Índice:

[1. Fuentes de información 5](#_Toc127522801)

[2. Atributos típicos usados en diferentes tablas de una base de datos 5](#_Toc127522802)

[2.1 Datos personales 5](#_Toc127522803)

[2.2 Dirección postal 5](#_Toc127522804)

[2.3 Tipo de Vía (en una dirección) 5](#_Toc127522805)

[3. mysqladmin: programa de consola para administrar el servidor mysql 7](#_Toc127522806)

[3.1 Establecer la contraseña del usuario root 7](#_Toc127522807)

[3.2 Cambiar la contraseña del usuario root 7](#_Toc127522808)

[3.3 Comprobar si el servidor MySQL está corriendo 7](#_Toc127522809)

[3.4 Conocer la versión de MySQL del servidor 8](#_Toc127522810)

[3.5 Conocer el estado actual del servidor MySQL 8](#_Toc127522811)

[3.6 Ver el estado de todas las variables del servidor MySQL y sus valores 8](#_Toc127522812)

[3.7 Ver las variables del servidor MySQL y sus valores 8](#_Toc127522813)

[3.8 Cambiar el valor de una variable del servidor 8](#_Toc127522814)

[3.8.1 Hacer que el cambio de la variable se aplique solo a un comando 8](#_Toc127522815)

[3.9 Ver todos los procesos que están corriendo del servidor MySQL 9](#_Toc127522816)

[3.10 Crear una base de datos en el servidor MySQL 9](#_Toc127522817)

[3.11 Eliminar una base de datos 9](#_Toc127522818)

[3.12 Recagar/refrescar los privilegios MySQL 9](#_Toc127522819)

[3.13 Apagar/iniciar/reiniciar el servidor MysQL 9](#_Toc127522820)

[3.14 Comandos FLUSH útiles 11](#_Toc127522821)

[3.15 matar un proceso del cliente MySQL que está durmiendo 11](#_Toc127522822)

[3.16 Ejecutar varios comandos mysqladmin a la vez 11](#_Toc127522823)

[3.17 Conectarse a un servidor MySQL remoto 11](#_Toc127522824)

[3.18 ejecutar un comando en un servidor MySQL remoto 12](#_Toc127522825)

[3.19 Almacenar la información de depuración (debug) del servidor MySQL hacia los logs 12](#_Toc127522826)

[3.20 Ver las opciones de uso de mysqladmin 12](#_Toc127522827)

[4. MySQL: cliente de consola para conectarse al servidor MySQL 12](#_Toc127522828)

[4.1 CONFIGURAR EL SERVIDOR PARA QUE PERMITA CONEXIONES REMOTAS A TRAVÉS DE LA RED: 12](#_Toc127522829)

[4.1.1 Servidor MySQL 8.0 sobre S.O Linux 12](#_Toc127522830)

[4.1.2 Servidor MySQL 5.6 sobre S.O Linux 13](#_Toc127522831)

[4.1.3 Servidor MySQL 5.6 sobre Windows 13](#_Toc127522832)

[4.2 Comando para Conectarse por consola al servidor, para desconectarse, mostrar BD existentes, mostrar tablas, ejecutar fichero de script,… 13](#_Toc127522833)

[4.2.1 Cuando intentamos conectarnos al servidor desde un programa cliente que está en el equipo servidor: ¿Qué usuario considera el servidor que está conectándose? CUIDADO!!!. 14](#_Toc127522834)

[4.2.2 Mostrar BD disponibles 15](#_Toc127522835)

[4.2.3 Usar una BD llamada bdempleados 15](#_Toc127522836)

[4.2.4 Mostrar las tablas de la BD 15](#_Toc127522837)

[4.2.5 Ejecutar un comando MySQL desde la consola del SO 15](#_Toc127522838)

[4.2.6 Ejecutar un fichero de script 15](#_Toc127522839)

[**4.2.6.1** desde la consola del SO 15](#_Toc127522840)

[**4.2.6.2** Desde el programa cliente mysql 15](#_Toc127522841)

[**4.2.6.3** Ejecutar todos los comandos de un script SQL aunque alguno produzca error 15](#_Toc127522842)

[4.2.7 Introducir comandos SQL multilínea por consola: 16](#_Toc127522843)

[4.2.8 Cancelar la introducción de un comando multilínea 16](#_Toc127522844)

[4.2.9 SELECT: no mostrar fila de encabezado / no mostrar caracteres que separan columnas: 16](#_Toc127522845)

[4.3 No puedo conectarme al servidor, posibles motivos 17](#_Toc127522846)

[4.4 Cambiar el PATH de Windows para incluir los programas de consola de MySQL 17](#_Toc127522847)

[5. Configuración del servidor MySQL 18](#_Toc127522848)

[5.1 Configuración del servidor en Windows 18](#_Toc127522849)

[5.1.1 Carpetas y ficheros utilizados en Windows 18](#_Toc127522850)

[5.2 Configuración del servidor en Linux 19](#_Toc127522851)

[5.2.1 Mysql 8.0 19](#_Toc127522852)

[5.2.2 Carpetas y ficheros utilizados 19](#_Toc127522853)

[**5.2.2.1** Archivo de configuración Del servidor 19](#_Toc127522854)

[**5.2.2.2** Caso especial: Los directorios 20](#_Toc127522855)

[6. SQL 20](#_Toc127522856)

[6.1 Crear comentarios en ficheros de script: #, /\* \*\*/, -- 20](#_Toc127522857)

[6.2 DDL (Data Definition Language) 21](#_Toc127522858)

[6.2.1 CREATE DATABASE 21](#_Toc127522859)

[6.2.2 DROP DATABASE 22](#_Toc127522860)

[6.2.3 CREATE TABLE 22](#_Toc127522861)

[**6.2.3.1** Ejemplo de sintaxis básica 22](#_Toc127522862)

[**6.2.3.2** COMENTARIOS: de columna, índice, tabla 22](#_Toc127522863)

[**6.2.3.3** Crear tabla que sea copia de otra e incluya la PRIMARY KEY y los índices (KEY) 23](#_Toc127522864)

[**6.2.3.4** Crear tabla que sea copia de otra, pero que no incluya las KEYs 24](#_Toc127522865)

[**6.2.3.5** Especificar el tipo de almacenamiento de la tabla como InnoDB 24](#_Toc127522866)

[**6.2.3.6** CREATE TEMPORARY TABLE: Crear una Tabla temporal en memoria RAM con los datos de un SELECT 25](#_Toc127522867)

[**6.2.3.7** SERIAL, AUTO\_INCREMENT 25](#_Toc127522868)

[**6.2.3.8** ON DELETE/UPDATE[RESTRICT | CASCADE | SET NULL | NO ACTION] 26](#_Toc127522869)

[**6.2.3.9** Índices: PRIMARY KEY, UNIQUE , INDEX 27](#_Toc127522870)

[6.2.3.9.1 Orden de Colocación en CREATE TABLE de los distintos índices: 27](#_Toc127522871)

[6.2.3.9.2 PRIMARY KEY 27](#_Toc127522872)

[**6.2.3.9.2.1.** 3 maneras distintas de definir la PRIMARY KEY: 27](#_Toc127522873)

[6.2.3.9.3 UNIQUE [INDEX|KEY] 28](#_Toc127522874)

[6.2.3.9.4 INDEX|KEY 29](#_Toc127522875)

[**6.2.3.10** FOREIGN KEY 32](#_Toc127522876)

[6.2.3.10.1 Errores típicos al crear una FOREIGN KEY 32](#_Toc127522877)

[6.2.3.10.2 Ejemplos correctos e incorrectos de FOREIGN KEY que provienen de PRIMARY KEY formada por más de un atributo 33](#_Toc127522878)

[6.2.3.10.3 Ejemplos correctos e incorrectos de FOREIGN KEY que provienden de PRIMARY KEY formada por más de un atributo 33](#_Toc127522879)

[**6.2.3.11** NULL / NOT NULL 33](#_Toc127522880)

[**6.2.3.12** DEFAULT 34](#_Toc127522881)

[6.2.3.12.1 Valor implícito por defecto de cada tipo de dato. 35](#_Toc127522882)

[6.2.3.12.2 DATETIME, DATE, TIME, TIMESTAMP: asignar valores DEFAULT 35](#_Toc127522883)

[**6.2.3.13** AUTO\_INCREMENT = 77 37](#_Toc127522884)

[**6.2.3.14** CHECK 37](#_Toc127522885)

[6.2.4 Comandos para añadir, modificar, borrar CHECKS 38](#_Toc127522886)

[**6.2.4.1** CHECK TABLE 38](#_Toc127522887)

[**6.2.4.2** CREAR UNA COPIA DE UNA TABLA 39](#_Toc127522888)

[6.2.5 DROP TABLE 39](#_Toc127522889)

[6.2.6 Copiar una BD / Cambiar el nombre de una base de datos (importar una BD desde un script SQL), hacer backup de una BD 39](#_Toc127522890)

[6.2.7 Longitud máxima de los identificadores: 40](#_Toc127522891)

[6.2.8 Storage Engines 40](#_Toc127522892)

[**6.2.8.1** InnoDB 40](#_Toc127522893)

[**6.2.8.2** MyISAM Storage Engine 40](#_Toc127522894)

[**6.2.8.3** MEMORY Storage Engine 41](#_Toc127522895)

[6.2.9 Partitions 41](#_Toc127522896)

[6.3 DML 42](#_Toc127522897)

[6.3.1 ISERT INTO 42](#_Toc127522898)

[**6.3.1.1** **Opción A**: Indicar valores para todos los atributos de la tabla: 42](#_Toc127522899)

[**6.3.1.2** **Opción B**: Indicar sólo algunos atributos de la tabla: 42](#_Toc127522900)

[**6.3.1.3** Insertar una fila en una tabla que incluye un atributo que tiene la propiedad **AUTO\_INCREMENT** 43](#_Toc127522901)

[**6.3.1.4** Insertar varias filas con un solo comando INSERT INTO 43](#_Toc127522902)

[**6.3.1.5** Insertar filas obteniendo los datos mediante una consulta a otras tablas 43](#_Toc127522903)

[**6.3.1.6** REPLACE: utiliza los datos de la tabla B para actualizar o insertar datos en la tabla A 44](#_Toc127522904)

[**6.3.1.7** SET foreign\_key\_checks = 0, Deshabilitar las claves ajenas temporalmente para introducir datos 44](#_Toc127522905)

[6.3.1.7.1 DISABLE KEYS (sólo funciona en tablas MyISAM) 45](#_Toc127522906)

[**6.3.1.8** Ejemplos: 45](#_Toc127522907)

[**6.3.1.9** LOAD DATA INFILE: Insertar filas procedentes de un fichero 46](#_Toc127522908)

[6.3.1.9.1 ERROR al importar los datos debido a violaciones de integridad referencial (clave ajena) 46](#_Toc127522909)

[6.3.2 Acelerar la inserción de datos en una tabla 47](#_Toc127522910)

[6.3.3 UPDATE 49](#_Toc127522911)

[6.3.4 Restricción del Workbench que impide UPDATES O DELETES debido al SAFE MODE cuando no se incluye una KEY en el WHERE o no se usa LIMIT 50](#_Toc127522912)

[6.3.5 DELETE 51](#_Toc127522913)

[**6.3.5.1** Ejemplos de DELETE 51](#_Toc127522914)

[**6.3.5.2** TRUNCATE 52](#_Toc127522915)

[6.3.6 LOCK TABLE / UNLOCK TABLE 53](#_Toc127522916)

[**6.3.6.1** Cuando bloquear una tabla 55](#_Toc127522917)

[6.4 CONSULTAS (SELECT) 55](#_Toc127522918)

[6.4.1 Orden de ejecución de las distintas cláusulas de una consulta SELECT: 55](#_Toc127522919)

[6.4.2 INNER JOIN, NATURAL JOIN, LEF JOIN, RIGHT JOIN, FULL JOIN … 56](#_Toc127522920)

[**6.4.2.1** INNER JOIN 56](#_Toc127522921)

[**6.4.2.2** Otros tipos de JOIN: CROSS JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN 57](#_Toc127522922)

[**6.4.2.3** Tipos de JOIN 57](#_Toc127522923)

[**6.4.2.4** Ejemplos de distintos tipos de JOIN: 58](#_Toc127522924)

[6.4.3 Cadenas de texto 60](#_Toc127522925)

[6.4.4 IFNULL / NZ / COALESCE 60](#_Toc127522926)

[6.4.5 AVG, SUM, MAX, MIN 60](#_Toc127522927)

[6.4.6 UNION / UNION ALL 60](#_Toc127522928)

[**6.4.6.1** Reunir las filas de 2 o más tablas (A, B) en una sola tabla (C) 61](#_Toc127522929)

[6.4.7 ORDER BY 61](#_Toc127522930)

[6.4.8 Vistas 62](#_Toc127522931)

[**6.4.8.1** Vistas en MySQL 62](#_Toc127522932)

[**6.4.8.2** Vistas en Access 62](#_Toc127522933)

[6.4.9 NULL 62](#_Toc127522934)

[6.4.10 CONCAT 63](#_Toc127522935)

[6.4.11 Cadena de texto 63](#_Toc127522936)

[6.4.12 Seleccionar todos los atributos de una tabla: \* 63](#_Toc127522937)

[6.4.13 LIKE y REGEXP (Patrones y Expresiones regulares) 63](#_Toc127522938)

[**6.4.13.1** LIKE (Patrones)Comodines de texto: % y \_ 63](#_Toc127522939)

[**6.4.13.2** REGEXP (Expresiones regulares): 64](#_Toc127522940)

[6.4.14 LENGTH 66](#_Toc127522941)

[6.4.15 COUNT 66](#_Toc127522942)

[6.4.16 Campo autonumérico 67](#_Toc127522943)

[6.4.17 GROUP BY 67](#_Toc127522944)

[6.4.18 LIMIT 69](#_Toc127522945)

[**6.4.18.1** Ejemplo: Encontrar a los empleados con el salario más alto:**Subconsulta** 70](#_Toc127522946)

[**6.4.18.2** Ejemplo: Encontrar a los empleados cuyos salarios están entre los 5 salarios más altos de la empresa: **LIMIT 4,1** 70](#_Toc127522947)

[**6.4.18.3** Ejemplo: Encontrar a los 3 empleados que más ganan en la empresa 71](#_Toc127522948)

[**6.4.18.4** TOP (Access) 71](#_Toc127522949)

[6.4.19 Fechas, como trabajar con fechas 72](#_Toc127522950)

[**6.4.19.1** DATE\_FORMAT() 72](#_Toc127522951)

[**6.4.19.2** Realizar cálculos con fechas 74](#_Toc127522952)

[6.4.20 SELECT INTO OUTFILE: guardar en un fichero el resultado de un SELECT 74](#_Toc127522953)

[**6.4.20.1** Especificar el formato del fichero 75](#_Toc127522954)

[**6.4.20.2** mysql -e "SELECT ..." > file\_name : Otra manera de volcar una tabla en un fichero 76](#_Toc127522955)

[**6.4.20.3** ERROR: The MySQL server is running with the --secure-file-priv option so it cannot execute this statement 76](#_Toc127522956)

[6.4.20.3.1 Soclución alternativa, cambiar configuración del servidor: 77](#_Toc127522957)

[6.5 Funciones en MySQL 77](#_Toc127522958)

[6.5.1 Table 12.2 Operators 77](#_Toc127522959)

[6.5.2 Table 12.3 Comparison Operators 79](#_Toc127522960)

[6.5.3 Table 12.4 Logical Operators 80](#_Toc127522961)

[6.5.4 Table 12.5 Assignment Operators 80](#_Toc127522962)

[6.5.5 Table 12.6 Flow Control Operators 80](#_Toc127522963)

[6.5.6 Table 12.7 String Operators 80](#_Toc127522964)

[6.5.7 Table 12.8 String Comparison Operators 82](#_Toc127522965)

[6.5.8 Table 12.9 String Regular Expression Operators 82](#_Toc127522966)

[6.5.9 Table 12.10 Numeric Functions and Operators 82](#_Toc127522967)

[6.5.10 Table 12.11 Arithmetic Operators 84](#_Toc127522968)

[6.5.11 Table 12.12 Mathematical Functions 84](#_Toc127522969)

[6.5.12 Table 12.13 Date/Time Functions 85](#_Toc127522970)

[6.5.13 Table 12.14 Cast Functions 87](#_Toc127522971)

[6.5.14 Table 12.15 XML Functions 87](#_Toc127522972)

[6.5.15 Table 12.16 Bit Functions and Operators 87](#_Toc127522973)

[6.5.16 Table 12.17 Encryption Functions 87](#_Toc127522974)

[6.5.17 Table 12.18 Information Functions 88](#_Toc127522975)

[6.5.18 Table 12.25 Miscellaneous Functions 89](#_Toc127522976)

[6.5.19 Table 12.26 Aggregate (GROUP BY) Functions 89](#_Toc127522977)

[6.5.20 Funciones de cadena 90](#_Toc127522978)

[6.6 Backup and Recovery 90](#_Toc127522979)

[7. Criterios para diseña una BD 90](#_Toc127522980)

[7.1.1 Por Qué es mejor evitar que los atributos permitan NULL 90](#_Toc127522981)

# Fuentes de información

<http://dev.mysql.com/doc/refman/5.6/en/create-table.html>

<http://www.w3schools.com/sql/>

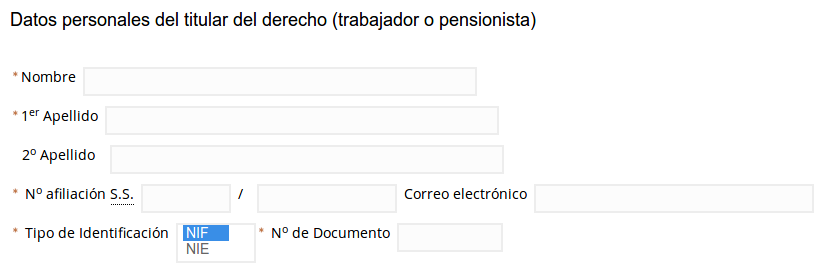
[www.jorgesanchez.net/bd/sgbd.html](http://www.jorgesanchez.net/bd/sgbd.html)

[www.nachocabanes.com/sql/curso/index.php](http://www.nachocabanes.com/sql/curso/index.php)

# Atributos típicos usados en diferentes tablas de una base de datos

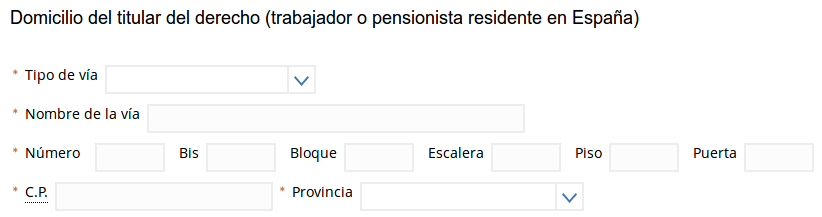
## Datos personales

<https://w6.seg-social.es/solTse/Inicio?Input1=Solicitar+%2F+Renovar++Tarjeta+Sanitaria>



## Dirección postal

<https://w6.seg-social.es/solTse/Inicio?Input1=Solicitar+%2F+Renovar++Tarjeta+Sanitaria>



## Tipo de Vía (en una dirección)

<https://w6.seg-social.es/solTse/Inicio?Input1=Solicitar+%2F+Renovar++Tarjeta+Sanitaria>

<select class="" size="1" name="idTipoVia" id="idTipoVia">

<option value="" selected></option>

<option value="CLCALLE">CALLE</option>

<option value="PZPLAZA">PLAZA</option>

<option value="AVAVENIDA">AVENIDA</option>

<option value="PPPASEO">PASEO</option>

<option value="GLGLORIETA">GLORIETA</option>

<option value=""></option>

<option value="">-----------------------</option>

<option value=""></option>

<option value="AQACEQUIA">ACEQUIA</option>

<option value="ACACERA">ACERA</option>

<option value="ALALAMEDA">ALAMEDA</option>

<option value="ADALDEA">ALDEA</option>

<option value="AMAMPLIACION">AMPLIACION</option>

<option value="ANANGOSTA">ANGOSTA</option>

<option value="ASAPARTADO DE CORREOS">APARTADO DE CORREOS</option>

<option value="APAPARTAMENTOS">APARTAMENTOS</option>

<option value="ATATAJO">ATAJO</option>

<option value="AVAVENIDA">AVENIDA</option>

<option value="BABAJADA">BAJADA</option>

<option value="BCBARRANCO">BARRANCO</option>

<option value="BDBARRIADA">BARRIADA</option>

<option value="BOBARRIO">BARRIO</option>

<option value="BLBLOQUES">BLOQUES</option>

<option value="CLCALLE">CALLE</option>

<option value="CACALLEJA">CALLEJA</option>

<option value="CJCALLEJON">CALLEJON</option>

<option value="CECALLEJUELA">CALLEJUELA</option>

<option value="CZCALZADA">CALZADA</option>

<option value="CMCAMINO">CAMINO</option>

<option value="CRCARRERA">CARRERA</option>

<option value="CTCARRETERA">CARRETERA</option>

<option value="CSCASERIO">CASERIO</option>

<option value="CHCHALET">CHALET</option>

<option value="COCOLONIA">COLONIA</option>

<option value="CPCOOPERATIVA">COOPERATIVA</option>

<option value="KOCORRAL">CORRAL</option>

<option value="CNCOSTANILLA">COSTANILLA</option>

<option value="CUCUESTA">CUESTA</option>

<option value="EDEDIFICIO">EDIFICIO</option>

<option value="EAESCALA">ESCALA</option>

<option value="ESESCALERA">ESCALERA</option>

<option value="ELESCALINATA">ESCALINATA</option>

<option value="ETESTRADA">ESTRADA</option>

<option value="GLGLORIETA">GLORIETA</option>

<option value="GRGRUPO">GRUPO</option>

<option value="LLLLANO">LLANO</option>

<option value="LGLUGAR">LUGAR</option>

<option value="MZMANZANA">MANZANA</option>

<option value="MCMERCADO">MERCADO</option>

<option value="MOMONTAñA">MONTAñA</option>

<option value="MNMUNICIPIO">MUNICIPIO</option>

<option value="PQPARQUE">PARQUE</option>

<option value="PCPARTICULAR">PARTICULAR</option>

<option value="PKPARTIDA">PARTIDA</option>

<option value="PDPASADIZO">PASADIZO</option>

<option value="PJPASAJE">PASAJE</option>

<option value="PPPASEO">PASEO</option>

<option value="PAPASEO ALTO">PASEO ALTO</option>

<option value="POPASEO BAJO">PASEO BAJO</option>

<option value="PIPASILLO">PASILLO</option>

<option value="PSPASO">PASO</option>

<option value="SPPASSEIG">PASSEIG</option>

<option value="PTPATIO">PATIO</option>

<option value="PLPLACETA">PLACETA</option>

<option value="PZPLAZA">PLAZA</option>

<option value="PEPLAZOLETA">PLAZOLETA</option>

<option value="PUPLAZUELA">PLAZUELA</option>

<option value="PBPOBLADO">POBLADO</option>

<option value="PGPOLIGONO">POLIGONO</option>

<option value="PRPORTALES">PORTALES</option>

<option value="PVPRIVADA">PRIVADA</option>

<option value="PNPROLONGACION">PROLONGACION</option>

<option value="RARAMAL">RAMAL</option>

<option value="RBRAMBLA">RAMBLA</option>

<option value="RPRAMPA">RAMPA</option>

<option value="RERESIDENCIA">RESIDENCIA</option>

<option value="RLRESIDENCIAL">RESIDENCIAL</option>

<option value="RRRIBERA">RIBERA</option>

<option value="RNRINCON">RINCON</option>

<option value="RCRINCONADA">RINCONADA</option>

<option value="RDRONDA">RONDA</option>

<option value="SCSECTOR">SECTOR</option>

<option value="SDSENDA">SENDA</option>

<option value="SRSENDERO">SENDERO</option>

<option value="XXSIN DATOS DOMICILIAR">SIN DATOS DOMICILIAR</option>

<option value="SUSUBIDA">SUBIDA</option>

<option value="TOTORRE">TORRE</option>

<option value="TTTORRENTE">TORRENTE</option>

<option value="TLTRANSVERSAL">TRANSVERSAL</option>

<option value="TSTRASERA">TRASERA</option>

<option value="TRTRAVESIA">TRAVESIA</option>

<option value="URURBANIZACION">URBANIZACION</option>

<option value="VIVIA">VIA</option>

<option value="VLVILLAS">VILLAS</option>

<option value="VVVIVIENDAS">VIVIENDAS</option>

<option value="ZZOTROS">OTROS</option>

</select>

# mysqladmin: programa de consola para administrar el servidor mysql

Fuente: <https://universo-digital.net/comandos-utiles-de-mysqladmin/>

## Establecer la contraseña del usuario root

mysqladmin -u root password TU-PASS

## Cambiar la contraseña del usuario root

mysqladmin -u root –pVIEJO-PASS password 'NUEVO-PASS'

## Comprobar si el servidor MySQL está corriendo

mysqladmin -u root -p ping

systemctl status mysql.service

## Conocer la versión de MySQL del servidor

# **mysqladmin -u root -p version**

Enter password:

mysqladmin Ver 8.42 Distrib 5.5.54, for debian-linux-gnu on i686

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

<strong>Server version 5.5.54-0+deb8u1</strong>

Protocol version 10

Connection Localhost via UNIX socket

UNIX socket /var/run/mysqld/mysqld.sock

Uptime: 11 min 6 sec

Threads: 1 Questions: 624 Slow queries: 0 Opens: 220 Flush tables: 1 Open tables: 72 Queries per second avg: 0.936

## Conocer el estado actual del servidor MySQL

Para conocer el tiempo que lleva ejecutándose el servidor MySQL, así como el número de hilos y consultas:

mysqladmin -u root -p status

## Ver el estado de todas las variables del servidor MySQL y sus valores

mysqladmin -u root -p extended-status

## Ver las variables del servidor MySQL y sus valores

mysqladmin -u root -p variables

## Cambiar el valor de una variable del servidor

### Hacer que el cambio de la variable se aplique solo a un comando

<https://dev.mysql.com/doc/refman/8.0/en/optimizer-hints.html#optimizer-hints-set-var>

The SET\_VAR hint sets the session value of a system variable temporarily (for the duration of a single statement). Examples:

SELECT /\*+ SET\_VAR(sort\_buffer\_size = 16M) \*/ name FROM people ORDER BY name;

INSERT /\*+ SET\_VAR(foreign\_key\_checks=OFF) \*/ INTO t2 VALUES(2);

SELECT /\*+ SET\_VAR(optimizer\_switch = 'mrr\_cost\_based=off') \*/ 1;

Syntax of the SET\_VAR hint:

SET\_VAR(var\_name = value)

With SET\_VAR, there is no need to save and restore the variable value. This enables you to replace multiple statements by a single statement. Consider this sequence of statements:

SET @saved\_val = @@SESSION.var\_name;

SET @@SESSION.var\_name = value;

SELECT ...

SET @@SESSION.var\_name = @saved\_val;

The sequence can be replaced by this single statement:

SELECT /\*+ SET\_VAR(var\_name = value) ...

## Ver todos los procesos que están corriendo del servidor MySQL

mysqladmin -u root -p processlist

## Crear una base de datos en el servidor MySQL

mysqladmin -u root -p create nombre-base-de-datos

## Eliminar una base de datos

mysqladmin -u root -p drop nombre-base-de-datos

## Recagar/refrescar los privilegios MySQL

El comando **reload** le dice al servidor que recargue las tablas grant;

mientras que el comando **refresh** limpia todas las tablas y reabre los archivos de logs.

mysqladmin -u root -p reload

mysqladmin -u root -p refresh

## Apagar/iniciar/reiniciar el servidor MysQL

Para apagar el servidor:

mysqladmin -u root -p shutdown

systemctl stop mysql.service

service mysql stop

Para iniciar el servidor:

systemctl start mysql.service

service mysql start

Para reiniciar el servidor:

systemctlrestart mysql.service

service mysql restart

## Comandos FLUSH útiles

flush-hosts: Limpia la información sobre hosts en la caché de hosts

flush-tables: Limpia todas las tablas.

flush-thereads: Limpia todos los hilos caché.

flush-logs: Limpia la información de logs.

flush-privileges: Recarga las tablas grant (lo mismo que reload).

flush-status: Limpia las variables de estado.

Para ejecutar los comandos anteriores, solo ejecuta:

mysqladmin -u root -p *flush-comando*

## matar un proceso del cliente MySQL que está durmiendo

Primero debes identificar el proceso del cliente MySQL que está durmiendo (sleeping):

mysqladmin -u root -p processlist

Enter password:

+----+------+-----------+----+---------+------+-------+------------------+

| Id | User | Host | db | Command | Time | State | Info |

+----+------+-----------+----+---------+------+-------+------------------+

| 5 | root | localhost | | Sleep | 14 | | |

| 8 | root | localhost | | Query | 0 | | show processlist |

+----+------+-----------+----+---------+------+-------+------------------+

De comando tenemos que el proceso con ID 5 se encuentra durmiend. Para matarlo ejecuta el siguiente comando:

mysqladmin -u root -p kill 5

Enter password:

+----+------+-----------+----+---------+------+-------+------------------+

| Id | User | Host | db | Command | Time | State | Info |

+----+------+-----------+----+---------+------+-------+------------------+

| 12 | root | localhost | | Query | 0 | | show processlist |

+----+------+-----------+----+---------+------+-------+------------------+

Para matar varios procesos a la vez, puedes hacerlo pasando el ID del proceso separado por comas, como se muestra a continuación:

mysqladmin -u root -p kill 4,12

## Ejecutar varios comandos mysqladmin a la vez

Es muy simple, solo tienes que separarlos por espacio, como se muestra a continuación:

mysqladmin -u root -p processlist status version

## Conectarse a un servidor MySQL remoto

La conexión a servidores remotos MySQL puede hacerse mediante la -h (host):

mysqladmin **-h 192.168.25.166** -u root -p

## ejecutar un comando en un servidor MySQL remoto

Se sigue el mismo procedimiento que el ejemplo anterior, solo debes adicionar el comando que quieres ejecutar. Por ejemplo, para ver el estado (status) del servidor MySQL remoto:

mysqladmin -h 192.168.25.166 -u root –p status

iniciar/parar la replicación MySQL en un servidor esclavo

Para iniciar la replicación:

mysqladmin -u root -p start-slave

Para parar la replicación:

mysqladmin -u root -p stop-slave

## Almacenar la información de depuración (debug) del servidor MySQL hacia los logs

Si activas la depuración de información, tendrás en los logs información sobre los locks en uso, memoria y consultas SQL, además de información sobre el programador de eventos:

mysqladmin -u root -p debug

## Ver las opciones de uso de mysqladmin

mysqladmin --help

# MySQL: cliente de consola para conectarse al servidor MySQL

## CONFIGURAR EL SERVIDOR PARA QUE PERMITA CONEXIONES REMOTAS A TRAVÉS DE LA RED:

### Servidor MySQL 8.0 sobre S.O Linux

<https://stackoverflow.com/questions/50570592/mysql-8-remote-access>

For MySQL 8

Crear copia de seguridad del fichero de configuración:

sudo cp /etc/mysql/mysql.conf.d/mysqld.cnf /etc/mysql/mysql.conf.d/mysqld.cnf.bak

Editar el fichero de configuración: mysqld.cnf

sudogedit /etc/mysql/mysql.conf.d/mysqld.cnf

And modify or add the bind-address option:

[mysqld]

bind-address = 0.0.0.0

Restart the mysql server

sudo service mysql restart

### Servidor MySQL 5.6 sobre S.O Linux

sudo gedit /etc/mysql/my.cnf

#COMENTAR LA LÍNEA SIGUIENTE DEL FICHERO DE CONFIGURACIÓN :

bind-address= 127.0.0.1

#PARA ELLO PRECEDERLA DEL SÍMBOLO #:

#bind-address = 127.0.0.1

#reiniciar servicio

Sudo srevice mysql restart

### Servidor MySQL 5.6 sobre Windows

Ídem al Servidor MySQL 5.6 sobre LINUX, pero el fichero de configuración del servidor se encuentra en:

C:\ProgramData\MySQL\MySQL Server 5.6\my.ini" MySQL56

Para averiguar cuál es el fichero de configuración del servidor:

<https://blog.sqlauthority.com/2014/01/07/mysql-locate-the-configuration-file-my-ini-or-my-cnf-on-windows-platform/>

Find MySQL Service in Services section. The MySQL services are usually named as MySQL56. Right click on it and go to properties. Ruta de acceso al ejecutable:

"C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqld.exe" --defaults-file="C:\ProgramData\MySQL\MySQL Server 5.6\my.ini" MySQL56

#el commando indicar cuál es el fichero de configuración utilizado

## Comando para Conectarse por consola al servidor, para desconectarse, mostrar BD existentes, mostrar tablas, ejecutar fichero de script,…

#abrir la consola

#conectarse al servidor con usuario root y pedir password

**mysql –u root –p**

#conectarse al servidor con usuario root y con la password**MiContraseña**

# la contraseña se pega a **-p** (sin espacios en blanco)

**mysql –u root –pMiContraseña**

#conectarse al servidor con usuario root y pedir password, servidor(host)**192.168.1.21**

**mysql –u root –p -h 192.168.1.21**

**mysql –u root –pMiContraseña -h 192.168.1.21**

#conectarse al servidor con usuario root y pedir password, servidor 192.168.1.21, puerto(P mayúscula)**3306**

**mysql –u root –p -h 192.168.1.21 -P3306**

#conectarse al servidor con usuario root, pedir password y convertir en activa la Base de Datos bdempleados (no se precede de modificador)

mysql –u root –p bdempelados

#conectarse al servidor usando la cuenta anónima (si está activa):

mysql

#desconectarse del servidor,2 opciones equivalentes:

quit

exit

#una vez que el cliente MySQL se ha conecatado al servidor puedo ejecutar comandos:

### Cuando intentamos conectarnos al servidor desde un programa cliente que está en el equipo servidor: ¿Qué usuario considera el servidor que está conectándose? CUIDADO!!!.

El servidor considera que el usuario que intenta conectarse es distinto dependiendo de:

* + si empleamos la opción ***–h*** o no
  + que ***ip*** indiquemos con opción ***–h***

Para probarlo **hemos intentado conectarnos al servidor desde una consola del propio equipo servidor** y **hemos escrito la contraseña incorrecta**, **para así ver el mensaje de error**, en el que **se muestra que usuario considera el servidor que está intentado conectarse**:

profesor@ubuntu:~$ **mysql -u root -pAbcd123456.**

mysql: [Warning] Using a password on the command line interface can be insecure.

ERROR 1045 (28000): Access denied for user **'root'@'localhost'** (using password: YES)

profesor@ubuntu:~$ **mysql -u root -pAbcd123456.-h 192.168.33.78**

mysql: [Warning] Using a password on the command line interface can be insecure.

ERROR 1045 (28000): Access denied for user **'root'@'192.168.33.78'** (using password: YES)

profesor@ubuntu:~$ **mysql -u root -pAbcd123456.-h 127.0.0.1**

mysql: [Warning] Using a password on the command line interface can be insecure.

ERROR 1045 (28000): Access denied for user **'root'@'localhost'** (using password: YES)

Consultamos los usuarios existentes:

**mysql> SELECT user, host FROM mysql.user;**

+------------------+-----------+

| user | host |

+------------------+-----------+

**| root | % |**

**| root | localhost |**

+------------------+-----------+

**¡SORPRESA!**: vemos que si indicamos la ip del servidor escribiendo: ***mysql -u root -pAbcd123456. -h 192.168.33.78*** , entonces el servidor considera que el usuario no es **'root'@'localhost'** sino que es el **'root'@'192.168.33.78'**, el cual , al no existir explícitamente en el servidor, será asimilado al usuario **'root'@'%'**

### Mostrar BD disponibles

#mostar BD disponibles

mysql>**show databases;**

### Usar una BD llamada bdempleados

#usar por defecto una BD llamada shakila, si algún comando no espcecificase que BD emplear, se utilizaría la indicada en el último USE

mysql>**use bdempleados;**

### Mostrar las tablas de la BD

#mostrar las tablas de la BD

mysql>**show tables;**

### Ejecutar un comando MySQL desde la consola del SO

Ejecutar un comando mysql **desde la consola del SO**:

**root@ubu:~# mysql -u root -p -e "SELECT \* FROM bdempleados.departamentos"**

### Ejecutar un fichero de script

#### desde la consola del SO

**root@ubu:~# mysql -u root -p < /backup/backupCompleto.sql**

**#ejecuta un USE bdempleados implicitamente antes de ejecutar el script**

**root@ubu:~# mysql -u root -p bdempleados < /backup/backupCompleto.sql**

**root@ubu:~# mysql -u root -p -e"/backup/backupCompleto.sql"**

#### Desde el programa cliente mysql

**mysql>source'/backup/backupCompleto.sql'**

**#NO se pone punto y coma al final del comando source**

**#NO se indica el fichero (ni el path) entre comillas, aunque incluya espacios en blanco; esto es así tanto para cliente Windows o Linux.**

#### Ejecutar todos los comandos de un script SQL aunque alguno produzca error

Opciones: -f , --force

**root@ubu:~# mysql -u root –p –f –e"/backup/backupCompleto.sql"**

### Introducir comandos SQL multilínea por consola:

El cliente de consola MySQL permite que un comando se extienda por más de una línea, hasta que se incluya el punto y coma final no considera finalizado el comando:

**mysql> SELECT \***

**FROM bdempleados.clientes**

**WHERE salario < 12000;**

### Cancelar la introducción de un comando multilínea

Cancelar la introducción de un comando multilínea: escribiremos en una nueva línea el texto:

**\c**

Ejemplo:

**mysql> SELECT \***

**FROM bdempleados.clientes**

**WHERE salario < 12000 AND**

**\c**

### SELECT: no mostrar fila de encabezado / no mostrar caracteres que separan columnas:

No mostrar fila de título (nombre de columnas): -N (--skip-column-names)

**#conectarse al servidor incluyendo el modificador –N**

profesor@ubuntu:~$ **mysql -N -u root -p**

mysql> **SELECT \* FROM bdempleados.clientes ;**

+-----+-----------+-------------------------+-----------+------+-----------+-----------+------------+

| 101 | 22427339Q | DISTRIBUCIONES GOMEZ | MADRID | 7499 | 300000.00 | 0.00 | 500000.00 |

| 102 | 27438931T | LOGITRONICA S.L | BARCELONA | 7654 | 0.00 | 120000.00 | 500000.00 |

| 103 | 27465913A | INDUSTRIAS LACTEAS S.A. | LAS ROZAS | 7844 | 0.00 | NULL | 1000000.00 |

| 104 | 22918037D | TALLERES ESTESO S.A. | SEVILLA | 7654 | 800000.00 | 0.00 | 1000000.00 |

| 105 | 22478873F | EDICIONES SANZ | BARCELONA | 7499 | NULL | NULL | 500000.00 |

| 106 | 27447854E | SIGNOLOGIC S.A. | MADRID | 7654 | 125.00 | 0.00 | NULL |

| 107 | 27440525F | MARTIN Y ASOCIADOS S.L. | ARAVACA | 7844 | 700000.00 | 0.00 | 1000000.00 |

| 108 | 22456294Z | MANUFACTURAS ALI S.A. | SEVILLA | 7654 | 0.00 | 1400.00 | 500000.00 |

+-----+-----------+-------------------------+-----------+------+-----------+-----------+------------+

8 rows in set (0,00 sec)

No mostrar líneas horizontales ni verticales): **-s** (--silent)

rofesor@ubuntu:~$ **mysql -s -u root -p**

mysql> SELECT \* FROM bdempleados.clientes ;

CLIENTE\_NO NIF NOMBRE LOCALIDAD VENDEDOR\_NO DEBE HABER LIMITE\_CREDITO

101 22427339Q DISTRIBUCIONES GOMEZ MADRID 7499 300000.00 0.00 500000.00

102 27438931T LOGITRONICA S.L BARCELONA 7654 0.00 120000.00 500000.00

103 27465913A INDUSTRIAS LACTEAS S.A. LAS ROZAS 7844 0.00 NULL 1000000.00

104 22918037D TALLERES ESTESO S.A. SEVILLA 7654 800000.00 0.00 1000000.00

105 22478873F EDICIONES SANZ BARCELONA 7499 NULL NULL 500000.00

106 27447854E SIGNOLOGIC S.A. MADRID 7654 125.00 0.00 NULL

107 27440525F MARTIN Y ASOCIADOS S.L. ARAVACA 7844 700000.00 0.00 1000000.00

108 22456294Z MANUFACTURAS ALI S.A. SEVILLA 7654 0.00 1400.00 500000.00

No mostrar ni fila de título ni líneas: **-sN**

**mysql -sN -u root -p**

No mostrar los datos en forma de tabla sino en una sola columna, en este caso escribit \G en el SELECT, no en la conexión a MySQL:

**SELECT \* FROM empleados \G;**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**EMP\_NO: 7499**

**APELLIDO: ALONSO**

**OFICIO: VENDEDOR**

**DIRECTOR: 7698**

**FECHA\_ALTA: 1981-02-20**

**SALARIO: 140000.00**

**COMISION: 40000.00**

**DEP\_NO: 30**

**TELEFONO: 654987412**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**EMP\_NO: 7521**

**APELLIDO: LOPEZ**

**OFICIO: EMPLEADO**

**DIRECTOR: 7782**

**FECHA\_ALTA: 1981-05-08**

**SALARIO: 135000.00**

**COMISION: NULL**

**DEP\_NO: 10**

**TELEFONO: 696547982**

## No puedo conectarme al servidor, posibles motivos

<http://dev.mysql.com/doc/refman/5.7/en/can-not-connect-to-server.html>

## Cambiar el PATH de Windows para incluir los programas de consola de MySQL

Para poder utilizar directamente desde la consola programas como:

mysqldump

mysqladmin

mysql

Es necesario añadir al PATH de Windows la siguiente ruta (escoger según la versión de MySQL):

C:\Program Files\MySQL\MySQL Server 5.5\bin

C:\Program Files\MySQL\MySQL Server 5.6\bin

1. antes de realizar el cambio el PATH tiene este valor:
2. %SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem;%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\MySQL\MySQL Fabric 1.5.2 & MySQL Utilities 1.5.2 1.5\;C:\Program Files (x86)\MySQL\MySQL Fabric 1.5.2 & MySQL Utilities 1.5.2 1.5\Doctrine extensions for PHP\
3. Debemos utilizar un punto y coma (**;)** para separar el texto Viejo del añadido.
4. cambiar el PATH en Windows:

Panel de Control - Sistema - configuración avanzada -variables de entorno -variables del sistema – PATH

# Configuración del servidor MySQL

## Configuración del servidor en Windows

### Carpetas y ficheros utilizados en Windows

<https://dev.mysql.com/doc/refman/5.6/en/windows-installation-layout.html>

<https://dev.mysql.com/doc/refman/8.0/en/windows-installation-layout.html>

Default MySQL Installation Layout for Microsoft Windows.

|  |  |  |
| --- | --- | --- |
| **Directory** | **Contents of Directory** | **Notes** |
| my.ini (fichero) | fichero de configuración del servidor |  |
| bin, scripts | mysqld server, client and utility programs |  |
| %PROGRAMDATA%\MySQL\MySQL Server 5.6\ | Log files, databases, my.ini | The Windows system variable %PROGRAMDATA% defaults to C:\ProgramData |
| examples | Example programs and scripts |  |
| include | Include (header) files |  |
| lib | Libraries |  |
| share | Miscellaneous support files, including error messages, character set files, sample configuration files, SQL for database installation |  |

## Configuración del servidor en Linux

### Mysql 8.0

### Carpetas y ficheros utilizados

<https://dev.mysql.com/doc/refman/8.0/en/binary-installation.html#binary-installation-layout>

#### Archivo de configuración Del servidor

FUENTE:

<https://wiki.cifprodolfoucha.es/index.php?title=Mysql_Instalaci%C3%B3n#Archivo_de_configuraci.C3.B3n>

El archivo de configuración de Mysql es my.cnf, que se encuentra en la carpeta /etc/mysql/. Este archivo es en realidad un acceso directo el cual va a acabar apuntando al archivo mysql.cnf del mismo directorio.

El contenido del archivo de configuración global tiene las siguientes líneas:

!includedir /etc/mysql/conf.d/

!includedir /etc/mysql/mysql.conf.d/

Estas líneas lo que indican es que van a buscar en los dos directorios indicados, todos los archivos de configuración que encuentren. Dichos archivos pueden tener cualquier nombre, pero deben acabar con la extensión .cnf

El archivo de configuración del servicio Mysql (mysql deamon=>mysqld) se encuentra en uno de estos dos sitios:

* O en la carpeta: /etc/mysql/mysql.conf.d/mysqld.cnf y tiene como nombre de sección [mysqld].
* O tiene la sección [mysqld] en el mismo archivo /etc/mysql/mssql.cnf

Podemos ver todas las opciones de configuración en este enlace:

<https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html>

o ejecuntando este comando:

sudo mysqld --verbose --help

Todas esas opciones pueden ser:

* 'escritas' en el archivo de configuración
* pasadas como 'parámetros' por la línea de comandos.

Para poder utilizar esta segunda opción, no podemos iniciar el servicio con la forma tradicional (service mysql start), sino que tenemos que iniciarlo ejecutando el archivo *mysqld* de la forma:

sudo mysqld --port=3308

En este ejemplo, el servicio estaría escuchando en el puerto 3308.

1. Siempre que se modifique el archivo de configuración es conveniente hacer una copia de seguridad del mismo. Dicha copia no debe de tener extensión .cnf si está situado en el directorio /etc/mysql/, ya que el gestor leerá todos los archivos que encuentre con dicha extensión y los 'unirá'.

#### Caso especial: Los directorios

Existen ciertos directorios que son utilizados por el servidor Mysql en su fichero de configuración.

Algunos de ellos son:

pid-file = /var/run/mysqld/mysqld.pid

socket = /var/run/mysqld/mysqld.sock

datadir = /var/lib/mysql

log\_error = /var/log/mysql/error.log

Si queremos cambiar esos directorios por otros, no vale sólo con cambiar las rutas en el archivo de configuración.

Recordar que si necesitamos crear el directorio tenemos que darle los permisos adecuados (como vimos anteriormente), permisos 755 y propieatario 'mysql:mysql'.

Además, la distribución Linux basada en Ubuntu incorpora en el Kernel un sistema de seguridad de acceso a archivos. Este sistema se llama AppArmor.

Su archivo de configuración, para Mysql, se encuentra en el archivo /etc/apparmor.d/usr.sbin.mysqld

Tendremos que editarlo y mirar los permisos que tiene para el directorio que queremos cambiar, y o bien, podemos crear un alias para que el 'viejo' apunte al nuevo directorio o bien copiamos y pegamos las líneas correspondientes cambiando el directorio.

# SQL

## Crear comentarios en ficheros de script: #, /\* \*\*/, --

En un grupo de sentencias SQL se pueden usar comentarios, el texto marcado como comentario es ignorado por el programa y no es ejecutado.

Podemos utilizar los comentarios con diversos cometidos:

* evitar que se ejecuten algunos comandos sin necesidad de borrarlos.
* Añadir comentarios para explicar el funcionamiento.

Tenemos 3 maneras de indicar un comentario:

* **#**: el comentario se extiende desde el símbolo hasta el final de la línea.
* **--** : Debemos dejar un espacio en blanco después de los 2 guiones. El comentario se extiende desde el símbolo hasta el final de la línea.
* **/\* \*/**: Es un comentario de bloque que se puede extender por varias líneas.

Ejemplos:

/\*! MySQL-specific code \*/

SELECT /\*! STRAIGHT\_JOIN \*/ col1 FROM table1,table2 WHERE ...

CREATE TABLE IF NOT EXISTS prueba ( #tabla para probar a los clientes

dni VARCHAR(9),

fecha DATETIME

);

/\*

CREATE TABLE IF NOT EXISTS contrata (

cif\_empresa\_de\_transportes VARCHAR(9) NOT NULL,

cif\_empresa\_cliente VARCHAR(9) NOT NULL,

UNIQUE INDEX (cif\_empresa\_de\_transportes, cif\_empresa\_cliente),

FOREIGN KEY (cif\_empresa\_de\_transportes) REFERENCES empresa\_de\_transportes (cif) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (cif\_empresa\_cliente) REFERENCES empresa\_cliente (cif) ON DELETE CASCADE ON UPDATE CASCADE

);

\*/

## DDL (Data Definition Language)

[**http://dev.mysql.com/doc/refman/5.6/en/sql-syntax-data-definition.html**](http://dev.mysql.com/doc/refman/5.6/en/sql-syntax-data-definition.html)

### CREATE DATABASE

CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] ***db\_name***

[***create\_specification***] ...

***create\_specification***:

[DEFAULT] CHARACTER SET [=] ***charset\_name***

| [DEFAULT] COLLATE [=] ***collation\_name***

IF NOT EXISTS: Evita un error cuando la base de datos ya existe.

No suelen ser necesarios:

CHARACTER SET: como se codifican los caracteres (depende del tipo del alfabeto utilizado por el lenguaje).

COLLATE: criterio para comparar cadenas de texto y decidir cuál va antes.

|  |  |  |
| --- | --- | --- |
| Character | utf8 | Juego de caracteres a utilizar en español, inglés , alemán y otros idiomas latinos o germánicos |
| Collation | latin1\_spanish\_ci | Cómo ordenar las palabras alfabéticamente |

Ejemplos:

CREATE DATABASE IF NOT EXISTS ejemplo1;

CREATE DATABASE ejemplo1;

### DROP DATABASE

DROP {DATABASE | SCHEMA} [IF EXISTS] ***db\_name***

IF EXISTS: Evita un error cuando la base de datos no existe.

### CREATE TABLE

#### Ejemplo de sintaxis básica

Definir Autonumérico, Clave Primaria, No duplicados, Clave Foránea:

CREATE TABLE direcciones (

direccion\_idINT UNSIGNED NOT NULL AUTO\_INCREMENT,

direccion VARCHAR(50) NOT NULL,

localidad VARCHAR(50) DEFAULT NULL,

provincia\_id VARCHAR(30) NOT NULL,

codigo\_postal VARCHAR(10) DEFAULT NULL,

telefono VARCHAR(13) NOT NULL,

fecha\_ultima\_actualizacion TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

PRIMARY KEY (direccion\_id),

UNIQUE KEY(telefono),

CONSTRAINT fk\_direcciones\_provincia\_id

FOREIGN KEY (provincia\_id) REFERENCES provincias (provincia\_id)

ON DELETE RESTRICT

ON UPDATE CASCADE

);

#

#### COMENTARIOS: de columna, índice, tabla

Existen varios tipos de comentarios: de columna (1024), índice (1024), tabla (2048).

<https://dev.mysql.com/doc/refman/5.7/en/create-table.html>

Entre paréntesis se indica la longitud máxima

A comment for a column can be specified with the COMMENT option, up to 1024 characters long. The comment is displayed by the SHOW CREATE TABLE and SHOW FULL COLUMNS statements.

CREATE TABLE direcciones (

direccion\_idINT UNSIGNED NOT NULL AUTO\_INCREMENT COMMENT 'Comentaraio de columna: atributo autonumérico' ,

direccion VARCHAR(50) NOT NULL,

telefono VARCHAR(13) NOT NULL,

PRIMARY KEY (direccion\_id),

UNIQUE (telefono) COMMENT 'Comentario de índice: atributo que no admite duplicados',

)

COMMENT 'Comentario de tabla: Esta tabla contiene las direcciones de los clientes'

;

Ver los comentarios:

#todos los comentarios:

SHOW CREATE TABLE direcciones;

#sólo los de tabla:

SHOW TABLE STATUS WHERE Name = 'direcciones';

#### Crear tabla que sea copia de otra e incluya la PRIMARY KEY y los índices (KEY)

CREATE TABLE pedidos\_viejos LIKE pedidos;

Al usar LIKE se crea una nueva tabla vacía, basada en la definición de la vieja:

* + incluyendo la PRIMARY KEY y los índices (KEY).
  + No incluye las FOREIGN KEY

Después se pueden insertar filas en la nueva tabla usando filas de la vieja:

INSERT INTO pedidos\_viejos SELECT \* FROM pedidos WHERE YEAR(fecha\_pedido) < 2014;

No se pueden hacer ambas operaciones (creación tabla, inserción datos) en una única instrucción.

1. Ejemplo

CREATE TABLE **`pedidos`** (

`PEDIDO\_NO` smallint(6) NOT NULL,

`PRODUCTO\_NO` smallint(6) DEFAULT NULL,

`CLIENTE\_NO` smallint(5) DEFAULT NULL,

`UNIDADES` decimal(8,2) DEFAULT NULL,

`FECHA\_PEDIDO` datetime DEFAULT NULL,

PRIMARY KEY (`PEDIDO\_NO`),

KEY `FK\_PEDIDOS\_CLIENTE\_NO` (`CLIENTE\_NO`),

KEY `FK\_PEDIDOS\_PRODUCTO\_NO` (`PRODUCTO\_NO`),

CONSTRAINT `FK\_PEDIDOS\_CLIENTE\_NO` FOREIGN KEY (`CLIENTE\_NO`) REFERENCES `clientes` (`CLIENTE\_NO`) ON UPDATE CASCADE,

CONSTRAINT `FK\_PEDIDOS\_PRODUCTO\_NO` FOREIGN KEY (`PRODUCTO\_NO`) REFERENCES `productos` (`PRODUCTO\_NO`) ON DELETE SET NULL ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE **`pedidos\_viejos`** (

`PEDIDO\_NO` smallint(6) NOT NULL,

`PRODUCTO\_NO` smallint(6) DEFAULT NULL,

`CLIENTE\_NO` smallint(5) DEFAULT NULL,

`UNIDADES` decimal(8,2) DEFAULT NULL,

`FECHA\_PEDIDO` datetime DEFAULT NULL,

PRIMARY KEY (`PEDIDO\_NO`),

KEY `FK\_PEDIDOS\_CLIENTE\_NO` (`CLIENTE\_NO`),

KEY `FK\_PEDIDOS\_PRODUCTO\_NO` (`PRODUCTO\_NO`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

1. KEY es sinónimo de INDEX

#### Crear tabla que sea copia de otra, pero que no incluya las KEYs

<http://dev.mysql.com/doc/refman/5.7/en/create-table.html>

CREATE TABLE IF NOT EXISTS empleadosTemporal

(SELECT \* FROM empleados);

CREATE TABLE IF NOT EXISTS empleadosTemporal

AS (SELECT \* FROM empleados);

La tabla se crea en la BD que esté activa en el servidor según estos criterios:

* Sí conserva los mismos atributos (nombre y tipo) y datos que la tabla original.
* No conserva ninguna de las KEYs de la table original (PRIMARY KEY, UNIQUE KEY, FOREIGN KEY, índice).
* añade a la nueva tabla los datos devueltos por la subconsulta.

Ejemplo:

CREATE TABLE IF NOT EXISTS *empleadosTemporal*  
(SELECT \* FROM empleados);

**#estructura de la tabla*empleadosTemporal*:**

CREATE TABLE `empleadostemporal` (

`EMP\_NO` smallint(4) NOT NULL,

`DNI` varchar(9) NOT NULL,

`NOMBRE` varchar(65) NOT NULL,

`APELLIDO1` varchar(65) NOT NULL,

`APELLIDO2` varchar(65) DEFAULT NULL,

`OFICIO` varchar(65) DEFAULT NULL,

`JEFE` smallint(4) DEFAULT NULL,

`FECHA\_ALTA` date DEFAULT NULL,

`SALARIO` decimal(8,2) DEFAULT NULL,

`COMISION` decimal(8,2) DEFAULT NULL,

`DEP\_NO` smallint(4) DEFAULT NULL,

`TELEFONO` varchar(15) DEFAULT NULL

);

#### Especificar el tipo de almacenamiento de la tabla como InnoDB

En nuestras BD siempre usaremos el tipo de almacenamiento InnoDB por tener actualmente las mejores propiedades, antiguamente sólo existían otros tipos de almacenamiento que adolecían de ciertas limitaciones.

No es necesario indicar explícitamente que la nueva tabla utilice el almacemaniento InnoDB , porque por defecto se ulitiza InnoDB.

De todas maneras, si deseamos especificarlo se hace así:

CREATE TABLE IF NOT EXISTS `departamentos` (

`DEP\_NO` SMALLINT(4) NOT NULL,

`DNOMBRE` VARCHAR(20) NOT NULL,

`LOCALIDAD` VARCHAR(20) ,

PRIMARY KEY (`DEP\_NO`)

) ENGINE = InnoDB;

Es equivalente a no especificarlo (gracias a que se le asigna el tipo InnoDB por defecto):

CREATE TABLE IF NOT EXISTS `departamentos` (

`DEP\_NO` SMALLINT(4) NOT NULL,

`DNOMBRE` VARCHAR(20) NOT NULL,

`LOCALIDAD` VARCHAR(20) ,

PRIMARY KEY (`DEP\_NO`)

);

#### CREATE TEMPORARY TABLE: Crear una Tabla temporal en memoria RAM con los datos de un SELECT

<http://dev.mysql.com/doc/refman/5.7/en/create-table.html>

CREATE TEMPORARY TABLE [IF NOT EXISTS] tbl\_name

Una tabla temporal sólo es visible para la sesión actual y se borra automáticamente al cerrarse la sesión que la crea.

You can use the TEMPORARY keyword when creating a table. A TEMPORARY table is visible only to the current session, and is dropped automatically when the session is closed. This means that two different sessions can use the same temporary table name without conflicting with each other or with an existing non-TEMPORARY table of the same name. (The existing table is hidden until the temporary table is dropped.) To create temporary tables, you must have the CREATE TEMPORARY TABLES privilege.

* + CREATE TABLE does not automatically commit the current active transaction if you use the TEMPORARY keyword.
  + TEMPORARY tables have a very loose relationship with databases (schemas). Dropping a database does not automatically drop any TEMPORARY tables created within that database.
* Also, you can create a TEMPORARY table in a nonexistent database if you qualify the table name with the database name in the CREATE TABLE statement. In this case, all subsequent references to the table must be qualified with the database name.

1. Ejemplos

CREATE TEMPORARY TABLE IF NOT EXISTS empleadosTemporal

(SELECT \* FROM empleados);

CREATE TEMPORARY TABLE IF NOT EXISTS empleadosTemporal

AS (SELECT \* FROM empleados);

#### SERIAL, AUTO\_INCREMENT

**SERIAL** es un alias de **BIGINT UNSIGNED NOT NULL AUTO\_INCREMENT**.

Al introducir una nueva fila en la tabla no podemos especificar ningún valor para los atributos que tienen la propiedad **AUTO\_INCREMENT**, el valor es asignado automáticamente por el SGBD.

Cuidado porque si un atributo es SERIAL, la clave ajena asociada en otra tabla tiene que ser **BIGINT UNSIGNED**

CREATE TABLE Persona

(

IdPersona SERIAL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),

PRIMARY KEY (ID)

)

CREATE TABLE Coche

(

IdPersona BIGINT UNSIGNED,

matriculaCoche varchar(7) NOT NULL

)

Si un atributo tiene un atributo SERIAL, hay que insertar datos tal y como se indica en[Insertar una fila en una tabla que incluye un atributo AUTO\_INCREMENT](#_Insertar_una_fila)

#### ON DELETE/UPDATE[RESTRICT | CASCADE | SET NULL | NO ACTION]

**<http://dev.mysql.com/doc/refman/5.6/en/create-table-foreign-keys.html>**

* **CASCADE**: Cuando se Borra o actualiza un registro en la tabla padre, automáticamente se elimina o actualizan las filas coincidentes en la tabla de la clave ajena. ON DELETE CASCADE y ON UPDATE CASCADE son compatibles.

1. Actualmente, CASCADE no activa TRIGGERS

* **SET NULL**: Cuando se Borra o actualiza un registro en la tabla padre, se establece la columna de clave ajena a NULL . ON DELETE SET NULL y ON UPDATE SET NULL son compatibles.

1. **CUIDADO**: Si se especifica un **SET NULL**en la clave ajena, entonces no se puede usar la restricción **NOT NULL** en los atributos de esa clave ajena.

* **RESTRICT**: Rechaza la operación de eliminación o actualización en la tabla padre cuando existe un registro relacionado en la tabla secundaria. Especificar RESTRICT (o NO ACTION ) es lo mismo que omitirON DELETE u ON UPDATE. Es preferible usar NO ACTION porque Workbench 6.2 y 6.3 no muestran correcamente la restricción en el diagrama gráfico si hacemos ingenería inversa sobre una BD.
* **NO ACTION**: Una palabra clave de SQL estándar. En MySQL, equivalente a RESTRICT .

1. Cuando no se especific a ON DELETE o ON UPDATE, la acción predeterminada es RESTRICT .

#### Índices: PRIMARY KEY, UNIQUE , INDEX

##### **Orden de Colocación** en CREATE TABLE de los distintos índices:

* Primero se coloca la restricción **PRIMARY KEY**
* Después se colocan los índices **UNIQUE**
* Por último se colocan los índices **INDEX**

1. This helps the MySQL optimizer to prioritize which index to use and also more quickly to detect duplicated UNIQUE keys.

##### PRIMARY KEY

###### 3 maneras distintas de definir la PRIMARY KEY:

CREATE TABLE actor (

actor\_id SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,

first\_name VARCHAR(45) NOT NULL,

last\_name VARCHAR(45) NOT NULL,

PRIMARY KEY (actor\_id)

);

CREATE TABLE actor (

actor\_id SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(45) NOT NULL,

last\_name VARCHAR(45) NOT NULL

);

CREATE TABLE actor (

actor\_id SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,

first\_name VARCHAR(45) NOT NULL,

last\_name VARCHAR(45) NOT NULL,

CONSTRAINT pk\_actor\_id PRIMARY KEY (actor\_id)

);

* + Una tabla puede tener **ninguna o una sola** PRIMARY KEY.
  + PRIMARY KEY lleva implícita la restricción UNIQUE.
  + PRIMARY KEY lleva implícita la restricción NOT NULL, pero deberíamos escribir explícitamente el NOT NULL para evitar inconvenientes, **por los siguientes motivos relacionados con el valor por defecto de un atributo**:
* Valor por defecto de un atributo

<http://dev.mysql.com/doc/refman/5.5/en/data-type-defaults.html>

* + Si insertamos una fila y no damos valor al atributo PRIMARY KEY, entonces MySQL **SÍ** le asigna al atributo PRIMARY KEY el [valor por defecto propio de su tipo de datos](#_Valor_por_defecto). Evidentemente si ese valor por defecto ya existiese previamente en otro registro, no podrá introducirse la fila (porque PRIMARY KEY no permite valores duplicados de la clave).

Veamos en un ejemplo, por qué debemos poner explícitamente NOT NULL a un atributo que forme parte de la PRIMARY KEY:

#El comando **SHOW CREATE TABLE** muestra la estructura real de la tabla (con las cláusulas que añade MySQL de forma implícita).

-- Ejemplo 1: tabla creada correctamente (especificando NOT NULL en el atributo que es PRIMARY KEY):

**-- SÍ hemos especificado NOT NULL en el atributo de la PRIMARY KEY**

CREATE TABLE t (i INT NOT NULL,

PRIMARY KEY (i)

);

SHOW CREATE TABLE t;

CREATE TABLE `t` (

`i` int(11) NOT NULL,

PRIMARY KEY (`i`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8

-- Ejemplo 2: la misma tabla creada INcorrectamente (NO especificando NOT NULL en el atributo que es PRIMARY KEY):

**-- NO hemos especificado NOT NULL en el atributo de la PRIMARY KEY**

**CREATE TABLE t** (i INT ,

PRIMARY KEY (i)

);

**SHOW CREATE TABLE t;**

CREATE TABLE `t` (

`i` int(11) NOT NULL **DEFAULT '0'**,

PRIMARY KEY (`i`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8

##### UNIQUE [INDEX|KEY]

[**http://www.w3schools.com/sql/sql\_unique.asp**](http://www.w3schools.com/sql/sql_unique.asp)

* + UNIQUE Crea una restricción que impide que el atributo (o atributos) se repita en varias filas.
  + UNIQUE **SÍ** permite que el atributo sea NULL en varias filas (en caso de que no exista una restricción NOT NULL que impida los nulos)
  + La restricción PRIMARY KEY lleva implícita la restricción UNIQUE y NOT NULL.
  + Se pueden tener varias restricciones UNIQUE en una tabla pero una sola restricción de tipo PRIMARY KEY
  + Si introducimos un registro y no damos valor al atributo UNIQUE, MySQL **NO** le asigna al atributo UNIQUE el [valor por defecto propio de su tipo de datos](#_Valor_por_defecto) (a diferencia de lo que sucede con PRIMARY KEY).

Comandos para añadir restricción UNIQUE, tenemos varias posibilidades:

SQL Server / Oracle / MS Access (también MySQL):

CREATE TABLE Persons

(

P\_Id int NOT NULL UNIQUE,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255)

)

MySQL:

CREATE TABLE Persons

(

P\_Id int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),

UNIQUE (P\_Id)

)

MySQL / SQL Server / Oracle / MS Access:

CREATE TABLE Persons

(

P\_Id int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),

CONSTRAINT uc\_PersonID UNIQUE (P\_Id,LastName)

)

MySQL / SQL Server / Oracle / MS Access:

ALTER TABLE Persons

ADD UNIQUE (P\_Id)

MySQL / SQL Server / Oracle / MS Access:

ALTER TABLE Persons

ADD CONSTRAINT uc\_PersonID UNIQUE (P\_Id,LastName)

MySQL:

ALTER TABLE Persons

DROP INDEX uc\_PersonID

SQL Server / Oracle / MS Access:

ALTER TABLE Persons

DROP CONSTRAINT uc\_PersonID

MySQL:

-- crear un índice sobre el atributo nombre

**CREATE INDEX index\_nombre ON clientes (nombre);**

--crear un índice sobre el atributo nombre, pero solamente utilizar los 10 primeros caracteres del nombre para formar el índice

**CREATE INDEX index\_nombre ON clientes (nombre(10));**

##### INDEX|KEY

<http://www.dimensis.com/consejos-1.html>

<http://stackoverflow.com/questions/3844899/difference-between-key-primary-key-unique-key-and-index-in-mysql>

<https://dev.mysql.com/doc/refman/5.6/en/optimization-indexes.html>

1. **KEY** e **INDEX** son sinónimos.
2. Los índices (INDEX) son usados para encontrar rápidamente los registros que tengan un determinado valor en alguna de sus columnas. Sin un índice, MySQL tiene que iniciar con el primer registro y leer a través de toda la tabla para encontrar los registros relevantes. Aún en tablas pequeñas, de unos 1000 registros, es por lo menos 100 veces más rápido leer los datos usando un índice, que haciendo una lectura secuencial.
3. Cuando indexamos una columna en particular, MySQL crea otra estructura de datos (un índice) que usa para almacenar información extra acerca de los valores en la columna indexada.
4. MySQL almacena todas las claves del índice en una estructura de datos de árbol. Esta estructura de datos de árbol le permite a MySQL encontrar claves muy rápidamente.
5. Cuando MySQL encuentre que hay un índice en una columna, lo usará en vez de hacer un escaneo completo de la tabla. Esto reduce de manera imporante los tiempos de CPU y las operaciones de entrada/salida en disco, a su vez que se mejora la concurrencia porque MySQL bloqueará la tabla únicamente para obtener las filas que necesite (en base a lo que encontró en el índice).
6. Cuando tenemos grandes cantidades de datos en nuestras tablas, la mejora en la obtención de los datos puede ser muy significativa.
7. A PRIMARY KEY is a unique index where all key columns must be defined as NOT NULL. If theyare not explicitly declared as NOT NULL, MySQL declares them so implicitly (and silently). A table can have only one PRIMARY KEY. The name of a PRIMARY KEY is always PRIMARY, which thus cannot be used as the name for any other kind of index.
8. a UNIQUE index permits multiple NULL values for columns that can contain NULL.
9. Un índice puede incluir hasta 16 atributos.
10. MySQL can use multiple-column indexes for queries that test all the columns in the index, or queries that test just the first column, the first two columns, the first three columns, and so on. If you specify the columns in the right order in the index definition, a single composite index can speed up several kinds of queries on the same table.

<https://dev.mysql.com/doc/refman/5.6/en/multiple-column-indexes.html>

Suppose that a table has the following specification:

CREATE TABLE test (

id INT NOT NULL,

last\_name CHAR(30) NOT NULL,

first\_name CHAR(30) NOT NULL,

PRIMARY KEY (id),

INDEX name (last\_name,first\_name)

);

The name index is an index over the last\_name and first\_name columns. The index can be used for lookups in queries that specify values in a known range for combinations of last\_name and first\_name values. It can also be used for queries that specify just a last\_name value because that column is a leftmost prefix of the index (as described later in this section).

Therefore, **the name index is used for lookups in the following queries**:

SELECT \* FROM test WHERE last\_name='Widenius';

SELECT \* FROM test

WHERE last\_name='Widenius' AND first\_name='Michael';

SELECT \* FROM test

WHERE last\_name='Widenius'

AND (first\_name='Michael' OR first\_name='Monty');

SELECT \* FROM test

WHERE last\_name='Widenius'

AND first\_name >='M' AND first\_name < 'N';

**However, the name index is not used for lookups in the following queries:**

SELECT \* FROM test WHERE first\_name='Michael';

SELECT \* FROM test

WHERE last\_name='Widenius' OR first\_name='Michael'

1. Cuando se crea un índice para un atributo BLOB o TEXT, es necesario especificar cuantos caracteres iniciales del atributo se utilizan para crear el índice (máximo 676 bytes). Ejemplo:

CREATE TABLE test (blob\_col BLOB, INDEX(blob\_col(10)));

1. Prefix limits are measured in bytes, while the prefix length in **CREATE TABLE**statements is interpreted as number of characters. Take this into account when specifying a prefix length for a column that uses a multibyte character set.
2. Ventajas:

* Concurrencia: mientras MySQL está leyendo datos de una tabla, esta queda bloqueada para escritura, aunque sí permite lecturas concurrentes. Si la consulta es más rápida entonces el bloqueo es más breve.
* Disco duro: cuantos menos datos se lean del disco duro menos cargamos el acceso al HD y puede hacer más tareas.
* CPU: analizar cada uno de los registros de una tabla da más trabajo que consultar el atributo de un índice. Cuanto menos cargamos la CPU más tareas podrá realizar servidor.

1. Inconvenientes

A pesar de que los índices aceleran la consulta (SELECT) de datos, tienen como inconveniente el que ralentizan las operaciones de inserción, borrado y modificación de datos:

Cada vez que insertamos, borramos un registro de la tabla, el SGBD debe modificar también todos los índices de esta tabla.

Si modificamos un atributo de un registro y ese atributo forma parte de un índice, el SGBD debe modificar ese índice.

1. En qué atributos crear índices:

* atributos utilizados en JOIN, WHERE, ORDER BY

1. Ejemplos:

CREATE TABLE IF NOT EXISTS autobus (

matricula VARCHAR(9) NOT NULL,

cif\_empresa\_propietaria VARCHAR(9) NOT NULL,

dni\_conductor VARCHAR(9),

PRIMARY KEY (matricula),

INDEX indice\_conductor (dni\_conductor)

);

#### FOREIGN KEY

##### Errores típicos al crear una FOREIGN KEY

Ejemplo de FOREIGN KEY correctas:

# Foreign Key formada por 1 atributo:

CONSTRAINT FK\_CAMA\_NUMAND FOREIGN KEY (Numero\_Andar) REFERENCES Andar (Numero) ON UPDATE CASCADE ON DELETE NO ACTION

# Foreign Key formada por 2 atributos:

CONSTRAINT fk\_doente\_numcama\_numplanta FOREIGN KEY (NUMERO\_CAMA, NUMERO\_PLANTA) REFERENCES CAMAS (NUMERO\_CAMA, NUMERO\_PLANTA) ON UPDATE CASCADE ON DELETE NO ACTION

1. No es obligatorio poner NOT NULL a un atributo que es FOREIGN KEY; se pondrá NOT NULL si deseamos impedir se pueda dejar el atributo en blanco y que por tanto tenga que casarse obligatoriamente con una fila de la tabla principal.
2. El orden es indiferente en:

#correcto:

ON UPDATE ... ON DELETE ...

#correcto:

ON DELETE ...ON UPDATE ...

1. La FOREIGN KEY debe apuntar a una PRIMARY KEY o a un UNIQUE en la tabla referenciada
2. Los tipos de dato en ambas tablas deben de ser similares:

* Los datos de tipo numérico deben ser del mismo tamaño y signo.
* La longitud de los tipos texto no tiene por qué ser la misma, aunque es recomendable.

1. Si ponemos ON DELETE SET NULL, el atributo no puede tener la restricción NOT NULL
2. Si la PRIMARY KEY de la tabla referenciada es SERIAL, el atributo que lo referencia debe de ser BIGINT UNSIGNED
3. Si La FOREIGN KEY está formada por 2 o más atributos, se crea una sola restricción con esos atributos, no varias restricciones:

# Foreign Key formada por 2 atributos:

CONSTRAINT fk\_doente\_numcama\_numplanta FOREIGN KEY (NUMERO\_CAMA, NUMERO\_PLANTA) REFERENCES CAMAS (NUMERO\_CAMA, NUMERO\_PLANTA) ON UPDATE CASCADE ON DELETE NO ACTION

1. MySQL requires indexes on foreign keys and referenced keys so that foreign key checks can be fast and not require a table scan. In the referencing table, there must be an index where the foreign key columns are listed as the first columns in the same order (el índice debe incluir los atributos de la FOREIGN KEY como sus prmieros atributos, y en el mismo orden, pero podría incluir atributos extra a su izquierda). Such an index is created on the referencing table automatically if it does not exist. This index might be silently dropped later, if you create another index that can be used to enforce the foreign key constraint. *index\_name*, if given, is used as described previously.

##### Ejemplos correctos e incorrectos de FOREIGN KEY que provienen de PRIMARY KEY formada por más de un atributo

##### Ejemplos correctos e incorrectos de FOREIGN KEY que provienden de PRIMARY KEY formada por más de un atributo

Si la PRIMARY KEY está formada por más de un atributo, entonces la FOREIGN KEY hay que crearla con una única cláusula FOREIGN KEY que debe traer juntos todos los atributos (y en el mismo orden en que se encuentran en la PRIMARY KEY).

Veamos este ejemplo:

**#dada la tabla A, ¿cuáles de las tablas A1, A2, A3 crean bien la FOREIGN KEY?. Respuesta: A2**

**CREATE TABLE A(**

**a INTEGER,**

**b INTEGER,**

**c INTEGER,**

**PRIMARY KEY (a,b));**

**CREATE TABLE A1(**

**a INTEGER,**

**b INTEGER,**

**d VARCHAR(33),**

**PRIMARY KEY (a,b,d),**

**FOREIGN KEY AA\_a (a) REFERENCES A(a) ON UPDATE CASCADE ON DELETE CASCADE,**

**FOREIGN KEY AA\_b (b) REFERENCES A(b) ON UPDATE CASCADE ON DELETE CASCADE);**

**CREATE TABLE A2(**

**a INTEGER,**

**b INTEGER,**

**d VARCHAR(33),**

**PRIMARY KEY (a,b,d),**

**FOREIGN KEY AA\_ab (a,b) REFERENCES A(a,b) ON UPDATE CASCADE ON DELETE CASCADE**

**);**

**CREATE TABLE A3(**

**a INTEGER,**

**b INTEGER,**

**d VARCHAR(33),**

**PRIMARY KEY (a,b,d),**

**FOREIGN KEY AA\_ba (b,a) REFERENCES A(b,a) ON UPDATE CASCADE ON DELETE CASCADE**

**);**

#### NULL / NOT NULL

**NULL** indica que un atributo puede tomar valores nulos.

Cuando de un atributo no se especifica si puede o no puede tomar valores nulos, entonces **por defecto** MySQL considera que puede tomar valores nulos:

**# son equivalentes**

**dni VARCHAR(9) NULL**

**dni VARCHAR(9)**

**NOT NULL** indica que un atributo NO puede tomar valores nulos:

**dni VARCHAR(9) NOT NULL**

#### DEFAULT

<http://dev.mysql.com/doc/refman/5.5/en/data-type-defaults.html>

DEFAULT indica el valor por defecto de un atributo. Es el valor que le asigna automáticamente MySQL a un atributo cuando nosotros introducimos un nuevo registro y no indicamos valor para ese atributo.

El valor por defecto de un atributo tiene que ser una constante (con la única excepción de los TIMESTAMP que admiten como valor por defecto CURRENT\_TIMESTAMP -fecha y hora del instante en que se introduce el registro-).

1. Para asignar por defecto la fecha actual a una variable de tipo fecha es necesario crear un TRIGGER.
2. Para asignar por defecto la hora actual a una variable de tipo hora es necesario crear un TRIGGER.

Ejemplo:

provincia VARCHAR(9) DEFAULT 'A CORUÑA',

numero\_de\_hijos TINYINT DEFAULT 0,

fecha DATETIME DEFAULT CURRENT\_TIMESTAMP,

salario DOUBLE()DOUBLE DEFAULT NULL,

salario DOUBLE()DOUBLE,

Si no se especifica un valor DEFAULT para un atributo entonces MySQL determina su **valor por defecto** del siguiente modo:

* Si el atributo **SÍ admite el valor NULL**: entonces se le asigna implícitamente una cláusula DEFAULT NULL.
* Si el atributo **NO admite el valor NULL**: entonces no se le asigna implícitamente una cláusula DEFAULT, y lo que sucederá al realizar un INSERT, REPLACE o UPDATE depende del modo activo de SQL:
* **Modo SQL estricto activado y tabla transaccional**: se produce un error y se produce un ROLL BACK de la sentencia problemática.
* **Modo SQL estricto activado y tabla NO transaccional**:se produce un error, pero si estamos realizando una tarea que afecta a más de una fila, la operación se realizará correctamente para todas las filas excepto la última.
* **Modo SQL estricto desactivado**: MySQL asigna al atributo el valor implícito de su tipo de dato.
* **Clave primaria**: Si un atributo forma parte de una clave primaria y no se indica explícitamente la cláusula NOT NULL, entonces MySQL le asigna implícitamente la cláusula NOT NULL (porque todos los atributos de una PRIMARY KEY tienen que ser NOT NULL) y además le asigna a ese atributo el [valor implícito por defecto de su tipo de dato](#_Valor_implícito_por); si queremos evitar esto, debemos incluir la cláusula NOT NULL en el atributo que forma parte de la clave primaria.

For data entry into a NOT NULL column that has no explicit DEFAULT clause, if an INSERT or REPLACE statement includes no value for the column, or an UPDATE statement sets the column to NULL, MySQL handles the column according to the SQL mode in effect at the time:

* If **strict SQL mode is enabled**, an error occurs for transactional tables and the statement is rolled back. For non-transactional tables, an error occurs, but if this happens for the second or subsequent row of a multiple-row statement, the preceding rows will have been inserted.
* If **strict mode is not enabled**, MySQL sets the column to the implicit default value for the column data type.

(ver apartado modos de MySQL: modo estricto y otros)

BLOB y TEXT no admiten DEFAULT.

##### Valor implícito por defecto de cada tipo de dato.

dev.mysql.com/doc/refman/5.0/en/data-type-defaults.html

Cuando MySQL debe asignar un valor por defecto a un atributo, el valor asignado depende del tipo de atributo.

El valor por defecto para cada tipo de atributo es el siguiente:

|  |  |
| --- | --- |
| **TIPO DE DATOS** | **VALOR POR DEFECTO PARA ESE TIPO DE DATOS** |
| Numéricos: | 0 |
| Texto: | cadena de texto vacía (se representa como 2 comillas simples seguidas):  '' |
| Fecha, hora (excepto TIMESTAMP) | 0, (codificado en forma de fecha u hora) |
| TIMESTAMP | *fecha y hora actuales*. |

##### DATETIME, DATE, TIME, TIMESTAMP: asignar valores DEFAULT

1. DATETIME

Para asignar por defecto la fecha y hora actuales a un atributo DATETIME:

fecha DATETIME DEFAULT CURRENT\_TIMESTAMP,

1. DATE

Asignar valor por defecto a DATE:

1. DATA y TIME necesitan un trigger, no admiten un método tan fácil como DATETIME

# borro el trigger si existe.

DROP TRIGGER IF EXISTS trigger\_muestra\_before;

DROP TRIGGER IF EXISTS ejemplo01.trigger\_muestra\_before; # puedo hacer referencia al trigger usando el schema.

DELIMITER//

CREATE TRIGGER trigger\_muestra\_before BEFORE INSERT ON muestra

FOR EACH ROW

BEGIN

IF ( ISNULL(new.dia) ) THEN

SET new.dia=CURDATE();

END IF;

END;

//

DELIMITER ;

1. TIME

Asignar valor por defecto a TIME:

# borro el trigger si existe.

DROP TRIGGER IF EXISTS trigger\_muestra\_before;

DELIMITER//

CREATE TRIGGER trigger\_muestra\_before BEFORE INSERT ON muestra

FOR EACH ROW

BEGIN

IF ( ISNULL(new.hora) ) THEN

SET new.hora=CURTIME();

END IF;

END;

//

DELIMITER;

1. DATE y TIME

Asignar valor por defecto a DATE y TIME: Hay que meterlos en el mismo trigger

1. Motivo: Before MySQL 5.7.2, there cannot be multiple triggers for a given table that have the same trigger event and action time. For example, you cannot have two BEFORE UPDATE triggers for a table. To work around this, you can define a trigger that executes multiple statements by using the BEGIN ... END compound statement construct after FOR EACH ROW. (An example appears later in this section.)

# borro el trigger si existe.

DROP TRIGGER IF EXISTS trigger\_muestra\_before;

DELIMITER//

CREATE TRIGGER trigger\_muestra\_before BEFORE INSERT ON muestra

FOR EACH ROW

BEGIN

IF ( ISNULL(new.dia) ) THEN

SET new.dia=CURDATE();

END IF;

IF ( ISNULL(new.hora) ) THEN

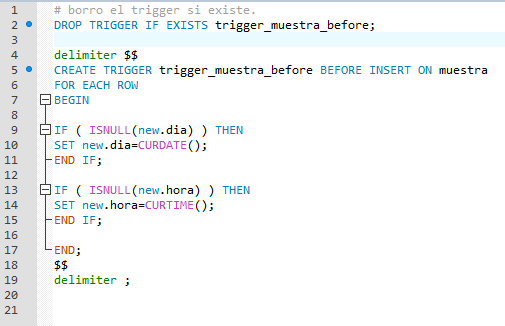
SET new.hora=CURTIME();

END IF;

END;

//

DELIMITER ;



1. TIMESTAMP

Asignar valor por defecto:

fechaYHora\_de\_entrada TIMESTAMP DEFAULT NOW(),

#### AUTO\_INCREMENT = 77

fuerza a que el valor inicial de un atributo de tipo auto\_increment sea 100

ALTER TABLE Personas AUTO\_INCREMENT=77

#### CHECK

<https://dev.mysql.com/doc/refman/8.0/en/create-table-check-constraints.html>

<https://mysqlserverteam.com/mysql-8-0-16-introducing-check-constraint/>

<https://www.w3resource.com/mysql/creating-table-advance/constraint.php>

Antes de MySQL 8.0.16 se permitía escribir restricciones mediante CHECK para ser compatible con el estándar SQL, pero MySQL **las ignoraba**, por tanto no tenía utilidad especificarlas. En esos casos podíamos intentar crear un **TRIGGER para realizar la función del CHECK**..

Desde la versión MySQL 8.0.16 los CHECK ya son ejecutados por MySQL.

Formato de la restricción CHECK:

[CONSTRAINT [symbol]] CHECK (expr) [[NOT] ENFORCED]

A CHECK constraint is specified as either a table constraint or column constraint:

* + A table constraint does not appear within a column definition and can refer to any table column or columns. Forward references are permitted to columns appearing later in the table definition.
  + A column constraint appears within a column definition and can refer only to that column.
* Ejemplos:

Ejemplo de CHECK usado en el SGBD MariaDB (que sí las permite):

ALTER TABLE Picnics ADD CONSTRAINT constraint\_1

CHECK (EXISTS (SELECT \* FROM Accounts WHERE balance > 0));

SALARIO DECIMAL(8,2) CHECK (SALARIO > 0),

SALARIO DECIMAL(8,2),

COMISION DECIMAL(8,2) CHECK (SALARIO > COMISION),

COMISION DECIMAL(8,2),

CHECK (SALARIO > COMISION),

DEP\_NO SMALLINT(4),

country varchar(25) NOT NULL CHECK (country IN ('USA','UK','India')),

CodigoPostal VARCHAR(5) CHECK (CodigoPostal LIKE '\_\_\_\_\_'),

FECHA\_NACIMIENTO date ,

CHECK ((PAIS='Portugal' AND CIUDAD='Porto')

OR (PAIS='España' AND CIUDAD='Ferrol')) ,

### Comandos para añadir, modificar, borrar CHECKS

ALTER TABLE Persons

ADD CHECK (Age>=18);

ALTER TABLE Persons

ADD CONSTRAINT CHK\_PersonAge CHECK (Age>=18 AND City='Sandnes');

ALTER TABLE Persons

DROP CHECK CHK\_PersonAge;

#### CHECK TABLE

CHECK TABLE comprueba si una tabla o vista tienen errores (por ejemplo una vista puede tener errores porque se ha borrado una tabla utilizada en esa vista).

CHECK TABLE empleados;

#### CREAR UNA COPIA DE UNA TABLA

CREATE TABLE empleadosb LIKE empleados;

Copia atributos,Primary Key, INDEX

No copia Foreign Key, datos.

### DROP TABLE

DROP [TEMPORARY] TABLE [IF EXISTS]

***tbl\_name*** [, ***tbl\_name***] ...

[RESTRICT | CASCADE]

### Copiar una BD / Cambiar el nombre de una base de datos (importar una BD desde un script SQL), hacer backup de una BD

<http://stackoverflow.com/questions/67093/how-do-i-quickly-rename-a-mysql-database-change-schema-name>

<http://dev.mysql.com/doc/sakila/en/sakila-installation.html>

Para **cambiar** el nombre de una BD primero hemos de crear una copia de la BD que tenga otro nombre y después ya borraremos la vieja.

1. Comandos para crear una copia ****junto con los datos**** de una BD:

Para **crear la copia** de la BD:

* indicamos a MySQL que cree un fichero de comandos SQL que permitan crear una base de datos igual a la existente, poblándola además con los datos.
* Usamos el fichero sql anterior para crear la BD.
  + Comandos a usar para crear una copia de una BD:

**-- Guarar los comandos que permiten crear la BD en un fichero sql llamado BDEmpleadosBACKUP.sql**

**-- La BD original se llama bdempleados**

**mysqldump -u root -p -v bdempleados > BDEmpleadosBACKUP.sql**

**-- Crear la nueva BD a mano llamada gestionsimple**

**mysqladmin -u root -p create gestionsimple**

**-- importar la estructura y los datos a la nueva BD usando el fichero sql**

**mysql -u root -p gestionsimple < DEmpleadosBACKUP.sql**

#Actualizar los permisos de los usuarios sobre la nueva BD, para ello copiamos los permisos que ya tenían en la BD vieja:

**mysql -u root -pAbcd1234. -e "UPDATE mysql.db SET Db = 'gestionsimple' WHERE Db = 'bdempleados'"**

**mysql -u root -pAbcd1234. -e "UPDATE mysql.tables\_priv SET Db = 'gestionsimple' WHERE Db = 'bdempleados';"**

**mysql -u root -pAbcd1234. -e "UPDATE mysql.columns\_priv SET Db = 'gestionsimple' WHERE Db = 'bdempleados';"**

**mysql -u root -pAbcd1234. -e "UPDATE mysql.procs\_priv SET Db = 'gestionsimple' WHERE Db = 'bdempleados';"**

#Borrar la BD vieja porque ahora usaremos la que tiene el nuevo nombre (queríamos renombrarla, no mantener 2 copias):

**mysql -u root -pAbcd1234. -e "DROP DATABASE IF EXISTS bdempleados;"**

si lo que deseábamos era cambiar el nombre de la base de datos, ahora borraremos la base de datos vieja

1. Comandos para crear una copia ****sin datos**** de una BD:

si deseamos que el fichero SQL **no contenga los comandos para importar datos**, debemos crear el fichero SQL con este comando (con el modificador --no-data):

**mysqldump --no-data -u root -p -v emple > BDEmpleados.sql**

### Longitud máxima de los identificadores:

<http://dev.mysql.com/doc/refman/5.7/en/identifiers.html>

|  |  |
| --- | --- |
| **Identifier** | **Maximum Length (characters)** |
| Database | 64 |
| Table | 64 |
| Column | 64 |
| Index | 64 |
| Constraint | 64 |
| Stored Program | 64 |
| View | 64 |
| Tablespace | 64 |
| Server | 64 |
| Log File Group | 64 |
| Alias | 256 (see exception following table) |
| Compound Statement Label | 16 |
| User-Defined Variable | 64 as of MySQL 5.7.5, no limit before that |

### Storage Engines

#### InnoDB

<http://dev.mysql.com/doc/refman/5.7/en/innodb-storage-engine.html>

CREATE TABLE IF NOT EXISTS `departamentos` (

`DEP\_NO` SMALLINT(4) NOT NULL,

`DNOMBRE` VARCHAR(20) NOT NULL,

`LOCALIDAD` VARCHAR(20) ,

PRIMARY KEY (`DEP\_NO`)

) ENGINE = INNODB;

#### MyISAM Storage Engine

<http://dev.mysql.com/doc/refman/5.7/en/myisam-storage-engine.html>

<http://dev.mysql.com/doc/refman/5.7/en/storage-engines.html>

#### MEMORY Storage Engine

<http://dev.mysql.com/doc/refman/5.7/en/memory-storage-engine.html>

### Partitions

<http://www.vertabelo.com/blog/technical-articles/everything-you-need-to-know-about-mysql-partitions>

<https://www.w3resource.com/mysql/mysql-partition.php>

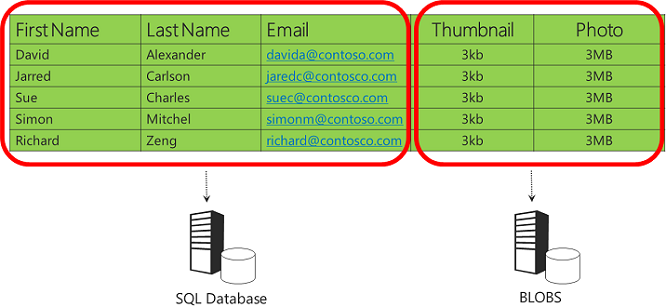
<https://dev.mysql.com/doc/refman/5.7/en/partitioning.html>

MySQL partitioning is about altering – ideally, optimizing – the way the database engine physically stores data. It allows you to distribute portions of table data (a.k.a. partitions) across the file system based on a set of user-defined rules (a.k.a. the “partitioning function”). In this way, if the queries you perform access only a fraction of table data and the partitioning function is properly set, there will be less to scan and queries will be faster.

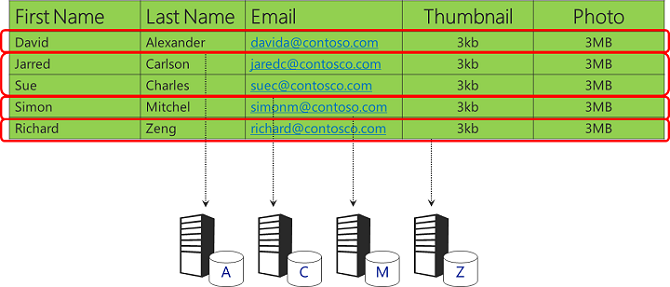
It is important to note that partitioning makes the most sense when dealing with large data sets. If you have fewer than a million rows or only thousands of records, partitioning will not make a difference.

1. Horizontal vs. Vertical Partitioning

Horizontal partitioning means that all rows matching the partitioning function will be assigned to different physical partitions. Vertical partitioning allows different table columns to be split into different physical partitions. Currently, MySQL supports horizontal partitioning but not vertical. The engine’s documentation clearly states it won’t support vertical partitions any time soon: ”There are no plans at this time to introduce vertical partitioning into MySQL.”



Vertical partitioning is about splitting up columns



Horizontal partitioning is about splitting up rows

1. Key Benefits of Partitioning

Some of the advantages of using partitions are:

* Storage: It is possible to store more data in one table than can be held on a single disk or file system partition.
* Deletion: Dropping a useless partition is almost instantaneous, but a classical DELETE query run in a very large table could take minutes.
* Partition Pruning: This is the ability to exclude non-matching partitions and their data from a search; it makes querying faster. Also, MySQL 5.7 supports explicit partition selection in queries, which greatly increases the search speed. (Obviously, this only works if you know in advance which partitions you want to use.) This also applies for DELETE, INSERT, REPLACE, and UPDATE statements as well as LOAD DATA and LOAD XML.

## DML

### ISERT INTO

#### **Opción A**: Indicar valores para todos los atributos de la tabla:

**INSERT INTO *table\_name* VALUES**

**(*valor1*,*valor2*,*valor3*,...);**

* En este caso hay que dar valor a todos los atributos.
* En caso de no querer darle valor a un atributo es obligatorio asignarle NULL

#### **Opción B**: Indicar sólo algunos atributos de la tabla:

Cuando sólo se va a especificar el valor de algunos atributos, entonces especificar el nombre de los atributos a los que se dará valor

**INSERT INTO table\_name (column1,column2,column3,...)**

**VALUES (value1,value2,value3,...);**

#### Insertar una fila en una tabla que incluye un atributo que tiene la propiedad **AUTO\_INCREMENT**

Como regla general a un atributo con la propiedad **AUTO\_INCREMENT** no debemos asignarle valor (es el propio servidor quien debe hacerlo); por tanto tenemo estas 3 opciones para insertar una fila que incluiye un atributo con la propiedad **AUTO\_INCREMENT**:

* + Opción A: Indicar el valor NULL para el atributo y el sistema le asignará a esa celda el número que le corresponda.
  + Opción B: no incluir el atributo en la lista de atributos que vamos a dar.
  + Opción C: indicar un valor para el atributo y entoces el servidor continuará a partir de ese número cuando en el futuro se inserten nuevas filas (el valor debe ser mayor que todos los existentes para que no haya colisines en el futuro).

CREATE TABLE Persona

(

IdPersonaINT NOT NULL AUTO\_INCREMENT,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),

PRIMARY KEY (IdPersona)

)

#Insertar datos:

**#opción A**

**INSERT INTO Persona**

**VALUES (NULL,'Rodriguez', 'Manuel','Calle del sol 45', 'Ferrol');**

**#opción B**

**INSERT INTO Persona (apellido1, mombre, direccion, localidad)**

**VALUES('Rodriguez', 'Manuel','Calle del sol 45', 'Ferrol');**

**#opción C**

**INSERT INTO Persona**

**VALUES (5467,'Rodriguez', 'Manuel','Calle del sol 45', 'Ferrol');**

#### Insertar varias filas con un solo comando INSERT INTO

Podemos insertar varios registros con un solo comando INSERT INTO

Si necesitamos insertar varias filas en una tabla, este es el método más rápido, porque reduce la comunicación entre cliente y servidor.

#**Opción A**:

INSERT INTO *table\_name* **VALUES**

**(value1,value2,value3,…, valueN),**

**(value1,value2,value3,…, valueN);**

#**Opción B**:

INSERT INTO table\_name (column1,column2,column3,...) **VALUES**

**(value1,value2,value3,…, valueJ),**

**(value1,value2,value3,…, valueJ);**

#### Insertar filas obteniendo los datos mediante una consulta a otras tablas

#**Opción A**:

INSERT INTO *table\_name*   
**sententcia SELECT**

#**Opción B**:

INSERT INTO table\_name (column1,column2,column3,...)  
**sententcia SELECT**

Ejemplos:

#**Opción A**: escoger todos los atributos

INSERT INTO AntiguosE*mpleados*  
**SELECT \***

**FROM empleados**

**WHERE emp\_no = 1234;**

#**Opción B**: escoger algunos atributos

INSERT INTO AntiguosE*mpleados (emp\_no, apellido1, apellido2, nombre, telefono)*  
**SELECT *emp\_no, apellido1, apellido2, nombre, telefono***

**FROM empleados**

**WHERE emp\_no = 1234**

#### REPLACE: utiliza los datos de la tabla B para actualizar o insertar datos en la tabla A

**REPLACE** inserta las filas de la tabla B en la tabla A, funciona como INSERT excepto cuando la TABLA A tiene una fila cuya **PRIMARY KEY** (o algún índice **UNIQUE)** coincide con la fila de la tabla B que se está insertando: en ese caso la fila original de la tabla A es borrada antes de insertar la fila procedente de la tabla B.

#**Opción A**: escoger todos los atributos

REPLACE INTO tabla\_A

SELECT \*

FROM tabla\_B

;

#**Opción B**: escoger algunos atributos

REPLACE INTO tabla\_A (dni, telefono, direccion)

SELECT (dni, telefono, direccion)

FROM tabla\_B

;

#### SET foreign\_key\_checks = 0, Deshabilitar las claves ajenas temporalmente para introducir datos

**SET foreign\_key\_checks = 0; #deshabilitar**

**INSERT…**

**SET foreign\_key\_checks = 1;#rehabilitar**

Deshabilitar las claves ajenas: el servidor deja de comprobar si se cumplen las restricciones de las claves ajenas en todas las tablas de todas las bases de datos.

**En algunas ocasiones se utilizan para introducir datos sin tener que preocuparse del orden en que se introducen las filas**: no importa que introduzcamos una fila que hace referencia a otra fila que todavía no hemos introducido.

Pero **cuidado**:

Al rehabililtar las FOREIGN KEY, el sistema **no comprueba si los datos ya existentes en la tabla cumplen las FOREIGN KEY que se acaban de rehabilitar, con lo que permanecerán los datos que no cumplen las FOREIGN KEY**.

Sin embargo: si una tabla ya contiene datos y después añadimos una nueva FOREIGN KEY, en este caso el sistema comprobará si todos los datos de la tabla cumplen esa FOREIGN KEY, y si alguno de ellos no la cumple entonces se produce un error y no se crea la FOREIGN KEY.

* + **SET foreign\_key\_checks = 0**:

1. Sí Deshabilita la comprobación de las claves FOREING KEY existentes en las tablas.
2. No Deshabilita la comprobación UNIQUE de las claves primarias.
3. No Deshabilita las restricciones UNIQUE.
   * **SET foreign\_key\_checks = 1**:
4. Vuelve a habilitar la comprobación de las claves FOREIGN KEY existentes en las tablas
5. Pero **no comprueba si los datos ya existentes en la tabla cumplen las FOREIGN KEY.**

Si Deshabilitamos las FOREING KEY con SET foreign\_key\_checks = 0;, permanecerán Deshabitadas hasta que se Habiliten con el comandoSET foreign\_key\_checks = 1; o hasta que se reinicie el servidor.

##### DISABLE KEYS (sólo funciona en tablas MyISAM)

El comando DISABLE KEYS sólo funciona con tablas MyISAM (no funciona con tablas InnoDB).

ALTER TABLE empleados DISABLE KEYS;

ALTER TABLE empleadosENABLE KEYS;

Es equivalente a:

SET foreign\_key\_checks = 0;

SET foreign\_key\_checks = 1;

#### Ejemplos:

CREATE TABLE IF NOT EXISTS conductor (

dni VARCHAR(9),

nombre VARCHAR(90),

direccion VARCHAR (200),

fecha\_nacimiento DATETIME,

telefono VARCHAR (15),

PRIMARY KEY (dni)

);

*#indicar* **todos los atributos***:*

*#tenemos que indicar NULL cuando deseamos dejar el atributo en blanco.*

INSERT INTO conductor

VALUES ('32669321y', NULL , NULL, NULL , '981326598');

*#indicar* **sólo algunos atributos***:*

INSERT INTO conductor (dni, telefono)

VALUES ('65269345y', '981326598');

*#introducir* **varios registros con un solo comando INSERT***:*

INSERT INTO conductor (dni, telefono)

VALUES ('65269345Y', '981326598'), ('1231234U', '675432134');

Error:

#tendríamos que indicar NULL, no vale con dejar los atributos vacíos.

INSERT INTO conductor

VALUES ('32669321y', , , , '981326598');

#### LOAD DATA INFILE: Insertar filas procedentes de un fichero

1. The LOAD DATA INFILE statement reads rows from a text file into a table at a very high speed.
2. LOAD DATA INFILE is the complement of [SELECT ... INTO OUTFILE](#_SELECT_INTO).

LOAD DATA INFILE '/var/lib/mysql-files/pedidos2000servidor.txt' INTO TABLE bdempleados.pedidos2000;

1. El fichero no puede tener líneas en blanco ni por el medio ni al final (no se puede meter carácter de nueva línea al final de la última línea).
2. For security reasons, when reading text files located on the server, the files must either reside in the database directory or be readable by the user account used to run the server.
3. Also, to use LOAD DATA INFILE on server files, you must have the FILE privilege.
4. Ejemplo

USE bdempleados;

LOAD DATA INFILE '/var/lib/mysql-files/pedidos2000servidor.txt' INTO TABLE bdempleados.pedidos2000;

;

##### ERROR al importar los datos debido a violaciones de integridad referencial (clave ajena)

Cuando existe una relación reflexiva de la tabla consigo misma, puede suceder que el orden de las filas en el fichero sea incorrecto y provoque un **error porque que al importarlo se violen restricciones de integridad referencial** (clave foránea), Para **solventar este problema** tenesmos 2 opciones:

* **sencilla**:
* deshabilitar las restricciones (SET FOREIGN\_KEY\_CHECKS=0;) antes de importar los datos del fichero
* importar los datos desde el fichero
* volver a habilitar las restricciones (SET FOREIGN\_KEY\_CHECKS=1;) paralas inserciones futuras (no comprueba los datos viejos)

1. El problema es que si alguna de las filas introducidas viola la FOREIGN KEY, el sistema no lo detecta, para ello habría que crear una consulta que buscase filas con la FOREIGN KEY errónea

* **compliacada**:
* reordenar las filas en el fichero antes de importarlo, de modo que no se violen las FOREIGN KEY

1. Ejemplo:

#deshabilitar las restricciones

SET FOREIGN\_KEY\_CHECKS=0;

# inserter las filas

INSERT INTO `bdempleados`.`empleados` (`EMP\_NO`, `DNI`, `NOMBRE`, `APELLIDO1`, `APELLIDO2`, `OFICIO`, `JEFE`, `FECHA\_ALTA`, `SALARIO`, `COMISION`, `DEP\_NO`, `TELEFONO`) VALUES ('9001', '14256821Y', 'Luís', 'Yañez', 'Rioboo', 'ADMINISTRATIVO', '9003', '2016-04-05', '150000', '0', '10', '652986532');

INSERT INTO `bdempleados`.`empleados` (`EMP\_NO`, `DNI`, `NOMBRE`, `APELLIDO1`, `APELLIDO2`, `OFICIO`, `JEFE`, `FECHA\_ALTA`, `SALARIO`, `COMISION`, `DEP\_NO`, `TELEFONO`) VALUES ('9002', '45325698P', 'Ana', 'Rinlo', 'Rois', 'INGENIERA', '8904', '2016-04-05', '150000', '0', '10', '547852146');

# volver a habilitar las restricciones para las inserciones futuras.

SET FOREIGN\_KEY\_CHECKS=1;

# Buscar las filas con FOREIGN KEY errónea:

SELECT \*

FROM empleados

WHERE jefe NOT IN (

SELECT emp\_no

FROM empleados)

;

#si aparece alguna fila errónea hay que arreglarla a mano:



#La fila mostrada indica que el empleado 9001 tiene como jefe al empleado 9003 (que no existe).

#La solución pasa por asignarle a mano otro jefe y volver a comprobar los datos mediante la consulta previa.

### Acelerar la inserción de datos en una tabla

<https://dev.mysql.com/doc/refman/5.7/en/insert-optimization.html>

<https://dev.mysql.com/doc/refman/5.7/en/optimizing-innodb-bulk-data-loading.html>

1. Reglas para acelerar la inserción de datos en tablas de cualquier tipo:
2. If you are inserting many rows from the same client at the same time, use [INSERT](https://dev.mysql.com/doc/refman/5.7/en/insert.html) statements with multiple VALUES lists to insert several rows at a time. This is considerably faster (many times faster in some cases) than using separate single-row [INSERT](https://dev.mysql.com/doc/refman/5.7/en/insert.html) statements. If you are adding data to a nonempty table, you can tune the [bulk\_insert\_buffer\_size](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_bulk_insert_buffer_size) variable to make data insertion even faster. See [Section 5.1.5, “Server System Variables”](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html).
3. When loading a table from a text file, use [LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html). This is usually 20 times faster than using [INSERT](https://dev.mysql.com/doc/refman/5.7/en/insert.html) statements. See [Section 13.2.6, “LOAD DATA INFILE Syntax”](https://dev.mysql.com/doc/refman/5.7/en/load-data.html).
4. Take advantage of the fact that columns have default values. Insert values explicitly only when the value to be inserted differs from the default. This reduces the parsing that MySQL must do and improves the insert speed.
5. Reglas para acelerar la inserción de datos en tablas de tipo innodb:
6. When importing data into InnoDB, turn off autocommit mode, because it performs a log flush to disk for every insert. To disable autocommit during your import operation, surround it with [SET autocommit](https://dev.mysql.com/doc/refman/5.7/en/commit.html) and [COMMIT](https://dev.mysql.com/doc/refman/5.7/en/commit.html) statements:

SET autocommit=0;

*... SQL import statements ...*

COMMIT;

The [**mysqldump**](https://dev.mysql.com/doc/refman/5.7/en/mysqldump.html) option [--opt](https://dev.mysql.com/doc/refman/5.7/en/mysqldump.html#option_mysqldump_opt) creates dump files that are fast to import into an InnoDB table, even without wrapping them with the [SET autocommit](https://dev.mysql.com/doc/refman/5.7/en/commit.html) and [COMMIT](https://dev.mysql.com/doc/refman/5.7/en/commit.html) statements.

1. If you have UNIQUE constraints on secondary keys, you can speed up table imports by temporarily turning off the uniqueness checks during the import session:

SET unique\_checks=0;

*... SQL import statements ...*

SET unique\_checks=1;

For big tables, this saves a lot of disk I/O because InnoDB can use its change buffer to write secondary index records in a batch. Be certain that the data contains no duplicate keys.

1. If you have FOREIGN KEY constraints in your tables, you can speed up table imports by turning off the foreign key checks for the duration of the import session:

SET foreign\_key\_checks=0;

*... SQL import statements ...*

SET foreign\_key\_checks=1;

For big tables, this can save a lot of disk I/O.

1. Use the multiple-row [INSERT](https://dev.mysql.com/doc/refman/5.7/en/insert.html) syntax to reduce communication overhead between the client and the server if you need to insert many rows:

INSERT INTO yourtable VALUES (1,2), (5,5), ...;

This tip is valid for inserts into any table, not just InnoDB tables.

1. When doing bulk inserts into tables with auto-increment columns, set [innodb\_autoinc\_lock\_mode](https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html#sysvar_innodb_autoinc_lock_mode) to 2 instead of the default value 1. See [Section 14.8.6, “AUTO\_INCREMENT Handling in InnoDB”](https://dev.mysql.com/doc/refman/5.7/en/innodb-auto-increment-handling.html) for details.
2. When performing bulk inserts, it is faster to insert rows in PRIMARY KEY order. InnoDB tables use a [clustered index](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_clustered_index), which makes it relatively fast to use data in the order of the PRIMARY KEY. Performing bulk inserts in PRIMARY KEY order is particularly important for tables that do not fit entirely within the buffer pool.
3. For optimal performance when loading data into an InnoDB FULLTEXT index, follow this set of steps:
4. Define a column FTS\_DOC\_ID at table creation time, of type BIGINT UNSIGNED NOT NULL, with a unique index named FTS\_DOC\_ID\_INDEX. For example:

CREATE TABLE t1 (

FTS\_DOC\_ID BIGINT unsigned NOT NULL AUTO\_INCREMENT,

title varchar(255) NOT NULL DEFAULT '',

text mediumtext NOT NULL,

PRIMARY KEY (`FTS\_DOC\_ID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE UNIQUE INDEX FTS\_DOC\_ID\_INDEX on t1(FTS\_DOC\_ID);

1. Load the data into the table.
2. Create the FULLTEXT index after the data is loaded.
3. Note
4. When adding FTS\_DOC\_ID column at table creation time, ensure that the FTS\_DOC\_ID column is updated when the FULLTEXT indexed column is updated, as the FTS\_DOC\_ID must increase monotonically with each [INSERT](https://dev.mysql.com/doc/refman/5.7/en/insert.html) or [UPDATE](https://dev.mysql.com/doc/refman/5.7/en/update.html). If you choose not to add the FTS\_DOC\_ID at table creation time and have InnoDB manage DOC IDs for you, InnoDB will add the FTS\_DOC\_ID as a hidden column with the next [CREATE FULLTEXT INDEX](https://dev.mysql.com/doc/refman/5.7/en/create-index.html) call. This approach, however, requires a table rebuild which will impact performance.
5. [PREV](https://dev.mysql.com/doc/refman/5.7/en/optimizing-innodb-logging.html)   [HOME](https://dev.mysql.com/doc/refman/5.7/en/index.html)   [UP](https://dev.mysql.com/doc/refman/5.7/en/optimizing-innodb.html)   [NEXT](https://dev.mysql.com/doc/refman/5.7/en/optimizing-innodb-queries.html)

### UPDATE

<http://mysql.conclase.net/curso/?sqlsen=UPDATE>

1. Actualiza algunos atributos de los registros que cumplan la condición impuesta por la cláusula WHERE

UPDATE [LOW\_PRIORITY] [IGNORE] table\_reference

SET col\_name1={expr1|DEFAULT} [, col\_name2={expr2|DEFAULT}] ...

[WHERE where\_condition]

[ORDER BY ...]

[LIMIT row\_count]

1. **Si no se indica cláusula WHERE** entonces se actualizarán **todos los registros** de la tabla.
2. **LIMIT**: nº de registros a modificar como máximo. En caso de que hubiese más registros que cumplen la condición WHERE para ser modificados de los que indicamos con LIMIT, se modificarán los N primeros y el resto no. Este comportamiento es **indeseable en un UPDATE**. o **DELETE**, porque dejaría registros sin modificar.
3. **ORDER BY**, indica en qué orden se modificarán los registros. Sólo tiene sentido combinado con LIMIT (se usa cuando no se quieren actualizar todos los registros que cumplan la condición WHERE).

-- cambiar 1 atributo

UPDATE TablaPersonas SET edad=edad+1;

-- cambiar 2 atributos

UPDATE TablaPersonas SET edad=edad\*2, numHijos = 2

WHERE dni LIKE (‘12345678A’);

-- cambiar datos en más de 1 tabla:

UPDATE Libros, Pedidos

SET Pedidos.CantidadLibrosVendidos=Pedidos.CantidadLibrosVendidos+2,

Libros.EnStock=Libros.EnStock-2

WHERE Libros.LibroID= Pedidos.BookID

AND Pedidos.PedidoID = 1002;

### Restricción del Workbench que impide UPDATES O DELETES debido al SAFE MODE cuando no se incluye una KEY en el WHERE o no se usa LIMIT

1. Por defecto Workbench tiene un comportamiento restrictivo porque impide el UPDATE o DELETE cuando se cumplen estas 2 condiciones simultáneamente:

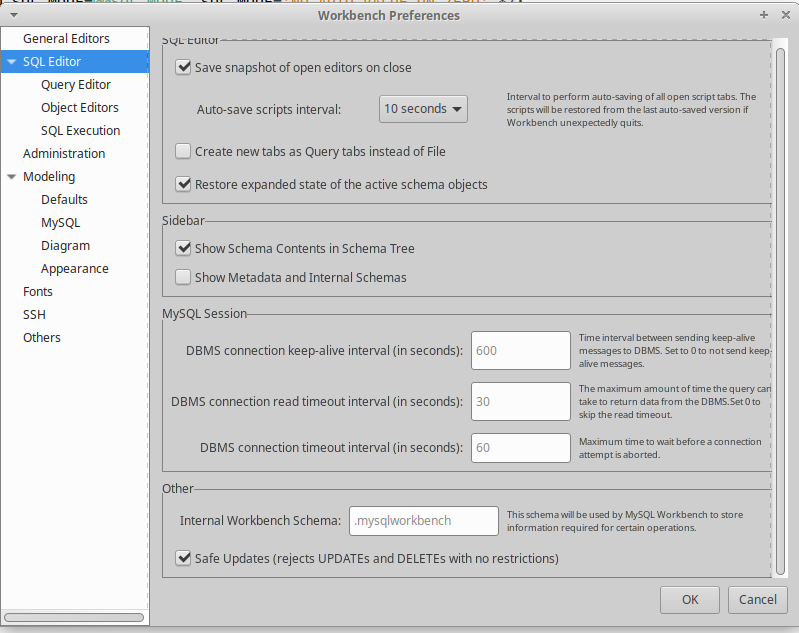
* El UPDATE (DELETE) no contiene un WHERE que incluya una KEY
* El UPDATE (DELETE) no incluye una cláusula LIMIT

El mensaje que muestra Workbench en este caso es éste:

You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column To disable safe mode, toggle the option in Preferences -> SQL Queries and reconnect.

1. La razón de esta restricción es evitar que un UPDATE (DELETE) pueda modificar más filas de las deseadas, eso podría suceder en caso de que no tengamos un buen dominio del lenguaje SQL y seleccionemos mal las filas a modificar. Una vez modificadas las filas, no hay manera de deshacer el cambio (salvo que hayamos iniciado una transacción y realicemos un ROLLBACK o que tengamos un BACKUP que restaurar).
2. Esta restricción es **exclusiva del Workbench**, no está presente si nos conectamos al servidor desde un terminal del S.O.
3. Podemos **configurar Workbench** para que no imponga esta restricción y permita cualquier tipo de UPDATE, esto se configura en:

WorkBench – Edit – Preferences – SQL Editor –“Save Updates” (Rejects UPDATES and DELETES with no restrictions)

1. 
2. **Reiniciar Workbench** para que surtan efecto los cambios.

### DELETE

<http://mysql.conclase.net/curso/?sqlsen=DELETE>

Borra los registros que cumplan la condición impuesta por la cláusula WHERE

DELETE FROM *tbl\_name*

[WHERE *where\_condition*]

[ORDER BY ...]

[LIMIT *row\_count*]

1. Si no se indica cláusula WHERE entonces se borrarán todos los registros de la tabla.
2. **LIMIT**: nº de registros a borrar como máximo.
3. **ORDER BY**, indica en qué orden se borrarán los registros. Sólo tiene sentido combinado con LIMIT (se usa cuando no se quieren borrar todos los registros que cumplan la condición WHERE).

#### Ejemplos de DELETE

Ejemplo de DELETE que utiliza una sola tabla: Borrar los pedidos anteriores al año 2015

DELETE

FROM pedidos

WHERE fecha\_pedido < '2015-01-01';

Ejemplo de DELETE que utiliza varias tablas: Borrar todos los pedidos excepto los 10 más recientes (según la fecha):

ROLLBACK;

START TRANSACTION;

#mostrar los pedidos existentes

SELECT \* FROM pedidos;

#crear en RAM una tabla temporal llamada pedidosNoQueremosBorrar,

#que sea copia de pedidos pero que sólo tenga los pedidos que no queremos borrar

#es decir contendrá los 10 pedidos más recientes.

#la tabla solo contendrá la PK (Primary Key) de los pedidos.

CREATE TEMPORARY TABLE pedidosNoQueremosBorrar

SELECT pedido\_no

FROM pedidos

ORDER BY fecha\_pedido DESC

LIMIT 10;

#crear en RAM una tabla temporal llamada pedidosSiQueremosBorrar,

#que sea copia de pedidos pero que solo tenga los pedidos que queremos borrar

#la tabla solo contendrá la PK (Primary Key) de los pedidos.

CREATE TEMPORARY TABLE pedidosSiQueremosBorrar

SELECT pedidos.pedido\_no

FROM pedidos LEFT JOIN pedidosNoQueremosBorrar

ON pedidos.pedido\_no = pedidosNoQueremosBorrar.pedido\_no

WHERE pedidosNoQueremosBorrar.pedido\_no IS NULL;

#borrar los pedidos de la tabla pedidos

#usando un LEFT JOIN para escoger los pedidos a borrar

#Ver diagrama de conjuntos para entenderlo

DELETE pedidos

FROM pedidos , pedidosSiQueremosBorrar

WHERE pedidos.pedido\_no = pedidosSiQueremosBorrar.pedido\_no ;

#### TRUNCATE

<http://dev.mysql.com/doc/refman/5.6/en/truncate-table.html>

<http://mysql.conclase.net/curso/?sqlsen=TRUNCATE>

1. Borra todos los datos de una tabla, pero No puede realizarse TRUNCATE sobre tablas InnoDB en las que otra tabla tiene claves ajenas que apuntan a la tabla a truncar. (por ejemplo no puedo truncar la tabla departamentos porque la tabla empleados tienen una clave ajena que hace referencia a la tabla departamentos).

TRUNCATE TABLE table\_name

1. TRUNCATE es similar en ciertos aspectos a esta otra sentencia:

DELETE FROM table\_name

1. Pero TRUNCATE tiene algunas **diferencias** frente a DELETE:

* Realmente se considera que TRUNCATE es una sentencia DDL, pues es equivalente a una sentencia DROP TABLE seguida por una sentencia CREATE TABLE. Por ello es más rápida que la sentencia DELETE FROM cuando queremos borrar todos los datos de una tabla.

1. Poniendo un símil:
2. DELETE abre un fichero y borrar línea a línea todo su contenido (imagínate si tiene miles di filas el tiempo que consume) mientras que TRUNCATE borra el fichero y crea otro nuevo vacío con el mismo nombre.

* No puede realizarse sobre tablas InnoDB en las que otra tabla tiene claves ajenas que apuntan a la tabla a truncar. Foreign key constraints between columns of the same table are permitted.
* No permite ROLL BACK
* No dispara TRIGGERS ON DELETE
* El valor *AUTO\_INCREMENT* se reinicializa a su valor inicial.
* No se devuelve el número de filas eliminadas.
* Mientras el fichero de definición de tabla 'table\_name.frm' sea válido, la tabla puede ser recreada como una tabla vacía con **TRUNCATE TABLE**, aunque los ficheros de datos o índices estén corruptos.

### LOCK TABLE / UNLOCK TABLE

<http://www.mysqltutorial.org/mysql-table-locking/>

<http://dev.mysql.com/doc/refman/5.7/en/lock-tables.html>

LOCK TABLES

tbl\_name [[AS] alias] lock\_type

[, tbl\_name [[AS] alias] lock\_type] ...

lock\_type:

READ [LOCAL]

| [LOW\_PRIORITY] WRITE

UNLOCK TABLES

The following lock types are available:

* + READ [LOCAL] lock:
* The session that holds the lock can read the table (but not write it).
* Multiple sessions can acquire a READ lock for the table at the same time.
* Other sessions can read the table without explicitly acquiring a READ lock.
* The LOCAL modifier enables nonconflicting INSERT statements (concurrent inserts) by other sessions to execute while the lock is held.
  + [LOW\_PRIORITY] WRITE lock:
* The session that holds the lock can read and write the table.
* Only the session that holds the lock can access the table. No other session can access it until the lock is released.
* Lock requests for the table by other sessions block while the WRITE lock is held.
* The LOW\_PRIORITY modifier has no effect. It is now deprecated.

1. MySQL enables client sessions to acquire table locks explicitly for the purpose of cooperating with other sessions for access to tables, or to prevent other sessions from modifying tables during periods when a session requires exclusive access to them. A session can acquire or release locks only for itself. One session cannot acquire locks for another session or release locks held by another session.
2. Locks may be used to emulate transactions or to get more speed when updating tables.
3. LOCK TABLES explicitly acquires table locks for the current client session.
4. Table locks can be acquired for base tables or views.
5. You must have the LOCK TABLES privilege, and the SELECT privilege for each object to be locked.
6. For view locking, LOCK TABLES adds all base tables used in the view to the set of tables to be locked and locks them automatically.
7. If you lock a table explicitly with LOCK TABLES, any tables used in triggers are also locked implicitly.
8. UNLOCK TABLES explicitly releases any table locks held by the current session.

* LOCK TABLES implicitly releases any table locks held by the current session before acquiring new locks.

1. A table lock protects only against inappropriate reads or writes by other sessions. A session holding a WRITE lock can perform table-level operations such as DROP TABLE or TRUNCATE TABLE.

* For sessions holding a READ lock, DROP TABLE and TRUNCATE TABLE operations are not permitted.

1. In TEMPORARY tables no lock is necessary because no other session can see the table.
2. If a session begins a transaction (for example, with START TRANSACTION), an implicit UNLOCK TABLES is performed, which causes existing locks to be released. (
3. A session that requires locks must acquire all the locks that it needs in a single LOCK TABLES statement.

* While the locks thus obtained are held, the session can access only the locked tables.
* For example, in the following sequence of statements, an error occurs for the attempt to access t2 because it was not locked in the LOCK TABLES statement:

mysql> LOCK TABLES t1 READ;

mysql> SELECT COUNT(\*) FROM t1;

+----------+

| COUNT(\*) |

+----------+

| 3 |

+----------+

mysql> SELECT COUNT(\*) FROM t2;

ERROR 1100 (HY000): Table 't2' was not locked with LOCK TABLES

1. Tables in the INFORMATION\_SCHEMA database are an exception. They can be accessed without being locked explicitly even while a session holds table locks obtained with LOCK TABLES

mysql> LOCK TABLE t WRITE, t AS t1 READ;

mysql> INSERT INTO t SELECT \* FROM t;

ERROR 1100: Table 't' was not locked with LOCK TABLES

mysql> INSERT INTO t SELECT \* FROM t AS t1;

#### Cuando bloquear una tabla

<http://www.databasejournal.com/features/mysql/the-pros-and-cons-of-mysql-table-locking.html>

1. No debemos bloquear las tablas de tipo InnoDB

En tablas que utilicen sistemas de almacenamiento InnoDB es preferible no utilizar bloqueo de tablas, puesto que no ofrece ninguna ventaja pero reduce la concurrencia (acceso simultáneo a una tabla desde sesiones distintas):

* + Desde la versión 5.5 de MySQL El sistema de almacenamiento por defecto de cualquier tabla es InnoDB.
  + InnoDB utiliza automáticamente bloqueo por filas, y con ello permite a varias sesiones leer y escribir en la misma tabla simultáneamente (cuando acceden a filas distintas) sin que tengan que esperar unas por otras.

El resto de sistemas de almacenamiento (MyISAM,…) sólo permiten bloqueo por tabla completa, y aquí sí podría tener sentido bloquear una tabla en ciertas circunstancias.

1. Beneficios del bloqueo de tablas

Cada vez que modificamos una tabla el sistema debe actualizar todos los índices de esa tabla.Si bloqueamos una tabla, la actualización de índices sólo se realizará cuando se desbloquee la tabla, lo cual puede ser más rápido.

## CONSULTAS (SELECT)

### Orden de ejecución de las distintas cláusulas de una consulta SELECT:

**(5) SELECT (6) DISTINCT (8) <TOP\_specification><select\_list>**

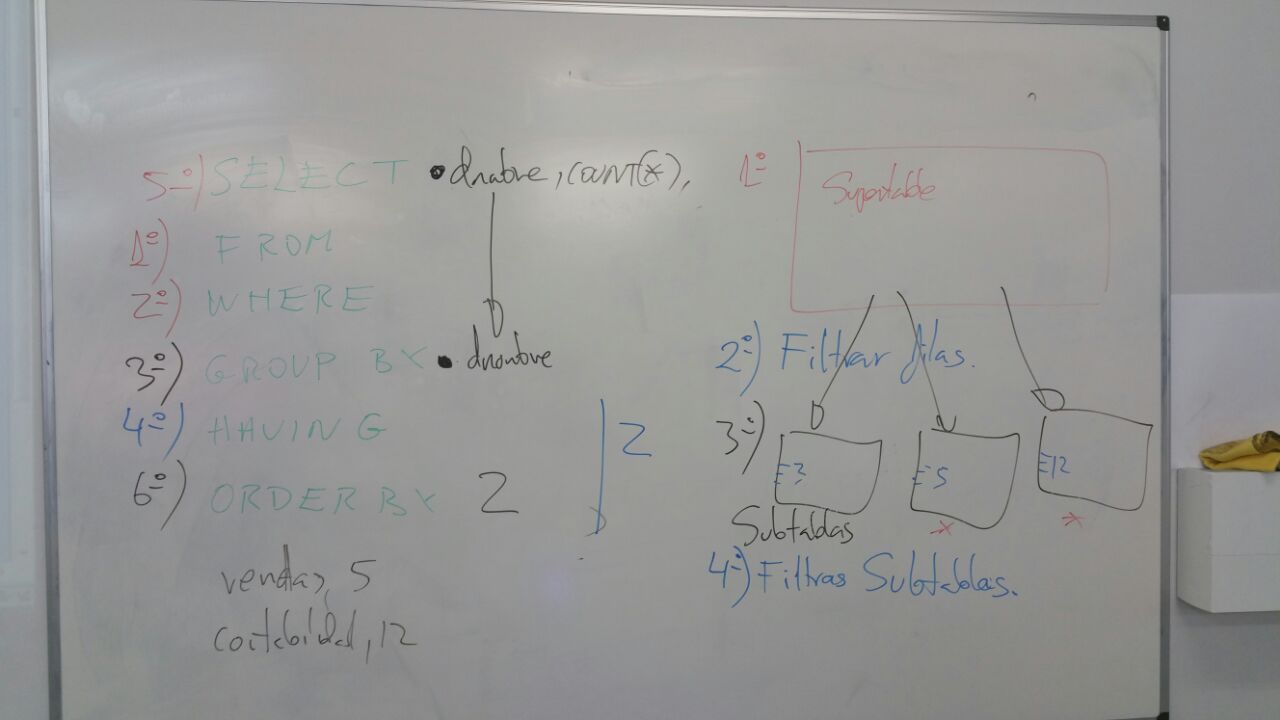
**(1) FROM**

**(2) WHERE <where\_condition>**

**(3) GROUP BY <group\_by\_list>**

**(4) HAVING <having\_condition>**

**(7) ORDER BY <order\_by\_list>**



1. Uso avanzado de SQL

Para Transact SQL de SQL Server el orden es:

(8) SELECT (9) DISTINCT (11) <TOP\_specification><select\_list>

(1) FROM <left\_table>

(3) <joint\_type> JOIN <right\_table>

(2) ON <join\_condition>

(4) WHERE <where\_condition>

(5) GROUP BY <group\_by\_list>

(6) WITH {CUBE | ROLLUP}

(7) HAVING <having\_condition>

(10) ORDER BY <order\_by\_list>

### INNER JOIN, NATURAL JOIN, LEF JOIN, RIGHT JOIN, FULL JOIN …

#### INNER JOIN

<http://stackoverflow.com/questions/8696383/difference-between-natural-join-and-inner-join>

<http://dev.mysql.com/doc/refman/5.7/en/join.html>

Hay 4 formas equivalentes de realizar INNER JOIN

SELECT apellido, empleados.dep\_no, localidad

FROM **empleados INNER JOIN departamentos**

**ON empleados.dep\_no = departamentos.dep\_no**

;

SELECT apellido, empleados.dep\_no, localidad

FROM **empleados INNER JOIN departamentos**

**USING (dep\_no)**

;

SELECT apellido, empleados.dep\_no, localidad

FROM **empleados NATURAL JOIN departamentos**

;

SELECT apellido, empleados.dep\_no, localidad

FROM **empleados, departamentos**

**WHERE empleados.dep\_no = departamentos.dep\_no**

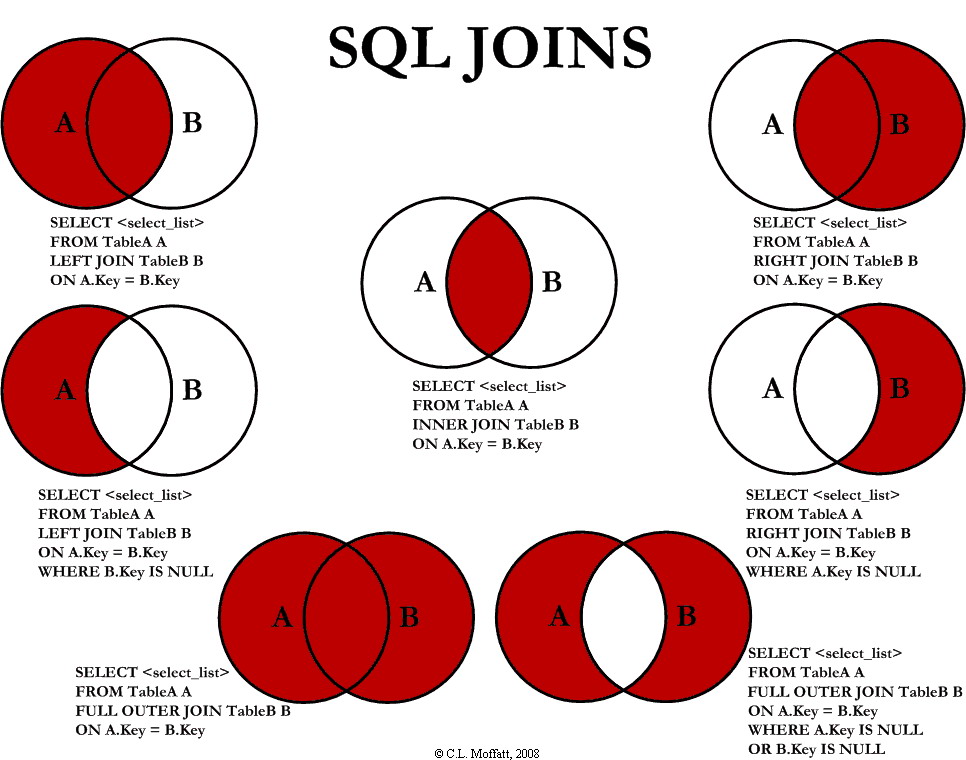
;

In MySQL, JOIN, CROSS JOIN, and INNER JOIN are syntactic equivalents (they can replace each other). In standard SQL, they are not equivalent. INNER JOIN is used with an ON clause, CROSS JOIN is used otherwise.

#### Otros tipos de JOIN: CROSS JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN

<http://stackoverflow.com/questions/448023/what-is-the-difference-between-left-right-outer-and-inner-joins>

<http://dev.mysql.com/doc/refman/5.7/en/join.html>



#### Tipos de JOIN

* + **INNER JOIN**junta dos tablasA y B, sólo devuelve las filaspara las que existe una coincidencia en los atributos de reunión.
  + **OUTER JOIN** compara dos tablas A y B,devuelve las filaspara las que existe una coincidencia en los atributos de reunión y además añade otras filas: cuando una fila de una tabla A no se corresponde con ninguna fila en la otra tabla B, se devuelve esa fila de A y en las columnas de B aparece NULL. Lo mismo sucede cuando una fila de la tabla B no se corresponde con ninguna fila en la otra tabla A, se devuelve esa fila de B y en las columnas de A aparece NULL. Existen **3 tipos de OUTER JOIN**:
* **LEFTOUTER JOIN (LEFTJOIN)**:Devuelve todos los registros de la tabla Ay NULL en los atributos de la tabla B si no tiene coincidencias
* **RIGHT OUTER JOIN (RIGHT JOIN)**:Devuelvetodos los registros de la tabla By NULL en los atributos de la tabla Asi no tiene coincidencias
* **FULLOUTER JOIN (FULLJOIN)**:Devuelvetodos los registros de ambas tablas, e insertar un valor NULL en cualquiera de las tablas si no hay ninguna coincidencia. Equivale a combinar LEFT y RIGHT

1. En MySQL no existe FULL JOIN, en su lugar debemosusar un truco: unir el resultado de un LEFT JOIN con el resultado de un RIGHT JOIN mediante el comando UNION
   * **CROSS JOIN** compara dos tablas y devuelve todas las combinaciones posibles de filas de ambas tablas. Se obtiene una gran cantidad de filas que notienen sentido.Tiene poca utilidad. Equivale a producto cartesiano de matemáticas.
2. En MySQL, JOIN, CROSS JOIN, y INNER JOIN son equivalentessintácticamente (sepueden sustituir unos por otros).En SQL estándar, esos comandos no son equivalentes: INNER JOIN se utilizaobligatoriamente con una cláusula ON yCROSS JOIN se utiliza en el resto de casos.[Http://dev.mysql.com/doc/refman/5.6/en/join.html](https://translate.google.com/translate?hl=es&prev=_t&sl=en&tl=es&u=http://dev.mysql.com/doc/refman/5.6/en/join.html)
   * A menudo se omite la palabra clave **OUTER**, de este modo los comandos serían: "LEFT JOIN", "RIGHT JOIN" o "FULL JOIN".

# **ejemplo** de cómo realizar algo equivalente al **FULLOUTER JOIN** en MySQL

#Combinanos el resultado de un LEFT y un RIGHT

SELECT \*

FROM empleados LEFT JOIN departamentos ON empleados.dep\_no = departamentos.dep\_no

UNION

SELECT \*

FROM empleados RIGHT JOIN departamentos ON empleados.dep\_no = departamentos.dep\_no

;

#[UNION](#_UNION_/_UNION): une el resultado de 2 consultas sin mostrar filas duplicadas

#[UNION ALL](#_UNION_/_UNION): une el resultado de 2 consultas (incluye filas duplicadas)

#### Ejemplos de distintos tipos de JOIN:

Analizaremos un caso práctico:

DESCRIPCIÓN DEL PROBLEMA:

**Se trata de un instituto con alumnos (alumnos) que pueden usar taquillas (taquillas)**

**Cada estudiante puede ser asignado a unataquilla, por lo que hay una clave ajena Numero\_Taquilla en la Tabla de alumnos.**

**Un alumno nunca tiene más de un taquilla.**

**En unataquilla podría estar Más de un estudiante (comparten la taquilla)**

**Es posible que algún estudiante no tengataquilla.**

**Puede haber taquillas que no tienen estudiantes asignados.**

**Ejemplo, digamos que: hay 100 estudiantes, 70 de los cuales tienen taquillas. Tiene un total de 50 taquillas, 40 de las cuales con al menos 1 estudiante y 10 taquillas no tienen ningún estudiante.**

1. INNER JOIN

**SELECT \***

**FROM alumnos INNER JOIN taquillas ON alumno.numero\_taquilla = taquilla.numero\_taquilla**

responde a la pregunta "muéstrame todos los estudiantes con armarios junto con los datos de su armario".

Cualquier estudiante sin armarios, o cualquier taquilla sin estudiantes desaparecen.

Devuelve 70 filas

1. LEFT JOIN (LEFT OUTER JOIN)

**SELECT \***

**FROM alumnos LEFT JOIN taquillas ON alumno.numero\_taquilla = taquilla.numero\_taquilla**

responde a la pregunta:

"muéstrame todos los estudiantes, con sus correspondientes armarios;muestra a todos los alumnos, aunque no tengan armario".

Esto podría ser una lista general de los estudiantes, o podría ser utilizado para identificar a los estudiantes sin casillero.

Devuelve 100 filas

1. RIGHT JOIN (RIGHT OUTER JOIN)

**SELECT \***

**FROM alumnos RIGHT JOIN taquillas ON alumno.numero\_taquilla = taquilla.numero\_taquilla**

Responde a la pregunta:

"muéstrame todas las taquillas (tengan o no estudiantes asignados), al lado de cada taquilla muestra los datos de los estudiantes asignados a la taquilla, y si no tiene ningún estudiante asignado muestra los atributos del estudiante como NULL".

Esto podría ser utilizado para conocer los armarios que tienen alumnos junto con los armarios sin alumnos.

Devuelve 80 filas (lista de 70 estudiantes en los casilleros 40, además de los 10 casilleros con ningún estudiante)

1. FULL JOIN (FULL OUTER JOIN)

**SELECT \***

**FROM alumnos FULL JOIN taquillas ON alumno.numero\_taquilla = taquilla.numero\_taquilla**

**#Como en MYSQL no existe FULL JOIN, lo solucionamos así:**

**SELECT \***

**FROM alumnos LEFT JOIN taquillas ON alumno.numero\_taquilla = taquilla.numero\_taquilla**

**UNION ALL**

**SELECT \***

**FROM alumnos RIGHT JOIN taquillas ON alumno.numero\_taquilla = taquilla.numero\_taquilla**

Responde a la pregunta:

"muéstrame todos los estudiantes y todos los armarios".

Devuelve 110 filas (100 todos los estudiantes, incluyendo los que no tienen taquillas, ademáslas 10 taquillas sin estudiante).

CROSS JOIN es absurdo en este escenario.

Muestra una lista gigante de toda posible vinculación entre estudiantes y el armario, aunque no sea real.

Devuelve 5000 filas (100 x 50 filas: 100 estudiantes x 50 taquillas).

### Cadenas de texto

1. **Las cadenas de texto(VARCHAR, CHAR,...) pueden escribirse indistintamente en mayúsculas o en minúsculas**, el SGBD no las distingue al buscar un texto.

### IFNULL / NZ / COALESCE

Función que sustituye un valor NULL por otro valor

SELECT apellido, salario, comision, salario + **IFNULL(comision,0)** AS 'salario + comision'

FROM empleados;

#si comisión es NULL se sustituye por 0

1. **La función IFNULL(atributo, valorSutituto ) convierte un NULL en otro valor sustituto que deseemos**.

En Access se usa la función **NZ(atributo, valorSutituto )**

En otros sistemas se usa la función **COALESCE( )**

[**http://www.w3schools.com/sql/sql\_isnull.asp**](http://www.w3schools.com/sql/sql_isnull.asp)

### AVG, SUM, MAX, MIN

En MySQLlas funciones de agregado no tienen en cuenta los valores nulos.

En Access si algún valor es nulo la función devuelve valor nulo.

### UNION / UNION ALL

<http://dev.mysql.com/doc/refman/5.7/en/union.html>

SELECT \*

FROM empleados

WHERE oficio<>"VENDEDOR" And salario>150000

UNION

SELECT \*

FROM empleados

WHERE dep\_no = 20

;

1. **El operador UNIONno devuelve filas duplicadas**.

SELECT \*

FROM empleados

WHERE oficio<>"VENDEDOR" And salario>150000

UNION ALL

SELECT \*

FROM empleados

WHERE dep\_no = 20

;

1. **El operador UNION ALLsí devuelve filas duplicadas**.
2. UNION permite que los SELECT implicados devuelvan columnas con distintos tipos de datos, la siguiete consulta no produce error (aunque dep\_no es INT, dnombre es VARCHAR y no tiene sentido mezclarlas):

SELECT dep\_no

FROM empleados

UNION

SELECT dnombre

FROM departamentos

;

1. UNION requiere que los SELECT devuelvan el mismo número de columnas.

# **PRODUCE ERROR PORQUE LOS 2 SELECT DEVUELVEN DISTINTO NÚMERO DE COLUMNAS**:

SELECT emp\_no, apellido

FROM empleados

WHERE oficio = ‘VENDEDOR’

UNION

SELECT emp\_no,

FROM empleados

WHERE oficio = ’DIRECTOR’

;

#### Reunir las filas de 2 o más tablas (A, B) en una sola tabla (C)

1. Podemos reunir las filas de 2 o más tablas (A, B) en una sola tabla (C) sobre la que se ejecutará una consulta. Ejemplo:

Tenemos 2 tablas:

empleadosActuales

empleadosJubilados

Queremos buscar aquellos que son de Ferrol.

Podríamos realizar 2 consultas separadas, una para cada tabla; pero en este caso vamos a realizar una única consulta y para ello vamos a juntar las filas de ambas tablas en una tabla virtual de nombre (ALIAS) empleadosTotales

SELECT \*

FROM (

SELECT dni, nombre, telefono, localidad FROM empleadosActuales

UNION ALL

SELECT dni, nombre, telefono, localidad FROM empleadosJubilados

) AS empleadosTotales

WHERE localidad = 'FERROL'

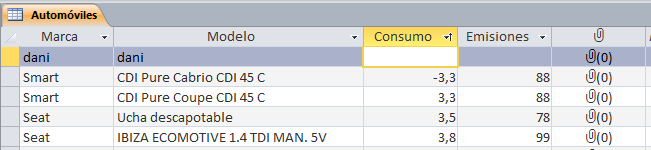
;

### ORDER BY

Los NULL se ponen al final de la lista ordenada si el orden es decreciente.

Los NULL se ponen al principio de la lista ordenada si el orden es creciente.

1. Ambas afirmaciones son ciertas independientemente de que los valores numéricos almacenados sean positivos o negativos.



Ordenando en creciente por el campo Consumo.

### Vistas

#### Vistas en MySQL

Las vistas en MySQL se explican en otro documento independiente

#### Vistas en Access

Una vista es una consulta que ha sido guardada con un nombre

Normalmente una consulta busca los datos dentro de una o varias tablas, pero también puede buscar los datos dentro de una o varias vistas.

1. Ejemplo

Creamos la siguiente consulta llamada **nombreMarcasDistintas**:

SELECT DISTINCT marca

FROM Automóviles

;

Creamos la siguiente consulta llamada **numeroMarcasDistintas**:

SELECT Count(\*) AS 'Núme -9 septiembre había perdido en un ro de Marcas distintas'

FROM **nombreMarcasDistintas**;

### NULL

<http://dev.mysql.com/doc/refman/5.7/en/working-with-null.html>

Problemas con los nulos en consultas: <http://dev.mysql.com/doc/refman/5.7/en/problems-with-null.html>

1. Un atributo (por ejemplo codigoDepartamento) con valor NULL produce como resultado NULL cuando se usa con estos operadores:

= <<= >= <> IN NOT IN < ALL <= ALL >= ALL <> ALL

1. Un atributo con valor NULL solo devuelve TRUE o FALSEsi se usa este operador:

IS NULL

IS NOT NULL

1. En MySQL: NULL, 0 y FALSE tienen el valor lógico FALSE al ser usados en una expresión lógica. Cualquier otro valor tiene el valor lógicoTRUE (El valor por defecto para TRUE en un operador booleano es 1).
2. Cuando se utiliza un ORDER BY*atributo*, todos los NULL del *atributo* se agrupan en un mismo grupo.
3. When doing an ORDER BY, NULL values are presented first if you do ORDER BY ... ASC and last if you do ORDER BY ... DESC.
4. A common error when working with NULL is to assume that it is not possible to insert a zero or an empty string into a column defined as NOT NULL, but this is not the case. These are in fact values, whereas NULL means “not having a value.” You can test this easily enough by using IS [NOT] NULL
5. When using DISTINCT, GROUP BY, or ORDER BY, all NULL values are regarded as equal
6. Aggregate (summary) functions such as COUNT(), MIN(), and SUM() ignore NULL values. The exception to this is COUNT(\*), which counts rows and not individual column values. For example, the following statement produces two counts. The first is a count of the number of rows in the table, and the second is a count of the number of non-NULL values in the age column:

mysql> SELECT COUNT(\*), COUNT(age) FROM person;

For some data types, MySQL handles NULL values specially:

* If you insert NULL into a TIMESTAMP column, the current date and time is inserted.
* If you insert NULL into an integer or floating-point column that has the AUTO\_INCREMENT attribute, the next number in the sequence is inserted.

### CONCAT

Concatena cadenas de texto.

SELECT CONCAT (apellido," es ", oficio) as "OFICIO DE LOS EMPLEADOS"

FROM empleados;

### Cadena de texto

Las cadenas de textose pueden escribir entre comillas dobles o comillas simples.

### Seleccionar todos los atributos de una tabla: \*

Para seleccionar todos los atributos de una tabla:

* + MySQL: \*
  + Access: \*, ALL

#MySQL

SELECT \*

#Acces

SELECT ALL

SELECT \*

### LIKE y REGEXP (Patrones y Expresiones regulares)

#### LIKE (Patrones)Comodines de texto: % y \_

[**http://dev.mysql.com/doc/refman/5.0/es/pattern-matching.html**](http://dev.mysql.com/doc/refman/5.0/es/pattern-matching.html)

**http://dev.mysql.com/doc/refman/5.6/en/pattern-matching.html**

-- PATTERN MATCHING:

-- SQL pattern matching enables you to use “**\_**” to match any single character and “**%**” to match an arbitrary number of characters (including zero characters).

-- In MySQL, SQL patterns are **case-insensitive** by default.

-- You **do not use = or <>**when you use SQL patterns; **use the LIKE or NOT LIKE** comparison operators instead.

MySQL:

* + **%**: 0 o varios caracteres
  + **\_**: 1 caracter

# MySQL buscar los empleados cuyo apellido comienza por M o m

SELECT \*

FROM empleados

WHERE apellido LIKE 'M%'

Access:

* + **\***: 0 o varios caracteres
  + **?** 1 caracter

<https://support.office.com/es-es/article/Ejemplos-de-caracteres-comod%C3%ADn-939e153f-bd30-47e4-a763-61897c87b3f4>

# Access buscar los empleados cuyo apellido comienza por M o m

SELECT \*

FROM empleados

WHERE apellido LIKE "M\*"

#### REGEXP (Expresiones regulares):

Fuentes:

<https://dev.mysql.com/doc/refman/5.6/en/pattern-matching.html>

<https://dev.mysql.com/doc/refman/5.6/en/pattern-matching.html>

<https://dev.mysql.com/doc/refman/5.6/en/regexp.html>

<https://dev.mysql.com/doc/refman/5.6/en/regexp.html>

-- **RLIKE** es un sinónimo de **REGEXP**; ejemplos :

# son sinónimos:

*expresión* REGEXP *plantilla*, *expresión*RLIKE *plantilla*

expr NOT REGEXP pat, expr NOT RLIKE pat, NOT (expr REGEXP pat)

-- If either expr or pat is NULL, the result is NULL

-- REGEXP is not case sensitive, except when used with binary strings.

-- Because MySQL uses the C escape syntax in strings (for example, “\n” to represent the newline character), you must double any “\” that you use in your REGEXP strings.

-- The REGEXP and RLIKE operators work in byte-wise fashion, so they are not multibyte safe and may produce unexpected results with multibyte character sets. In addition, these operators compare characters by their byte values and accented characters may not compare as equal even if a given collation treats them as equal.

<https://dev.mysql.com/doc/refman/5.6/en/pattern-matching.html>

-- MySQL provides standard SQL pattern matching as well as a form of pattern matching based on extended regular expressions similar to those used by Unix utilities such as vi, grep, sed.

-- REGULAR EXPRESSIONS, ESTRUCTURA:

-- expr REGEXP pat, expr RLIKE pat

-- expr NOT REGEXP pat, expr NOT RLIKE pat, NOT (expr REGEXP pat)

-- REGULAR EXPRESSION METACHARACTERS:

A REGEXP pattern match succeeds if the pattern matches anywhere in the value being tested. (This differs from a LIKE pattern match, which succeeds only if the pattern matches the entire value.)

. matchone character, any character. Acepta un caracter, cualquier character.

? match zero or one character

+ match one or more characters

\* match zero or more characters. “\*” matches zero or more instances of the thing preceding it. For example, “p\*” matches any number of “p” character, “[0-9]\*” matches any number of digits, and “.\*” matches any number of anything.

{n} match n times

{m,n} match m through n times

{n,} match n or more times

To be more precise:

a{n} matches exactly n instances of a

a{n,} matches n or more instances of a

a{m,n} matches m through n instances of a, inclusive.

a\*Can be written as a{0,}.

a+Can be written as a{1,}.

a?Can be written as a{0,1}.

^ beginning of line

$ end of line

To anchor a pattern so that it must match the beginning or end of the value being tested, use “^” at the beginning or “$” at the end of the pattern.

[abc] match one of enclosed chars

[a-m] matches any character from a to m

[a-mw-z] matches any character from a to m or from w to z

[0-5] matches any digit from 0 to 5

[^xyz] match any char not enclosed

| separates alternatives

*[[:<:]] match beginning of words [[:>:]] match ending of words*

*[:class:] match a character class*

*i.e., [:alpha:] for letters*

*[:space:] for whitespace*

*[:punct:] for punctuation*

*[:upper:] for upper case letters*

If you really want to force a REGEXP comparison to be case sensitive, use the BINARY keyword to make one of the strings a binary string. This query matches only lowercase “b” at the beginning of a name:

mysql> SELECT \* FROM pet WHERE name REGEXP BINARY '^b';

-- EJEMPLOS:

To find names beginning with “b”:

SELECT \* FROM pet WHERE name LIKE 'b%';

SELECT \* FROM pet WHERE name REGEXP '^b';

+--------+--------+---------+------+------------+------------+

| name | owner | species | sex | birth | death |

+--------+--------+---------+------+------------+------------+

| Buffy | Harold | dog | f | 1989-05-13 | NULL |

| Bowser | Diane | dog | m | 1989-08-31 | 1995-07-29 |

+--------+--------+---------+------+------------+------------+

To find names ending with “fy”:

SELECT \* FROM pet WHERE name LIKE '%fy';

SELECT \* FROM pet WHERE name REGEXP 'fy$';

+--------+--------+---------+------+------------+-------+

| name | owner | species | sex | birth | death |

+--------+--------+---------+------+------------+-------+

| Fluffy | Harold | cat | f | 1993-02-04 | NULL |

| Buffy | Harold | dog | f | 1989-05-13 | NULL |

+--------+--------+---------+------+------------+-------+

To find names containing a “w”:

SELECT \* FROM pet WHERE name LIKE '%w%';

SELECT \* FROM pet WHERE name REGEXP 'w';

+----------+-------+---------+------+------------+------------+

| name | owner | species | sex | birth | death |

+----------+-------+---------+------+------------+------------+

| Claws | Gwen | cat | m | 1994-03-17 | NULL |

| Bowser | Diane | dog | m | 1989-08-31 | 1995-07-29 |

| Whistler | Gwen | bird | NULL | 1997-12-09 | NULL |

+----------+-------+---------+------+------------+------------+

To find names containing exactly five characters, use five instances of the “\_” pattern character:

SELECT \* FROM pet WHERE name LIKE '\_\_\_\_\_';

SELECT \* FROM pet WHERE name REGEXP '^.....$';

SELECT \* FROM pet WHERE name REGEXP '^.{5}$';

+-------+--------+---------+------+------------+-------+

| name | owner | species | sex | birth | death |

+-------+--------+---------+------+------------+-------+

| Claws | Gwen | cat | m | 1994-03-17 | NULL |

| Buffy | Harold | dog | f | 1989-05-13 | NULL |

+-------+--------+---------+------+------------+-------+

### LENGTH

Longitud de un texto.

Equivale al LEN de Access

### COUNT

SELECT COUNT(\*)

FROM emple.clientes;

1. MySQL: El paréntesis tiene que estar pegado a COUNT (sin espacios en blanco entre ambos), caso contrario falla.

Ejemplo:

SELECT COUNT(DISTINCT LOCALIDAD)

FROM emple.clientes;

Existen 3 posibilidades:

* + COUNT (\*) cuenta el número de filas de la tabla.
  + COUNT (*atributo*) cuenta el número de filas en las que el *atributo* no es NULL.
  + COUNT (DISTINCT *atributo*) cuenta el número de valores distintos y no nulos de *atributo*.

### Campo autonumérico

En un campo autonumérico, si metemos un valor a mano (se puede hacer), depende de nosotros que se haya utilizado antes o no. Si el campo es clave, tampoco se puede insertar un valor que ya existe, pero si un valor ya utilizado y borrado.

**# estas 2 lineas son equivalentes**

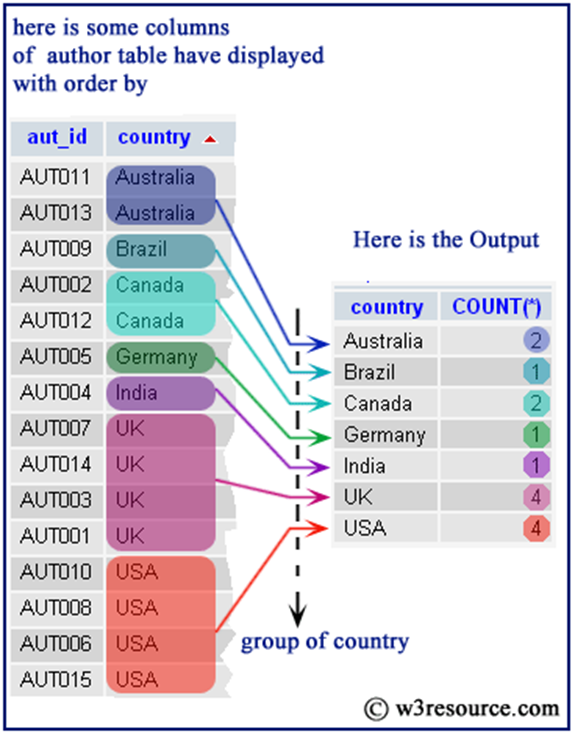
**idordentrabajo BIGINT UNSIGNED NOT NULL AUTO\_INCREMENT UNIQUE,**

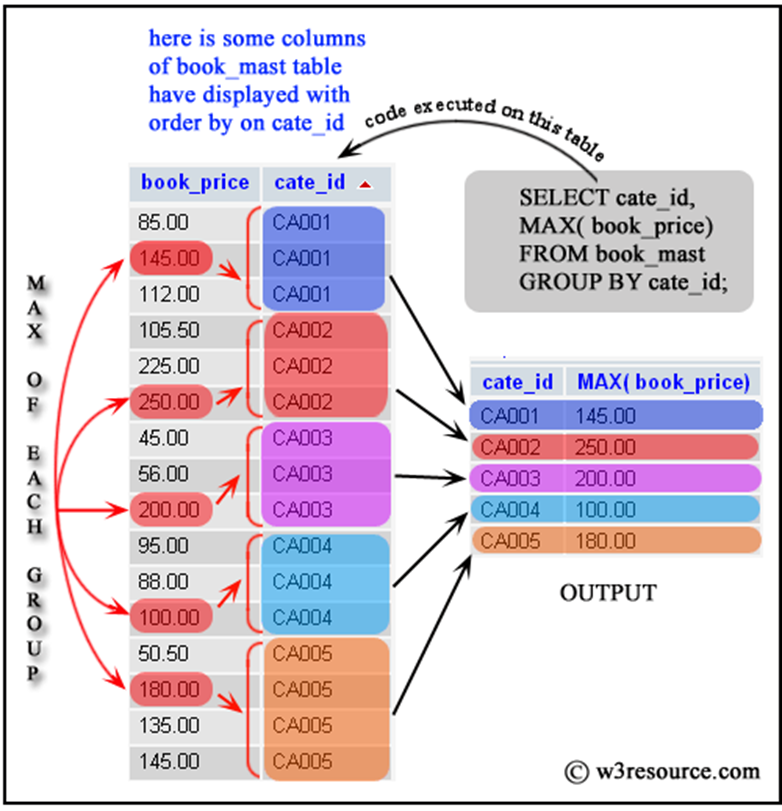
**idordentrabajo SERIAL,**

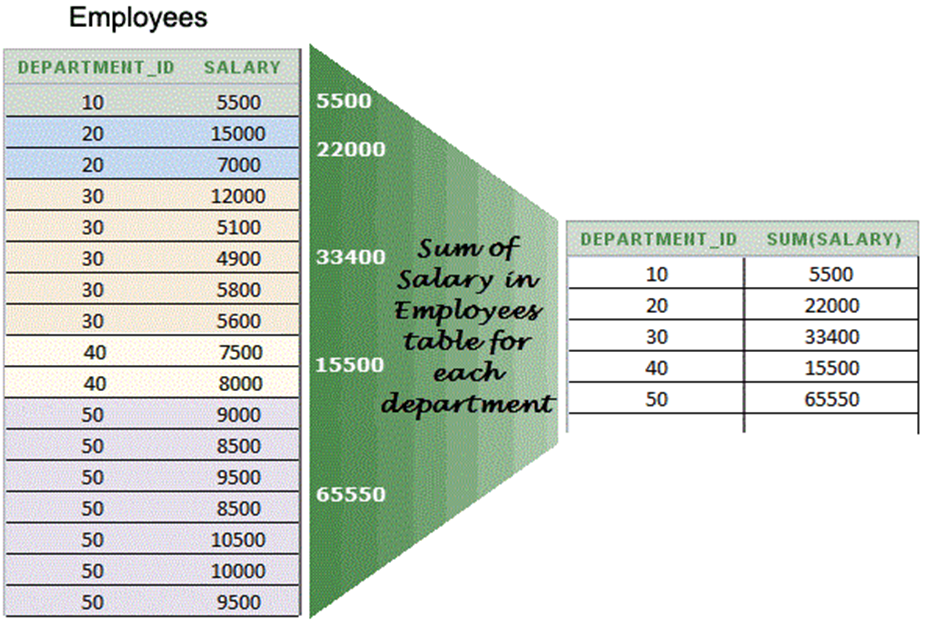
### GROUP BY

Crea una subtabla para cada valor del atributo (o atributos) indicados en GROUP BY.

1. Además crea una tabla para el valor NULL, si es que en algún registro el atributo toma el valor NULL







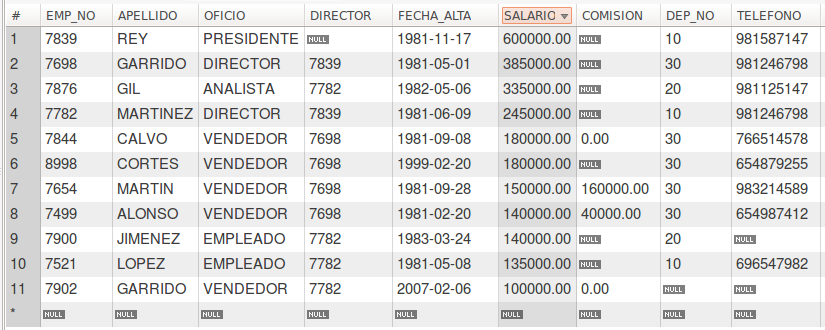
### LIMIT

[LIMIT {[offset,] row\_count | row\_count OFFSET offset}]

ORDER BY salario DESC **LIMIT 4,1**;

ORDER BY salario DESC **LIMIT 1**;

1. Mostar las row\_countfilas, que se encuentran después (sin incluir) de la posición offset.
2. Podemos utilizarla, por ejemplo, para mostrar las N primeras filas que han sido ordenadas en orden creciente (decreciente), de este modo podemos ver los N valores más bajos (altos).



#### Ejemplo: Encontrar a los empleados con el salario más alto:**Subconsulta**

Supongamos que ordenamos a los empleados por el atributo salario, en orden descendente y deseamos mostrar a las personas con el salario más alto. Con este método si 3 personas tuviesen el salario más alto, solamente se mostrará a 1 de ellas, por este motivo esta solución no vale para encontrar a los empleados que tienen el salario más alto:

SELECT \*

FROM empleados

ORDER BY salario

LIMIT 1;

Esta sí es una solución correcta:

SELECT \*

FROM empleados WHERE salario =

(

SELECT MAX(salario)

FROM empleados

);

#### Ejemplo: Encontrar a los empleados cuyos salarios están entre los 5 salarios más altos de la empresa: **LIMIT 4,1**

Supongamos que ordenamos en descendente los registros por el atributo salario y deseamos mostrar a los empleados con los 5 salarios más altos. Si hay 6 personas cuyos salarios están entre los 5 salarios más altos, este método solo mostrará a 5 de ellas, por lo que este método no sirve como solución:

SELECT salario, emp\_no, apellido

FROM empleados

ORDER BY salario DESC

LIMIT 5;

Esta sí es una solución correcta:

#guardar en la variable @quintoSalarioMasAlto el valor del quinto salario distinto más alto

SELECT **DISTINCT**salario INTO @quintoSalarioMasAlto

FROM empleados

ORDER BY salario DESC **LIMIT 4,1**;

#mostrar a los empleados cuyo salario es mayor o igual a @quintoSalarioMasAlto

SELECT \* FROM empleados

WHERE salario >= @quintoSalarioMasAlto;

Otra solución correcta:

SELECT \*

FROM empleados

WHERE salario >= (

SELECT salario

FROM empleados

ORDER BY salario DESC LIMIT 4 ,1

);

Otra solución correcta (utiliza una variable):

**#guardar en la variable @quintoSalarioMasAlto el valor del tercer salario distinto más alto**

SELECT MIN(salario) INTO @quintoSalarioMasAlto FROM

(

SELECT DISTINCT salario

FROM empleados

ORDER BY 1 DESC LIMIT 3

) AS minimos;

#mostrar a los empleados cuyo salario es mayor o igual a @quintoSalarioMasAlto

SELECT \*

FROM empleados

WHERE salario >= @quintoSalarioMasAlto;

Esta solución no funciona:

#esta no funciona porque MySQL 5.6 no permite incluir LIMIT junto con IN/ANY/SOME/ALL en una subconsulta

SELECT \* FROM empleados WHERE salario IN (

SELECT salario

FROM empleados

ORDER BY 1 DESC LIMIT 5

);

#### Ejemplo: Encontrar a los 3 empleados que más ganan en la empresa

Si hay varios tuviesen el mismo salario y eso hiciese que por ejemplo 4 de ellos estuviesen en el grupo de los 3 que más ganan, entonces debería mostrarse a los 4 empleados y no a 3 de ellos (eso sería engañoso).

SELECT salario INTO @tercerSalario

FROM empleados

ORDER BY salario DESC **LIMIT 2,1**;

SELECT @tercerSalario;

SELECT \* FROM empleados

WHERE salario >= @tercerSalario;

Aunque esta solución parece igual a la del apartado (Encontrar a los empleados cuyos salarios están entre los 5 salarios más altos de la empresa: LIMIT 4,1), pero no es así, esta no incluye DISTINCT

#### TOP (Access)

TOP es el comando usado en Access en lugar de LIMIT

[**http://www.w3schools.com/sql/sql\_top.asp**](http://www.w3schools.com/sql/sql_top.asp)

TOP **n** muestra **n** o más registros. Si hay m registros (m > n) con los n valores TOP (debido a repeticiones del valor), entonces TOP los muestra todos (muestra los m registros).

1. Ejemplo:

SELECT TOP 5 consumo

FROM Automóviles

ORDER BY 1 ASC

;

Si existen 6 coches con un consumo más bajo, entonces muestran los 6 coches, la base de datos no se atreve a descartar uno, pues no sabría cuál.

### Fechas, como trabajar con fechas

<http://forums.mysql.com/read.php?71,154336,154380>

<https://dev.mysql.com/doc/refman/5.6/en/date-and-time-functions.html>

Realizar cálculos con fechas: <http://dev.mysql.com/doc/refman/5.7/en/date-calculations.htm>

Elúnico formato con el que se pueden almacenar las fechas enMySQL es el americano:

**YYYY-MM-DD**

Para insertar una fecha es obligatorio proporcionarla en el formato YYYY-MM-DD, como separador se pueden usar símbolos: (**-./@**). Ejemplo:

INSERT INTO mytabla (fecha) VALUES ('2013-05-21'), ('2013.05.22'), ('2013/05/23'), (''2013@05@24'));

Para mostrar una fecha almacenada tenemos varias opciones:

* Mostrarla en el **formato americano**
* Especificar un **formato distinto**: función **DATE\_FORMAT**

Código de ejemplo:

CREATE TABLE `mytabla` (

`id` int(11) NOT NULL auto\_increment,

`fecha` varchar(10) default NULL,

PRIMARY KEY (`id`)

);

INSERT INTO mytabla (fecha) VALUES ('2013/5/23'), ('2013/5/24'), ('2013/5/25'), ('2013/5/26'), ('2013/5/27');

SELECT \* FROM mytabla;

SELECT id, DATE\_FORMAT(fecha, '%d/%m/%y') AS fecha FROM mytabla;

SELECT id, DATE\_FORMAT(fecha, '%d/%m/%Y') AS fecha FROM mytabla;

SELECT id, DATE\_FORMAT(fecha, '%e/%c/%Y') AS fecha FROM mytabla;

#### DATE\_FORMAT()

<http://www.w3schools.com/sql/func_date_format.asp>

Función que cambia el formato en que se muestra una fecha.

DATE\_FORMAT(date,format)

Ejemplo:

SELECT DATE\_FORMAT(NOW(),'%b %d %Y %h:%i %p')

SELECT DATE\_FORMAT(NOW(),'%m-%d-%Y')

SELECT DATE\_FORMAT(NOW(),'%d %b %y')

SELECT DATE\_FORMAT(NOW(),'%d %b %Y %T:%f')

-- salida:

Nov 04 2014 11:45 PM

11-04-2014

04 Nov 14

04 Nov 2014 11:45:34:243

Códigos de formato:

| **Formato** | **Descripción** |
| --- | --- |
| **%a** | Abbreviated weekday name (Sun-Sat) |
| **%b** | Abbreviated month name (Jan-Dec) |
| **%c** | Month, numeric (0-12) |
| **%D** | Day of month with English suffix (0th, 1st, 2nd, 3rd, ) |
| **%d** | Day of month, numeric (00-31) |
| **%e** | Day of month, numeric (0-31) |
| **%f** | Microseconds (000000-999999) |
| **%H** | Hour (00-23) |
| **%h** | Hour (01-12) |
| **%I** | Hour (01-12) |
| **%i** | Minutes, numeric (00-59) |
| **%j** | Day of year (001-366) |
| **%k** | Hour (0-23) |
| **%l** | Hour (1-12) |
| **%M** | Month name (January-December) |
| **%m** | Month, numeric (00-12) |
| **%p** | AM or PM |
| **%r** | Time, 12-hour (hh:mm:ss followed by AM or PM) |
| **%S** | Seconds (00-59) |
| **%s** | Seconds (00-59) |
| **%T** | Time, 24-hour (hh:mm:ss) |
| **%U** | Week (00-53) where Sunday is the first day of week |
| **%u** | Week (00-53) where Monday is the first day of week |
| **%V** | Week (01-53) where Sunday is the first day of week, used with %X |
| **%v** | Week (01-53) where Monday is the first day of week, used with %x |
| **%W** | Weekday name (Sunday-Saturday) |
| **%w** | Day of the week (0=Sunday, 6=Saturday) |
| **%X** | Year for the week where Sunday is the first day of week, four digits, used with %V |
| **%x** | Year for the week where Monday is the first day of week, four digits, used with %v |
| **%Y** | Year, numeric, four digits |
| **%y** | Year, numeric, two digits |

#### Realizar cálculos con fechas

<http://dev.mysql.com/doc/refman/5.7/en/date-calculations.htm>

### SELECT INTO OUTFILE: guardar en un fichero el resultado de un SELECT

1. query result to be stored in **variables** or written to a **file**:

#selects column values and stores them into variables.

**SELECT ... INTO var\_list**

#writes the selected rows to a file. Column and line terminators can be specified to produce a specific output

**SELECT ... INTO OUTFILE format.**

#writes a single row to a file without any formatting.

**SELECT ... INTO DUMPFILE**

1. An INTO clause should not be used in a nested SELECT
2. The INTO clause can name a list of one or more variables,
3. The selected **values are assigned to the variables:**

* The **number of variables must match the number of columns**.
* The query should return a **single row**.
* If the query returns **no rows**, a **warning** with error code 1329 occurs (No data), and the **variable** values remain **unchanged**.
* If the query returns **multiple rows**, **error** 1172 occurs (Result consisted of more than one row).
* If it is possible that the statement may retrieve multiple rows, you can use **LIMIT 1** to limit the result set to a single row.

SELECT id, data INTO @x, @y FROM test.t1 LIMIT 1;

1. The **SELECT ... INTO OUTFILE 'file\_name'** form of SELECT writes the selected rows to a file.

* The file is created on the server host, so you must have the FILE privilege to use this syntax.
* **file\_name cannot be an existing file**, which among other things prevents files such as */etc/passwd* and database tables from being destroyed.

1. If you want to create the resulting file on some other host than the server host, you normally cannot.

* It is also possible to create the resulting file on a different host other than the server host, if the location of the file on the remote host can be accessed using a network-mapped path on the server's file system.
* However, if the MySQL client software is installed on the remote machine, you can instead use this client command to generate the file on the client host.

mysql -e "SELECT ..." > file\_name

1. SELECT ... INTO OUTFILE is the **complement** of [LOAD DATA INFILE](#_LOAD_DATA_INFILE).
2. Ejemplos

Los siguientes comandos:

#nos situamos en una BD

USE bdempleados;

#volcamos el SELECT

SELECT \*

INTO OUTFILE 'empleados.txt'

FROM bdempleados.empleados

;

Crean el fichero en la carpeta (Windows/Linux):

/var/lib/mysql/bdempleados/empleados.txt

C:\ProgramData\MySQL\MySQL Server 5.6\data\bdempleados\empleados.txt

Si no hemos predefinido ninguna BD con USE, entonces el fichero se crea en la carpeta (Windows/Linux):

/var/lib/mysql/empleados.txt

C:\ProgramData\MySQL\MySQL Server 5.6\data\empleados.txt

Contenido delfichero:

1000 TOUCEDA VENDEDOR 7698 2007-02-06 00:00:00 100000 0 \N \N

7499 ALONSO VENDEDOR 7698 1981-02-20 00:00:00 140000 40000 30 \N

En el fichero de salida:

* + los **NULL** se escriben con **\N**
  + los **atributos se separan** con carácter de **tabulación**.

#### Especificar el formato del fichero

Es posible especificar el formato del fichero:

* + el carácter de separación de columnas: típico ;
  + el carácter de separación de filas: típico \n
  + delimitador de texto: entre que caracteres (normalmente comillas simples o dobles) se guardan los tipos de datos texto y fecha.

#nos situamos en una BD

USE bdempleados;

#volcamos el SELECT

SELECT \*

INTO OUTFILE 'empleadosConSeparadoresEspeciales.txt'

FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY '\''

LINES TERMINATED BY '\n'

FROM bdempleados.empleados

;

Para cargar ese fichero con formato especial en una tabla se ejecutaría este comando:

LOAD DATA INFILE 'empleadosConSeparadoresEspeciales.txt'

INTO TABLE empleados

FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY '\''

LINES TERMINATED BY '\n'

;

#### mysql -e "SELECT ..." > file\_name : Otra manera de volcar una tabla en un fichero

However, if the MySQL client software is installed on the remote machine, you can instead use a client command such as

mysql -e "SELECT ..." > file\_name

to generate the file on the client host.

1. Ejemplo

profesor@ubu:~$ **mysql -u root -pAbcd1234. bdempleados -e "SELECT \* FROM empleados" > empleadosDesdeConsola.txt**

#### ERROR: The MySQL server is running with the --secure-file-priv option so it cannot execute this statement

El comando falla y produce un error:

**#volcamos el resultado del SELECT en un fichero con la opción INTO OUTFILE**

**SELECT \***

**FROM bdempleados.pedidos**

**WHERE YEAR(FECHA\_PEDIDO) = '2000'**

**INTO OUTFILE '*pedidos2000servidor.txt*'**

**;**

**#produce este error:**

**Error Code: 1290. The MySQL server is running with the --secure-file-priv option so it cannot execute this statement**

explicación del error: buscamos el significado de esa variable de sistema en:

[**https://dev.mysql.com/doc/refman/8.0/en/server-system-variable-reference.html**](https://dev.mysql.com/doc/refman/8.0/en/server-system-variable-reference.html)

**#pulsamos en el hiperenlace de esa variable y no lleva aquí:**

[**https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar\_secure\_file\_priv**](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_secure_file_priv)

consultamos al servidor cuál es el valor de la variable global de sistema que produce este error:

**SELECT @@secure\_file\_priv;**

**#nos indica que la carpeta donde se pueden guardar y leer ficheros es: '/var/lib/mysql-files/'**

por tanto debemos cambiar el comando anterior para que escriba el fichero en esa carpeta:

**SELECT \***

**FROM bdempleados.pedidos**

**WHERE YEAR(FECHA\_PEDIDO) = '2000'**

**INTO OUTFILE '/var/lib/mysql-files/pedidos2000servidor.txt'**

**;**

Ahora sí funciona, vemos que se ha creado el fichero y es correcto:

**sudo ls /var/lib/mysql-files/ -al**

**drwxrwx--- 2 mysql mysql 4096 ene 15 09:18 .**

**drwxr-xr-x 67 root root 4096 oct 15 09:18 ..**

**-rw-r----- 1 mysql mysql 186 ene 15 09:18 pedidos2000servidor2.txt**

**-rw-r----- 1 mysql mysql 246 ene 15 09:15 pedidos2000servidor.txt**

##### Soclución alternativa, cambiar configuración del servidor:

Una solución alternativa sería cambiar el valor de la variable del servidor:

<http://stackoverflow.com/questions/32737478/how-should-i-tackle-secure-file-priv-in-mysql>

1. Solución en servidor sobre Windows:
2. Stop the MySQL server service by going into services.msc.
3. Go to C:\ProgramData\MySQL\MySQL Server 5.6 (ProgramData was a hidden folder in my case).
4. Open the my.ini file in Notepad.
5. Search for 'secure-file-priv'.
6. Comment the line out by adding '#' at the start of the line.
7. Save the file.
8. Start the MySQL server service by going into services.msc
9. Solución en servidor sobre Linux:

On Ubuntu 14 and Mysql 5.5.53 this setting seems to be enabled by default. To disable it you need to add secure-file-priv = "" to your **my.cnf** file under the mysqld config group. eg:

[mysqld]

secure-file-priv=""

## Funciones en MySQL

<http://dev.mysql.com/doc/refman/5.7/en/functions.html>

<http://dev.mysql.com/doc/refman/5.7/en/func-op-summary-ref.html>

### Table 12.2 Operators

|  |  |
| --- | --- |
| Name | Description |
| AND, && | Logical AND |
| = | Assign a value (as part of a SET statement, or as part of the SET clause in an UPDATE statement) |
| := | Assign a value |
| BETWEEN ... AND ... | Check whether a value is within a range of values |
| BINARY | Cast a string to a binary string |
| & | Bitwise AND |
| ~ | Bitwise inversion |
| | | Bitwise OR |
| ^ | Bitwise XOR |
| CASE | Case operator |
| DIV | Integer division |
| / | Division operator |
| <=> | NULL-safe equal to operator |
| = | Equal operator |
| >= | Greater than or equal operator |
| > | Greater than operator |
| IS NOT NULL | NOT NULL value test |
| IS NOT | Test a value against a boolean |
| IS NULL | NULL value test |
| IS | Test a value against a boolean |
| -> | Return value from JSON column after evaluating path |
| << | Left shift |
| <= | Less than or equal operator |
| < | Less than operator |
| LIKE | Simple pattern matching |
| - | Minus operator |
| %, MOD | Modulo operator |
| NOT BETWEEN ... AND ... | Check whether a value is not within a range of values |
| !=, <> | Not equal operator |
| NOT LIKE | Negation of simple pattern matching |
| NOT REGEXP | Negation of REGEXP |
| NOT, ! | Negates value |
| ||, OR | Logical OR |
| + | Addition operator |
| REGEXP | Pattern matching using regular expressions |
| >> | Right shift |
| RLIKE | Synonym for REGEXP |
| SOUNDS LIKE | Compare sounds |
| \* | Multiplication operator |
| - | Change the sign of the argument |
| XOR | Logical XOR |
| Name | Description |

### Table 12.3 Comparison Operators

|  |  |
| --- | --- |
| Name | Description |
| BETWEEN ... AND ... | Check whether a value is within a range of values |
| COALESCE() | Return the first non-NULL argument |
| <=> | NULL-safe equal to operator |
| = | Equal operator |
| >= | Greater than or equal operator |
| > | Greater than operator |
| GREATEST() | Return the largest argument |
| IN() | Check whether a value is within a set of values |
| INTERVAL() | Return the index of the argument that is less than the first argument |
| IS NOT NULL | NOT NULL value test |
| IS NOT | Test a value against a boolean |
| IS NULL | NULL value test |
| IS | Test a value against a boolean |
| ISNULL() | Test whether the argument is NULL |
| LEAST() | Return the smallest argument |
| <= | Less than or equal operator |
| < | Less than operator |
| LIKE | Simple pattern matching |
| NOT BETWEEN ... AND ... | Check whether a value is not within a range of values |
| !=, <> | Not equal operator |
| NOT IN() | Check whether a value is not within a set of values |
| NOT LIKE | Negation of simple pattern matching |
| STRCMP() | Compare two strings |
| Name | Description |

### Table 12.4 Logical Operators

|  |  |
| --- | --- |
| Name | Description |
| AND, && | Logical AND |
| NOT, ! | Negates value |
| ||, OR | Logical OR |
| XOR | Logical XOR |

### Table 12.5 Assignment Operators

|  |  |
| --- | --- |
| Name | Description |
| = | Assign a value (as part of a SET statement, or as part of the SET clause in an UPDATE statement) |
| := | Assign a value |

### Table 12.6 Flow Control Operators

|  |  |
| --- | --- |
| Name | Description |
| CASE | Case operator |
| IF() | If/else construct |
| IFNULL() | Null if/else construct |
| NULLIF() | Return NULL if expr1 = expr2 |

### Table 12.7 String Operators

|  |  |
| --- | --- |
| Name | Description |
| ASCII() | Return numeric value of left-most character |
| BIN() | Return a string containing binary representation of a number |
| BIT\_LENGTH() | Return length of argument in bits |
| CHAR\_LENGTH() | Return number of characters in argument |
| CHAR() | Return the character for each integer passed |
| CHARACTER\_LENGTH() | Synonym for CHAR\_LENGTH() |
| CONCAT\_WS() | Return concatenate with separator |
| CONCAT() | Return concatenated string |
| ELT() | Return string at index number |
| EXPORT\_SET() | Return a string such that for every bit set in the value bits, you get an on string and for every unset bit, you get an off string |
| FIELD() | Return the index (position) of the first argument in the subsequent arguments |
| FIND\_IN\_SET() | Return the index position of the first argument within the second argument |
| FORMAT() | Return a number formatted to specified number of decimal places |
| FROM\_BASE64() | Decode to a base-64 string and return result |
| HEX() | Return a hexadecimal representation of a decimal or string value |
| INSERT() | Insert a substring at the specified position up to the specified number of characters |
| INSTR() | Return the index of the first occurrence of substring |
| LCASE() | Synonym for LOWER() |
| LEFT() | Return the leftmost number of characters as specified |
| LENGTH() | Return the length of a string in bytes |
| LIKE | Simple pattern matching |
| LOAD\_FILE() | Load the named file |
| LOCATE() | Return the position of the first occurrence of substring |
| LOWER() | Return the argument in lowercase |
| LPAD() | Return the string argument, left-padded with the specified string |
| LTRIM() | Remove leading spaces |
| MAKE\_SET() | Return a set of comma-separated strings that have the corresponding bit in bits set |
| MATCH | Perform full-text search |
| MID() | Return a substring starting from the specified position |
| NOT LIKE | Negation of simple pattern matching |
| NOT REGEXP | Negation of REGEXP |
| OCT() | Return a string containing octal representation of a number |
| OCTET\_LENGTH() | Synonym for LENGTH() |
| ORD() | Return character code for leftmost character of the argument |
| POSITION() | Synonym for LOCATE() |
| QUOTE() | Escape the argument for use in an SQL statement |
| REGEXP | Pattern matching using regular expressions |
| REPEAT() | Repeat a string the specified number of times |
| REPLACE() | Replace occurrences of a specified string |
| REVERSE() | Reverse the characters in a string |
| RIGHT() | Return the specified rightmost number of characters |
| RLIKE | Synonym for REGEXP |
| RPAD() | Append string the specified number of times |
| RTRIM() | Remove trailing spaces |
| SOUNDEX() | Return a soundex string |
| SOUNDS LIKE | Compare sounds |
| SPACE() | Return a string of the specified number of spaces |
| STRCMP() | Compare two strings |
| SUBSTR() | Return the substring as specified |
| SUBSTRING\_INDEX() | Return a substring from a string before the specified number of occurrences of the delimiter |
| SUBSTRING() | Return the substring as specified |
| TO\_BASE64() | Return the argument converted to a base-64 string |
| TRIM() | Remove leading and trailing spaces |
| UCASE() | Synonym for UPPER() |
| UNHEX() | Return a string containing hex representation of a number |
| UPPER() | Convert to uppercase |
| WEIGHT\_STRING() | Return the weight string for a string |
| Name | Description |

### Table 12.8 String Comparison Operators

|  |  |
| --- | --- |
| Name | Description |
| LIKE | Simple pattern matching |
| NOT LIKE | Negation of simple pattern matching |
| STRCMP() | Compare two strings |

### Table 12.9 String Regular Expression Operators

|  |  |
| --- | --- |
| Name | Description |
| NOT REGEXP | Negation of REGEXP |
| REGEXP | Pattern matching using regular expressions |
| RLIKE | Synonym for REGEXP |

### Table 12.10 Numeric Functions and Operators

|  |  |
| --- | --- |
| Name | Description |
| ABS() | Return the absolute value |
| ACOS() | Return the arc cosine |
| ASIN() | Return the arc sine |
| ATAN2(), ATAN() | Return the arc tangent of the two arguments |
| ATAN() | Return the arc tangent |
| CEIL() | Return the smallest integer value not less than the argument |
| CEILING() | Return the smallest integer value not less than the argument |
| CONV() | Convert numbers between different number bases |
| COS() | Return the cosine |
| COT() | Return the cotangent |
| CRC32() | Compute a cyclic redundancy check value |
| DEGREES() | Convert radians to degrees |
| DIV | Integer division |
| / | Division operator |
| EXP() | Raise to the power of |
| FLOOR() | Return the largest integer value not greater than the argument |
| LN() | Return the natural logarithm of the argument |
| LOG10() | Return the base-10 logarithm of the argument |
| LOG2() | Return the base-2 logarithm of the argument |
| LOG() | Return the natural logarithm of the first argument |
| - | Minus operator |
| MOD() | Return the remainder |
| %, MOD | Modulo operator |
| PI() | Return the value of pi |
| + | Addition operator |
| POW() | Return the argument raised to the specified power |
| POWER() | Return the argument raised to the specified power |
| RADIANS() | Return argument converted to radians |
| RAND() | Return a random floating-point value |
| ROUND() | Round the argument |
| SIGN() | Return the sign of the argument |
| SIN() | Return the sine of the argument |
| SQRT() | Return the square root of the argument |
| TAN() | Return the tangent of the argument |
| \* | Multiplication operator |
| TRUNCATE() | Truncate to specified number of decimal places |
| - | Change the sign of the argument |
| Name | Description |

### Table 12.11 Arithmetic Operators

|  |  |
| --- | --- |
| Name | Description |
| DIV | Integer division |
| / | Division operator |
| - | Minus operator |
| %, MOD | Modulo operator |
| + | Addition operator |
| \* | Multiplication operator |
| - | Change the sign of the argument |

### Table 12.12 Mathematical Functions

|  |  |
| --- | --- |
| Name | Description |
| ABS() | Return the absolute value |
| ACOS() | Return the arc cosine |
| ASIN() | Return the arc sine |
| ATAN2(), ATAN() | Return the arc tangent of the two arguments |
| ATAN() | Return the arc tangent |
| CEIL() | Return the smallest integer value not less than the argument |
| CEILING() | Return the smallest integer value not less than the argument |
| CONV() | Convert numbers between different number bases |
| COS() | Return the cosine |
| COT() | Return the cotangent |
| CRC32() | Compute a cyclic redundancy check value |
| DEGREES() | Convert radians to degrees |
| EXP() | Raise to the power of |
| FLOOR() | Return the largest integer value not greater than the argument |
| LN() | Return the natural logarithm of the argument |
| LOG10() | Return the base-10 logarithm of the argument |
| LOG2() | Return the base-2 logarithm of the argument |
| LOG() | Return the natural logarithm of the first argument |
| MOD() | Return the remainder |
| PI() | Return the value of pi |
| POW() | Return the argument raised to the specified power |
| POWER() | Return the argument raised to the specified power |
| RADIANS() | Return argument converted to radians |
| RAND() | Return a random floating-point value |
| ROUND() | Round the argument |
| SIGN() | Return the sign of the argument |
| SIN() | Return the sine of the argument |
| SQRT() | Return the square root of the argument |
| TAN() | Return the tangent of the argument |
| TRUNCATE() | Truncate to specified number of decimal places |
| Name | Description |

### Table 12.13 Date/Time Functions

|  |  |
| --- | --- |
| Name | Description |
| ADDDATE() | Add time values (intervals) to a date value |
| ADDTIME() | Add time |
| CONVERT\_TZ() | Convert from one timezone to another |
| CURDATE() | Return the current date |
| CURRENT\_DATE(), CURRENT\_DATE | Synonyms for CURDATE() |
| CURRENT\_TIME(), CURRENT\_TIME | Synonyms for CURTIME() |
| CURRENT\_TIMESTAMP(), CURRENT\_TIMESTAMP | Synonyms for NOW() |
| CURTIME() | Return the current time |
| DATE\_ADD() | Add time values (intervals) to a date value |
| DATE\_FORMAT() | Format date as specified |
| DATE\_SUB() | Subtract a time value (interval) from a date |
| DATE() | Extract the date part of a date or datetime expression |
| DATEDIFF() | Subtract two dates |
| DAY() | Synonym for DAYOFMONTH() |
| DAYNAME() | Return the name of the weekday |
| DAYOFMONTH() | Return the day of the month (0-31) |
| DAYOFWEEK() | Return the weekday index of the argument |
| DAYOFYEAR() | Return the day of the year (1-366) |
| EXTRACT() | Extract part of a date |
| FROM\_DAYS() | Convert a day number to a date |
| FROM\_UNIXTIME() | Format UNIX timestamp as a date |
| GET\_FORMAT() | Return a date format string |
| HOUR() | Extract the hour |
| LAST\_DAY | Return the last day of the month for the argument |
| LOCALTIME(), LOCALTIME | Synonym for NOW() |
| LOCALTIMESTAMP, LOCALTIMESTAMP() | Synonym for NOW() |
| MAKEDATE() | Create a date from the year and day of year |
| MAKETIME() | Create time from hour, minute, second |
| MICROSECOND() | Return the microseconds from argument |
| MINUTE() | Return the minute from the argument |
| MONTH() | Return the month from the date passed |
| MONTHNAME() | Return the name of the month |
| NOW() | Return the current date and time |
| PERIOD\_ADD() | Add a period to a year-month |
| PERIOD\_DIFF() | Return the number of months between periods |
| QUARTER() | Return the quarter from a date argument |
| SEC\_TO\_TIME() | Converts seconds to 'HH:MM:SS' format |
| SECOND() | Return the second (0-59) |
| STR\_TO\_DATE() | Convert a string to a date |
| SUBDATE() | Synonym for DATE\_SUB() when invoked with three arguments |
| SUBTIME() | Subtract times |
| SYSDATE() | Return the time at which the function executes |
| TIME\_FORMAT() | Format as time |
| TIME\_TO\_SEC() | Return the argument converted to seconds |
| TIME() | Extract the time portion of the expression passed |
| TIMEDIFF() | Subtract time |
| TIMESTAMP() | With a single argument, this function returns the date or datetime expression; with two arguments, the sum of the arguments |
| TIMESTAMPADD() | Add an interval to a datetime expression |
| TIMESTAMPDIFF() | Subtract an interval from a datetime expression |
| TO\_DAYS() | Return the date argument converted to days |
| TO\_SECONDS() | Return the date or datetime argument converted to seconds since Year 0 |
| UNIX\_TIMESTAMP() | Return a UNIX timestamp |
| UTC\_DATE() | Return the current UTC date |
| UTC\_TIME() | Return the current UTC time |
| UTC\_TIMESTAMP() | Return the current UTC date and time |
| WEEK() | Return the week number |
| WEEKDAY() | Return the weekday index |
| WEEKOFYEAR() | Return the calendar week of the date (1-53) |
| YEAR() | Return the year |
| YEARWEEK() | Return the year and week |
| Name | Description |

### Table 12.14 Cast Functions

|  |  |
| --- | --- |
| Name | Description |
| BINARY | Cast a string to a binary string |
| CAST() | Cast a value as a certain type |
| CONVERT() | Cast a value as a certain type |

### Table 12.15 XML Functions

|  |  |
| --- | --- |
| Name | Description |
| ExtractValue() | Extracts a value from an XML string using XPath notation |
| UpdateXML() | Return replaced XML fragment |

### Table 12.16 Bit Functions and Operators

|  |  |
| --- | --- |
| Name | Description |
| BIT\_COUNT() | Return the number of bits that are set |
| & | Bitwise AND |
| ~ | Bitwise inversion |
| | | Bitwise OR |
| ^ | Bitwise XOR |
| << | Left shift |
| >> | Right shift |

### Table 12.17 Encryption Functions

|  |  |
| --- | --- |
| Name | Description |
| AES\_DECRYPT() | Decrypt using AES |
| AES\_ENCRYPT() | Encrypt using AES |
| COMPRESS() | Return result as a binary string |
| DECODE() | Decodes a string encrypted using ENCODE() |
| DES\_DECRYPT() (deprecated 5.7.6) | Decrypt a string |
| DES\_ENCRYPT() (deprecated 5.7.6) | Encrypt a string |
| ENCODE() | Encode a string |
| ENCRYPT() (deprecated 5.7.6) | Encrypt a string |
| MD5() | Calculate MD5 checksum |
| OLD\_PASSWORD() | Return the value of the pre-4.1 implementation of PASSWORD |
| PASSWORD() (deprecated 5.7.6) | Calculate and return a password string |
| RANDOM\_BYTES() | Return a random byte vector |
| SHA1(), SHA() | Calculate an SHA-1 160-bit checksum |
| SHA2() | Calculate an SHA-2 checksum |
| UNCOMPRESS() | Uncompress a string compressed |
| UNCOMPRESSED\_LENGTH() | Return the length of a string before compression |
| VALIDATE\_PASSWORD\_STRENGTH() | Determine strength of password |

### Table 12.18 Information Functions

|  |  |
| --- | --- |
| Name | Description |
| BENCHMARK() | Repeatedly execute an expression |
| CHARSET() | Return the character set of the argument |
| COERCIBILITY() | Return the collation coercibility value of the string argument |
| COLLATION() | Return the collation of the string argument |
| CONNECTION\_ID() | Return the connection ID (thread ID) for the connection |
| CURRENT\_USER(), CURRENT\_USER | The authenticated user name and host name |
| DATABASE() | Return the default (current) database name |
| FOUND\_ROWS() | For a SELECT with a LIMIT clause, the number of rows that would be returned were there no LIMIT clause |
| LAST\_INSERT\_ID() | Value of the AUTO\_INCREMENT column for the last INSERT |
| ROW\_COUNT() | The number of rows updated |
| SCHEMA() | Synonym for DATABASE() |
| SESSION\_USER() | Synonym for USER() |
| SYSTEM\_USER() | Synonym for USER() |
| USER() | The user name and host name provided by the client |
| VERSION() | Return a string that indicates the MySQL server version |

### Table 12.25 Miscellaneous Functions

|  |  |
| --- | --- |
| Name | Description |
| ANY\_VALUE() | Suppress ONLY\_FULL\_GROUP\_BY value rejection |
| DEFAULT() | Return the default value for a table column |
| GET\_LOCK() | Get a named lock |
| INET\_ATON() | Return the numeric value of an IP address |
| INET\_NTOA() | Return the IP address from a numeric value |
| INET6\_ATON() | Return the numeric value of an IPv6 address |
| INET6\_NTOA() | Return the IPv6 address from a numeric value |
| IS\_FREE\_LOCK() | Checks whether the named lock is free |
| IS\_IPV4\_COMPAT() | Return true if argument is an IPv4-compatible address |
| IS\_IPV4\_MAPPED() | Return true if argument is an IPv4-mapped address |
| IS\_IPV4() | Return true if argument is an IPv4 address |
| IS\_IPV6() | Return true if argument is an IPv6 address |
| IS\_USED\_LOCK() | Checks whether the named lock is in use. Return connection identifier if true. |
| MASTER\_POS\_WAIT() | Block until the slave has read and applied all updates up to the specified position |
| NAME\_CONST() | Causes the column to have the given name |
| RAND() | Return a random floating-point value |
| RELEASE\_ALL\_LOCKS() | Releases all current named locks |
| RELEASE\_LOCK() | Releases the named lock |
| SLEEP() | Sleep for a number of seconds |
| UUID\_SHORT() | Return an integer-valued universal identifier |
| UUID() | Return a Universal Unique Identifier (UUID) |
| VALUES() | Defines the values to be used during an INSERT |

### Table 12.26 Aggregate (GROUP BY) Functions

|  |  |
| --- | --- |
| Name | Description |
| AVG() | Return the average value of the argument |
| BIT\_AND() | Return bitwise AND |
| BIT\_OR() | Return bitwise OR |
| BIT\_XOR() | Return bitwise XOR |
| COUNT(DISTINCT) | Return the count of a number of different values |
| COUNT() | Return a count of the number of rows returned |
| GROUP\_CONCAT() | Return a concatenated string |
| MAX() | Return the maximum value |
| MIN() | Return the minimum value |
| STD() | Return the population standard deviation |
| STDDEV\_POP() | Return the population standard deviation |
| STDDEV\_SAMP() | Return the sample standard deviation |
| STDDEV() | Return the population standard deviation |
| SUM() | Return the sum |
| VAR\_POP() | Return the population standard variance |
| VAR\_SAMP() | Return the sample variance |
| VARIANCE() | Return the population standard variance |

### Funciones de cadena

**<file:///M:/__Clase/__Curso_2014-15/_BASES_DE_DATOS/SQL.y.MySQL-www.nachocabanes.com/sql09.html>**

## Backup and Recovery

<https://dev.mysql.com/doc/refman/5.6/en/backup-and-recovery.html>

# Criterios para diseña una BD

### Por Qué es mejor evitar que los atributos permitan NULL

<https://www.safaribooksonline.com/library/view/high-performance-mysql/9781449332471/ch04.html>

Avoid NULL if possible.

A lot of tables include nullable columns even when the application does not need to store NULL (the absence of a value), merely because it’s the default. It’s usually best to specify columns as NOT NULL unless you intend to store NULL in them.

It’s harder for MySQL to optimize queries that refer to nullable columns, because they make indexes, index statistics, and value comparisons more complicated. A nullable column uses more storage space and requires special processing inside MySQL. When a nullable column is indexed, it requires an extra byte per entry and can even cause a fixed-size index (such as an index on a single integer column) to be converted to a variable-sized one in MyISAM.

The performance improvement from changing NULL columns to NOT NULL is usually small, so don’t make it a priority to find and change them on an existing schema unless you know they are causing problems. However, if you’re planning to index columns, avoid making them nullable if possible.

There are exceptions, of course. For example, it’s worth mentioning that InnoDB stores NULL with a single bit, so it can be pretty space-efficient for sparsely populated data. This doesn’t apply to MyISAM, though.