  | Gabriel | Gabrielazo  | 1985-06-10 | 7        | M    |
| 9  | Pedro   | de la Piedra | 1984-04-04 | 6        | M    |
| 6  | Daniel  | Pachon      | 1996-04-07 | 5        | M    |
| 10 | Gustavo | Hernandez   | 1980-03-07 | 5        | M    |
+----+-----+-----+-----+-----+-----+

10 rows in set (0.02 sec)

mysql>
mysql>
mysql>
```

Introducción a MySQL

4) En lugar de la columna podemos utilizar el número ordinal de la misma en la lista SELECT. En el siguiente ejemplo ordenamos por apellidos, que es la segunda columna en el SELECT:



```
mysql> SELECT nombre, apellidos, sexo
-> FROM alumnos
-> ORDER BY 2;
+-----+-----+-----+
| nombre | apellidos | sexo |
+-----+-----+-----+
| Pedro  | de la Piedra | M    |
| Fernando | Fernandez  | M    |
| Francisco | Franciscanio | M    |
| Gabriel | Gabrielazo  | M    |
| Hernando | Hernandez   | M    |
| Gustavo | Hernandez   | M    |
| Laura  | Laureles    | F    |
| Linda  | Mendez      | F    |
| Daniel | Pachon      | M    |
| Martha | Rico        | F    |
+-----+-----+-----+
10 rows in set (0.00 sec)

mysql> mysql> Aborted
```

Los renglones con valores NULL se consideran los más bajos en el ordenamiento. No hay límite de columnas dentro de la sentencia **ORDER BY**.

GROUP BY

Especifica las agrupaciones que se van a realizar en las filas de salida y, en caso de incluir funciones de agregado como COUNT o MAX en la cláusula SELECT *lista_selección* calcula el valor de resumen de cada grupo.

La expresión GROUP BY debe de coincidir exactamente con la expresión de la lista de selección. Su sintaxis es:

[GROUP BY expresión_de_agrupamiento]

La expresión de agrupamiento especifica la expresión a la que se realiza el agrupamiento o columna de agrupamiento y puede ser una columna o una expresión diferente a una de agregado

Introducción a MySQL

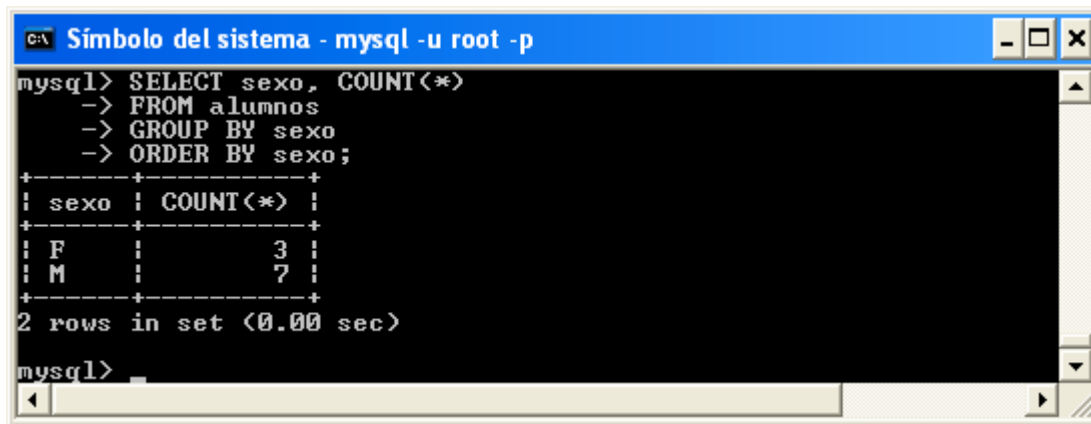
que hace referencia a una columna.

No se puede utilizar un alias de columna en la lista de selección para especificar una columna de agrupamiento.

Si no se especifica la cláusula ORDER BY, los grupos devueltos con la cláusula GROUP BY no estarán en un orden particular.

Se recomienda que siempre utilice la cláusula ORDER BY con GROUP BY para especificar un orden determinado de datos.

1) Supongamos que queremos contar a las personas por sexo. Necesitamos ordenarlas y agruparlas por esa columna:



```
C:\> Símbolo del sistema - mysql -u root -p
mysql> SELECT sexo, COUNT(*)
-> FROM alumnos
-> GROUP BY sexo
-> ORDER BY sexo;
+-----+-----+
| sexo | COUNT(*) |
+-----+-----+
| F    | 3        |
| M    | 7        |
+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

HAVING

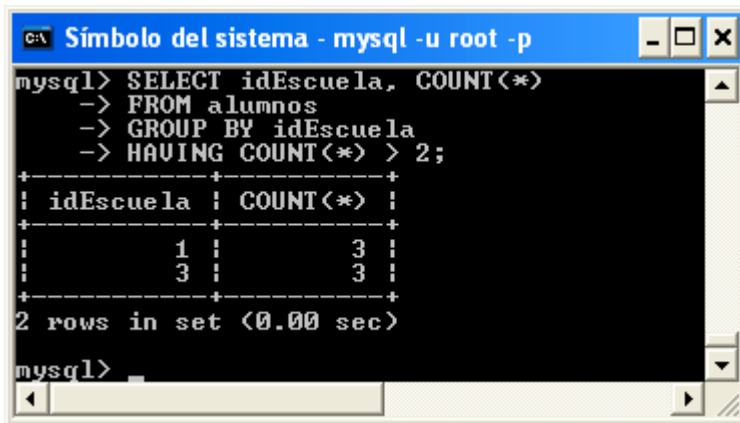
HAVING es similar a WHERE, pero se aplica después de que los resultados de la consulta hayan sido seleccionados y usados para reducir los resultados realmente enviados por el servidor al cliente.

Por lo general se utiliza después de la sentencia GROUP BY y antes que ORDER BY.

HAVING y WHERE no son excluyentes, simplemente HAVING funciona después de la selección de WHERE. Es mejor utilizar los criterios de selección en WHERE, ya que el optimizador trabajará sobre esta cláusula.

Introducción a MySQL

1) Por ejemplo, queremos saber las escuelas que tengan más de dos alumnos dentro de nuestra tabla. Para ello podemos utilizar la sentencia HAVING:



```
C:\> Símbolo del sistema - mysql -u root -p
mysql> SELECT idEscuela, COUNT(*)
-> FROM alumnos
-> GROUP BY idEscuela
-> HAVING COUNT(*) > 2;
+-----+-----+
| idEscuela | COUNT(*) |
+-----+-----+
| 1         | 3        |
| 3         | 3        |
+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

LIMIT

La cláusula LIMIT nos permite seleccionar un conjunto de los renglones regresados por el select. Esto es de mucha utilidad cuando estamos mostrando registros por Internet, ya que nos permite "paginar" la selección y no saturar ni al cliente ni al servidor. Su sintaxis es:

[LIMIT n]

[LIMIT m,n]

Donde m es a partir del registro que accederemos y n es el número de registros a extraer.

1) Seleccionar solo tres registros de toda la tabla de alumnos:



```
C:\> Símbolo del sistema - mysql -u root -p
mysql> SELECT *
-> FROM alumnos
-> LIMIT 3;
+----+-----+-----+-----+
| id | Nombre | Apellidos | Nacimiento |
+----+-----+-----+-----+
| 1  | Hernando | Hernandez | 1990-01-01 |
| 2  | Fernando | Fernandez | 1985-12-28 |
| 3  | Laura   | Laureles  | 1970-10-03 |
+----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

Introducción a MySQL

2) Seleccionar tres registros a partir del cuarto:

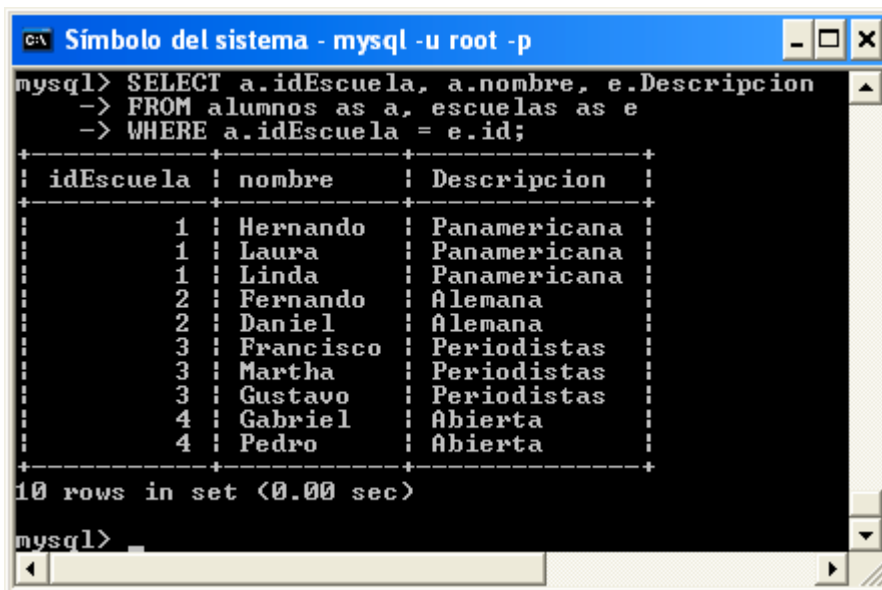


```
C:\> Símbolo del sistema - mysql -u root -p
mysql> SELECT *
-> FROM alumnos
-> LIMIT 4, 3;
+----+-----+-----+-----+
| id | Nombre | Apellidos | Nacimiento |
+----+-----+-----+-----+
| 5 | Gabriel | Gabrielazo | 1985-06-10 |
| 6 | Daniel | Pachon | 1996-04-07 |
| 7 | Linda | Mendez | 1990-01-01 |
+----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

ALIAS

Recuerde que podemos utilizar los alias tanto para renombrar tablas en el FROM como para renombrar el nombre de las columnas. Esto es de gran utilidad porque algunas veces el nombre de las tablas son muy largos y hacen a la sentencia muy compleja. Para generar un "alias" en la sentencia FROM necesitamos la palabra AS como se muestra en el siguiente ejemplo:



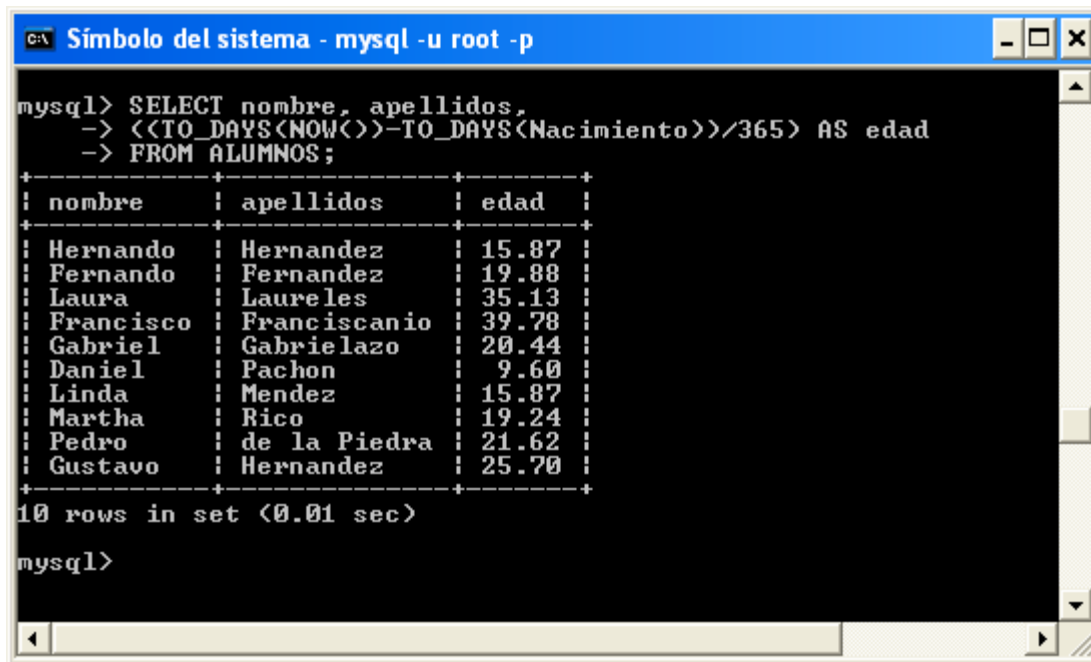
```
C:\> Símbolo del sistema - mysql -u root -p
mysql> SELECT a.idEscuela, a.nombre, e.Descripcion
-> FROM alumnos as a, escuelas as e
-> WHERE a.idEscuela = e.id;
+----+-----+-----+
| idEscuela | nombre | Descripcion |
+----+-----+-----+
| 1 | Hernando | Panamericana |
| 1 | Laura | Panamericana |
| 1 | Linda | Panamericana |
| 2 | Fernando | Alemana |
| 2 | Daniel | Alemana |
| 3 | Francisco | Periodistas |
| 3 | Martha | Periodistas |
| 3 | Gustavo | Periodistas |
| 4 | Gabriel | Abierta |
| 4 | Pedro | Abierta |
+----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

Algunas veces las columnas calculadas son muy largas o simplemente la fórmula no nos sirve como encabezado en nuestros resultados. Para ello podemos utilizar un alias en la sentencia

Introducción a MySQL

SELECT usando la sentencia AS:



```
C:\> Símbolo del sistema - mysql -u root -p

mysql> SELECT nombre, apellidos,
-> <<(TO_DAYS(NOW())-TO_DAYS(Nacimiento))>>/365> AS edad
-> FROM ALUMNOS;

+-----+-----+-----+
| nombre | apellidos | edad |
+-----+-----+-----+
| Hernando | Hernandez | 15.87 |
| Fernando | Fernandez | 19.88 |
| Laura | Laureles | 35.13 |
| Francisco | Franciscanio | 39.78 |
| Gabriel | Gabrielazo | 20.44 |
| Daniel | Pachon | 9.60 |
| Linda | Mendez | 15.87 |
| Martha | Rico | 19.24 |
| Pedro | de la Piedra | 21.62 |
| Gustavo | Hernandez | 25.70 |
+-----+-----+-----+
10 rows in set (0.01 sec)

mysql>
```

USO DE OPERADORES DE COMPARACIÓN ESPECIALES

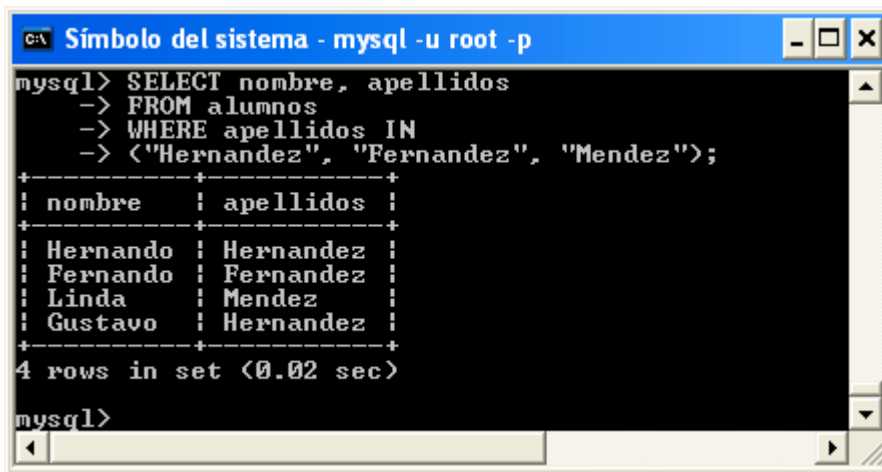
Existen ciertos operador que por lo general no existen en otros lenguajes de programación. Veremos como funcionan con la sentencia SELECT. Estos son:

Introducción a MySQL

IN

IN efectúa una selección entre un conjunto de posibles resultados. También se tiene su contraparte NOT IN.

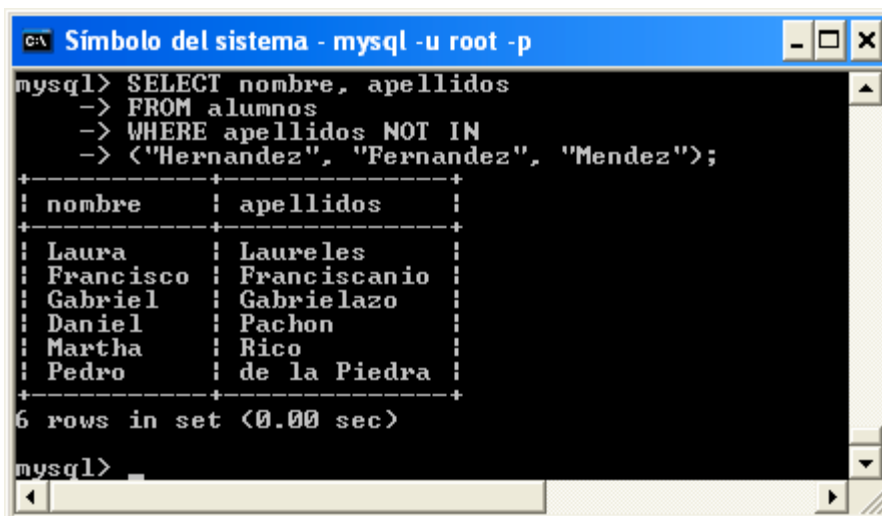
1) Seleccionamos a los alumnos que se apelliden como Hernandez, Fernandez o Mendez:



```
C:\> Símbolo del sistema - mysql -u root -p
mysql> SELECT nombre, apellidos
      -> FROM alumnos
      -> WHERE apellidos IN
      -> ("Hernandez", "Fernandez", "Mendez");
+-----+-----+
| nombre | apellidos |
+-----+-----+
| Hernando | Hernandez |
| Fernando | Fernandez |
| Linda    | Mendez    |
| Gustavo  | Hernandez |
+-----+-----+
4 rows in set (0.02 sec)

mysql>
```

2) de igual modo podemos seleccionar renglones excluyendo los de un grupo con el NOT IN:



```
C:\> Símbolo del sistema - mysql -u root -p
mysql> SELECT nombre, apellidos
      -> FROM alumnos
      -> WHERE apellidos NOT IN
      -> ("Hernandez", "Fernandez", "Mendez");
+-----+-----+
| nombre | apellidos |
+-----+-----+
| Laura  | Laureles  |
| Francisco | Franciscanio |
| Gabriel | Gabrielazo |
| Daniel | Pachon    |
| Martha | Rico      |
| Pedro  | de la Piedra |
+-----+-----+
6 rows in set (0.00 sec)

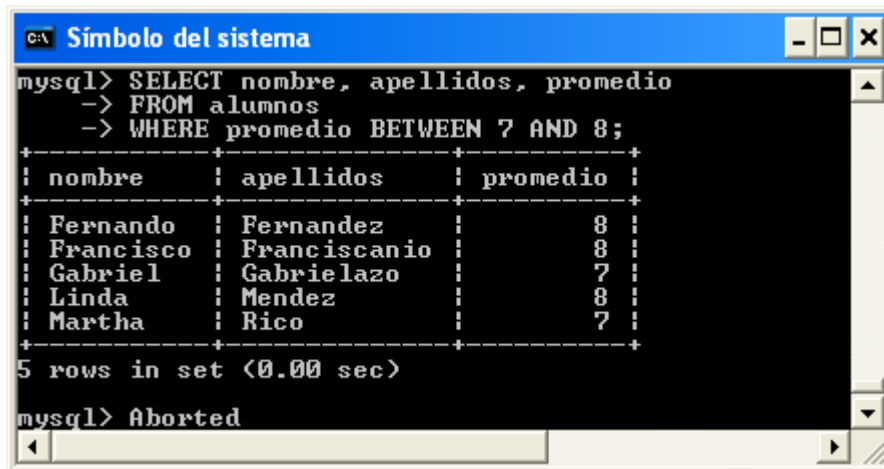
mysql>
```

BETWEEN x AND y

Introducción a MySQL

Con este operador de comparación podremos seleccionar registros en un rango.

1) Por ejemplo, queremos a los alumnos que tengan promedio entre 7y 8:



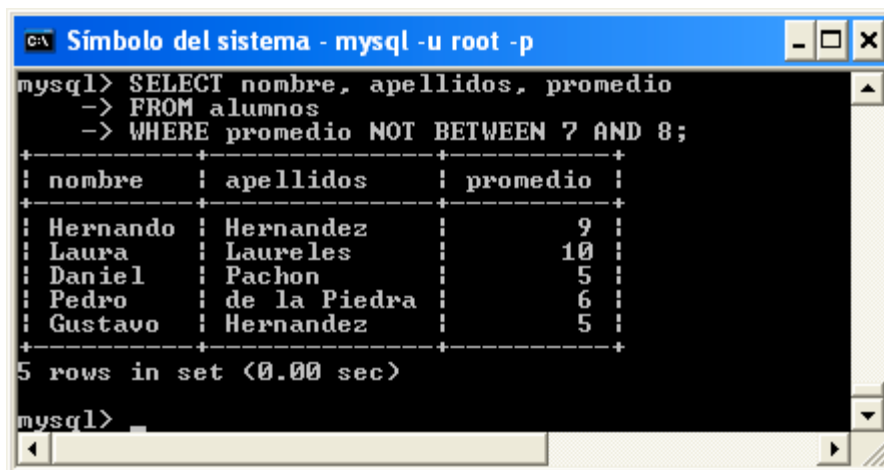
```
mysql> SELECT nombre, apellidos, promedio
-> FROM alumnos
-> WHERE promedio BETWEEN 7 AND 8;
```

nombre	apellidos	promedio
Fernando	Fernandez	8
Francisco	Franciscanio	8
Gabriel	Gabrielazo	7
Linda	Mendez	8
Martha	Rico	7

```
5 rows in set (0.00 sec)

mysql> Aborted
```

2) También se puede utilizar la negación, es decir NOT BETWEEN:



```
mysql> SELECT nombre, apellidos, promedio
-> FROM alumnos
-> WHERE promedio NOT BETWEEN 7 AND 8;
```

nombre	apellidos	promedio
Hernando	Hernandez	9
Laura	Laureles	10
Daniel	Pachon	5
Pedro	de la Piedra	6
Gustavo	Hernandez	5

```
5 rows in set (0.00 sec)

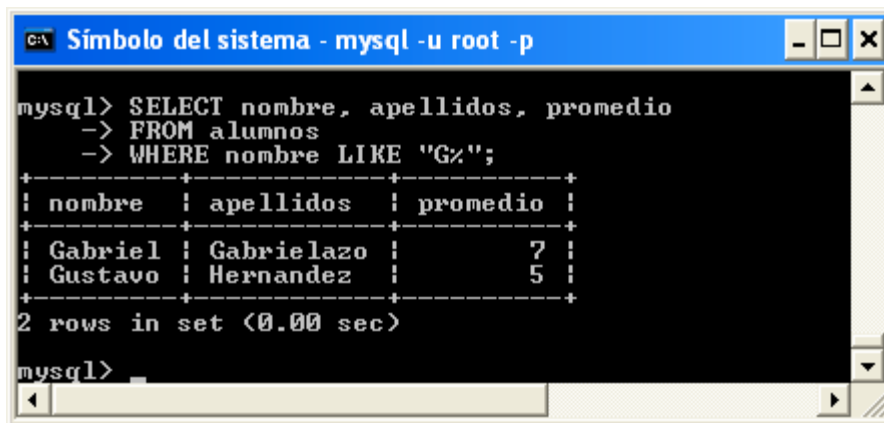
mysql>
```

LIKE

Por medio de esta comparación podemos buscar patrones dentro de las cadenas, ya que se podemos utilizar comodines, por ejemplo:

1) Mostrar a las persona que su nombre inicie con "G":

Introducción a MySQL



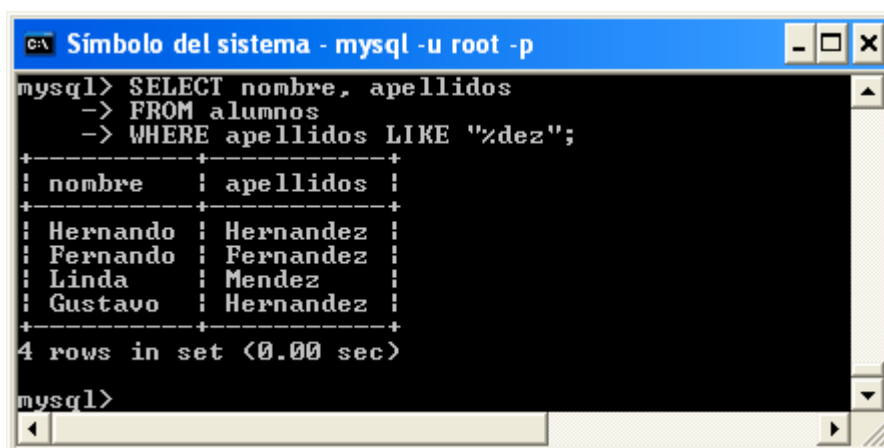
```
C:\> Símbolo del sistema - mysql -u root -p

mysql> SELECT nombre, apellidos, promedio
-> FROM alumnos
-> WHERE nombre LIKE "G%";

+-----+-----+-----+
| nombre | apellidos | promedio |
+-----+-----+-----+
| Gabriel | Gabrielazo | 7 |
| Gustavo | Hernandez | 5 |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

2) Buscar a los apellidos que terminen en "dez":



```
C:\> Símbolo del sistema - mysql -u root -p

mysql> SELECT nombre, apellidos
-> FROM alumnos
-> WHERE apellidos LIKE "%dez";

+-----+-----+
| nombre | apellidos |
+-----+-----+
| Hernando | Hernandez |
| Fernando | Fernandez |
| Linda | Mendez |
| Gustavo | Hernandez |
+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

IS NULL

También se puede seleccionar los campos que se encuentren con el valor NULL, o seleccionar exclusivamente aquellos que sean diferentes a NULL.

FUNCIONES DE AGREGADO

Las funciones de agregado realizan un cálculo sobre un conjunto de valores y devuelven un solo valor. Con la excepción de COUNT,, las funciones de agregado omiten los valores NULL. Las funciones de agregado se suelen utilizar con la cláusula GROUP BY de la instrucción SELECT.

Las funciones de agregado sólo se aceptan como expresiones en la lista de selección de una

Introducción a MySQL

instrucción SELECT. Las funciones de agregado son:

AVG(expresión): Devuelve la media de los valores de un grupo definido por la expresión. Por ejemplo, podemos obtener el promedio de las calificaciones de los alumnos:

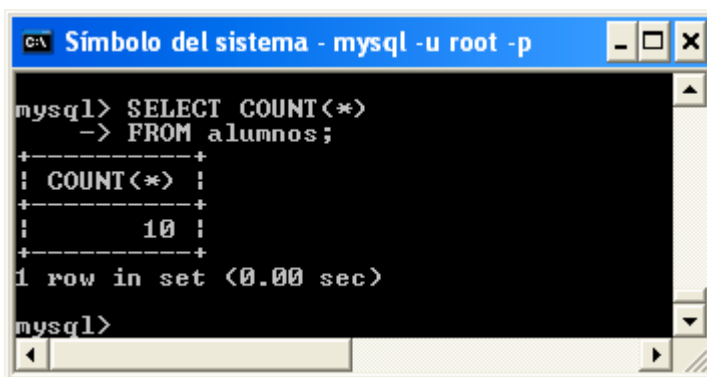


```
Símbolo del sistema - mysql -u root -p

mysql> SELECT AVG(promedio)
-> FROM alumnos
-> ;
+-----+
| AVG(promedio) |
+-----+
|          7.3000 |
+-----+
1 row in set (0.00 sec)

mysql>
```

COUNT(expresión): Número de elementos que cumplen la expresión, por ejemplo, contar los alumnos de la tabla respectiva:



```
Símbolo del sistema - mysql -u root -p

mysql> SELECT COUNT(*)
-> FROM alumnos;
+-----+
| COUNT(*) |
+-----+
|         10 |
+-----+
1 row in set (0.00 sec)

mysql>
```

MAX(expresión): Extrae el número máximo dependiendo la expresión.



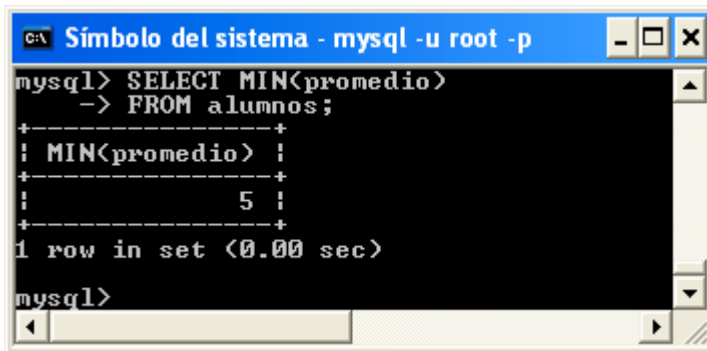
```
Símbolo del sistema - mysql -u root -p

mysql> SELECT MAX(promedio)
-> FROM alumnos;
+-----+
| MAX(promedio) |
+-----+
|         10 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Introducción a MySQL

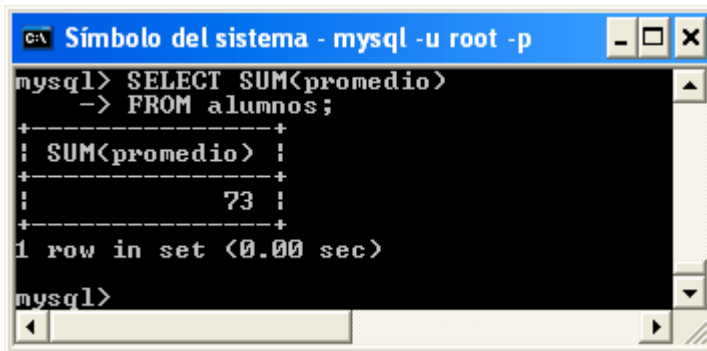
MIN(expresión): Extrae el número mínimo de la expresión.



```
Símbolo del sistema - mysql -u root -p
mysql> SELECT MIN(promedio)
-> FROM alumnos;
+-----+
| MIN(promedio) |
+-----+
|             5 |
+-----+
1 row in set (0.00 sec)

mysql>
```

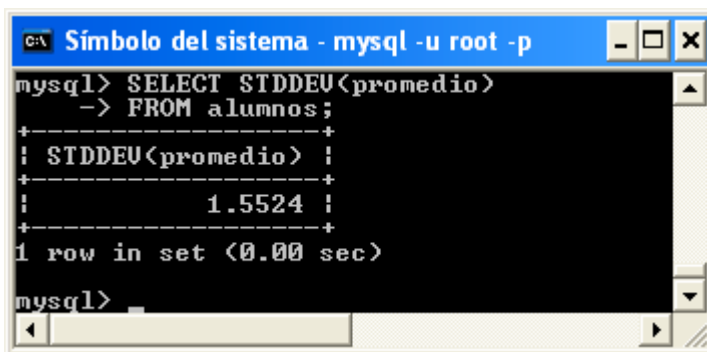
SUM(expresión): Suma la columna según el criterio.



```
Símbolo del sistema - mysql -u root -p
mysql> SELECT SUM(promedio)
-> FROM alumnos;
+-----+
| SUM(promedio) |
+-----+
|            73 |
+-----+
1 row in set (0.00 sec)

mysql>
```

STDDEV(expresión): Desviación estándar de la evaluación de la expresión.



```
Símbolo del sistema - mysql -u root -p
mysql> SELECT STDDEV(promedio)
-> FROM alumnos;
+-----+
| STDDEV(promedio) |
+-----+
|          1.5524 |
+-----+
1 row in set (0.00 sec)

mysql>
```


Introducción a MySQL

Actualización de datos

En muchas ocasiones es necesario modificar los datos que ya se encuentran en nuestras tablas.

Para ello contamos con la siguiente sentencia:

```
UPDATE [LOW_PRIORITY] tabla  
SET columna=valor [,columna2=valor2...]  
[WHERE condición]  
[LIMIT n]
```

Donde SET contiene una lista separada por comas de las columnas que deben actualizarse y el nuevo valor de cada columna, con el formato columna=valor. El valor suministrado por las expresiones incluye elementos tales como constantes, valores seleccionados de una columna de otra tabla o vista, o valores calculados por una expresión compleja.

WHERE: Especifica la condición de búsqueda que define las filas de las tablas de origen que están calificadas para proporcionar valores para las expresiones de la cláusula SET.

LOW_PRIORITY indica que no se ejecuta la actualización hasta que ningún cliente esté leyendo la tabla.

LIMIT n establece en n el número máximo de filas que se pueden actualizar.

Si quisiéramos modificar los datos añadidos en el punto anterior, la sentencia sería:

```
UPDATE contactos SET email="paco@mail.com" WHERE nombre="Paco";
```

La cláusula WHERE en la sentencia UPDATE es opcional, y si se omite, entonces se actualizan TODAS las filas de la tabla destino. Mucho cuidado: Debido a que puede cambiar en forma definitiva la información de sus tablas, siempre utilice el UPDATE con la sentencia WHERE (a menos que desee cambiar todas las columnas por un mismo valor).

Ejercicio:

Use la base de datos ESCUELA y modifique el nombre y el salón de tres alumnos.

Introducción a MySQL

Borrando datos

Para borrar un registro, o grupos de registros, se utilizará la sentencia DELETE:

```
DELETE [LOW_PRIORITY] FROM tabla  
[WHERE condicion]  
[LIMIT n]
```

En esta instrucción también es necesario utilizar la sentencia WHERE, a menos que desee borrar todos los datos de la tabla, por ejemplo:

```
DELETE FROM contactos;
```

Para borrar solo un registro, puede utilizar la siguiente sentencia:

```
DELETE FROM contactos WHERE nombre = "Paco";
```

Introducción a MySQL

Modificación de tablas con ALTER TABLE

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

- Agregar, modificar o eliminar columnas. Así se puede cambiar el nombre, longitud, tipo de datos, la precisión, la escala y la aceptación de valores NULL de la columna, aunque existen algunas restricciones.
- Agregar o eliminar restricciones PRIMARY KEY y FOREIGN KEY.
- Agregar o eliminar restricciones UNIQUE y CHECK, así como definiciones (y objetos) DEFAULT.
- Registrar una tabla y las columnas seleccionadas de una tabla para la indexación de texto.

La sintaxis general es:

ALTER TABLE tabla

```
{      [CHANGE nombre_columna nombre_columna_nuevo
      { nuevo_tipo_datos [ (precisión [, escala] ) ]
        [NULL | NOT NULL] [SET DEFAULT valor | DROP DEFAULT]]
      | ADD [COLUMN] declaración_columna [FIRST | AFTER nombre_columna]
      | ADD INDEX nombre_índice (columnas_índice)
      | ADD PRIMARY KEY (columnas_índice)
      | ADD UNIQUE nombre_índice (columnas_índice)
      | CHANGE [COLUMN] nombre_columna declaración_columna
      | DROP [COLUMN] nombre_columna
      | DROP INDEX nombre_índice (lista_columnas)
      | DROP PRIMARY KEY
      | MODIFY [COLUMN] declaración_columna
      | RENAME [AS] nombre_tabla_nueva
}
```

Donde "tabla" indica la tabla que se va a modificar.

Introducción a MySQL

Cambiar una columna

CHANGE especifica que la columna dada va a cambiarse de nombre_columna a nombre_columna_nuevo, o a modificarse al nuevo tipo de datos (si es que los nombres de las columnas coinciden).

Por ejemplo:

1) Cambiar el nombre y la longitud de la columna "apellidos" de la tabla "alumnos" a "apellidoPaterno":

```
ALTER TABLE alumnos CHANGE apellidos apellidoPaterno VARCHAR(30);
```

Observe que incluimos la longitud del nuevo nombre de la columna, de lo contrario marca MySQL error.

2) Cambie la longitud de la columna nombres a VARCHAR(40):

```
ALTER TABLE alumnos CHANGE nombres nombres VARCHAR(40);
```

Añadir una columna

Para ello usaremos la sentencia ADD [COLUMN], donde la declaración de columna comprende el nombre de la columna, así como su longitud, tipo y otras características, como UNSIGNED, NOT NULL, etc.

Con las opciones FIRST la columna se incluirá al inicio de la tabla, con AFTER nombre_columna, se insertará después de la columna señalada. Si no se incluye ninguna de estas opciones, se incluirá la nueva columna hasta el fondo de la tabla.

Por ejemplo:

1) Añadir una columna en la tabla SALONES con nombre "numAsientos", tipo

Añadir un índice

Introducción a MySQL

Para ello necesitamos la opción ADD INDEX [nombre_indice] (columnas_indice). Si no se especifica el valor del índice, MySQL tomará el nombre de la primer columna. Si desea indexar con campos del tipo BLOB y/o TEXT deberá especificar un número determinado de caracteres. Por ejemplo, podemos indexar la tabla "alumnos" por el campo "ID" de la siguiente manera:

```
ALTER TABLE alumnos ADD INDEX alumnos (id);
```

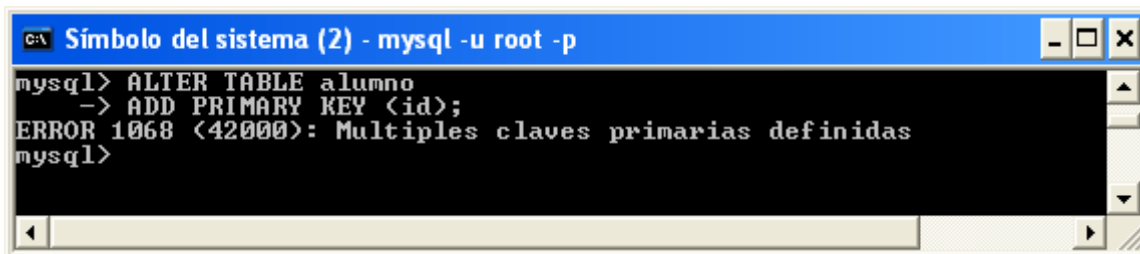
Ejemplo:

Indexe las tablas salones y escuelas por su indentificador.

También se puede añadir un índice por medio de la sentencia ADD UNIQUE, que tiene las mismas reglas que ADD INDEX.

Añadir una llave principal

Necesitamos la sentencia ADD PRIMARY KEY (columnas_indice). Añade una llave principal a las columnas de "columnas_indice". Si existe una clave principal previa MySQL enviará un mensaje de error.

A screenshot of a Windows command prompt window titled "Símbolo del sistema (2) - mysql -u root -p". The window has a blue title bar and standard window controls. The command prompt shows the following text:

```
mysql> ALTER TABLE alumno  
-> ADD PRIMARY KEY <id>;  
ERROR 1068 (42000): Multiples claves primarias definidas  
mysql>
```

Cambio del valor predeterminado

Para cambiar el valor por default de una columna necesitamos la sentencia ALTER columna SET DEFAULT valor, por ejemplo:

```
ALTER TABLE alumno ALTER idEscuela SET DEFAULT 1;
```

CUIDADO: Si la columna tiene valores los sustituirá por el DEFAULT.

Introducción a MySQL

Para quitar el valor por default, es necesaria la sentencia ALTER *nombre_columna* DROP DEFAULT.

Por ejemplo:

Quitemos el default que definimos en el punto anterior:

```
ALTER TABLE alumno ALTER idEscuela DROP DEFAULT;
```

NOTA: Esta opción no cambia los valores de la columna.

Eliminar una columna

Para eliminar una columna necesitamos la sentencia DROP [COLUMN] *nombre_columna*. Si esta columna es parte del índice, se elimina del índice sin borrar este. Si elimina todas las columnas del índice, este también se borra.

NOTA: No hay manera de recuperar la información borrada con este comando.

```
ALTER TABLE alumno DROP COLUMN idEscuela;
```

Borrar un índice

Para borrar un índice hay que utilizar el comando DROP INDEX *nombre_indice*. No hay manera de recuperar un índice después de ejecutado este comando. Tendría que volver a crearlo.

```
ALTER TABLE alumnos DROP INDEX alumnos;
```

Borrar una llave principal

Para borrar una llave principal se necesita el comando DROP PRIMARY KEY. Si una tabla no tiene un índice único creado como PRIMARY KEY, pero tiene uno o más índices creados como UNIQUE, se borrará el primero de los índices creados como UNIQUE.

Introducción a MySQL

Modificar la declaración de una columna

Necesitamos la sentencia `MODIFY [COLUMN] declaración_columna`. La "declaración_columna" sigue el mismo formato que la sentencia `CREATE TABLE`, con el nombre de la columna en un inicio.

```
ALTER TABLE alumnos MODIFY COLUMN apellidoPaterno VARCHAR(40);
```

Renombrar una tabla

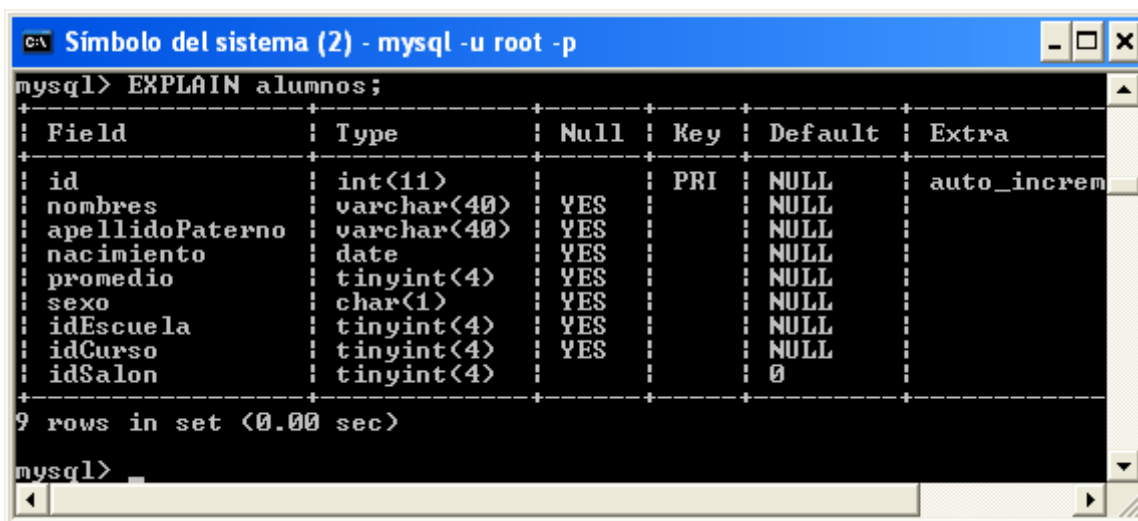
Para renombrar una tabla necesitamos la sentencia `RENAME [AS] nuevo_nombre_tabla`, por ejemplo:

```
ALTER TABLE alumno RENAME AS alumnos;
```

Ver la estructura de una tabla

Para ver la estructura interna de una tabla utilizamos la sentencia `EXPLAIN nombre_tabla`. Por ejemplo:

```
EXPLAIN alumnos;
```



Field	Type	Null	Key	Default	Extra
id	int(11)		PRI	NULL	auto_increment
nombres	varchar(40)	YES		NULL	
apellidoPaterno	varchar(40)	YES		NULL	
nacimiento	date	YES		NULL	
promedio	tinyint(4)	YES		NULL	
sexo	char(1)	YES		NULL	
idEscuela	tinyint(4)	YES		NULL	
idCurso	tinyint(4)	YES		NULL	
idSalon	tinyint(4)			0	

9 rows in set (0.00 sec)

También se puede utilizar la sentencia `SHOW COLUMNS FROM nombre_tabla`;

Introducción a MySQL

Otra opción es mediante el comando `DESCRIBE nombre_tabla;`

Optimización y mantenimiento de una base de datos

Optimizar el espacio de una tabla

Para optimizar el espacio de una tabla necesitamos el comando

```
OPTIMIZE TABLE nombre_tabla;
```

Con ellos se compactará lo más posible la tabla.

Cambiar de contraseña a un usuario

Para cambiar la contraseña de un usuario, necesitamos la sentencia SET, por ejemplo

```
SET PASSWORD FOR root = PASSWORD("hola");
```

Bloquear el acceso a una tabla

Para bloquear el acceso a una tabla necesitamos la sentencia

```
LOCK TABLES nombre_tabla1 [AS alias] {READ | [LOW PRIORITY] WRITE}[, nombre_tabla2 ...]
```

Desbloquear una tabla

Para desbloquear una tabla se utiliza la sentencia:

```
UNLOCK TABLES tabla1, tabla2,...
```

Copias de seguridad y copias de bases de datos

Es importante hacer copias de seguridad de sus bases de datos por si sus tablas se pierden o se deterioran. Si se produce un colapso del sistema, debe ser capaz de restaurar sus tablas al

Introducción a MySQL

estado en que estaban cuando se produjo dicho colapso con la menor pérdida de datos posible. Igualmente debemos de prever un error en el uso u operación del sistema, por ejemplo, un usuario que utilice un DROP DATABASE.

Existen dos formas de respaldar una base de datos:

- 1) Con la herramienta mysqldump
- 2) Copiando directamente los archivos desde el sistema operativo.

mysqldump

Este comando opera en cooperación con el servidor MySQL. Los métodos de copia directa son externos al servidor, y usted debe tomar las medidas para asegurar que ningún cliente modifique las tablas mientras usted las copia. Es el mismo problema que se produce si intenta usar copias de seguridad del sistema de archivos en base de datos de copias de seguridad: si una base de datos está siendo actualizada durante la copia de seguridad del sistema de archivo, los archivos de tabla que entran en dicha copia están en un estado poco consistente y no valen para restaurar la tabla más tarde.

mysqldump es más lento que las técnicas de copia directa.

mysqldump genera archivos de texto transportables a otras máquinas, incluso aquellos con una arquitectura de hardware diferente. Los archivos de copia directa no lo son, a no ser que las tablas que esté copiando usen el formato de almacenamiento MyISAM.

Algunos lineamientos al efectuar un respaldo de la base de datos son:

Realizar copias de seguridad regularmente. Establezca un plan y cúmplalo.

Indique al servidor que realice logging de actualización. Los logs de actualización le ayudarán restaurar una base de datos tras un colapso. Después de restaurarla, puede volver a aplicar los cambios que se hicieron después de la copia, ejecutando las consultas en los logs de actualización.

En el lenguaje de las copias de seguridad del sistema de archivo, los archivos de copia de seguridad de base de datos representan el volcado completo, y los logs de actualización representan los volcados adicionales (DELTA) de los incrementos.

Use un esquema para nombrar archivos de copia de seguridad coherente y comprensible. De

Introducción a MySQL

preferencia indique el nombre de la base de datos y su fecha.

Asegura tener una copia externa al sistema de sus respaldos. No los deje en la misma máquina que la base de datos.

Cuando utiliza el comando **mysqldump** generará un archivo con las sentencias necesarias para crear una base de datos (CREATE TABLES) y los datos vendrán en sentencias INSERT. Tendrá que utilizar estos archivos como archivos batch para restaurar la información. Por ejemplo:

```
-- MySQL dump 10.9
--
-- Host: localhost      Database: escuela
-- -----
-- Server version      4.1.9-max

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE="NO_AUTO_VALUE_ON_ZERO" */;

--
-- Table structure for table `alumnos`
--

DROP TABLE IF EXISTS `alumnos`;
CREATE TABLE `alumnos` (
  `id` int(11) NOT NULL auto_increment,
  `nombres` varchar(40) default NULL,
  `apellidoPaterno` varchar(40) default NULL,
  `nacimiento` date default NULL,
  `promedio` tinyint(4) default NULL,
  `sexo` char(1) default NULL,
  `idEscuela` tinyint(4) default NULL,
  `idCurso` tinyint(4) default NULL,
  `idSalon` tinyint(4) NOT NULL default '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Dumping data for table `alumnos`
--
```

Introducción a MySQL

```
/*!40000 ALTER TABLE `alumnos` DISABLE KEYS */;  
LOCK TABLES `alumnos` WRITE;  
INSERT INTO `alumnos` VALUES (1,'Paco','Arce','1996-02-  
10',8,'M',1,1,1),(2,'Alex','Alexander','1976-03-  
11',9,'M',1,2,2),(3,'Monica','Arellano','1986-04-  
10',10,'F',1,2,2),(4,'Laura','Vieyra','1966-04-  
01',10,'F',1,1,2),(5,'Hernando','Hernández','1956-05-  
05',5,'M',1,1,2),(6,'Laura','Murcio','1976-07-07',7,'F',1,1,1);  
UNLOCK TABLES;
```

La sintaxis general de la instrucción es:

mysqldump [opciones] db_nombre [tabla1, tabla2...]