**MySql**

MySQL es un [sistema de gestión de bases de datos](https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_bases_de_datos).Esta es muy rápida en la lectura cuando utiliza el motor no transaccional [MyISAM](https://es.wikipedia.org/wiki/MyISAM" \o "MyISAM), pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

\*La mayoría de su código se escribe en mayúscula, aunque puede llegar a ingresar el mismo pero en minúscula.

\*Cuando hablo de seleccionar el **elemN** no se refiere a conseguir el nombre de la columna, sino los **dataN** en ella.

Cogidos Basicos

**FROM:** Se usa para clarificar de que **tblN** estoy sacando mis datos.

**SELECT:** Selecciona los **elemN** que quieras, sin necesitar comparación. Puede usarse **\***

**WHERE:** Este sirve para conseguir los datos de la columnas donde tal **elemN** sea igual o valide con tal propiedad.

Tipos de datos

Existen 40 tipos de datos en MySql, entre los que se encuentran:

**-Números**. Los números enteros tienen diferentes formas de ser escritos, cada una teniendo su rango. Ejemplo:

**INT:** DESDE -2.1b a 2.1b.

**TINYINT:** DESDE -128 a 127

**SAMLLINT:** DESDE -32,768 a 32,767

**MEDIUMINT:** DESDE -8.3m a 8.3m.

**BIGINT:** DESDE -2^63 a 2^63 - 1.

Puedes aplicarle un **UNSIGNED** a este rango de números, deshabilitando los números negativos, y obteniendo este espacio doble en los positivos.

Existen también códigos para números decimales:

**FLOAT:** Básico.

**DOUBLE:** Este tiene más espacio, por lo tanto son más grandes.

**DECIMAL:** En este no se aceptan redondos, sirve para guardar datos en forma de pago o dinero.

**-Cadenas de texto y datos binarios.** Las cadenas y datos binarios tienen diferentes formas de ser escritas en código. Ejemplo:

**VARCHAR:** Se puede usar para texto corto, pero no está limitado.

**CHAR:** Este es corto y es limitado (usarse para passwords)

**TEXT:** Este es un texto más largo (entrada de blog).

**ENUM:** Un valor de una lista enumerada

**BLOB:** Archivos comprimidos, imágenes, sonidos (es mejor leer la ruta del archivo en ves del archivo en sí, dado que relentiza la base de datos por lo que este comando es una mala práctica.).

**-Fechas y horas.** Las fechas y horas tienen diferentes formas de ser escritas en código. Ejemplo:

**DATE:** AAAA-MM-DD.

**TIME:** hh:mm:ss.

**DATETIME:** AAAA-MM-DD hh:mm:ss.

**TIMESTAMP:** AAAA-MM-DD hh:mm:ss.

**YEAR:** AAAA.

**-Los mas utilizados.** Estos son los más utilizados a la hora de hacer nuestra base de datos:

**INT:** Para números enteros.

**TINYINT:** Números pequeños, como edades.

**DECIMAL:** Dinero y medidas.

**VARCHAR:** Textos cortos.

**TEXT:** Textos largos.

**DATETIME:** Fechas.

**Database**

Esta es el primer paso, para luego ingresarle tablas. Esta se crea con **CREATE DATABASE dbN;**

Siendo **dbN** = database nombre.

**Table**

En ellas ingresamos distintos tipos de datos para guardarlos, son las que forman al **database.** Primero creamos el molde para luego ir ingresando los datos.

\*Debe hacerse dentro de un database obligatoriamente\*

Tal se crea con:

**CREATE TABLE `tblN` (**

**`elemNID ` typeDATA AUTO\_INCREMENT,**

**`elemN2 ` typeDATA**

**) ENGINE=INNODB DEFAULT CHARSET=utf8** (UTF8MB4)

Donde **tblN**= tabla nombre y **elemN** = es el nombre del elemento.

Luego de delimitar el nombre del elemento se le tiene que asignar que tipo de dato es, por ejemplo: **INT**, **DATATIME**, **TEXT**,etc. Se le puede ingresar un paréntesis luego del tipo de dato, con un número dentro de este para delimitar cuantos números o caracteres pueden ingresar. Ejemplo: **INT(2)** (Siendo solo 2).

El primer **elemN** tiene como nombre id ya que este será el numero de id por el cual podremos elegir el conjunto de datos.

**Codigos de la tabla**

**AUTO\_INCREMENT:** Este sirve para darle una id a cada conjunto de datos sin tener que ir poniendo 1, 2, 3 etc. Comienza en 1 y avanza 1+ por elemento.

**NOT NULL:** No deja que se ingrese un valor vacío a la tabla.

**PRIMARY KEY:** Se utiliza en la primera key o elemento. Esto nos ayuda cuando utilicemos muchas tablas en conjunto.

**CHARSET=utf8:** cambiarse a (UTF8MB4) dado que mysql sino no lo detecta.

**Modificación de la tabla**

**ALTER TABLE `tblN`:** Modificamos una tabla para generar, eliminar, o otras propiedades. Se usa para todas las modificaciones.

**ALTER TABLE `tblN` ADD `elemN` typeDATA AFTER `elemN `:** Esto agrega un elemento a la table. El **AFTER** se utiliza para colocarlo en la posición después de tal elemento.

Las **datan** van entre comillas si son texto.

**Ordenar la tabla**

Con este código podemos ordenar la tabla a través de su data.

**SELECT \* FROM tblN ORDER BY elemN typeOrder.**

Selecciona todas las filas y ordena sus filas a partir de tal elemento en la forma que le digamos.

Los **typeOrder** que existen son:

**ASC:** De A a Z en texto y de menor a mayor en número.

**DESC:** De Z a A en texto y de mayor a menor en número.

**typeOrder LIMIT** n: Limita el orden aplicado a solo los primeros n elementos.

**Contar elementos**

**SELECT elemN, COUNT(\*) FROM tablN GROUP BY elemN**

Con este código contaremos la cantidad de data en tal elemento y cuantos **datan** se repiten en la misma.

Operadores

Existen operadores que nos dejan comparar la data dentro de los **elemN** para recibir solo la que comparta tal estamento. Un ejemplo es con:

**SELECT elemN FROM tblN WHERE elemN > 1**

De esta forma solo recibimos el **dataN** que tenga un valor mayor a 1. Se pueden poner cuantos **elemN** cuantos quieras, o un **\*** para que nos devualva todas las columnas

**BETWEEN** **dataN AND dataN:** Este es un operador que nos da todos los **elemN** que estén entre esos dos **dataN**

**OR**: Este sirve para recibir dato de un tipo O de otra comparación. Ejem: **dataN=”hola” OR dataN=”chau”**

Funciones de Texto Y Numeros

Existen funciones para cadenas que nos dejan modificarlas:

**CONCAT:** Se utiliza entre dos **elemN** para que su **dataN** vuelva concatenada. Ejemplo: **SELECT CONCAT(elemN, “ ”, elemN2) FROM tblN WHERE elemNID=”id”**

**CONCAT\_WS:** Funciona como su pariente pero este nos evita tener que poner el “ ” todo el tiempo entre los **elemN.** Se puede utilizar para agregarle texto también.

**SELECT CONCAT(“ ”, elemN, elemN2) FROM tblN WHERE elemNID=”id”**

Existen funciones para números que nos dejan modificarlas:

**AVG:** Nos da el promedio de todas las **dataN** del **elemN** que elijamos. Ejemplo:

**SELECT AVG(elemN) FROM tblN WHERE elemNID=”id”**

**MIN=** Nos da el minimo numero en tal **elemN.** Ejemplo:

**SELECT \*FROM tblN WHERE elemN= (SELECT MIN(elemN) FROM tblN)**

En este codigo estoy realizando una subconsulta, dado a que el min se relaiza sobre la data en si, y no el **elemN**, es necesario una doble comparación.

**MAX=** Nos da el máximo numero en tal **elemN.** Funciona de la misma manera que **MIN**

**CRUD applications (create, read, update, delete)**

Con estos subconjuntos ya podríamos crear una base de datos funcional que realiza funciones complejas y las muestra sobre nuestra aplicación.

**Seleccionar contenido de la tabla**

Podemos seleccionar un conjunto de datos de la tabla con:

**SELECT \* FROM tblN:** Selecciona todas las columnas y las filas de la tabla

**SELECT elemN FROM tbln:** Muestra solo la columna de datos que estén en ese elemento. También funciona con **WHERE** así elegimos solo una fila de todas las columnas.

**SELECT \* FROM tblN WHERE elemNID = id:** Este nos sirve para elegir a partir de la id, que usualmente es el primer elemento de la tabla.

**SELECT \* FROM tblN WHERE elemN = dataN:** Este nos muestra todas las columnas de un elemento que tengan tal tipo de data.

\*El **select** nos filtra que columna (el \* elige todas las filas), mientras que el **where** nos deja elegir que fila.

Dónde:

|  |  |
| --- | --- |
| **elemNID** | **elemN2** |
| **id** | **dataN** |

**Agregar valores a la tabla**

Con esto ingresamos datos a la tabla, utilizando nuestra estructura ya propuesta. Se crean datos con:

**INSERT INTO `tblN` (`elemN`, `elemN2`) VALUES (NULL, ‘dataN2’)**

Se pueden ingresar cuantos elementos quieras, respondiendo cada **dataN** a el numero de posición del **elemN**

**NULL:** Este se utiliza cuando el **AUTO\_INCREMENT**  esta en nuestra tabla, dado que este anterior nos coloca los id de forma incrementativa y no hay que declararlos. Este está en la posición **dataN1** (correlativo a **elemN1**)dado a que tal en su tabla es el que representa el id de la misma.

**Actualizar datos de una tabla**

Este cambia los datos de una fila para nuestra conveniencia. Se hace con:

**UPDATE tblN SET elemN = datan WHERE elemNID = id.**

En este código elegimos la tabla, luego modificamos un dato de una columna y con el **where** especificamos la fila.

Dónde:

|  |  |
| --- | --- |
| **elemNID** | **elemN2** |
| **id** | **dataN** |

**Eliminar datos de una tabla**

Elimina datos específicos de una tabla.

**DELETE FROM tblN WHERE elemNID = id**