



## Mysql. Nivel I

mayo, 2019



## Objetivos del nivel

- Aprender a crear bases de datos
- Crear tablas
- Manipular datos con instrucciones SQL
- Administrar la perisología en el servidor

## Prerrequisitos del nivel

Este nivel del curso no tiene pre requisitos

## Acerca de este manual

Este manual pertenece al Centro de Asesoramiento y Desarrollo Informático C.A. (CADI F1). Para obtener más información sobre este u otros cursos visite nuestro sitio Web [www.cadif1.com](http://www.cadif1.com), escribanos a la dirección de correo [cadi@cadif1.com](mailto:cadi@cadif1.com) o visítenos en nuestra sede ubicada en la Av. Pedro León Torres con calle 59, Centro Comercial Sotavento, piso 2 oficina 27, Barquisimeto estado Lara, Venezuela. Tlf. 0251-7179247, 0251-4410268.

Las marcas mencionadas en este manual son propiedad de sus respectivos dueños.  
Copyright 2019. Todos los derechos reservados.

ACADEMIA DE SOFTWARE



## Contenido del nivel

### Capítulo 1. MySQL

- 1.1.- Conociendo Mysql.
- 1.2.- Conexión Local.
- 1.3.- Creación de Bases de Datos.

### Capítulo 2. MySQL Workbench

- 2.1.- Conociendo la Interfaz.
- 2.2.- Administrador de Conexiones.
- 2.3.- Crear Bases de Datos.

### Capítulo 3. Configuración de MySQL

- 3.1.- Archivo de Configuración.
- 3.2.- Contraseña de Root.
- 3.3.- Crear Usuarios.

### Capítulo 4. Trabajando Con Modelos. Parte 1

- 4.1.- Crear un Modelo.
- 4.2.- Crear Tablas.
- 4.3.- Forward Engineer.

### Capítulo 5. Creación de Tablas

- 5.1.- Parámetros Básicos.
- 5.2.- Definición de Campos.
- 5.3.- Creación de índices.

### Capítulo 6. Manipulando Registros

- 6.1.- Listado de Registros.
- 6.2.- Agregar, Modificar y Eliminar Registros.
- 6.3.- Reiniciar el Auto Incremento.

## Capítulo 7. Tipos de Datos

- 7.1.- Tipos de Datos Numéricos.
- 7.2.- Datos de Fecha.
- 7.3.- Datos de Texto.
- 7.4.- Datos en Binario.
- 7.5.- Conjuntos y Enumerados.

## Capítulo 8. Exportar e Importar

- 8.1.- Mysqldump.
- 8.2.- Exportar en Mysql Workbench.
- 8.3.- Importar en Mysql Workbench.

## Capítulo 9. Trabajando Con Modelos. Parte 2

- 9.1.- Crear un Mer.
- 9.2.- Relaciones Entre Tablas.
- 9.3.- Sincronizar la Base de Datos.

## Capítulo 10. Definición de Claves Foráneas

- 10.1.- Definir Una Clave Foránea.
- 10.2.- Restricciones de Clave Foránea.
- 10.3.- Ingeniería Inversa.

## Capítulo 11. Trabajando Con Instrucciones Sql

- 11.1.- Ejecutar Instrucciones Sql.
- 11.2.- Trabajando Con Scripts.
- 11.3.- Trabajando Con Vistas.

## Capítulo 12. Funciones de Mysql

- 12.1.- Funciones y Operadores.
- 12.2.- Funciones de Cadena.

- 12.3.- Funciones de Fecha.
- 12.4.- Funciones de Números.

#### Capítulo 13. Stored Procedures. Parte 1

- 13.1.- Crear Store Procedures.
- 13.2.- El Delimitador.
- 13.3.- Ejecutar Procedures.

#### Capítulo 14. Stored Procedures. Parte 2

- 14.1.- Trabajando Con Variables.
- 14.2.- Select Into.
- 14.3.- Paso de Parámetros.
- 14.4.- Funciones.



## Capítulo 1. MYSQL

### 1.1.- Conociendo Mysql

Es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado lo ofrece bajo la GNU GPL, pero, empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia que les permita ese uso. Está desarrollado en su mayor parte en ANSI C.

MySQL funciona sobre múltiples plataformas, incluyendo AIX, BSD, FreeBSD, HP-UX, GNU/Linux, Mac OS X, NetBSD, Novell Netware, OpenBSD, OS/2 Warp, QNX, SGI IRIX, Solaris, SunOS, SCO OpenServer, SCO UnixWare, Tru64, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows Vista y otras versiones de Windows.



MariaDB es un sistema de gestión de bases de datos derivado de MySQL con licencia GPL (General Public License). Es desarrollado por Michael Widenius —fundador de MySQL—, la fundación MariaDB y la comunidad de desarrolladores de software libre. Tiene una alta compatibilidad con MySQL ya que posee las mismas órdenes, interfaces, API y bibliotecas, siendo su objetivo poder cambiar un servidor por otro directamente.

Este SGBD surge a raíz de la compra de Sun Microsystems —compañía que había comprado previamente MySQL AB— por parte de Oracle. MariaDB es una bifurcación directa de MySQL que asegura la existencia de una versión de este producto con licencia GPL. Widenius decidió crear esta variante porque estaba convencido de que el único interés de Oracle en MySQL era reducir la competencia que MySQL suponía para el mayor proveedor de bases de datos relacionales del mundo, que es Oracle.

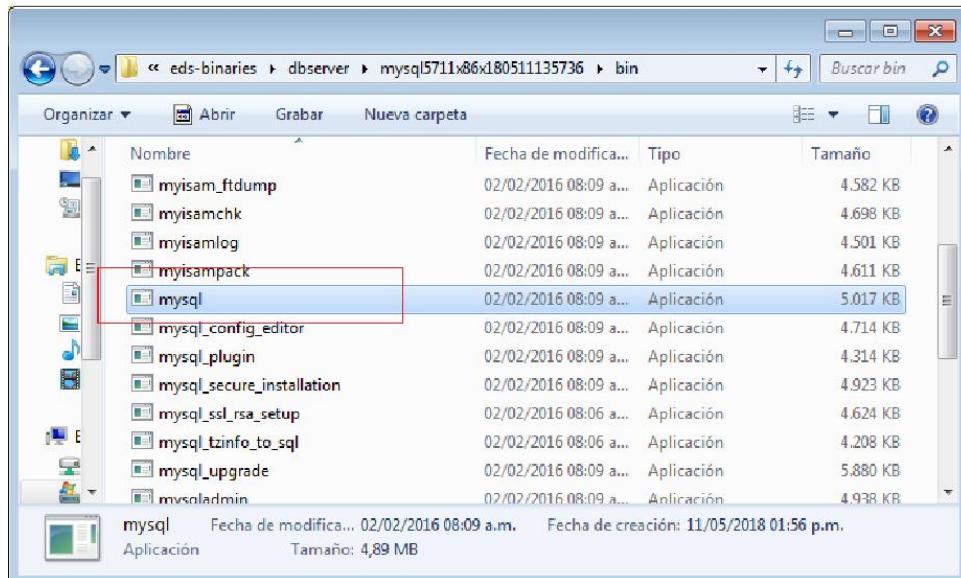


#### 1.2.- Conexión Local

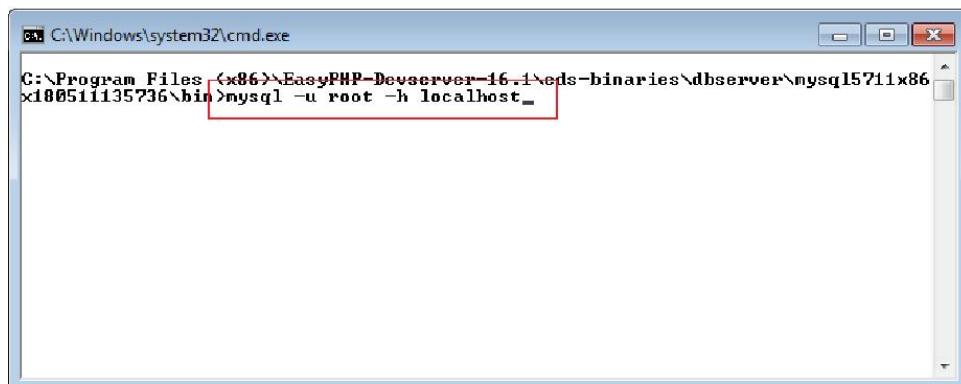
### ACADEMIA DE SOFTWARE

Al instalar el servidor MySQL lo único que se instala para crear bases de datos es un programa de consola llamado MySQL. El problema de usar este programa es que todas las tareas se realizan ejecutando instrucciones SQL, lo que puede hacer muy complicada y lenta la tarea de crear bases de datos, tablas y manipular registros.

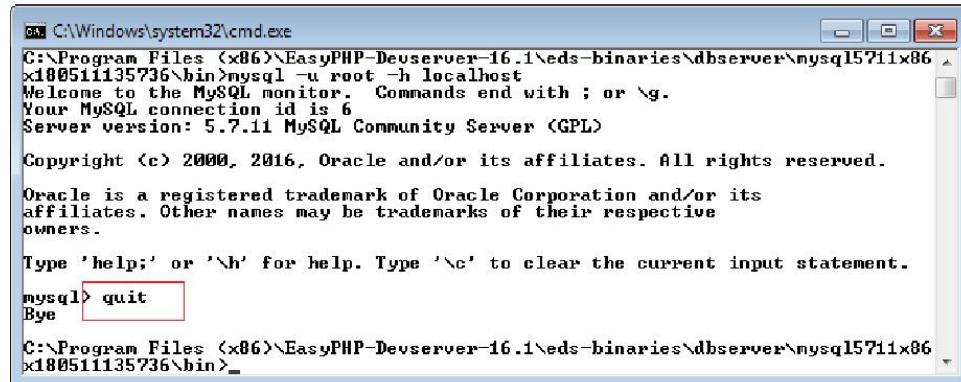
Se debe conocer la ruta donde está ubicado el cliente de MySQL. Conociendo donde se instaló el servidor, se ubica la carpeta "bin":



Para conectar con un servidor MySQL, se ejecuta el comando "mysql" usando los parámetros "-u" y "-h", que permite indicar el nombre de usuario y el host donde se está ejecutando el servidor. En el siguiente ejemplo se intenta conectar al servidor que se ejecuta localmente (localhost, también se puede usar la dirección 127.0.0.1), con las credenciales del usuario "root":



Para salir de la consola de MySQL se utiliza el comando "quit" o "exit":



```
C:\Windows\system32\cmd.exe
C:\Program Files (<x86>)\EasyPHP-Devserver-16.1\eds-binaries\dbserver\mysql15711x86
>180511135736\bin>mysql -u root -h localhost
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.7.11 MySQL Community Server (GPL)

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

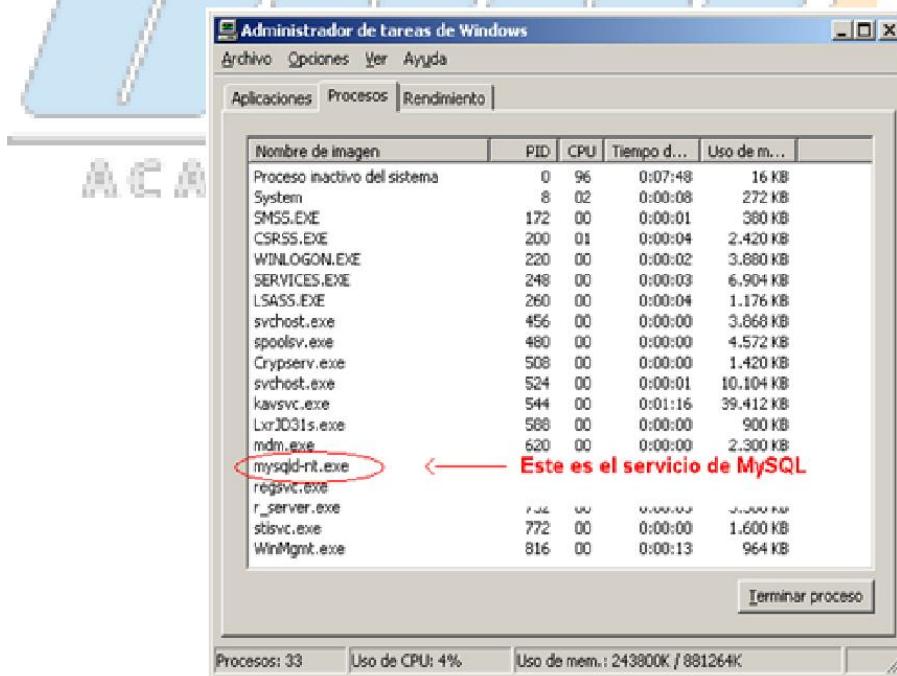
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> quit
Bye

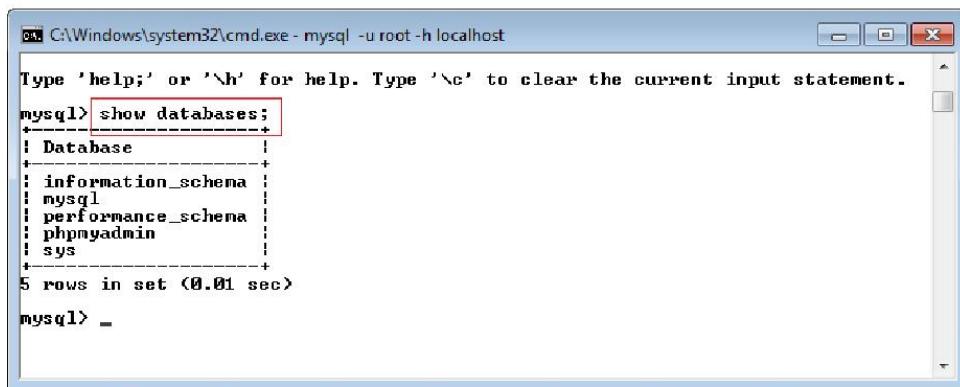
C:\Program Files (<x86>)\EasyPHP-Devserver-16.1\eds-binaries\dbserver\mysql15711x86
>180511135736\bin>
```

Si ocurre un error de conexión, puede ser por varios motivos. Uno de ellos es que el servidor MySQL no se está ejecutando en la dirección especificada. Por ello, debe verificarse si se está ejecutando el servidor, utilizando el administrador de tareas:



### 1.3.- Creación de Bases de Datos

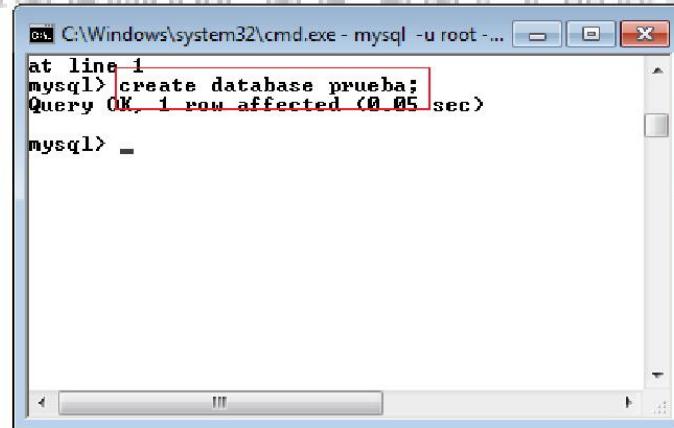
Se pueden visualizar las bases de datos a las que tiene acceso el usuario que está conectado usando el comando "show databases":



```
C:\Windows\system32\cmd.exe - mysql -u root -h localhost
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show databases;
+ Database
| information_schema |
| mysql |
| performance_schema |
| phpmyadmin |
| sys |
+-----+
5 rows in set (0.01 sec)

mysql> _
```

Para crear la base de datos se ejecuta el comando "create database ...", como se puede apreciar en el siguiente ejemplo, donde se crea una base de datos con el nombre "prueba":



```
at line 1
mysql> create database prueba;
Query OK, 1 row affected (0.05 sec)

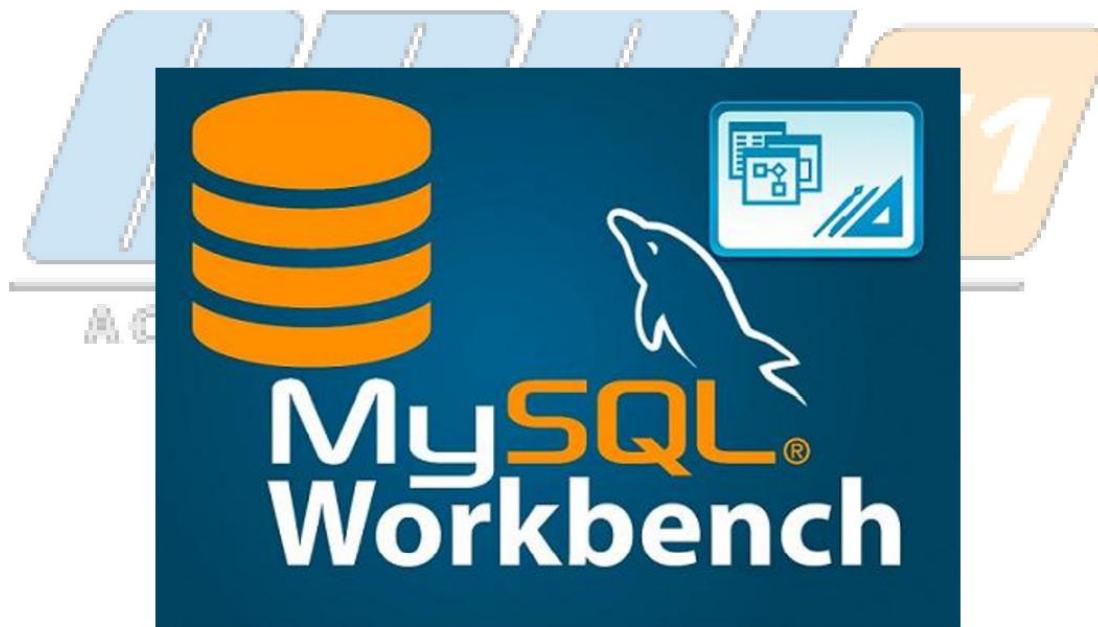
mysql> _
```

## Capítulo 2. MYSQL WORKBENCH

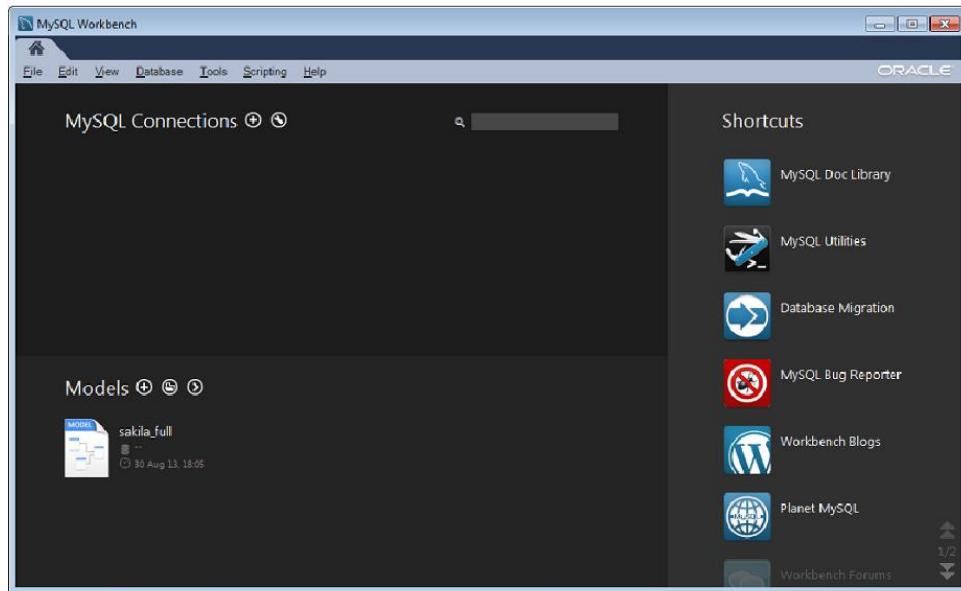
### 2.1.- Conociendo la Interfaz

MySQL Workbench es una herramienta visual de diseño de bases de datos que integra desarrollo de software, administración de bases de datos, diseño de bases de datos, gestión y mantenimiento para el sistema de base de datos MySQL. Las características destacadas de MySQL Workbench:

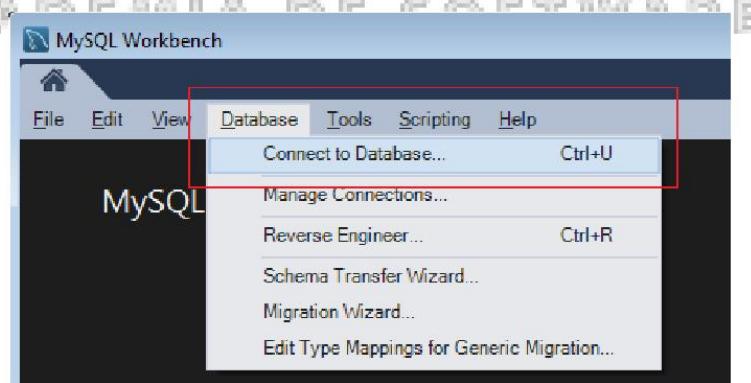
- Conexión y gestión de instancia de base de datos.
- Editor de SQL.
- Modelado de datos.
- Administración de base de datos.



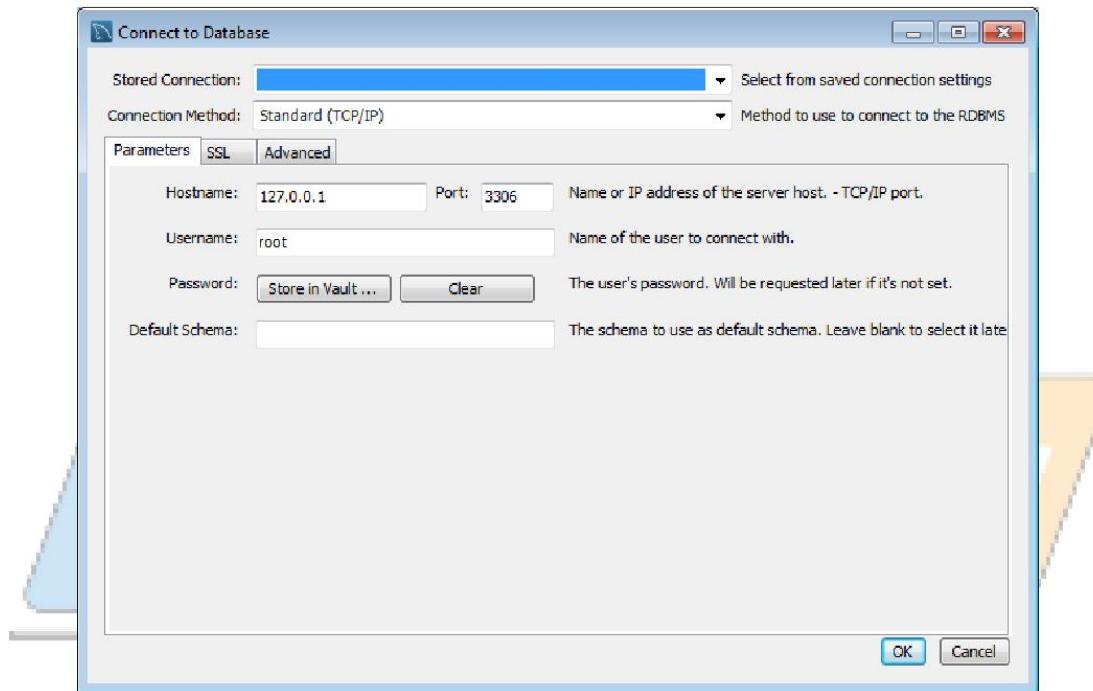
La interfaz de MySQL Workbench es simple e intuitiva:



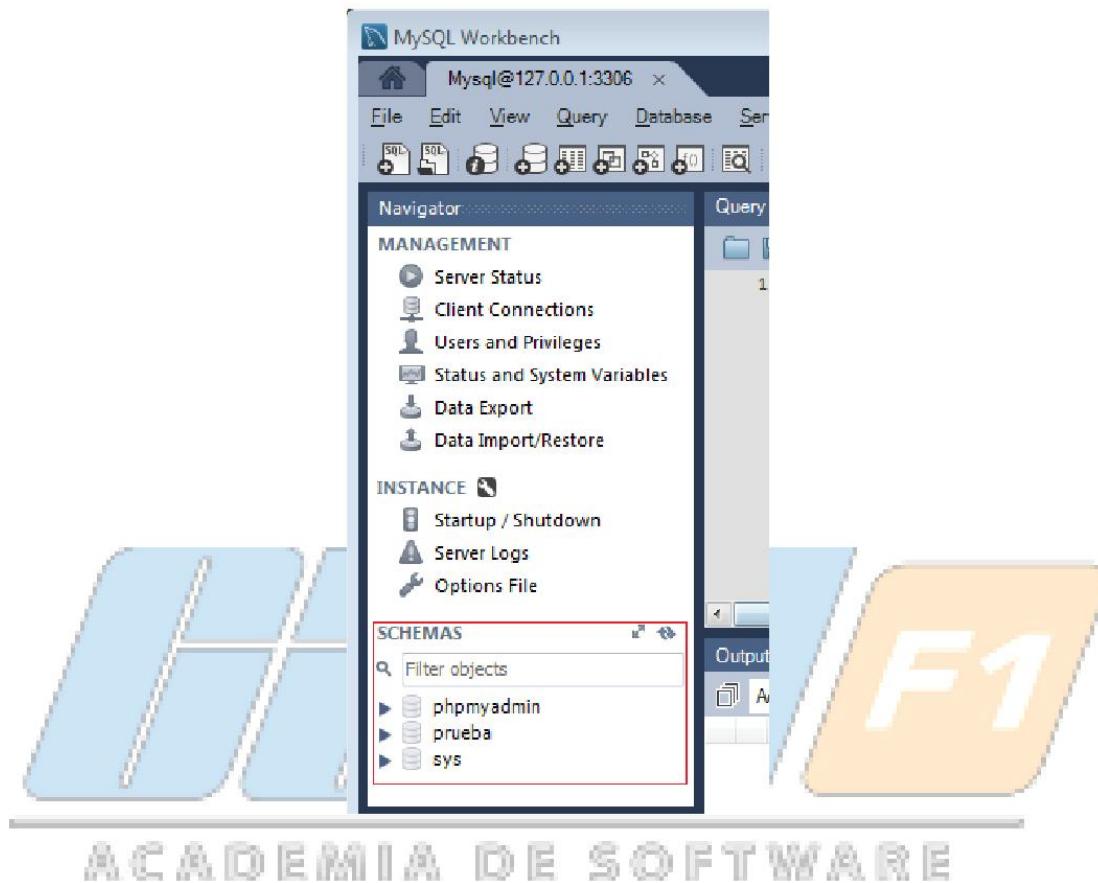
Una de las principales funciones de MySQL Workbench es permitir conectar a servidores MySQL. Este objetivo se puede lograr de varias formas. Una de ellas es haciendo clic en la opción "Database" y luego en "Connect to Database":



Se muestra una ventana donde se pueden establecer los parámetros de la conexión: dirección IP, puerto, usuario y la clave. La clave se solicita luego de hacer clic en el botón "Ok":



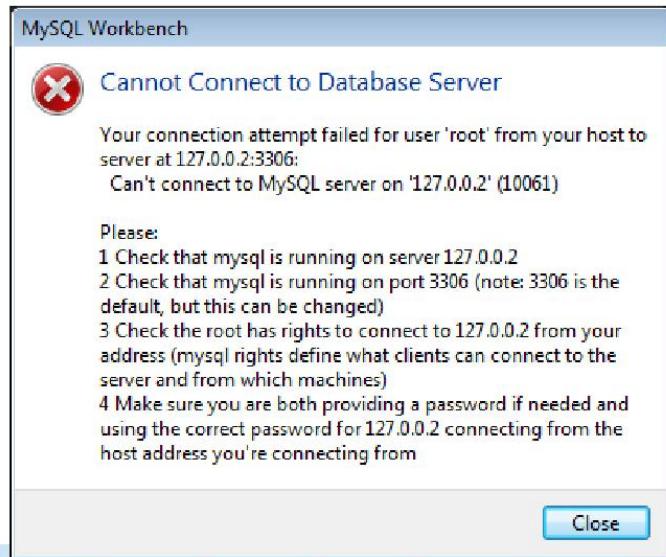
Si la conexión es exitosa, se abre una pestaña nueva y se muestra en el panel de la izquierda las bases de datos a las cuales puede acceder el usuario con el cual se ha conectado al servidor:



## ACADEMIA DE SOFTWARE

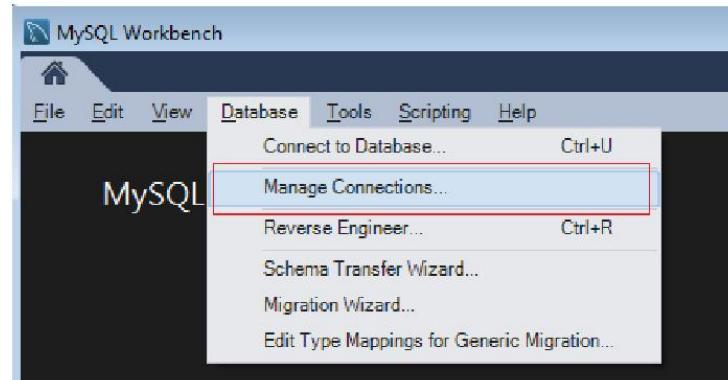
Si ocurriera algún error en la conexión, se muestra una ventana informando el error. Las causas por las cuales la conexión no se logra son muy variadas, entre ellas:

- dirección IP incorrecta.
- el servidor no se está ejecutando.
- puerto incorrecto.
- usuario incorrecto.
- clave incorrecta.

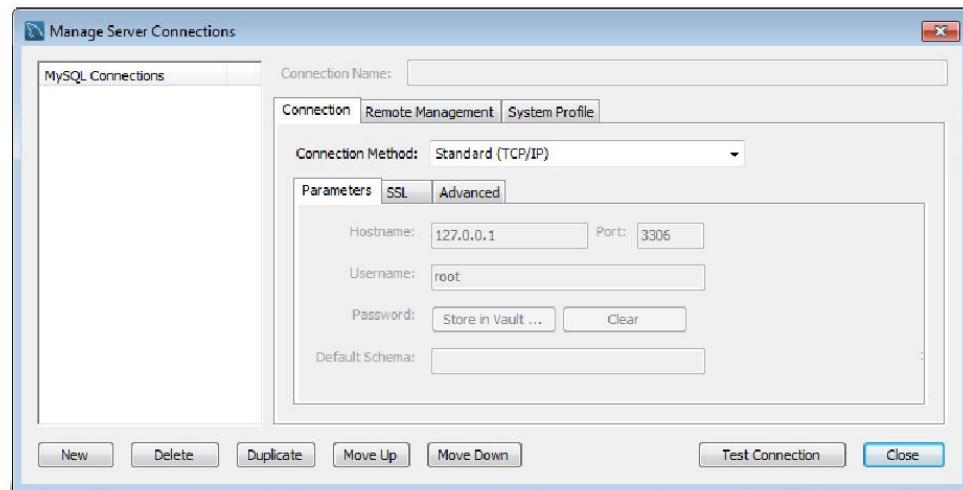


## 2.2.- Administrador de Conexiones

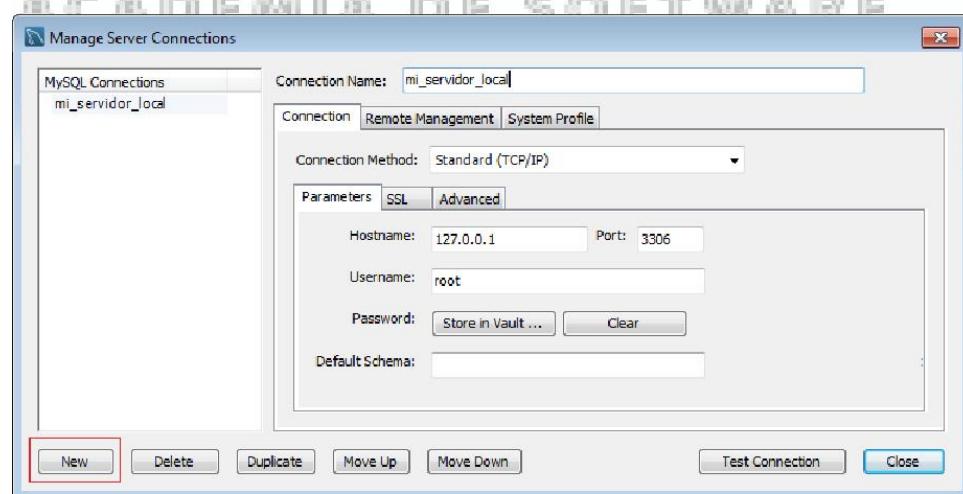
Si una conexión a un servidor se hace con mucha frecuencia, es conveniente guardar los datos de la conexión para usarlos cada vez que se necesite. Para esto, MySQL Workbench tiene una herramienta llamada "Administrador de Conexiones". A este se puede acceder con la opción de menú "Database" y luego la opción "Manage Connections.":



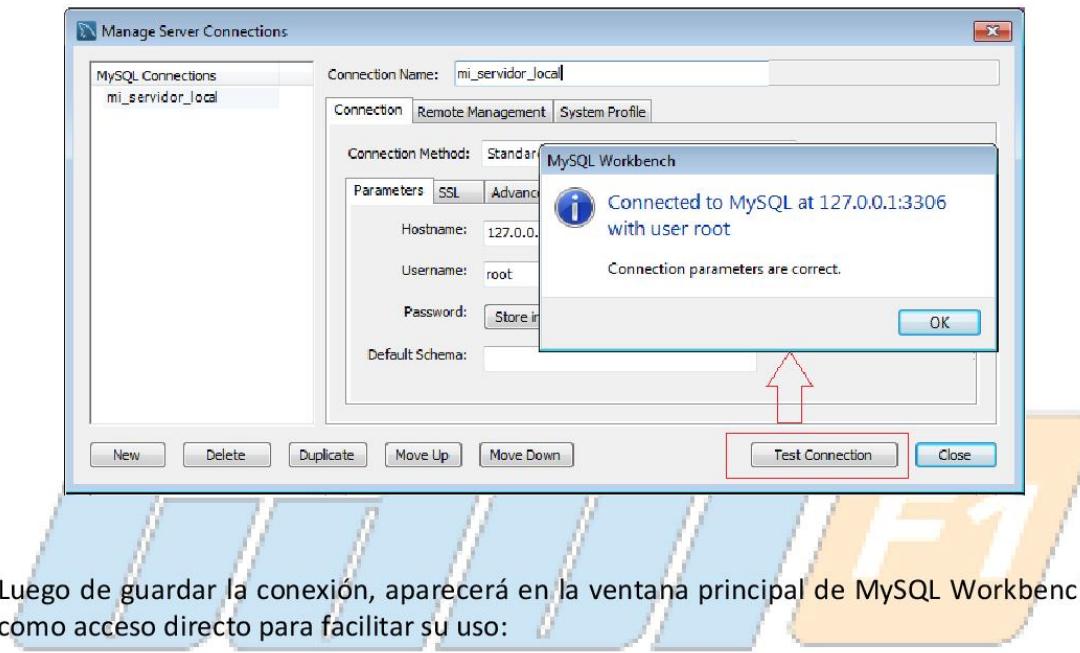
Se abre una ventana donde se muestran las conexiones guardadas. Inicialmente no hay conexiones.



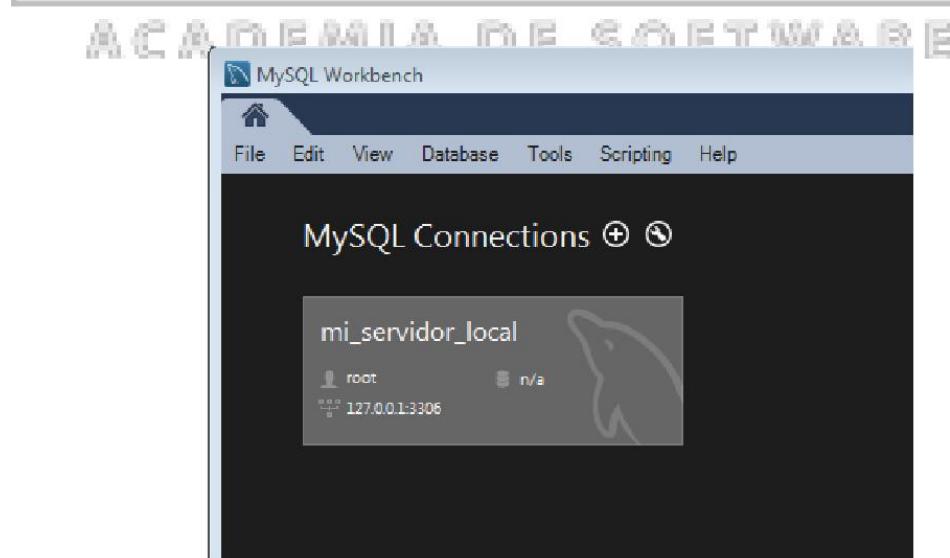
Al hacer clic en el botón "New" se habilitan los cuadros de texto para especificar los parámetros de conexión, bajo un nombre, como se ve en el ejemplo, "mi\_servidor\_local":



Se prueba la conexión antes de guardarla para asegurarse que todos los datos son correctos, en cuyo caso aparece el mensaje "Connection parameters are correct":

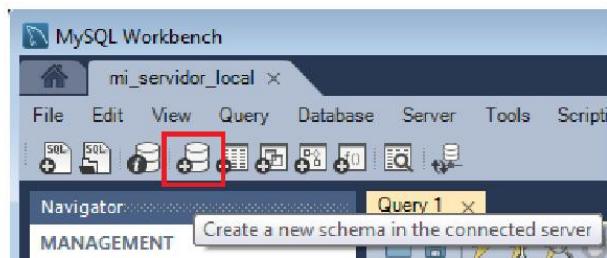


Luego de guardar la conexión, aparecerá en la ventana principal de MySQL Workbench como acceso directo para facilitar su uso:

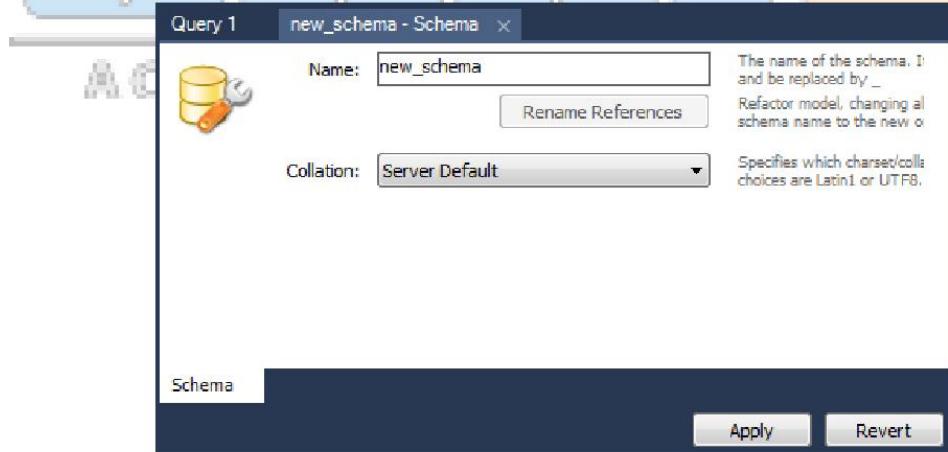


### 2.3.- Crear Bases de Datos

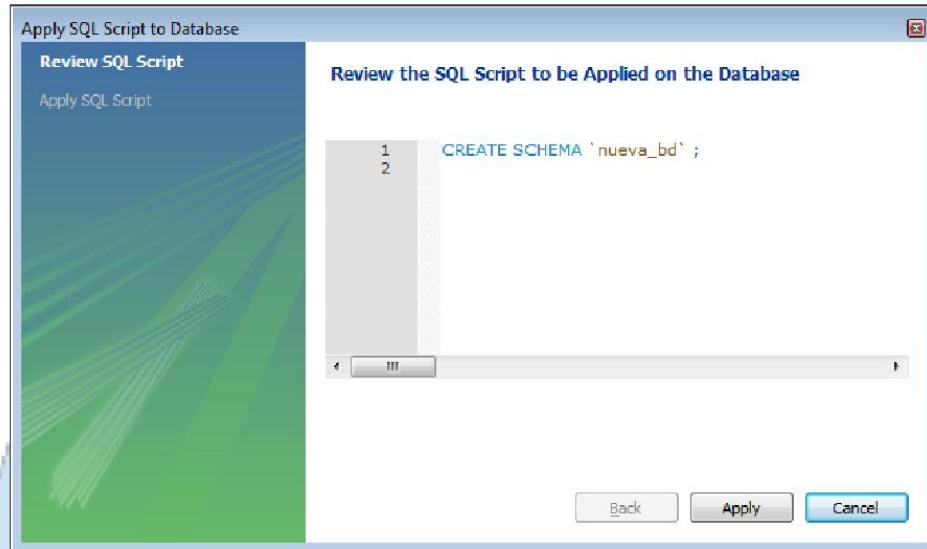
Luego de conectado al servidor, se pueden explorar las bases de datos existentes o crear una nueva. Para crear una base de datos se debe hacer clic en el botón "Create New schema", como se visualiza en la imagen:



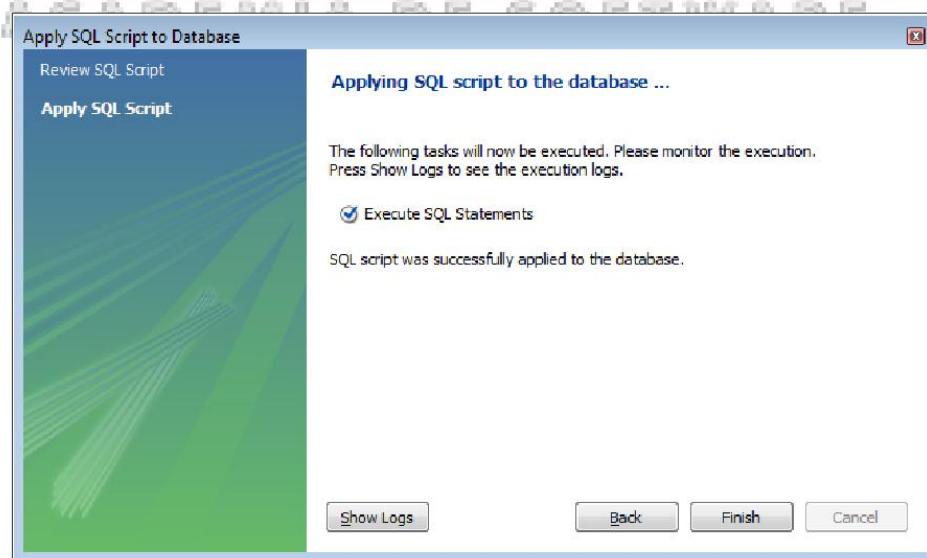
Aparecerá una pestaña para establecer el nombre de la base de datos que se va a crear, así como el conjunto de caracteres que se utilizaran en la base de datos (Collation), que al no ser especificado, se utiliza el que por defecto tiene establecido el servidor:



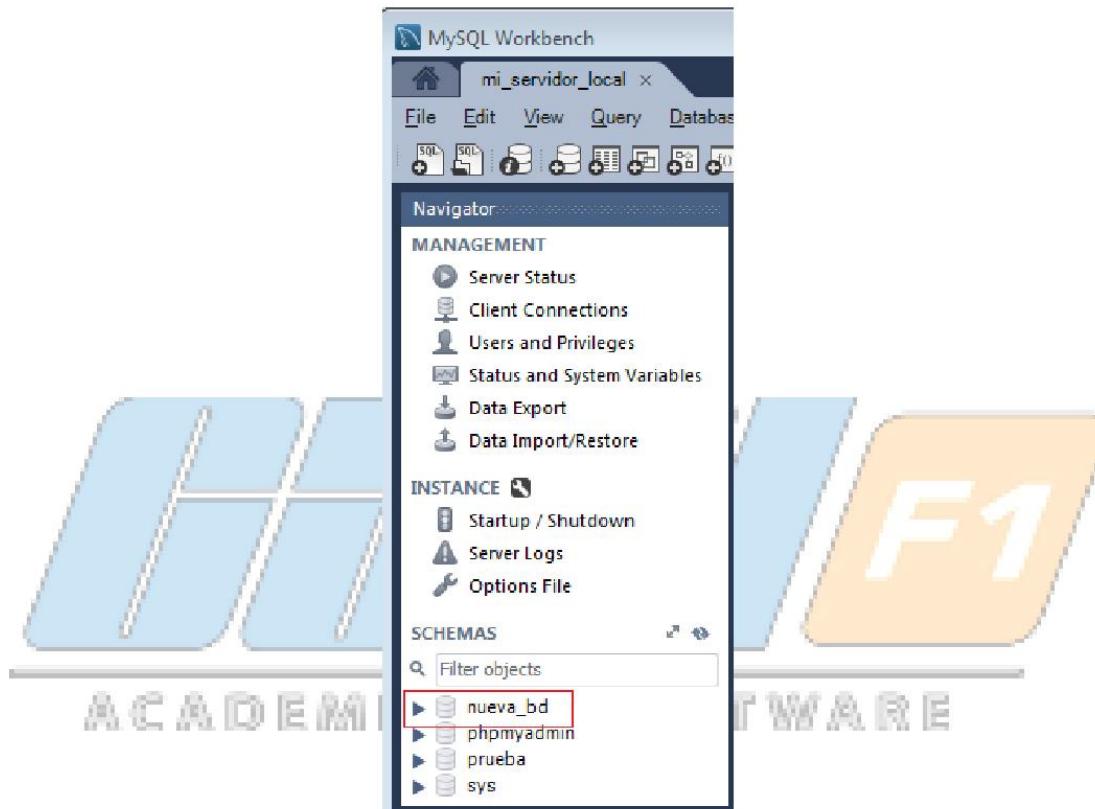
Al hacer click en el botón "Apply", aparecerá una ventana que muestra la instrucción SQL (CREATE SCHEMA, que es un sinónimo de CREATE DATABASE) que está a punto de ejecutarse, para hacer la confirmación:



Si todo va bien, aparecerá el mensaje "SQL script was successfully ...", como se muestra en la imagen:



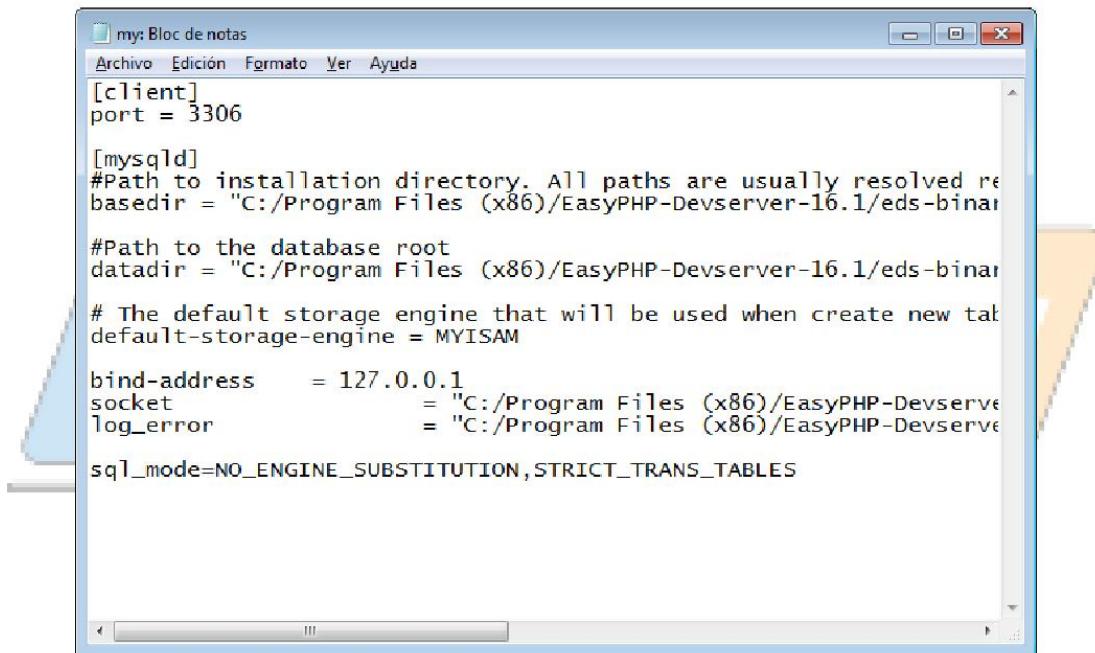
Y la base de datos podrá visualizarse en el panel lateral. Si la base de datos no aparece, se debe hacer clic en el botón para refrescar la lista:



## Capítulo 3. CONFIGURACIÓN DE MYSQL

### 3.1.- Archivo de Configuración

MySQL tiene un archivo de configuración que controla su comportamiento. Este archivo se encuentra en la carpeta de instalación del servidor y luce como la siguiente imagen:



```
[client]
port = 3306

[mysqld]
#Path to installation directory. All paths are usually resolved relative to this.
basedir = "C:/Program Files (x86)/EasyPHP-Devserver-16.1/eds-binaries"

#Path to the database root
datadir = "C:/Program Files (x86)/EasyPHP-Devserver-16.1/eds-binaries/mysql"

# The default storage engine that will be used when create new tables
default-storage-engine = MYISAM

bind-address      = 127.0.0.1
socket            = "C:/Program Files (x86)/EasyPHP-Devserver-16.1/eds-binaries/mysql"
log_error         = "C:/Program Files (x86)/EasyPHP-Devserver-16.1/eds-binaries/mysql/error.log"

sql_mode=NO_ENGINESUBSTITUTION,STRICT_TRANS_TABLES
```

Las líneas que comienzan con # son comentarios y son ignorados. Entre las opciones que más frecuentemente se modifican están:

- datadir: la ruta donde se almacenan las bases de datos.
- port: el puerto que utiliza para escuchar las peticiones.
- bind-address: la dirección desde donde aceptara conexiones.

### 3.2.- Contraseña de Root

En MySQL el usuario "root" (el administrador) tiene una clave por defecto, o es establecida durante el proceso de instalación. Esta clave se puede modificar en MySQL Workbench a través de la opción de administración "Users and Privileges". Al marcar esta opción, se muestran todos los usuarios registrados en el servidor:

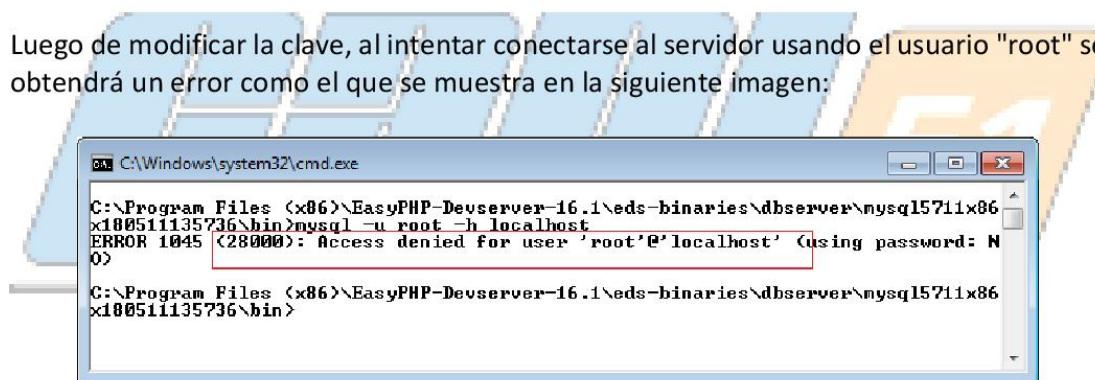
The screenshot shows the MySQL Workbench interface. On the left, the Navigator panel has a 'MANAGEMENT' section with several options: Server Status, Client Connections, Users and Privileges (which is highlighted with a red box), Status and System Variables, Data Export, and Data Import/Restore. Below that is an 'INSTANCE' section with Startup / Shutdown, Server Logs, and Options File. At the bottom of the Navigator are tabs for Management and Schemas. The main window title is 'Administration - Users and Privileges'. It displays a list of 'User Accounts' with the following data:

User	From Host
<anonymous>	localhost
mysql.sys	localhost
root	localhost
root	127.0.0.1
root	::1

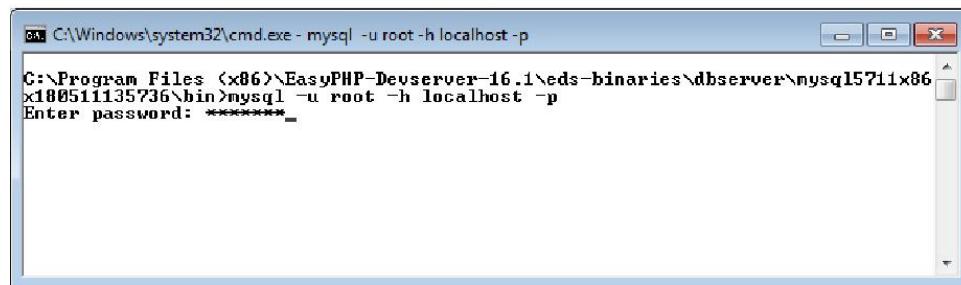
Al seleccionar el usuario "root" (cualquiera de ellos), aparece en el panel de la derecha la opción para modificar la clave:

The screenshot shows the MySQL Workbench interface. On the left, there's a sidebar with a user icon and the text 'mi\_servidor\_local'. Below it is the 'Users and Privileges' section. On the right, a detailed configuration dialog is open for the 'root' user at 'localhost'. The 'User Accounts' table lists several users with their host information. The 'root' row is selected and highlighted with a red border. The configuration dialog has tabs for 'Login', 'Account Limits', 'Administrative Roles', and 'Schema Privileges'. The 'Login' tab is active, showing fields for 'Login Name' (set to 'root'), 'Authentication Type' (set to 'Standard'), 'Limit Connectivity to Hosts Matching' (set to 'localhost'), and password fields for 'Password' and 'Confirm Password'.

Luego de modificar la clave, al intentar conectarse al servidor usando el usuario "root" se obtendrá un error como el que se muestra en la siguiente imagen:



En adelante, para hacer la conexión se debe agregar el parámetro "-p", que solicita la clave del usuario para intentar conectarse:



En MySQL Workbench, al intentar conectarse utilizando el usuario "root", aparecerá un dialogo que solicita la contraseña:



### 3.3.- Crear Usuarios

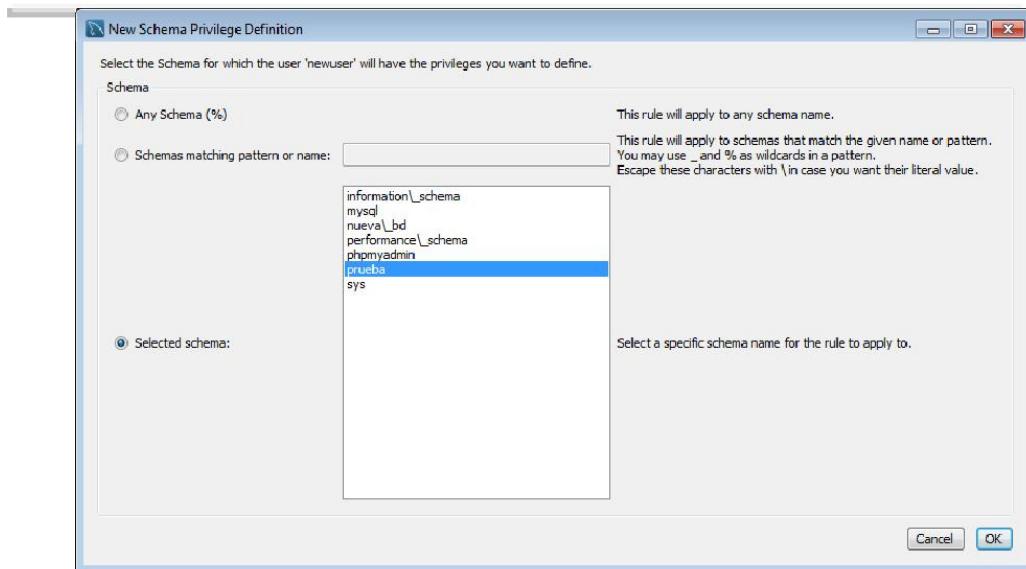
Se pueden crear otras cuentas de usuario además de root, que se debe dejar solo para conexiones locales del DBA. La conexión remota debe hacerse con otros usuarios:

User	From Host
<anonymous>	localhost
mysql.sys	localhost
root	localhost
root	127.0.0.1
root	::1
newuser	%

Al crear el usuario, se puede especificar a cuáles bases de datos tendrá acceso:

The screenshot shows the 'Schema Privileges' tab of the 'Details for account newuser@%' dialog. At the top, there are tabs for Login, Account Limits, Administrative Roles, and Schema Privileges. The Schema Privileges tab is selected. Below the tabs is a table with two columns: 'Schema' and 'Privileges'. A note below the table states: 'Schema and Host fields may use % and \_ wildcards. The server will match specific entries before wildcarded ones.' To the right of the table are 'Delete Entry' and 'Add Entry...' buttons. Below the table are three groups of checkboxes: 'Object Rights' (SELECT, INSERT, UPDATE, DELETE, EXECUTE, SHOW VIEW), 'DDL Rights' (CREATE, ALTER, REFERENCES, INDEX, CREATE VIEW, CREATE ROUTINE, ALTER ROUTINE, DROP, TRIGGER), and 'Other Rights' (GRANT OPTION, CREATE TEMPORARY TABLES, LOCK TABLES). At the bottom of the group are 'Unselect All' and 'Select "All"' buttons.

Seleccionándolas todas, algunas o una en particular:



Inclusive, indicar cuales acciones puede realizar el usuario sobre esa base de datos:

**Details for account newuser@%**

Schema Privileges

Schema	Privileges
prueba	ALTER, CREATE, CREATE VIEW, INDEX, SELECT

Schema and Host fields may use % and \_ wildcards.  
The server will match specific entries before wildcarded ones.

The user 'newuser'@'%' will have the following access rights to the schema 'prueba':

Object Rights      DDL Rights      Other Rights

<input checked="" type="checkbox"/> SELECT	<input checked="" type="checkbox"/> CREATE	<input type="checkbox"/> GRANT OPTION
<input type="checkbox"/> INSERT	<input checked="" type="checkbox"/> ALTER	<input type="checkbox"/> CREATE TEMPORARY TABLES
<input type="checkbox"/> UPDATE	<input type="checkbox"/> REFERENCES	<input type="checkbox"/> LOCK TABLES
<input type="checkbox"/> DELETE	<input checked="" type="checkbox"/> INDEX	
<input type="checkbox"/> EXECUTE	<input checked="" type="checkbox"/> CREATE VIEW	
<input type="checkbox"/> SHOW VIEW	<input type="checkbox"/> CREATE ROUTINE	
	<input type="checkbox"/> ALTER ROUTINE	
	<input type="checkbox"/> DROP	
	<input type="checkbox"/> TRIGGER	

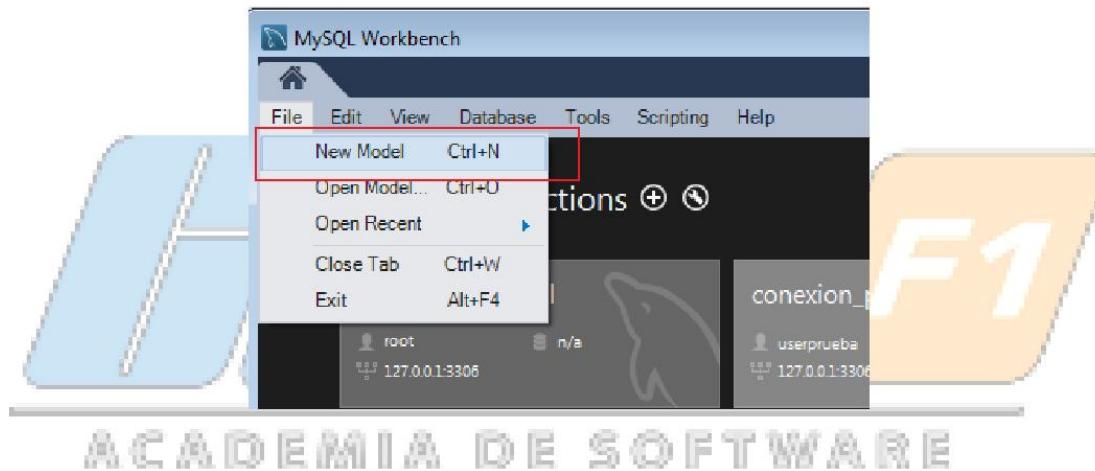
**ACADEMIA DE SOFTWARE**

Buttons: Delete Entry, Add Entry..., Unselect All, Select "All", Revoke All Privileges, Expire Password, Revert, Apply, Refresh

## Capítulo 4. TRABAJANDO CON MODELOS

### 4.1.- Crear un Modelo

MySQL Workbench incorpora una herramienta que permite crear el modelo de una base de datos sin aún crearla físicamente. Es una herramienta muy usada por diseñadores. Para crear un nuevo modelo debe hacer clic en la opción "File" y luego en la opción "New Model"



Se abre un panel donde se pueden agregar todos los elementos del modelo: bases de datos, tablas, vistas, procedimientos almacenados, entre otros.

The screenshot shows the MySQL Model Overview interface. At the top, there is a toolbar with various icons. Below it, a section titled "Physical Schemas" is expanded, showing a single entry: "mydb MySQL Schema". This entry has three sub-options: "Tables (0 items)", "Add Table", "Views (0 items)", and "Add View".

Below this, there is a large blue rectangular area labeled "Physical Schemas" containing several small, overlapping windows. One window is highlighted in orange and titled "nueva\_bd - Schema". It contains the following fields:

- Name:  (with a wrench icon next to it)
- Collation:

To the right of these fields, there is explanatory text:

Name: The name of the schema. It is recommended to use only underscores and digits. Refactor model, changing all references found in view, triggers and functions from the old schema name to the new one.

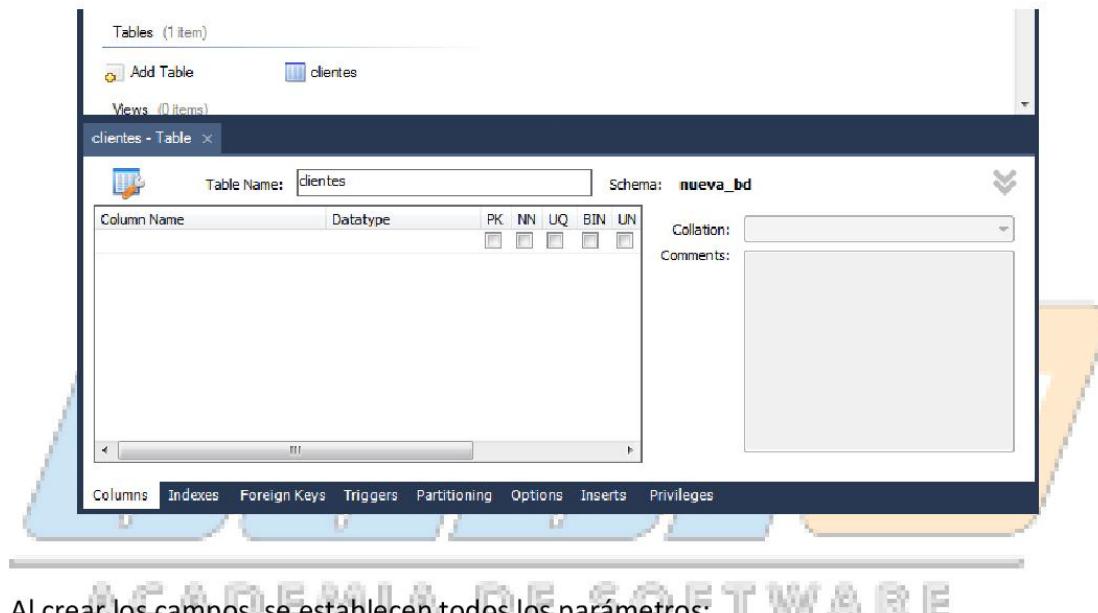
Collation: Specifies which charset/collations the schema's tables will use. Common choices are Latin1 or UTF8.

A tooltip for the "Rename References" button states: "Refactor model, changing all references found in view, triggers and functions from the old schema name to the new one."

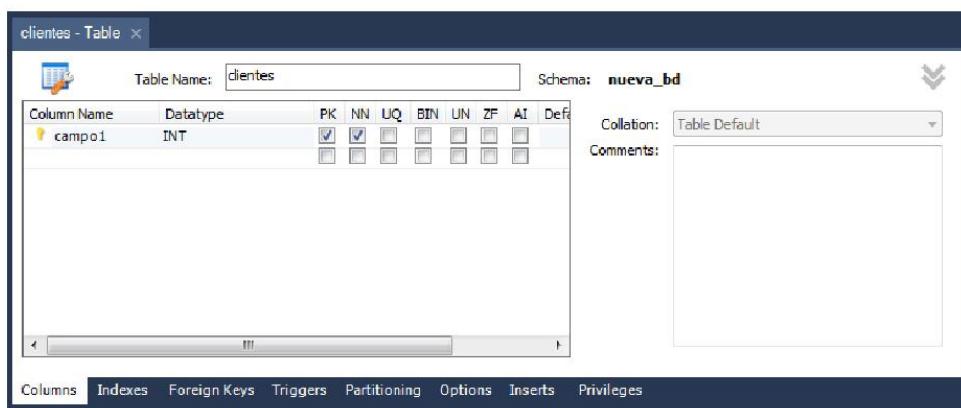
Below the schema creation window, there is a message: "Un modelo incorpora al menos una base de datos. La base de datos tendrá un nombre por defecto (Mydb), que puede ser modificado al hacer doble click sobre ésta:"

## 4.2.- Crear Tablas

Se pueden crear las tablas de la base de datos al hacer clic en la opción "Add Table", que abre un panel donde se establecen todos los parámetros de la tabla: nombre, comentario, campos, claves, registros, entre otros.

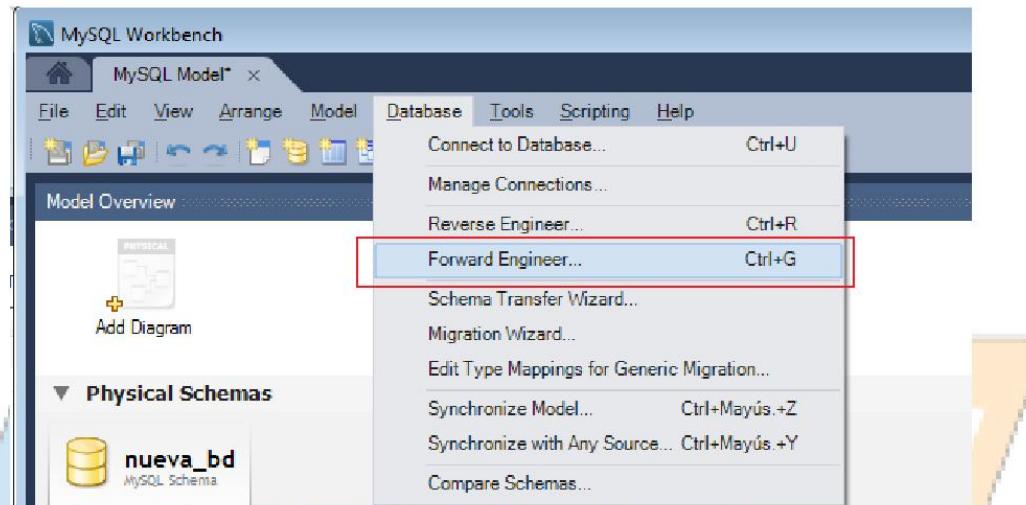


Al crear los campos, se establecen todos los parámetros:

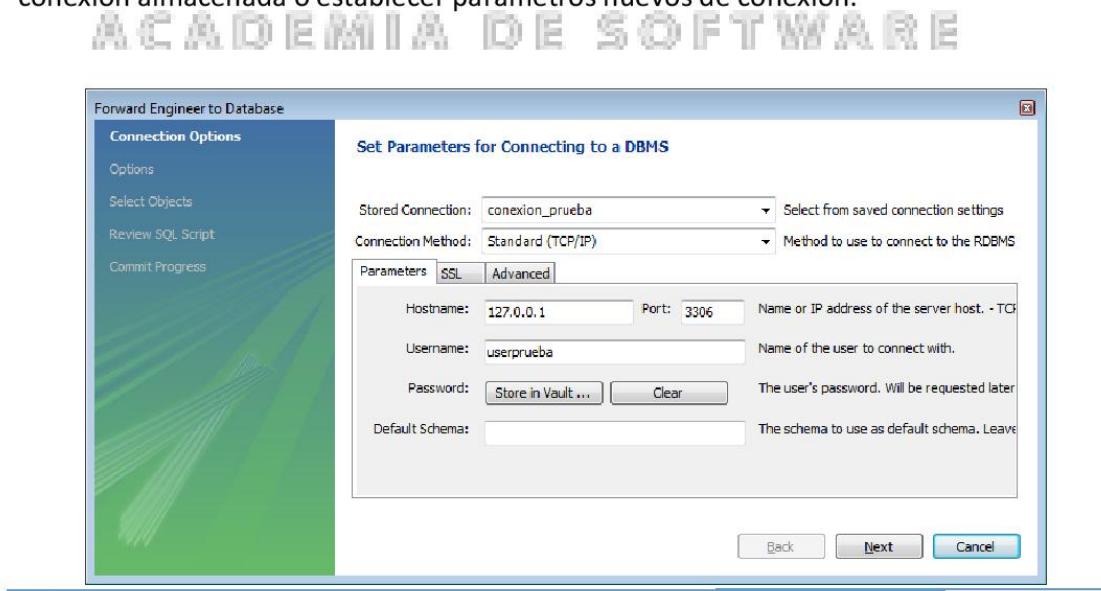


### 4.3.- Forward Engineer

Un modelo se puede convertir en una base de datos física utilizando la herramienta "Forward Engineer":



El primer paso del asistente es establecer la conexión con el servidor. Se puede usar una conexión almacenada o establecer parámetros nuevos de conexión:

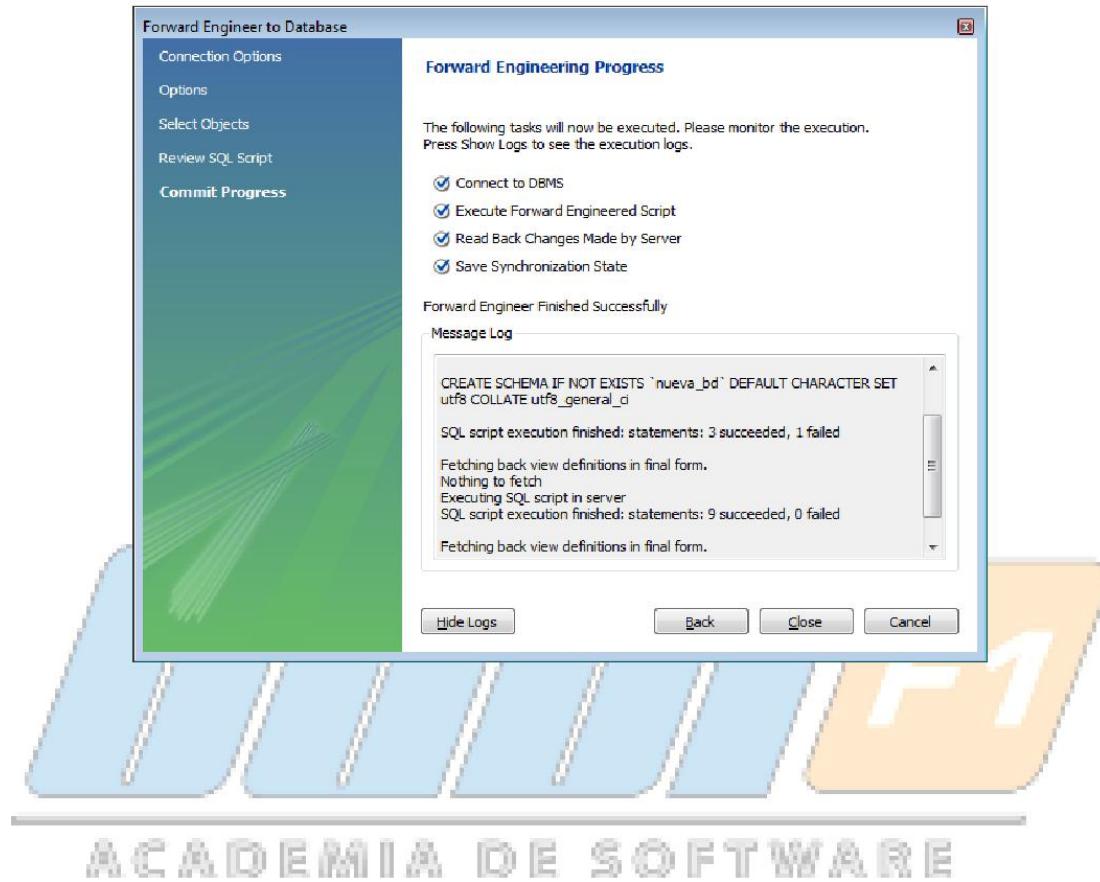


Luego de varios pasos, aparece las instrucciones SQL que serán ejecutadas en el servidor:

```
1 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
2 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
3 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
4
5 CREATE SCHEMA IF NOT EXISTS `nueva_bd` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
6 USE `nueva_bd`;
7
8 -- Table: `nueva_bd`.`clientes`
9
10 CREATE TABLE IF NOT EXISTS `nueva_bd`.`clientes` (
11   `campo1` INT NOT NULL,
12   PRIMARY KEY (`campo1`)
13 ) ENGINE = InnoDB;
14
15
16
17 SET SQL_MODE=@OLD_SQL_MODE;
18 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
19 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Si todo se ejecuta correctamente, aparecerá el mensaje "Forward Engineer Successfully":

ACADEMIA DE SOFTWARE

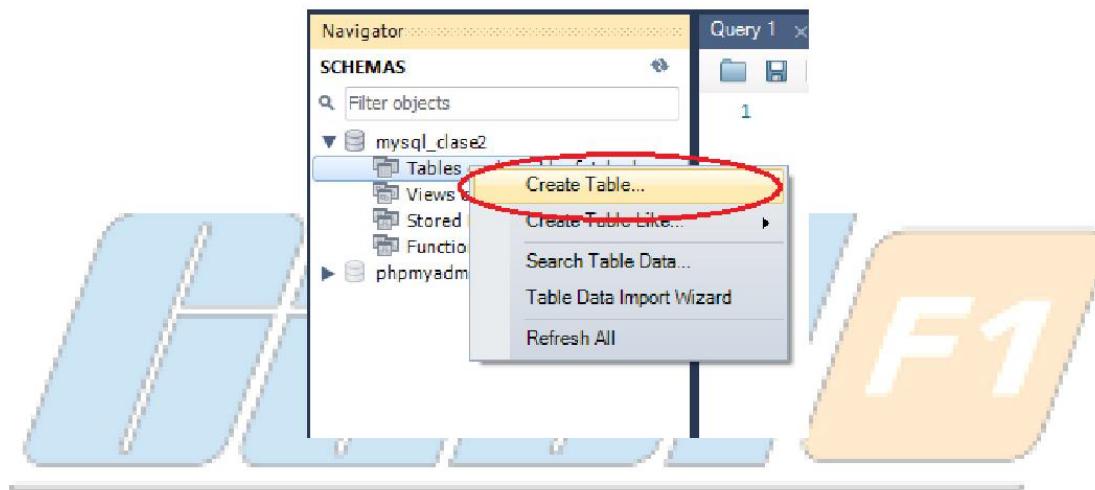


ACADEMIA DE SOFTWARE

## Capítulo 5. CREACIÓN DE TABLAS

### 5.1.- Parámetros Básicos

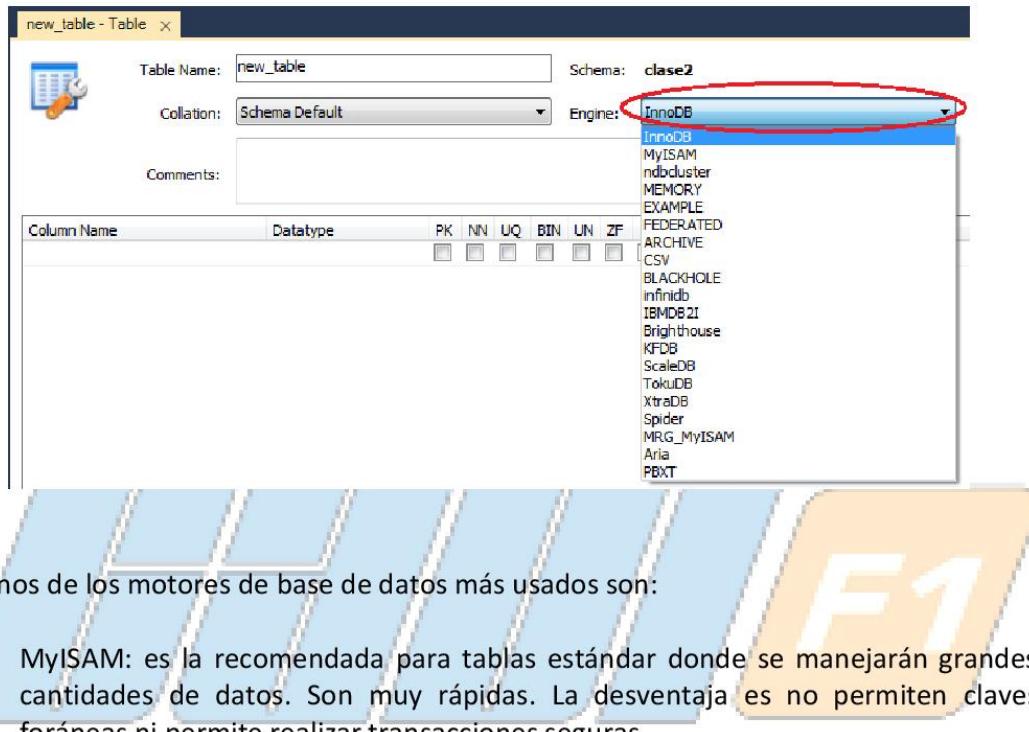
Luego de crear una base de datos, se deben crear las tablas. Para crear una tabla, se hace click en el botón derecho sobre la opción "Create Table":



Se abre una pestaña donde se establece el nombre de la tabla, el conjunto de caracteres y algún comentario:

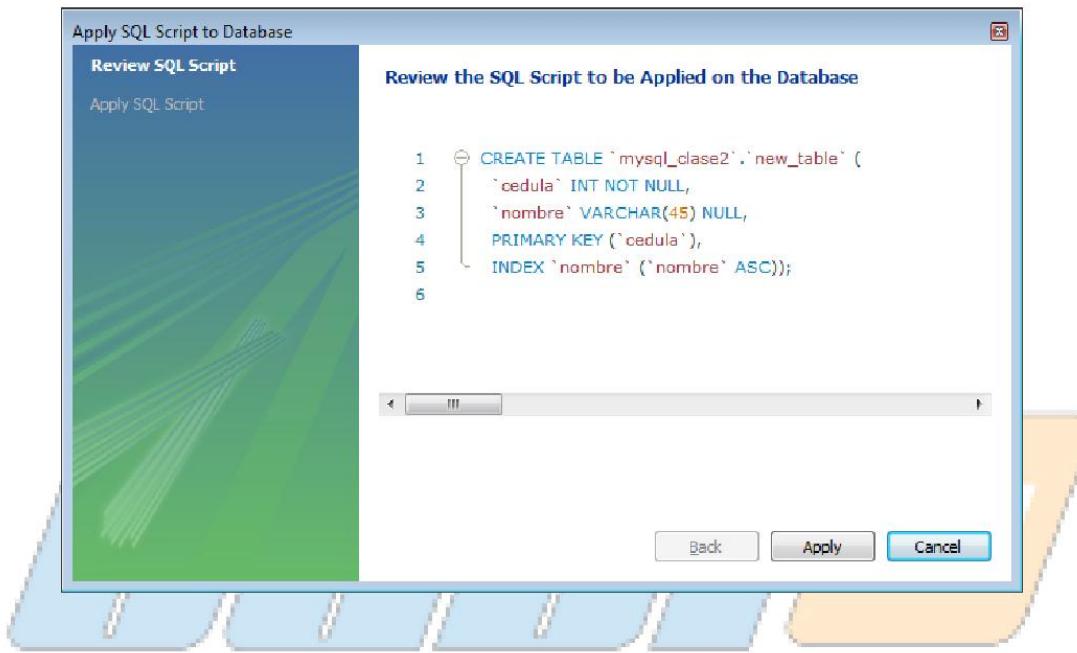
A screenshot of the 'Query 1' tab in phpMyAdmin, titled 'new\_table - Table'. It shows the configuration for creating a new table. The 'Table Name' is set to 'new\_table', 'Schema' is 'mysql\_clase2', 'Charset/Collation' is 'Default Charset', 'Engine' is 'InnoDB', and the 'Comments:' field is empty.

Otro de los parámetros que se establecen al crear una tabla es el tipo de motor (Engine). El motor es clave a la hora de evaluar la rapidez y las funcionalidades que se pueden lograr con la tabla. En la siguiente imagen se muestran los posibles motores que se pueden seleccionar al crear una tabla:



Para más información consultar <https://dev.mysql.com/doc/refman/8.0/en/storage-engines.html>

Luego de establecer todos los parámetros, se presiona el botón "Apply", que muestra una ventana con la instrucción SQL "create table" que se va a ejecutar:



## 5.2.- Definición de Campos

### ACADEMIA DE SOFTWARE

Luego de que la tabla esta creada, se puede modificar en cualquier momento, seleccionando la tabla, haciendo click con el botón derecho sobre el nombre de la tabla y luego seleccionando la opción "Alter table":

Al definir los campos de una tabla, se definen varios parámetros: el nombre, el tipo de dato, el valor por defecto y algún comentario.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idnew_table	INT	<input type="checkbox"/>								

Column Name: idnew\_table Data Type: INT  
 Charset/Collation: Default Charset: Default Collation: Default:  
 Comments:  
 Storage:  Virtual  Stored  
 Primary Key  Not Null  Unique  
 Binary  Unsigned  Zero Fill  
 Auto Increment  Generated

Los tipos de datos más usados en MySQL son:

- int: para valores numéricos enteros.
- float: para tipos de datos numéricos reales.
- varchar: para tipos de datos alfanuméricos.
- date: para tipos de datos fecha.

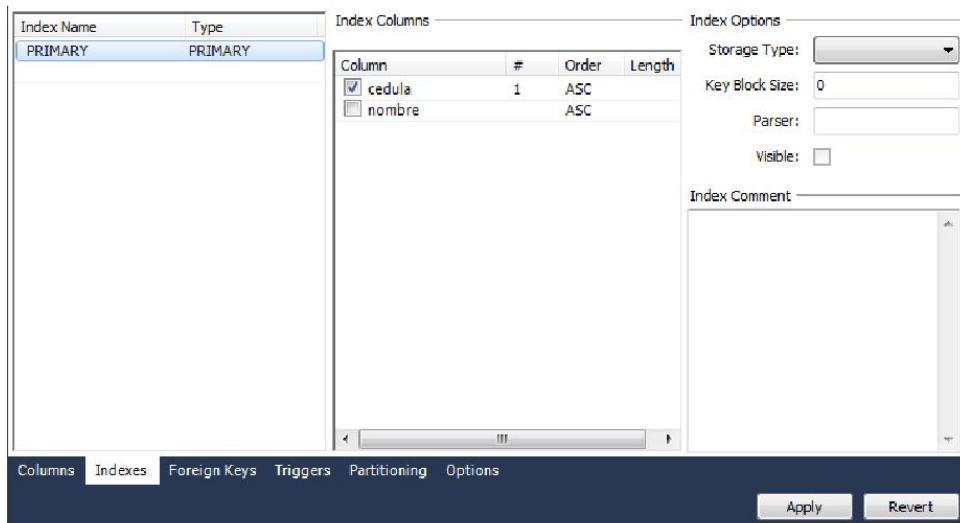
En un capítulo posterior se explicarán con detalles todos los tipos de datos disponibles y cuales rangos de valores maneja cada uno.

Entre los atributos que puede tener un campo están:

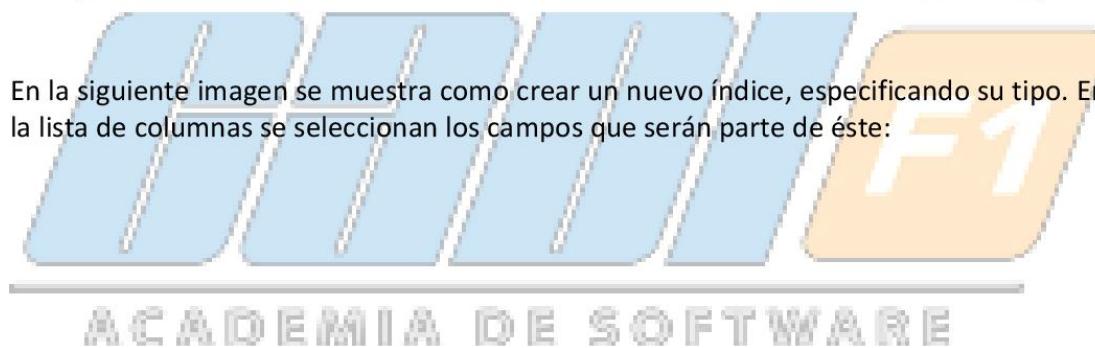
- PK (Primary Key): indica que el campo es la clave primaria.
- NN (not null): indica si el campo permite valores null.
- UQ (Unique): indica que el campo es clave única.
- B (Binary): indica que el campo se establece a binaria el conjunto de caracteres. No aplica a tipos de datos numéricos.
- UN (unsigned): indica que el campo no tendrá signo, es decir, que sólo almacenará valores positivos.
- ZF (zero fill): llena de ceros a la izquierda el valor. Solo aplica a tipos de datos numéricos.
- AI (auto incremental): indica que el campo será auto incremental. Obliga que el campo sea la clave primaria.
- G (generated): indica que la columna es generada. Puede ser virtual o stored. Virtual indica que se calcula pero no se almacena. Stored indica que se calcula y se almacena.

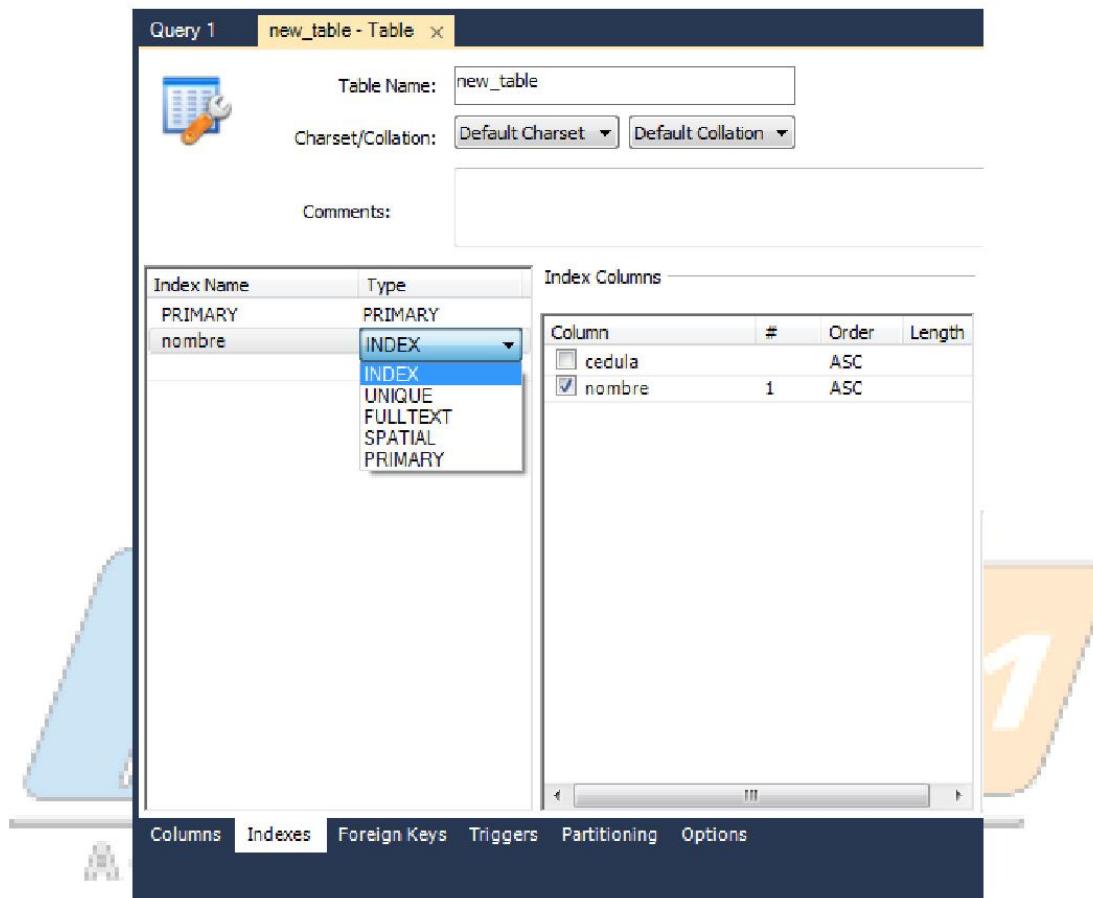
### 5.3.- Creación de índices

Los índices se crean automáticamente al establecer campos como clave primaria o clave única, pero también se pueden crear índices manualmente.



En la siguiente imagen se muestra como crear un nuevo índice, especificando su tipo. En la lista de columnas se seleccionan los campos que serán parte de éste:





Los tipos de índices que se pueden usar son:

- INDEX : se refiere a un índice normal, no único. Esto implica que admite valores duplicados para la columna (o columnas) que componen el índice.
- UNIQUE: no admite valores duplicados para la columna (o columnas) que componen el índice. Puede haber varios índices UNIQUE.
- PRIMARY: todas las columnas deben tener un valor único (al igual que en el caso del índice UNIQUE) pero con la limitación de que sólo puede existir un índice PRIMARY en cada una de las tablas.
- FULLTEXT: estos índices se emplean para realizar búsquedas sobre texto (CHAR, VARCHAR y TEXT). Estos índices se componen por todas las palabras que están

contenidas en la columna (o columnas) que contienen el índice. Este tipo de índices sólo están soportados por InnoDB y MyISAM.

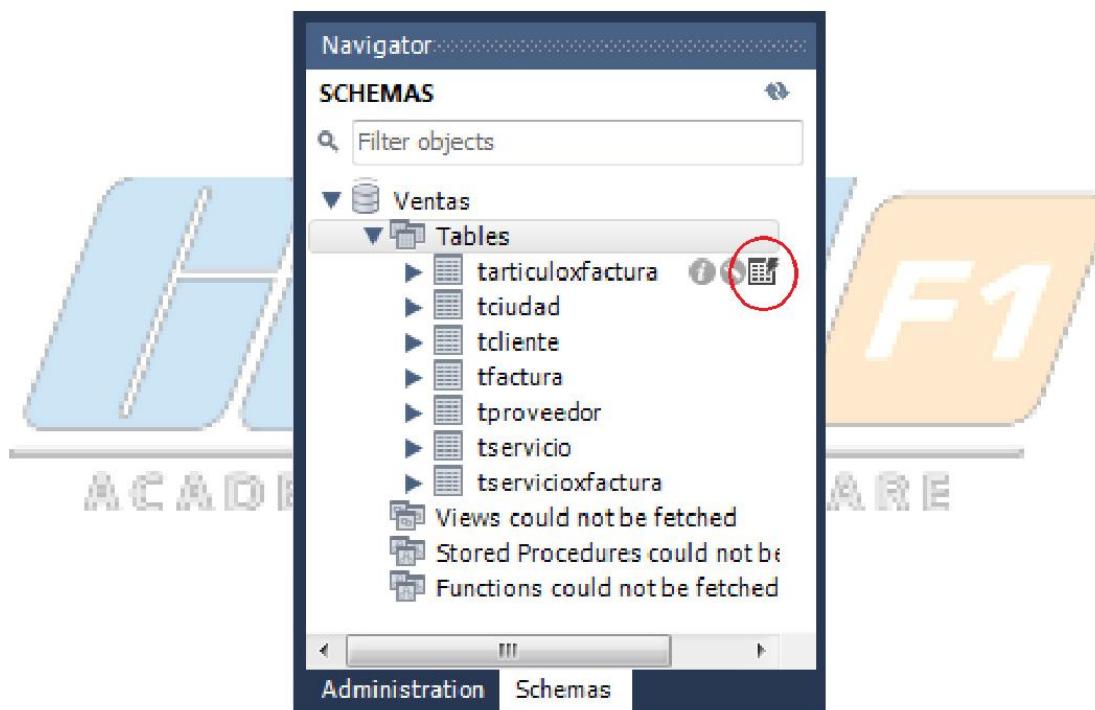
- SPATIAL: se emplean para realizar búsquedas sobre datos que componen formas geométricas representadas en el espacio.



## Capítulo 6. MANIPULANDO REGISTROS

### 6.1.- Listado de Registros

Para listar los registros almacenados en una tabla, se debe ubicar la base de datos en el panel "Schemas", desplegar la opción "Tables", ubicar el nombre de la tabla de la cual se necesitan visualizar los registros y hacer click en el botón señalado en la siguiente imagen:



Se abre una pestaña con una instrucción SQL "select", que muestra todos los campos y todos los registros de la tabla, como se muestra en la siguiente imagen:

A screenshot of the MySQL Workbench interface. The title bar says 'tcliente'. The main area shows a 'Result Grid' with the following data:

codcliente	cedula	nombre	comollego	telefono	email
43	16090221	Jose Miguel Rojas	1	4824744	paramocende@hotmail.com
46	14483352	Adelis Mendez	1	2551663	adelis18@hotmail.com
60	16138319	Alejandra Yépez	1	2559341	lmcst1@hotmail.com
561	9903190	Luisa Colón	1	4451362-445...	migolara@cantv.net
575	J-309737...	MIGO LARA, C.A.	0	74700461	lmcst1@hotmail.com
627	150037529	Irma Elenis Tchirivana	1	74700461	lmcst1@hotmail.com

The bottom right corner of the results grid has a context menu with options like 'Result Grid', 'Table', and 'Text'.

El listado de registros se puede limitar para que se muestre solo una cantidad máxima:

A screenshot of the MySQL Workbench interface showing the 'Don't Limit' dropdown menu. The menu items are:

- Don't Limit
- Limit to 10 rows
- Limit to 50 rows
- Limit to 100 rows
- Limit to 200 rows
- Limit to 300 rows
- Limit to 400 rows
- Limit to 500 rows
- Limit to 1000 rows
- Limit to 2000 rows
- Limit to 3000 rows
- Limit to 4000 rows
- Limit to 5000 rows
- Limit to 10000 rows
- Limit to 50000 rows

A tooltip for the 'Limit' button states: 'Number of rows returned by query. Click to automatically add the LIMIT clause'.

## 6.2.- Agregar, Modificar y Eliminar Registros

Visualizando los registros de la tabla, se pueden realizar las operaciones sobre ésta: agregar registros, modificarlos o eliminarlos. Para agregar registros, sólo se debe escribir en cualquier columna de la última fila, para que automáticamente MySQL WorkBench agregue una fila nueva:

cliente >

1 • `SELECT * FROM clase2.cliente;`

Cuando se está editando una fila se muestra un lápiz en la columna de indicador

id	rif	nombre	tipo	codigo	fecha	estado	fecha_registro
1	J310934401	Cadi F1 CA	1	12	2019-03-15	1	2019-06-18

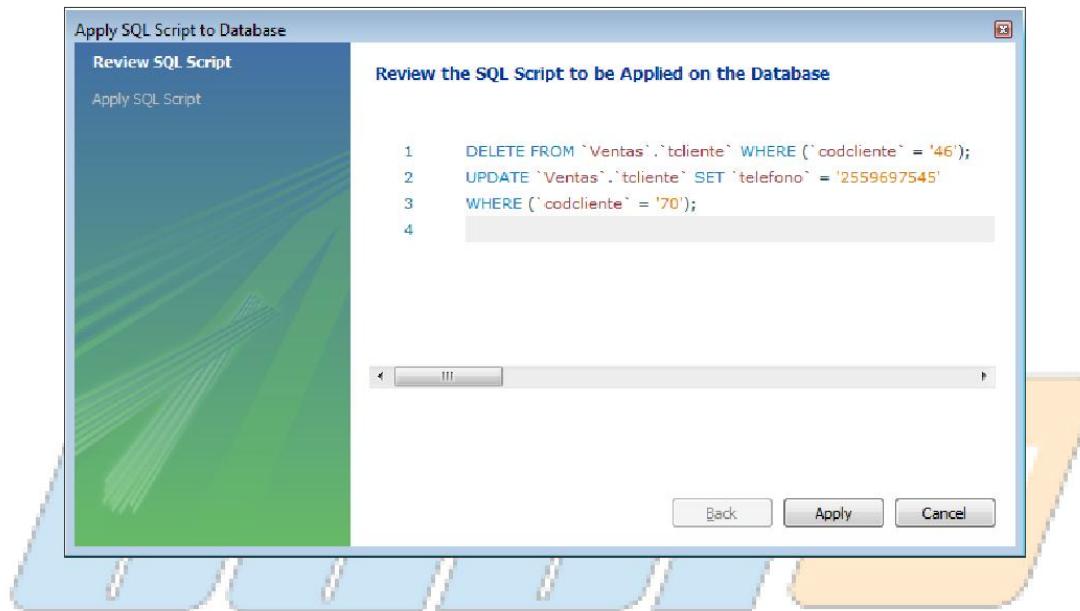
Para eliminar un registro (o varios), se marca el registro haciendo click en la primera columna (donde aparece el indicador del registro actual) y luego se hace click en el botón derecho del ratón. En el menú contextual que aparece se selecciona la opción "Delete Row(s)". La fila desaparece del listado, aunque aún no ha sido eliminada efectivamente.

Result Grid | Filter Rows: Edit: Wrap Cell Content:

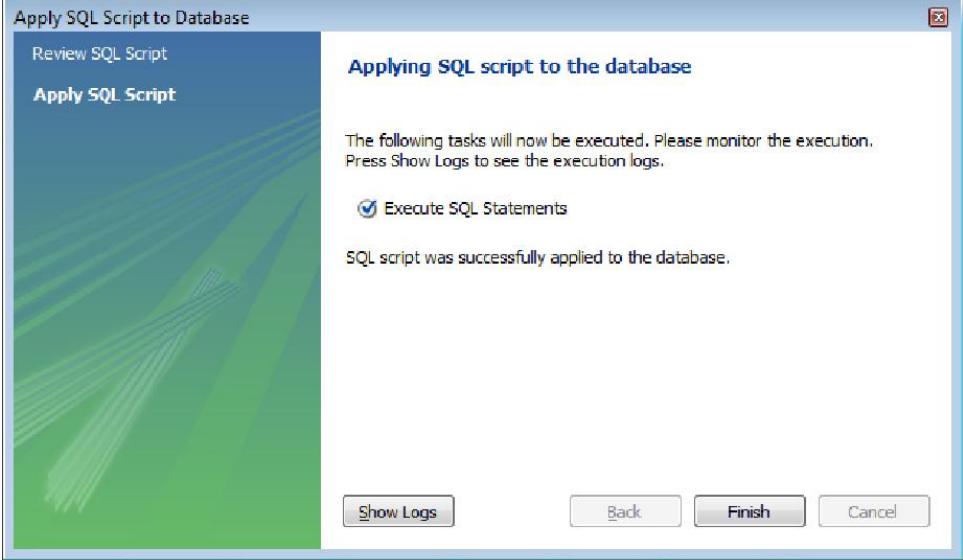
codcliente	cedula	nombre	comollego
43	16090221	Jose Miguel Rojas	1

Open Value in Editor  
Set Field to NULL  
Mark Field Value as a Function/literal  
**Delete Row(s)**  
Load Value From File...  
Save Value To File...  
Copy Row  
Copy Row (with names)  
Copy Row (unquoted)  
Copy Row (with names, unquoted)  
Copy Row (tab separated)  
Copy Field

Las operaciones sobre la tabla no se llevan a cabo hasta que se hace click en el botón "Apply", lo que hace aparecer una ventana con el lote de instrucciones SQL para ejecutar las operaciones realizadas.

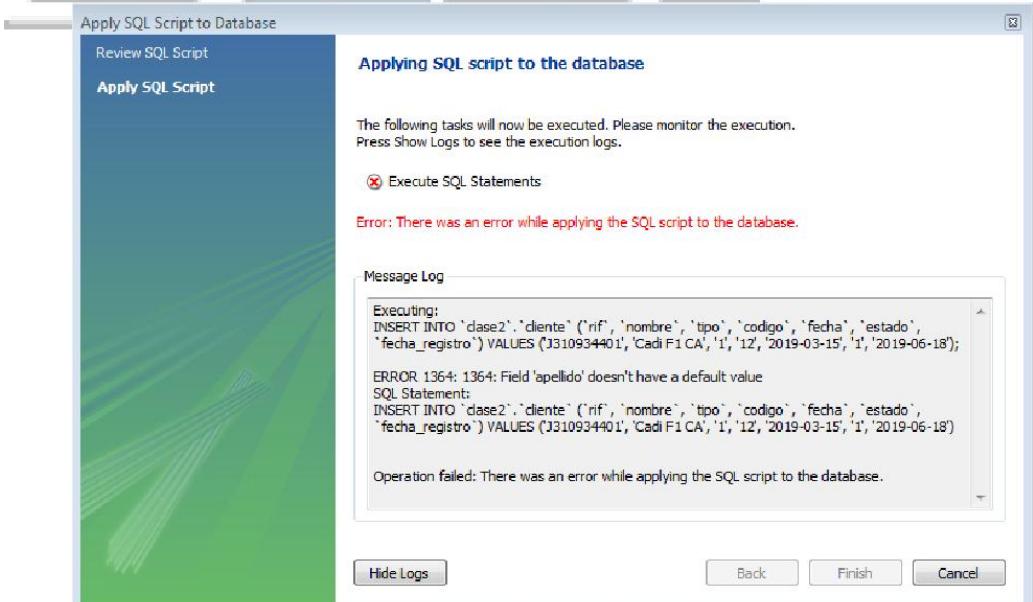


Luego de hacer click en el botón "Apply" en esta ventana, los datos de la tabla si se verán afectados:



The screenshot shows the 'Apply SQL Script to Database' dialog box. On the left, there's a sidebar with 'Review SQL Script' and 'Apply SQL Script' buttons. The main area is titled 'Applying SQL script to the database'. It contains the message: 'The following tasks will now be executed. Please monitor the execution. Press Show Logs to see the execution logs.' Below this is a checked checkbox for 'Execute SQL Statements'. At the bottom, there are 'Show Logs', 'Back', 'Finish', and 'Cancel' buttons.

Es posible que al ejecutar las instrucciones SQL se produzca un error. Los errores son mostrados en la ventana de resultados:



This screenshot shows the same dialog box after an error occurred. The 'Execute SQL Statements' checkbox is unchecked. A red error message at the top right says: 'Error: There was an error while applying the SQL script to the database.' Below this is a 'Message Log' section containing the SQL statement and the error message: 'ERROR 1364: Field 'apellido' doesn't have a default value'. At the bottom, there are 'Hide Logs', 'Back', 'Finish', and 'Cancel' buttons.

### 6.3.- Reiniciar el Auto Incremento

MySQL almacena en un atributo de cada tabla el próximo valor a usar en los campos auto incrementales. Si en algún momento se eliminan todos los registros de la tabla, este valor se puede reiniciar manualmente. Visualizando la ventana de edición de la tabla, se ubica la pestaña "Options" y luego la opción "Auto Increment". Luego de cambiar el valor se hace click en el botón "Apply":



ACADEMIA DE SOFTWARE

## Capítulo 7. TIPOS DE DATOS

### 7.1.- Tipos de Datos Numéricicos

Los datos numéricos se dividen en enteros y reales. Como se explicó en la definición de los campos, los tipos de datos numéricos pueden tener signo (signed) o no (unsigned). Por defecto tienen signo. Para los números enteros hay varios rangos, entre los más pequeños están:

- bit(m): un valor de bit. El parámetro "M" indica el número de bits que puede almacenar un valor, que va desde 1 hasta 64. Si se omite, se asume que es 1. Los valores de un campo tipo bit se pueden guardar como un número decimal que esté en el rango de valores permitido según la cantidad de bits, o se guarda con un número binario con la notación b'1100'.
- Tinyint[(M)]: un entero muy pequeño. Si tiene signo el rango va desde -128 to 127, sin signo el rango va desde 0 hasta 255. M es opcional.
- SMALLINT[(M)]: un entero pequeño. El rango cuando tiene signo es desde -32768 hasta 32767. Cuando no tiene signo el rango va desde 0 hasta 65535. M es opcional.
- Los rangos para enteros más grandes son los siguientes:
- MEDIUMINT[(M)] : un entero de tamaño medio. El rango cuando tiene signo va desde -8388608 hasta 8388607. Sin signo el rango va desde 0 hasta 16777215. M es opcional.
- INT[(M)] o INTEGER[(M)]: un entero de tamaño normal. El rango cuando tiene signo va desde -2147483648 hasta 2147483647. The unsigned range is 0 to 4294967295. M es opcional.
- BIGINT[(M)]: un entero grande. El rango cuando tiene signo va desde -9223372036854775808 hasta 9223372036854775807. Sin signo el rango va desde 0 hasta 18446744073709551615. M es opcional.

- SERIAL: es un alias para la combinación BIGINT UNSIGNED NOT NULL AUTO\_INCREMENT UNIQUE.

En resumen, los posibles valores que pueden tomar los tipos de datos enteros se pueden visualizar en la siguiente tabla:

Tipo de dato	Espacio Requerido Bytes	Mínimo valor con Signo	Mínimo valor sin Signo	Máximo valor con Signo	Máximo valor sin Signo
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	- $2^{63}$	0	$2^{63}-1$	$2^{64}-1$

MySQL permite especificar opcionalmente el ancho (display width) de los tipos de datos enteros, colocado entre paréntesis luego del tipo de dato. Por ejemplo, INT(4) especifica un ancho de 4 dígitos para un INT. Este ancho no restringe el rango de valores que puede almacenar un campo, ni tampoco impide que se guarden valores más anchos que el especificado. Por ejemplo, si a un campo se le asigna el tipo de dato SMALLINT(3) (que tiene un rango entre -32768 y 32767) se le asigna un valor de más de 3 dígitos pero dentro del rango permitido, se almacenará el valor sin ningún problema.

El mayor uso del ancho se obtiene cuando se combina con el atributo ZEROFILL (llenar con ceros). Si la cantidad de dígitos de un valor entero es menor que el ancho, se llenará con ceros a la izquierda hasta alcanzar el ancho. Por ejemplo, para un campo declarado INT(4) ZEROFILL, un valor 5 se llenará con cuatro ceros (0005), el valor 78 se llenará con dos ceros (0078), el valor 160 se llenará con un cero (0160) y así sucesivamente.

Los tipos de datos para números con decimales o reales para mayor precisión son: DECIMAL (equivalente a dec, fixed y numeric), FLOAT y DOUBLE.

- DECIMAL[(M[,D])]: Los tipos de datos DECIMAL y NUMERIC almacenan valores numéricos exactos. Estos tipos de datos son usados cuando se necesita preservar precisión, por ejemplo, para datos monetarios.

M es el número total de dígitos (la precisión) y D es el número de dígitos después del punto decimal (la escala). El punto decimal y el signo "-" no son considerados para la precisión. El máximo valor para M es 65. El valor por defecto para M es 10. El máximo valor para D es 30. El valor por defecto para D es 0.

El Standard SQL requiere que un valor DECIMAL(5,2) sea capaz de almacenar cualquier valor con 5 dígitos (incluyendo 2 decimales), por lo tanto podría almacenar valores en un rango entre -999.99 y 999.99.

Los otros tipos de datos para valores reales son:

- FLOAT[(M,D)]: un número pequeño de punto flotante. Permite almacenar valores desde -3.402823466E+38 hasta -1.175494351E-38, 0, y desde 1.175494351E-38 hasta 3.402823466E+38. Estos son los límites teóricos, aunque el rango podría ser menos dependiendo del hardware y del sistema operativo.
- DOUBLE[(M,D)]: un número de punto flotante de tamaño normal. Los valores permitidos son -1.7976931348623157E+308 hasta -2.2250738585072014E-308, 0, y 2.2250738585072014E-308 hasta 1.7976931348623157E+308.

Al igual que el tipo DECIMAL, M es el número total de dígitos y D es el número de dígitos luego del punto decimal. Si M y D son omitidos, los valores que se pueden almacenar son los valores permitidos por el hardware.

## 7.2.- Datos de Fecha

Para almacenar fechas y horas MySQL provee varias opciones. Entre ellas:

- DATE: es usado para valores de fechas sin horas. Los valores de fechas deben guardarse en el formato "YYYY-MM-DD". El rango de fechas soportado va desde "1000-01-01" hasta "9999-12-31".
- DATETIME: es usado para valores que contienen fecha y hora. El formato para los valores es "YYYY-MM-DD hh:mm:ss". El rango soportado es "1000-01-01 00:00:00" hasta "9999-12-31 23:59:59".

- **TIMESTAMP:** es usado para almacenar valores que contienen fecha y hora en un rango entre "1970-01-01 00:00:01" UTC y "2038-01-19 03:14:07" UTC.
- **TIME[(fsp)]:** una hora. El rango de horas es "-838:59:59.000000" hasta "838:59:59.000000". MySQL muestra los valores de hora en el formato "hh:mm:ss[.fraction]", y permite valores usando String o números. Existe un valor opcional "fsp" en el rango entre 0 to 6 para especificar la precisión de fracciones de segundos. Si se omite, se asume el valor 0.
- **YEAR[(4)]:** un año en formato de cuatro dígitos. MySQL muestra los valores de años YEAR en el formato YYYY. Permite asignar valores usando Strings o números. El rango de valores es desde 1901 hasta 2155, y 0000.

### 7.3.- Datos de Texto

MySQL maneja varios tipos de datos para almacenar cadenas de texto: CHAR, VARCHAR, TEXT (y sus variantes), ENUM y SET. Las más usadas son char y varchar. Estos tipos son similares, pero difieren en cómo se almacenan y recuperan. Desde MySQL 5.0.3, también difieren en la longitud máxima y en cómo se tratan los espacios finales.

Los tipos CHAR y VARCHAR se declaran con una longitud que indica el máximo número de caracteres que quiere almacenar. Por ejemplo, CHAR(30) puede almacenar hasta 30 caracteres.

La longitud de una columna CHAR se fija a la longitud que se declara al crear la tabla. La longitud puede ser cualquier valor de 0 a 255. Cuando los valores CHAR se almacenan, se añaden espacios a la derecha hasta la longitud específica. Cuando los valores CHAR se recuperan, estos espacios se borran.

Los valores en columnas VARCHAR son cadenas de caracteres de longitud variable. En MySQL 5.0, la longitud puede especificarse de 0 a 255 antes de MySQL 5.0.3, y de 0 a 65,535 en 5.0.3 y versiones posteriores. (La máxima longitud efectiva de un VARCHAR en MySQL 5.0 se determina por el tamaño de registro máximo y el conjunto de caracteres usados. La longitud máxima total es de 65,532 bytes.)

En contraste con CHAR, VARCHAR almacena los valores usando sólo los caracteres necesarios, más un byte adicional para la longitud (dos bytes para columnas que se declaran con una longitud superior a 255).

**Importante:**

- Durante el almacenamiento y la recuperación de valores no hace ninguna conversión de mayúsculas y minúsculas.
- Si se asigna un valor a una columna CHAR o VARCHAR que excede la longitud máxima de la columna, el valor se trunca. Si los caracteres truncados no son espacios, se genera una advertencia. Puede hacer que aparezca un error en lugar de una advertencia usando modo SQL estricto.

La siguiente tabla ilustra las diferencias entre los dos tipos de columnas mostrando el resultado de almacenar varios valores de cadenas de caracteres en columnas CHAR(4) y VARCHAR(4) :

Valor	CHAR(4)	Almacenamiento necesario	VARCHAR(4)	Almacenamiento necesario
''	'  '	4 bytes	'  '	1 byte
'ab'	'ab '	4 bytes	'ab '	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefg'h	'abcd'	4 bytes	'abcd'	5 bytes

Si es necesario almacenar valores de texto más largos, MySQL provee el tipo de dato TEXT y sus variantes:

- TINYTEXT: es una columna de texto con una longitud máxima de 255 caracteres ( $2^8 - 1$ ).
- TEXT[ (M)]: es una columna de texto con una longitud máxima de 65,535 ( $2^{16} - 1$ ) caracteres. Se utiliza un valor opcional M. Si se usa este valor, MySQL crea la columna usando el tipo de TEXT más pequeño que pueda contener a los M caracteres.
- MEDIUMTEXT: una columna de tipo texto con una longitud máxima de 16,777,215 ( $2^{24} - 1$ ) caracteres.

- LONGTEXT: una columna de tipo texto con una longitud máxima de 4,294,967,295 o 4GB ( $2^{32} - 1$ ) caracteres. La longitud máxima efectiva de una columna puede depender del tamaño máximo del paquete configurado en el protocolo cliente/servidor y la memoria disponible.

MySQL para la definición de columnas de tipo carácter (char, varchar y text) permite especificar el conjunto de caracteres (CHARACTER SET) y el collation. El conjunto de caracteres es un conjunto de símbolos y su codificación. El "collation" es un conjunto de reglas para comparar los caracteres en un conjunto de caracteres. La comparación de los caracteres y el ordenamiento esta basado en el "collation" asignado a la columna.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
rif	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
nombre	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
tipo	INT(10)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

En la imagen anterior, con el número 1 se resalta donde se selecciona el "charset" y el "collate" para la tabla completa (que aplicaría a todos los campos de tipo carácter) y con el número 2 se resalta el "charset" para un campo (el seleccionado) y el "collate". Si en la definición de un campo no se selecciona un "charset" y "collate", se asume que usará el de la tabla.

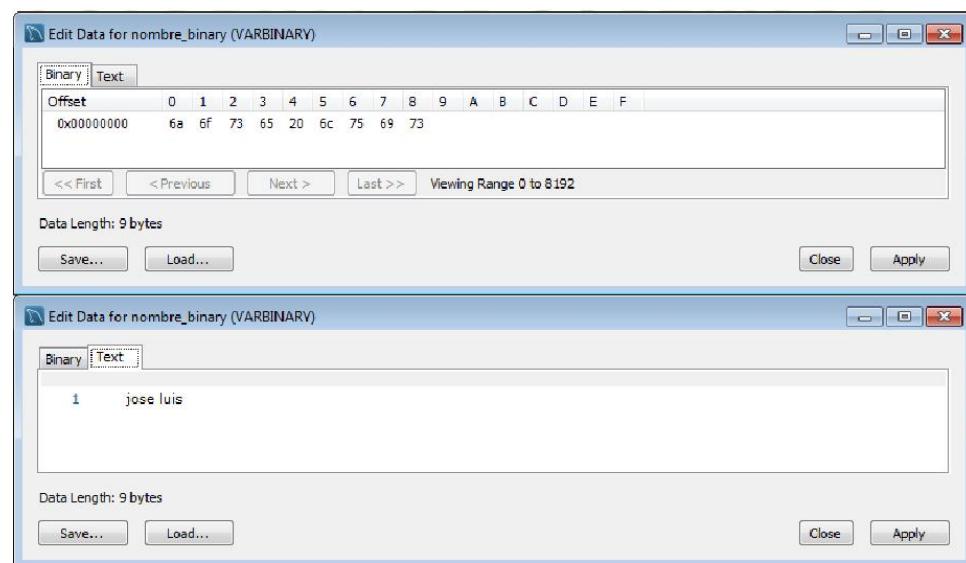
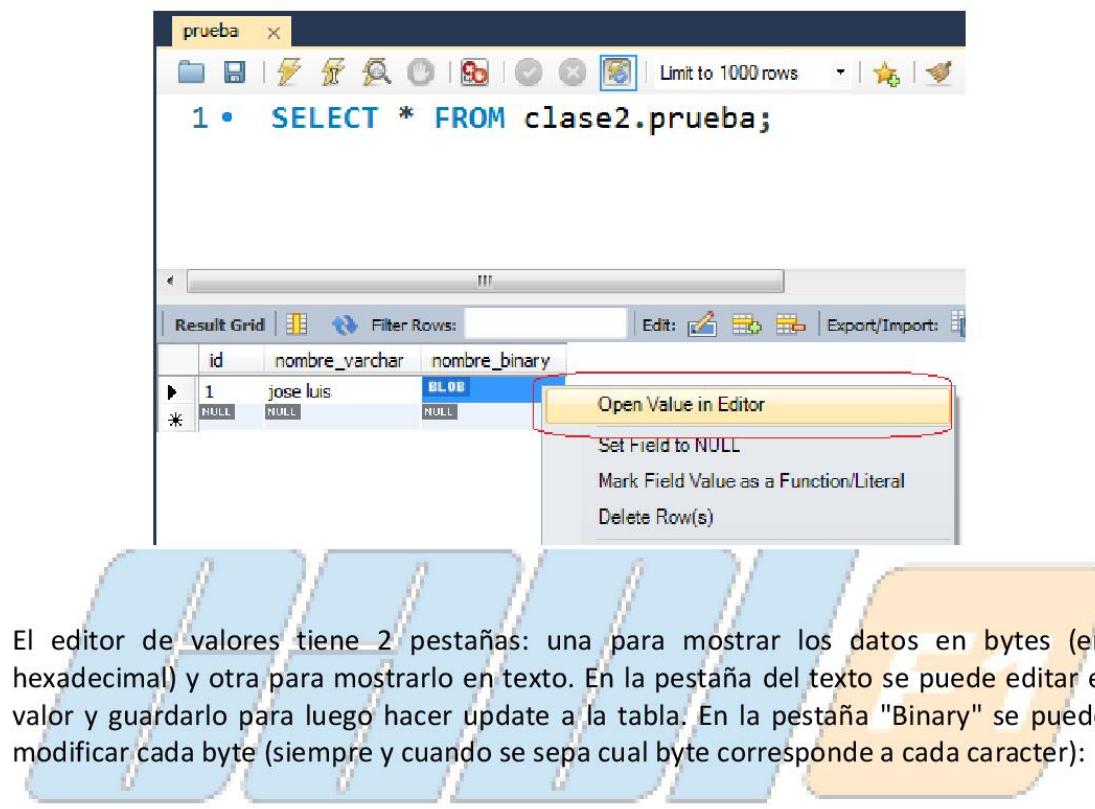
#### 7.4.- Datos en Binario

Los tipos BINARY y VARBINARY son similares a CHAR y VARCHAR, excepto que contienen cadenas de caracteres binarias en lugar de cadenas de caracteres no binarias. Esto es,

contienen cadenas de bytes en lugar de cadenas de caracteres (cada carácter tiene una representación en byte). Esto significa que no tienen conjunto de caracteres asignado, por lo tanto, la comparación y ordenación se basa en los valores numéricos de los valores de los bytes. En la siguiente imagen se muestra un campo con el tipo "varchar" y otro con el tipo "varbinary":

The screenshot shows the 'prueba - Table' configuration window in MySQL Workbench. The table name is set to 'prueba'. The Charset/Collation is 'latin1' with 'latin1\_bin' selected. The table structure includes three columns: 'id' (datatype INT(11), primary key, auto-increment), 'nombre\_varchar' (datatype VARCHAR(45)), and 'nombre\_binary' (datatype VARBINARY(45)). The 'nombre\_binary' column is highlighted with a red box.

Al visualizar los datos de esta tabla, el campo de tipo "varchar" puede visualizarse directamente, pero el "varbinary" no. Para ver el contenido del campo "varbinary" debe abrirse un editor de valores haciendo click con el botón derecho del ratón sobre el valor que se desea visualizar, como se muestra en la siguiente imagen:



El tratamiento de los espacios finales es el mismo para BINARY y VARBINARY como lo es para CHAR y VARCHAR. Cuando se almacenan los valores BINARY, se rellenan con espacios a la derecha hasta la longitud especificada. Cuando los valores BINARY se recuperan, los espacios finales se eliminan. Para VARBINARY, los espacios finales se eliminan cuando los valores se almacenan. Desde MySQL 5.0.3, los espacios finales se mantienen. Debe considerar estas características si planea usar estos tipos de datos para almacenar datos binarios que deban acabar con espacios.

Los campos de tipo char o varchar pueden ser marcados como "binary" (B). A partir de MySQL 5.0, BINARY y VARBINARY son tipos de datos distintos. Para CHAR(M) BINARY y VARCHAR(M) BINARY, el atributo BINARY hace que se use el collation binaria para la columna, pero la columna no contiene cadenas de caracteres no binarios en lugar de cadenas binarias de bytes. Por ejemplo CHAR(5) BINARY se trata como CHAR(5) CHARACTER SET latin1 COLLATE latin1\_bin, asumiendo que el conjunto de caracteres por defecto es latin1.

Un BLOB es un objeto binario que puede tratar una cantidad de datos variables. Las columnas BLOB se tratan como cadenas de caracteres binarias (de bytes). Las columnas BLOB no tienen conjunto de caracteres, y la ordenación y la comparación se basan en los valores numéricos de los bytes. En la mayoría de aspectos, puede tratar una columna BLOB como VARBINARY que puede ser tan grande como desee.

Si asigna un valor a una columna BLOB que excede la longitud máxima del tipo de la columna, el valor se trunca. Si los caracteres truncados no son espacios, aparece una advertencia. Puede hacer que aparezca un error en lugar de una advertencia usando el modo SQL estricto. Los cuatro tipos BLOB son:

- TINYBLOB
- BLOB
- MEDIUMBLOB
- LONGBLOB.

### 7.5.- Conjuntos y Enumerados

MySQL tiene dos tipos de datos en los que se puede definir el sub conjunto de valores que puede tomar el campo. Los tipos son: "Enum" (enumerados) y "Set" (conjuntos).

El tipo de dato ENUM("value1","value2",...) es un objeto string que puede tener un valor seleccionado de una lista de valores, puede ser NULL o el valor especial " ". Los valores del enum son representados internamente como un entero. Puede tener hasta 65535 elementos distintos. La longitud máxima de un elemento individual es 255 caracteres. Por ejemplo:

```
CREATE TABLE sizes (
    size ENUM("small", "medium", "large") NOT NULL DEFAULT "medium"
);
```

En el ejemplo anterior, el enumerado tiene tres posibles valores: "small", "medium" y "large". Cada valor de la lista es numerado con un índice (empieza desde el 1), pudiendo usar el índice en vez del valor de la lista (en el ejemplo de arriba: 1=>"small", 2=>"medium" y 3=>"large"). En caso de introducir un valor no perteneciente a la lista, el campo pasará a valer "" (una cadena vacía) que tiene el índice zero.

El tipo de dato SET("value1","value2",...) es muy parecido al ENUM, solo que puede tener cero, uno o más valores, cada uno de los cuales debe ser escogido de una lista de valores representados internamente como números enteros. Un campo de tipo SET puede tener un máximo de 64 valores distintos. Por ejemplo:

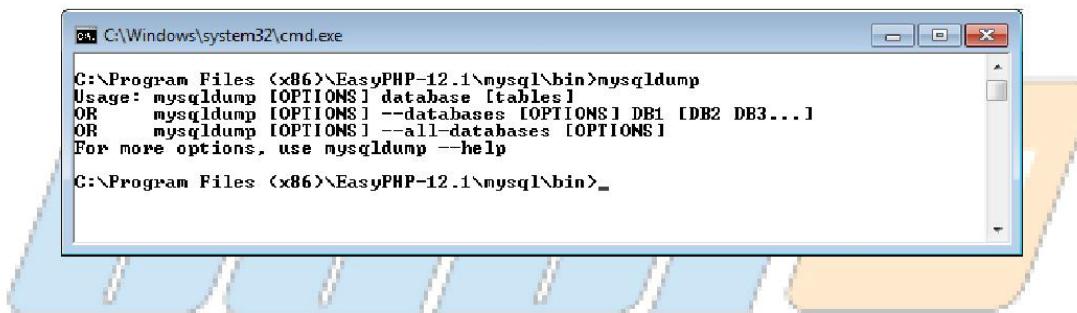
```
CREATE TABLE letters (
    letter SET("a", "b", "c", "d")
);
```

En el ejemplo anterior, los valores del conjunto son: "a", "b", "c" y "d". La longitud máxima de un elemento individual es de 255 caracteres. Los valores no pueden contener comas, ya que los valores asignados son separados por comas

## Capítulo 8. EXPORTAR E IMPORTAR

### 8.1.- Mysqldump

Para pasar una base de datos de un servidor a otro o para hacer un respaldo de los datos, la estructura de la base de datos o ambos, se ejecuta el proceso de exportación de la base de datos. Para exportar, la instalación del servidor MySQL incluye una herramienta llamada "mysqldump", que se encuentra en la carpeta "bin" de la instalación. Se ubica la ventana de comandos del sistema operativo para luego ubicarse en la carpeta "bin". Al ejecutar el comando, se muestran las posibles opciones que se pueden usar:



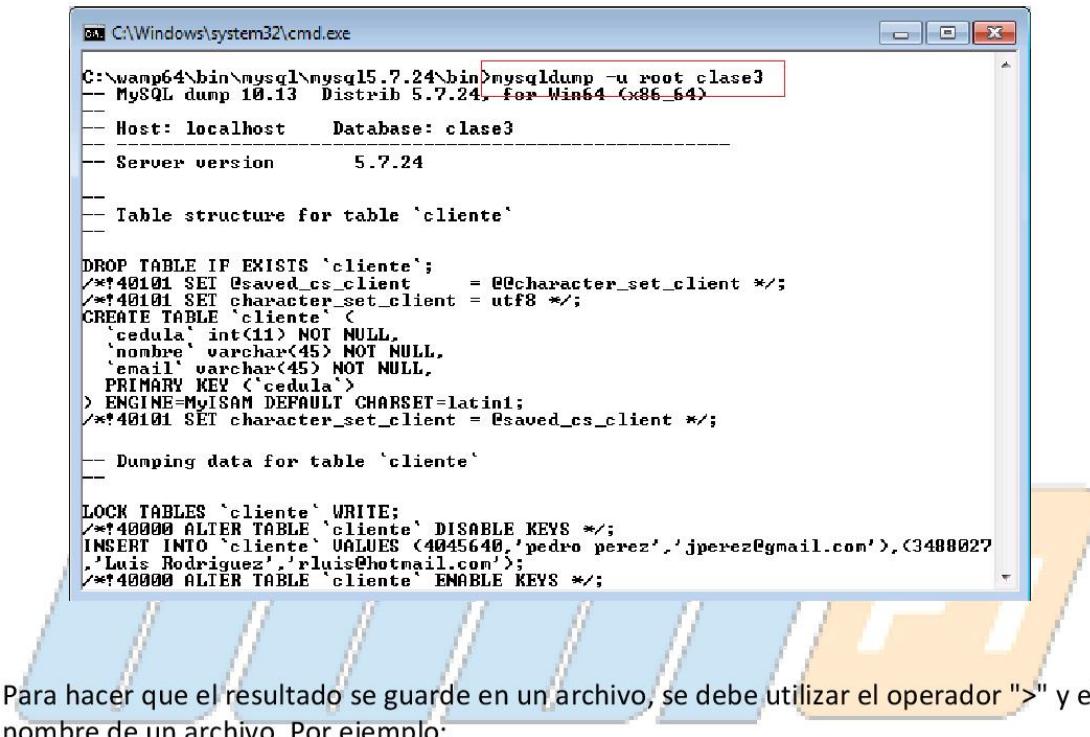
Al igual que el cliente "mysql", mysqldump necesita los parámetros para conectarse al servidor donde están las bases de datos que se desean exportar. Los parámetros son:

- host (-h): dirección de la computadora donde está ejecutándose MySQL.
- user (-u): nombre de usuario.
- password (-p): contraseña del usuario.

Se pueden exportar:

- todas las bases de datos: mysqldump [opciones de conexión] --all-databases.
- una o algunas bases de datos completas: mysqldump [opciones de conexión] nombreBd1 [nombreBd2]
- una o algunas tablas de una base de datos: mysqldump [opciones de conexión] nombreBd1 tabla1

Por defecto, el resultado de mysqldump se muestra en pantalla. Por ejemplo:

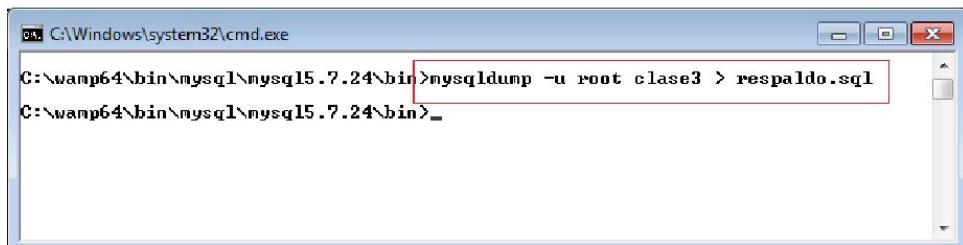


```
C:\Windows\system32\cmd.exe
C:\wamp64\bin\mysql\mysql5.7.24\bin>mysqldump -u root clase3
-- MySQL dump 10.13 Distrib 5.7.24, for Win64 (x86_64)
...
-- Host: localhost      Database: clase3
...
-- Server version      5.7.24
...
-- Table structure for table 'cliente'
...
DROP TABLE IF EXISTS `cliente`;
/* 40101 SET @saved_cs_client      = @@character_set_client */;
/* 40101 SET character_set_client = utf8 */;
CREATE TABLE `cliente` (
  `cedula` int(11) NOT NULL,
  `nombre` varchar(45) NOT NULL,
  `email` varchar(45) NOT NULL,
  PRIMARY KEY (`cedula`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
/* 40101 SET character_set_client = @saved_cs_client */;

-- Dumping data for table 'cliente'
...
LOCK TABLES `cliente` WRITE;
/* 40000 ALTER TABLE `cliente` DISABLE KEYS */;
INSERT INTO `cliente` VALUES (4045640,'pedro perez','jperez@gmail.com'),(3488027
,'Luis Rodriguez','rluis@hotmail.com');
/* 40000 ALTER TABLE `cliente` ENABLE KEYS */;
```

Para hacer que el resultado se guarde en un archivo, se debe utilizar el operador ">" y el nombre de un archivo. Por ejemplo:

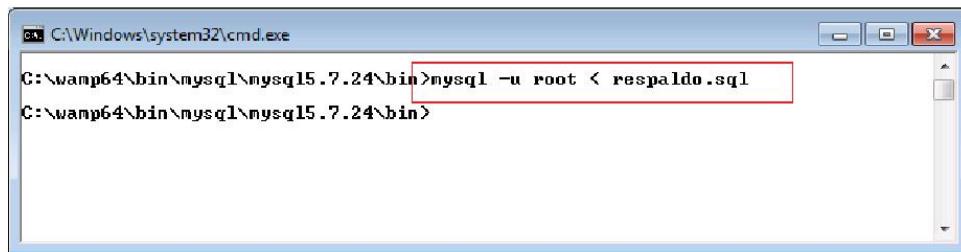
## ACADEMIA DE SOFTWARE



```
C:\Windows\system32\cmd.exe
C:\wamp64\bin\mysql\mysql5.7.24\bin>mysqldump -u root clase3 > respaldo.sql
C:\wamp64\bin\mysql\mysql5.7.24\bin>
```

Un archivo SQL puede ejecutarse por consola usando el cliente de interfaz de línea de comandos utilizando el operador "<", que le indica a la aplicación que debe leer del archivo que se coloque la lado. Se deben usar todos los parámetros necesarios para la conexión al servidor (como se explica en el capítulo 1). Por ejemplo, si se necesita

importar el archivo "respaldo.sql" en el servidor local, se puede hacer la conexión usando las credenciales de root de la siguiente forma:

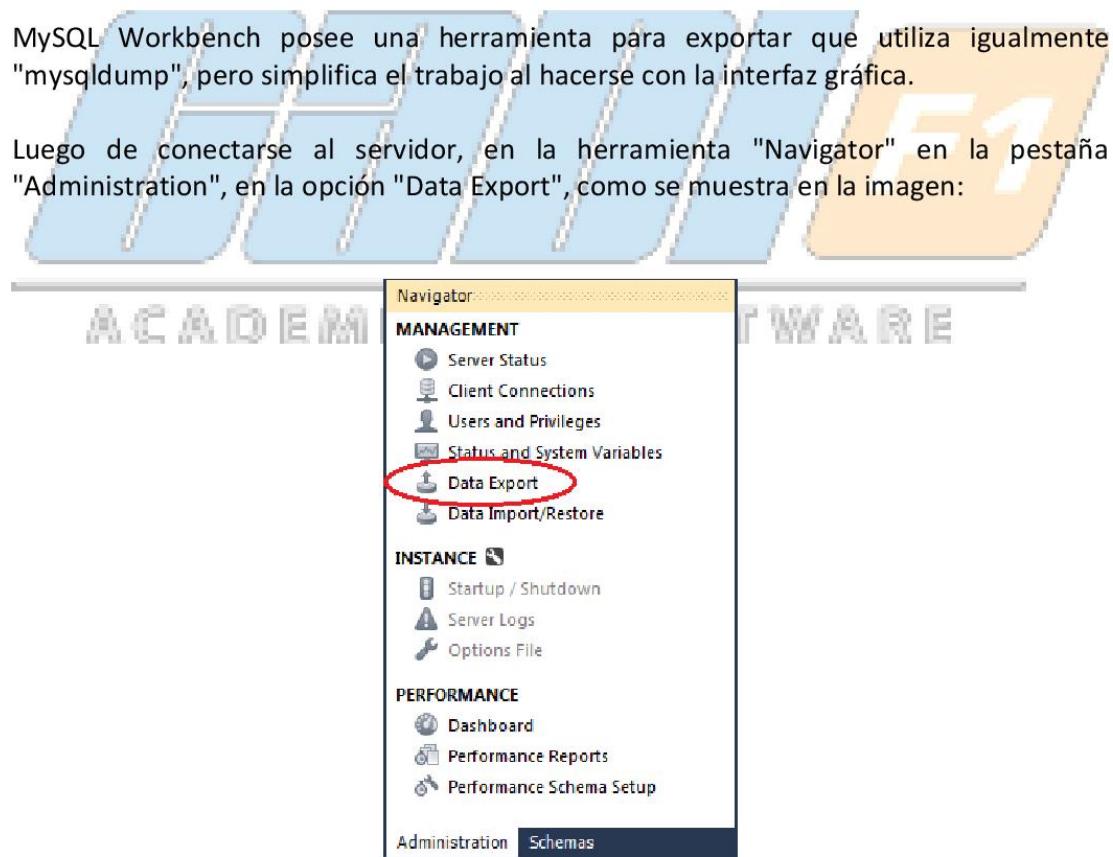


```
C:\Windows\system32\cmd.exe
C:\wamp64\bin\mysql\mysql15.7.24\bin>mysql -u root < respaldo.sql
C:\wamp64\bin\mysql\mysql15.7.24\bin>
```

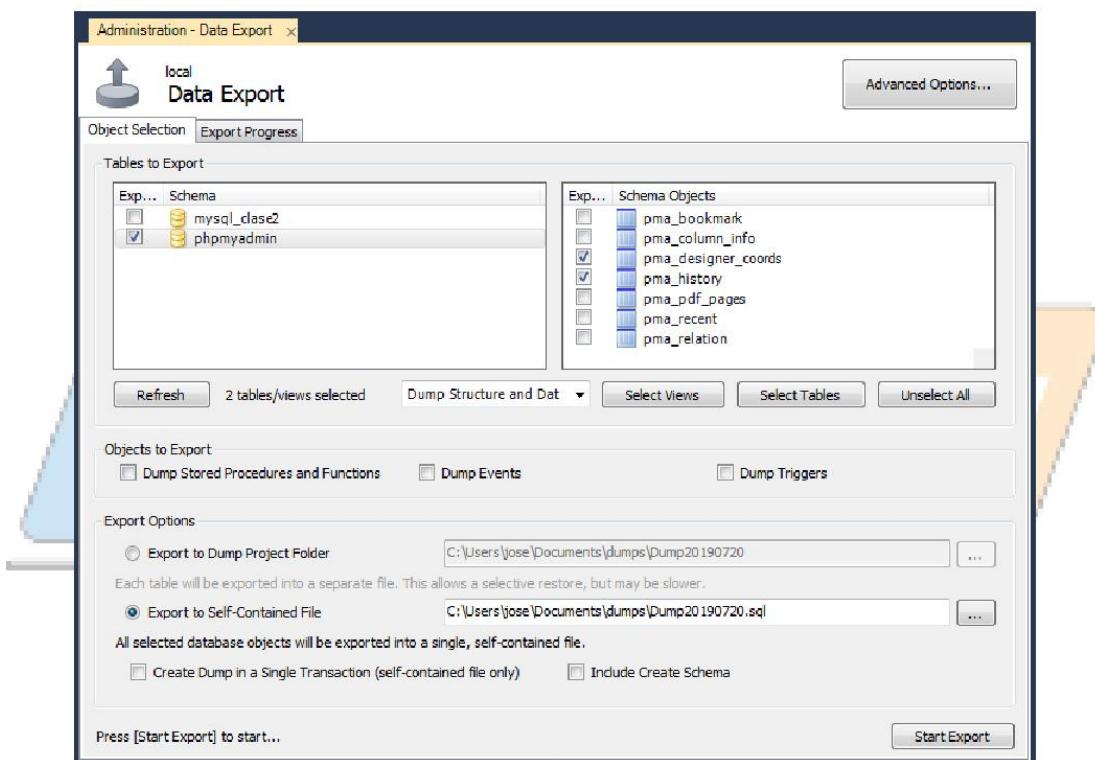
## 8.2.- Exportar en Mysql Workbench

MySQL Workbench posee una herramienta para exportar que utiliza igualmente "mysqldump", pero simplifica el trabajo al hacerse con la interfaz gráfica.

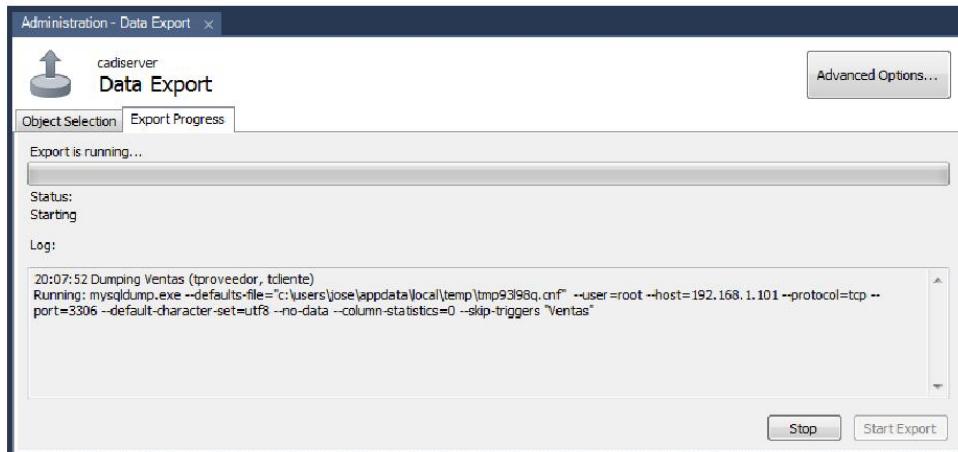
Luego de conectarse al servidor, en la herramienta "Navigator" en la pestaña "Administration", en la opción "Data Export", como se muestra en la imagen:



Se abre una pestaña donde se muestran las bases de datos alojadas en el servidor al cual se está conectado. Se puede marcar una o varias bases de datos. De una base de datos se puede seleccionar una o varias tablas. Se puede exportar cada tabla en un archivo por separado con la opción "Export to Dump Project Folder" o se puede exportar todo en un único archivo con la opción "Export to Self-Contained File":

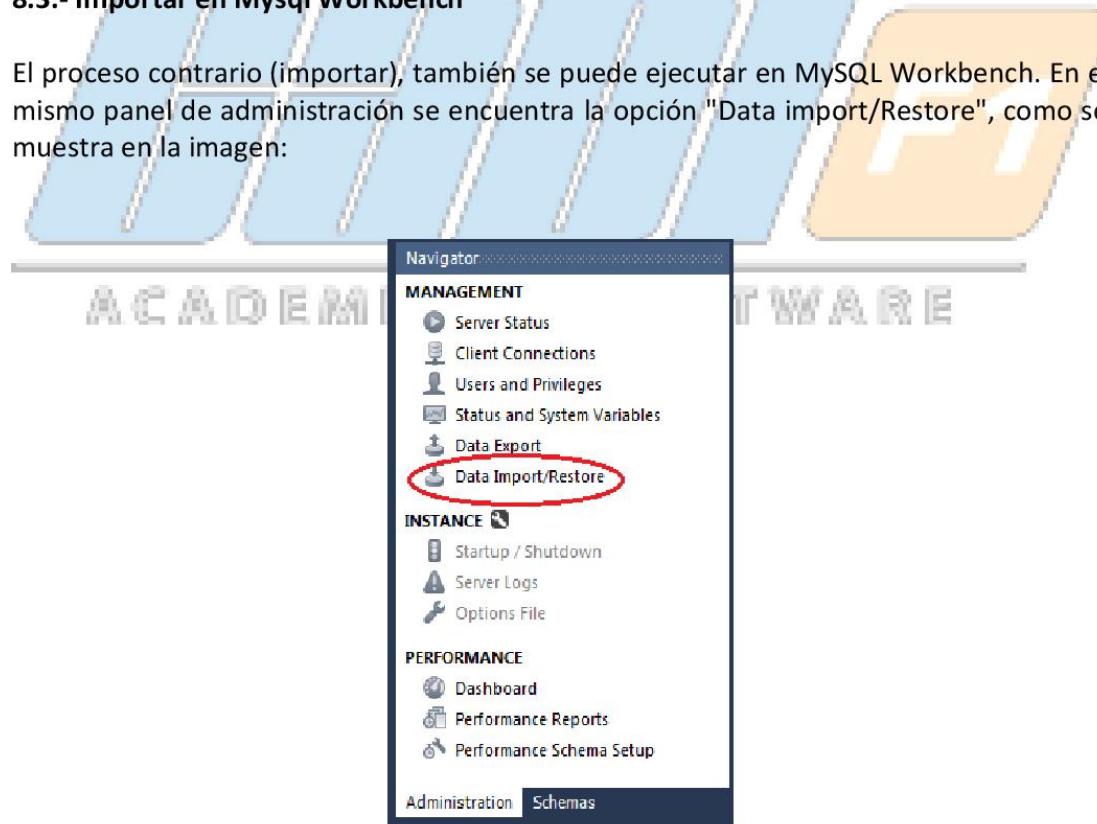


Al hacer click en el botón "Start Export", se abre una pestaña donde se muestra el progreso de la exportación. El proceso puede tardar si son muchas tablas y si las tablas tienen muchos datos. Si ocurre un error, se mostrará en esta ventana:

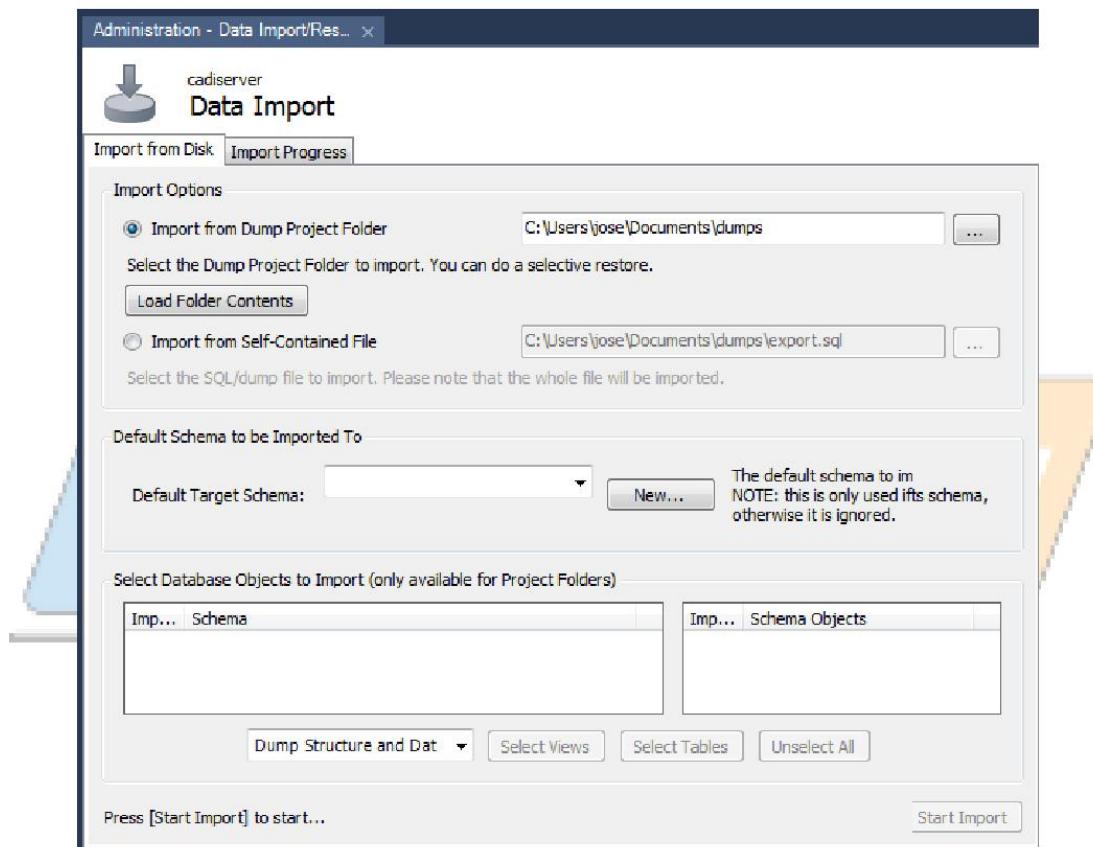


### 8.3.- Importar en Mysql Workbench

El proceso contrario (importar), también se puede ejecutar en MySQL Workbench. En el mismo panel de administración se encuentra la opción "Data import/Restore", como se muestra en la imagen:



Se abre una pestaña, donde se puede seleccionar entre cargar los archivos de una carpeta (Import from Dump Project Folder) o cargar un archivo auto contenido (Import Self Contained File):



Si se selecciona la opción "Importar from Dump Project", se debe seleccionar la base de datos existente en el servidor que será el destino de la importación, es decir, la base de datos en la cual se ejecutarán las instrucciones SQL contenidas en los archivos SQL que están en la carpeta. Se especifica la base de datos destino en la opción "Default Target Schema".

Si se selecciona la opción "Import from Self Contained File" y si el archivo que se esta importando NO contiene la instrucción "create database" o "create schema", se debe

seleccionar la base de datos en donde se van a ejecutar las instrucciones SQL que están en el archivo. Si el archivo que se está importando contiene la instrucción para crear la base de datos, no hace falta seleccionar la base de datos destino.

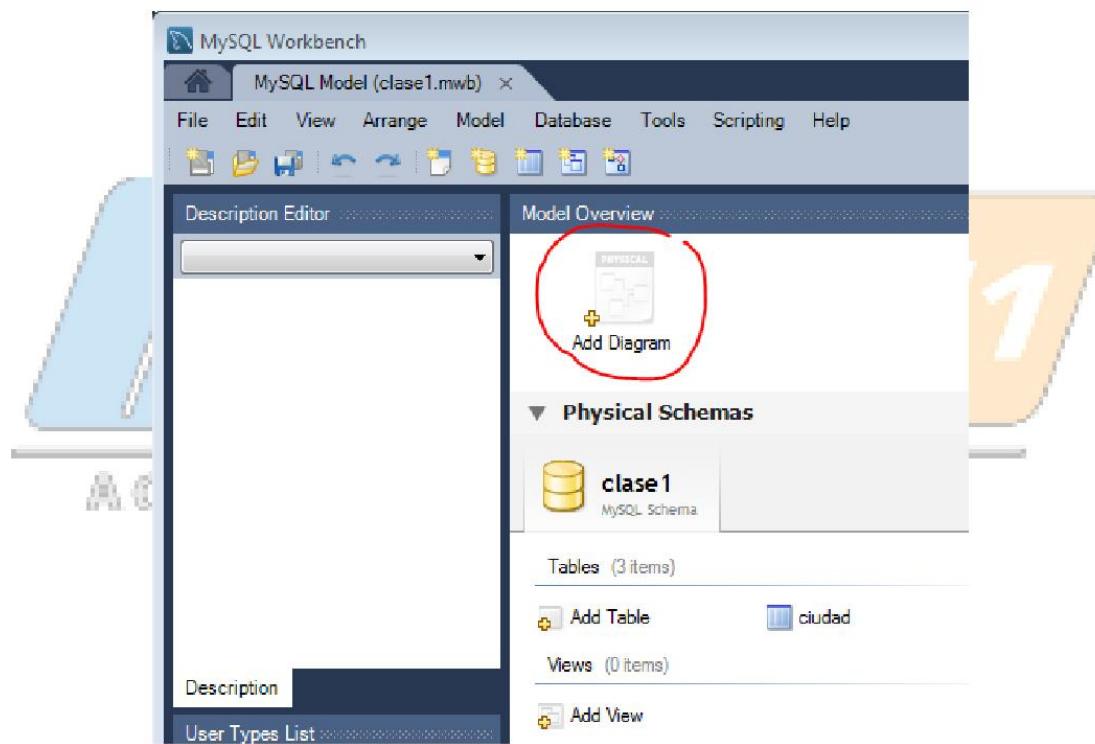
Al presionar el botón "Start Import" se abre otra pestaña con el progreso del proceso de importación. Si ocurre algún error en el proceso, se mostrará en esta pestaña.



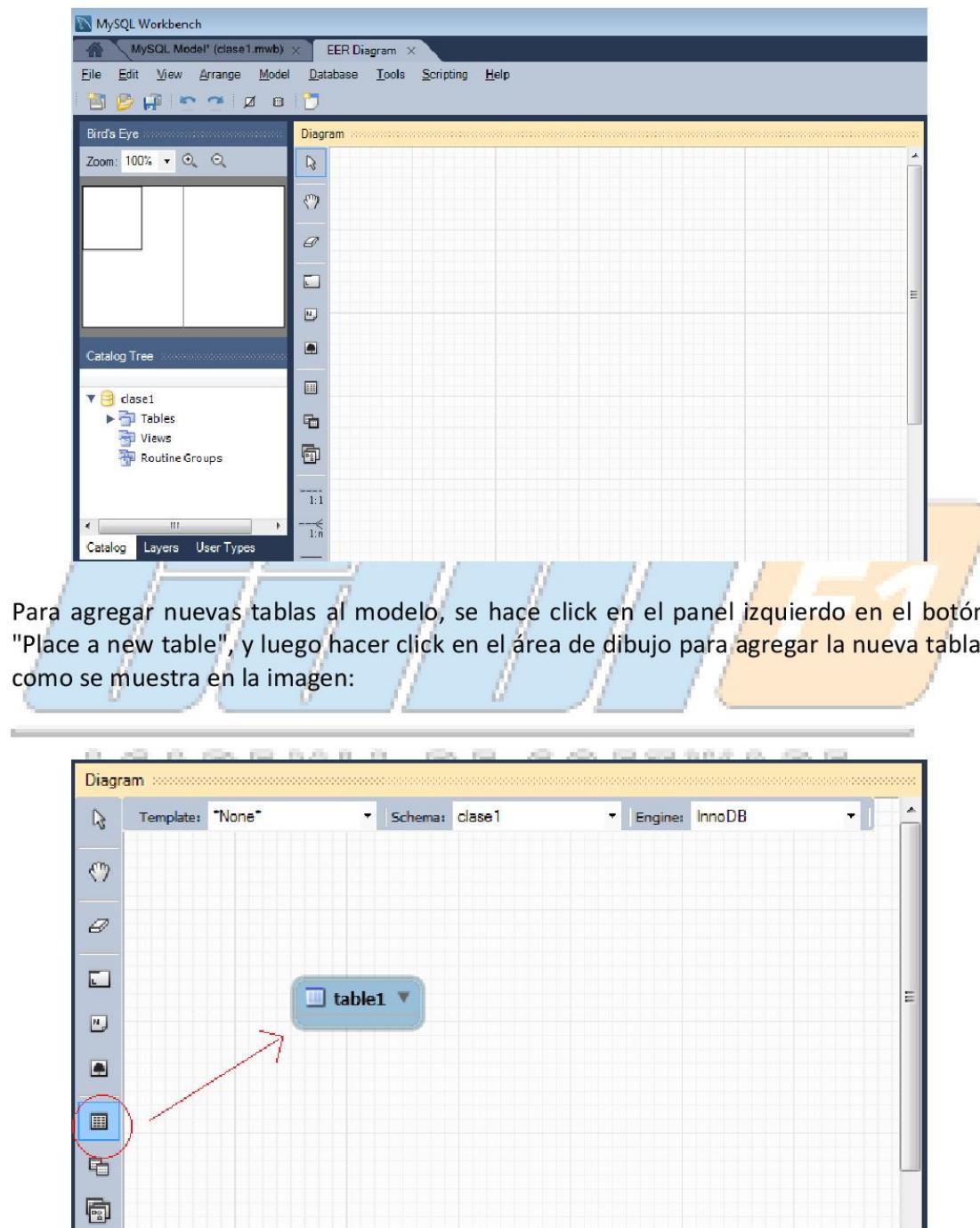
## Capítulo 9. TRABAJANDO CON MODELOS. PARTE 2

### 9.1.- Crear un Mer

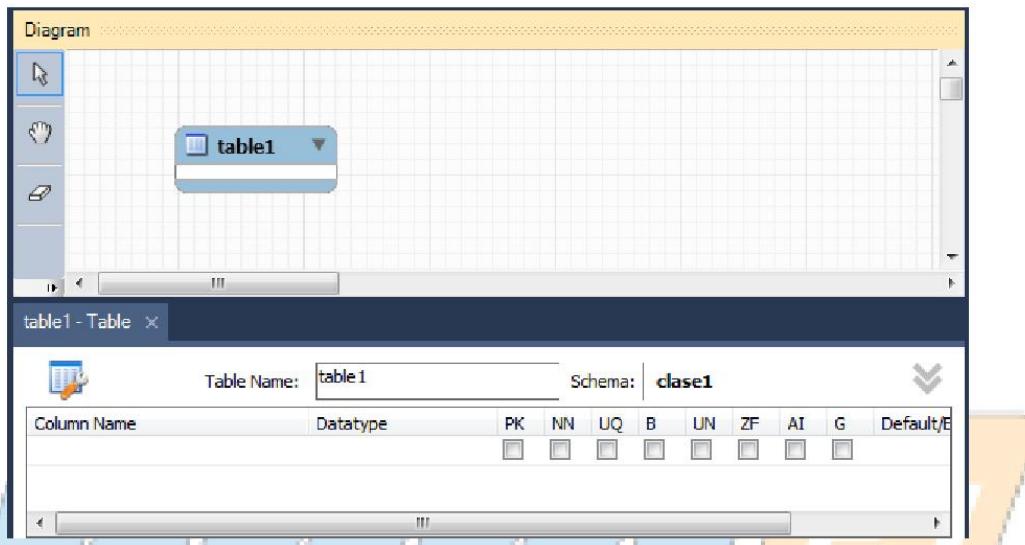
Al crear un modelo en MySQL Workbench se puede crear un Modelo Entidad Relación (MER) o EER Diagram (Enhanced entity–relationship). En la parte superior, aparece un botón con el texto "Add Diagram", como se muestra en la imagen:



Se abre una pestaña adicional, donde se muestra un área de dibujo para crear el MER:

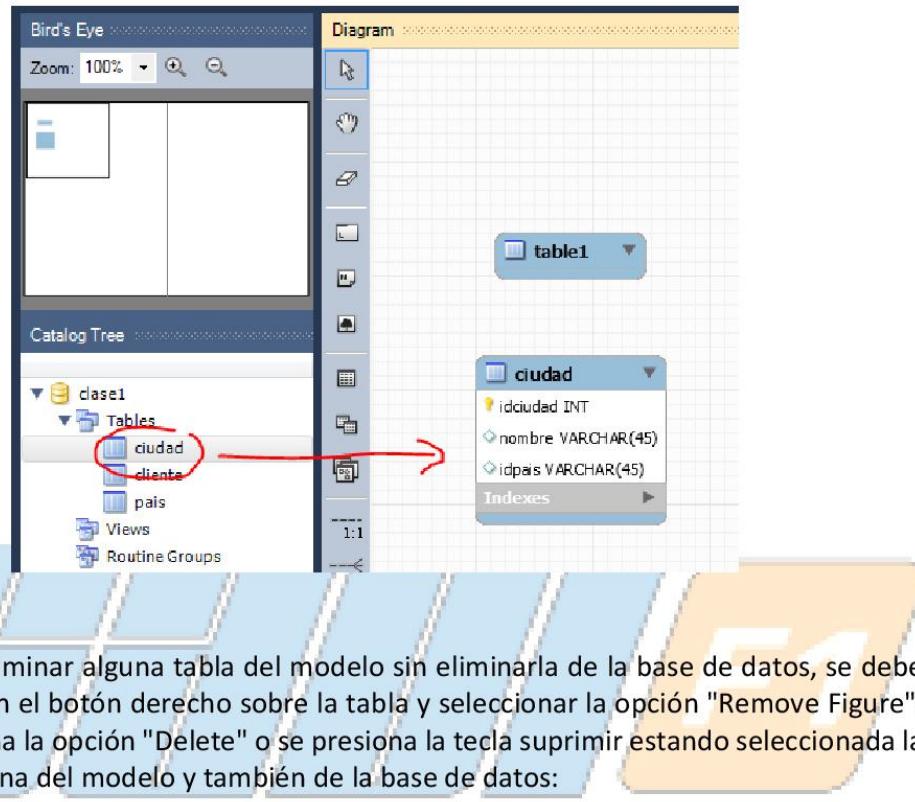


Al crear una tabla nueva, haciendo doble click sobre ésta se abre el panel de edición de la tabla para especificar su nombre y los campos que ésta tendrá:

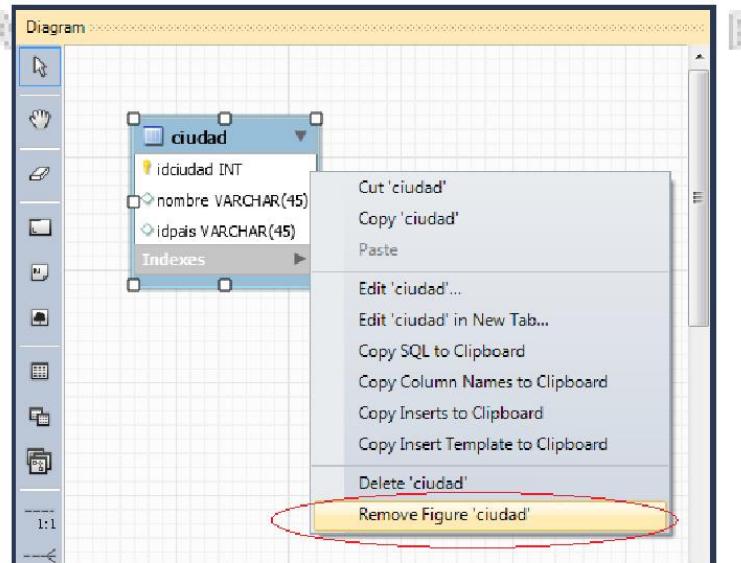


Si ya existen tablas en el modelo, se pueden visualizar en el panel izquierdo y se pueden arrastrar hacia el área de dibujo:

ACADEMIA DE SOFTWARE



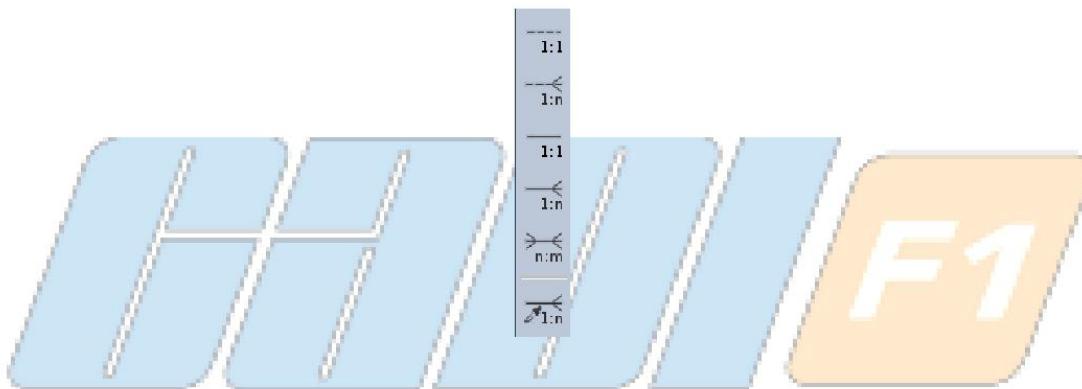
Si se desea eliminar alguna tabla del modelo sin eliminarla de la base de datos, se debe hacer click con el botón derecho sobre la tabla y seleccionar la opción "Remove Figure". Si se selecciona la opción "Delete" o se presiona la tecla suprimir estando seleccionada la tabla, se elimina del modelo y también de la base de datos:



## 9.2.- Relaciones Entre Tablas

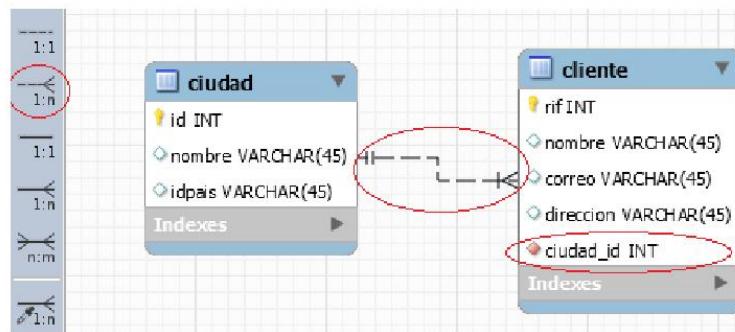
El objetivo principal de un MER es mostrar las relaciones entre las tablas. Las relaciones entre las tablas se crean utilizando el panel de herramientas, que permite crear varios tipos de relaciones:

- 1:1: relación uno a uno.
- 1:n: relación uno a muchos.
- n:m: relación muchos a muchos.



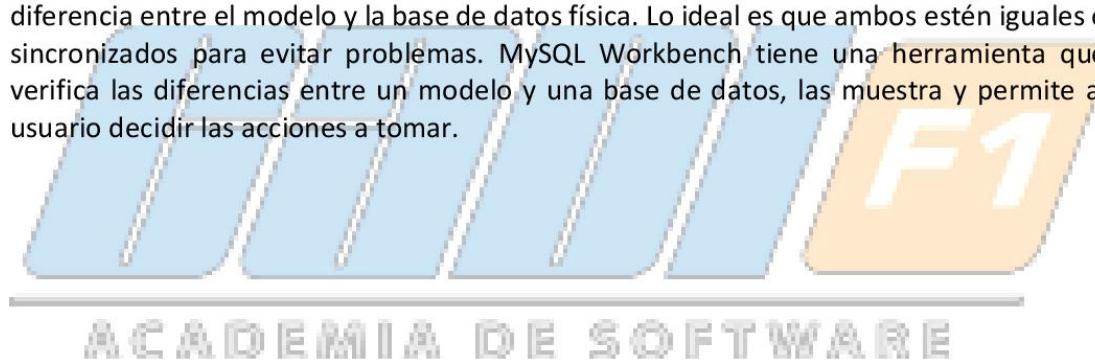
Las relaciones se establecen en las tablas de 2 en 2. Para establecer la relación entre 2 tablas, primero se hace click en el tipo de relación que se desea crear, luego en una de las tablas y luego en la otra tabla. El orden en que se hacen click en las tablas es muy relevante, porque indicará a MySQL Workbench donde debe crear la clave foránea.

En el siguiente ejemplo, para crear una relación 1 a muchos entre las tablas "ciudad" y "cliente" (en una ciudad viven muchos clientes), primero se hace click en la tabla "cliente" y luego en la tabla "ciudad". MySQL Workbench crea la clave foránea en la tabla "cliente", usando el nombre de la tabla "ciudad" y el nombre de la clave primaria:



### 9.3.- Sincronizar la Base de Datos

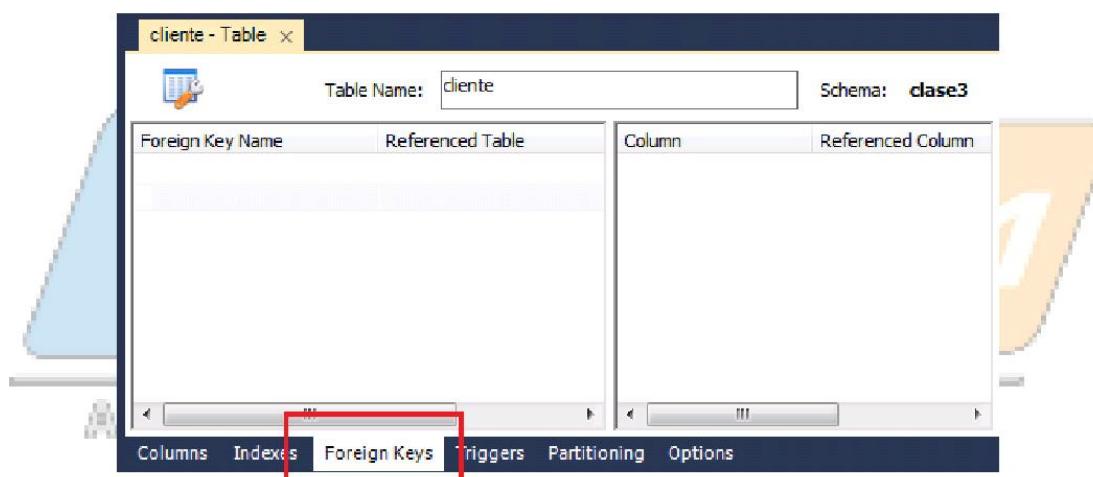
Al hacer cambios en la base de datos o en el MER unilateralmente, se puede crear una diferencia entre el modelo y la base de datos física. Lo ideal es que ambos estén iguales o sincronizados para evitar problemas. MySQL Workbench tiene una herramienta que verifica las diferencias entre un modelo y una base de datos, las muestra y permite al usuario decidir las acciones a tomar.



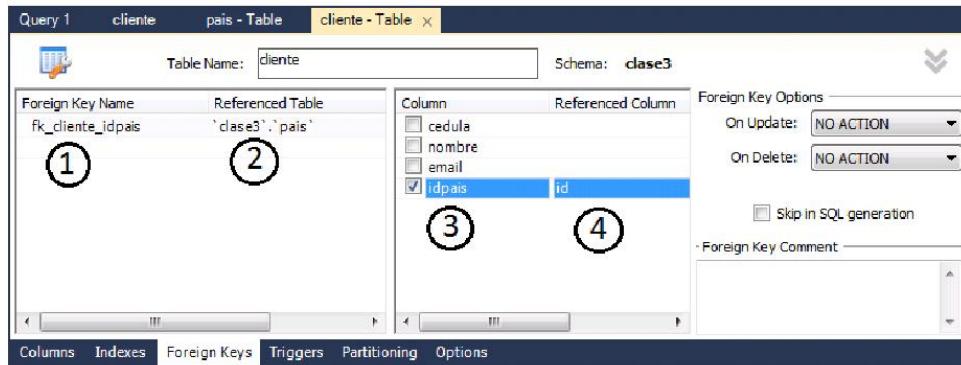
## Capítulo 10. DEFINICIÓN DE CLAVES FORÁNEAS

### 10.1.- Definir Una Clave Foránea

Al crear relaciones en los Mer, MySQL Workbench crea automáticamente las claves foráneas en las tablas destino. Pero, si se crearon las tablas manualmente no partiendo de un modelo y usando la herramienta Forward Engineer, se deben crear manualmente las claves foráneas de las tablas. Para crear una clave foránea, se abre la ventana de edición de la tabla que contendrá la clave foránea y se ubica la pestaña "Foreign Keys":



Al establecer una clave foránea manualmente se deben establecer 4 parámetros obligatorios:



Los parámetros son:

- 1- el nombre del índice de clave foránea: debe ser único en toda la base de datos.
- 2- la tabla referencia: tabla donde se encuentran la clave primaria.
- 3- campo clave foránea: el campo de la tabla actual que será la clave foránea.
- 4- clave primaria: campo que es clave primaria en la tabla referencia.

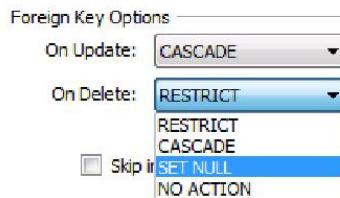
Ambas tablas deben tener el Engine "Innodb" y ambos campos deben ser del mismo tipo de dato. En el ejemplo de la imagen anterior, se desea crear en una tabla con el nombre "cliente" una clave foránea que proviene de la tabla "pais" (en un país hay muchos clientes y un cliente está asociado a un país). En la tabla "cliente" ya debe existir el campo que será la clave foránea. En el ejemplo, este campo se llama "idpais". Se asignó al índice el nombre "fk\_cliente\_idpais" (al usar la nomenclatura fk\_tabla\_campo se puede garantizar que el nombre sea único en la base de datos), se seleccionó la tabla "pais" como referencia, se seleccionó el campo "idpais" de la tabla "cliente" y el campo que es clave primaria de la tabla "pais", el campo "id".

## 10.2.- Restricciones de Clave Foránea

Al crear una clave foránea, se pueden establecer restricciones para cuando ocurran las operaciones "update" o "delete" en la tabla referencia, sobre un registro que tenga asociados uno o más registros en la tabla donde está la clave foránea.

Cuando se intenta actualizar un registro (on update) y/o cuando se intenta eliminar un registro (on delete). Existen 4 opciones para cada uno:

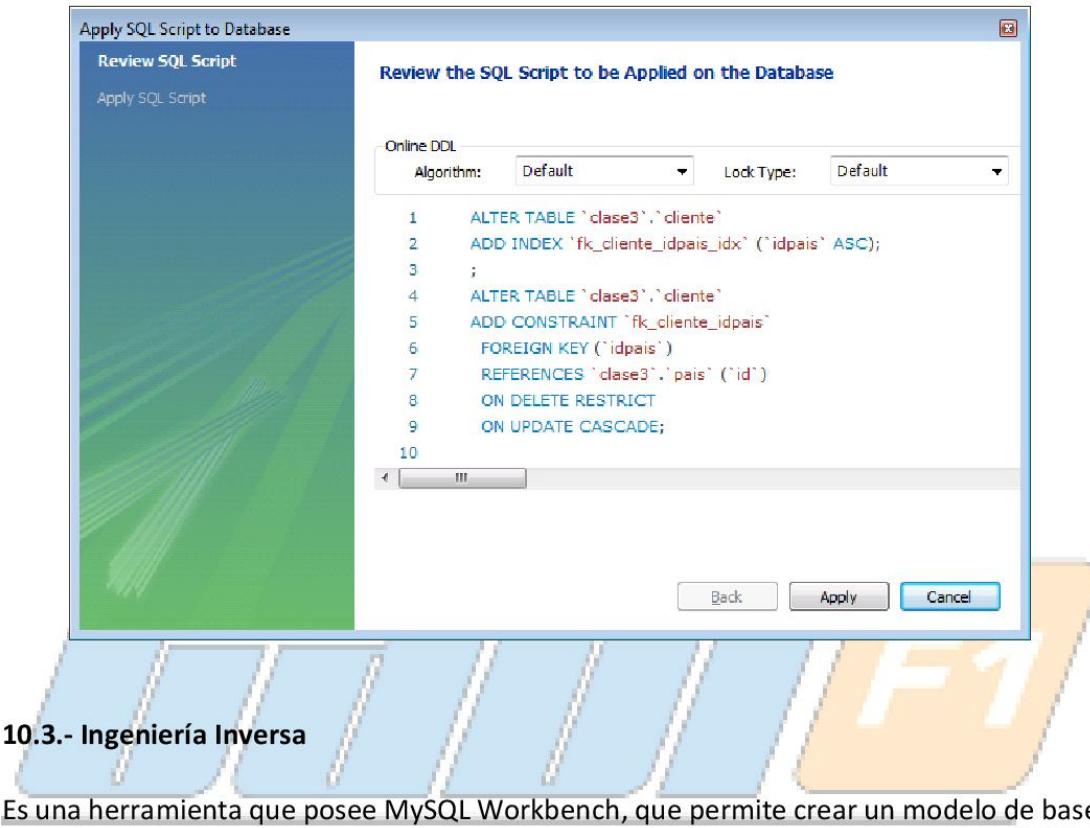
Restrict, Cascade, Set null, No action. Por defecto, se asigna el valor No action. Este valor se cambia en el área identificada con el texto "Foreign Key Options":



La explicación de cada opción es la siguiente:

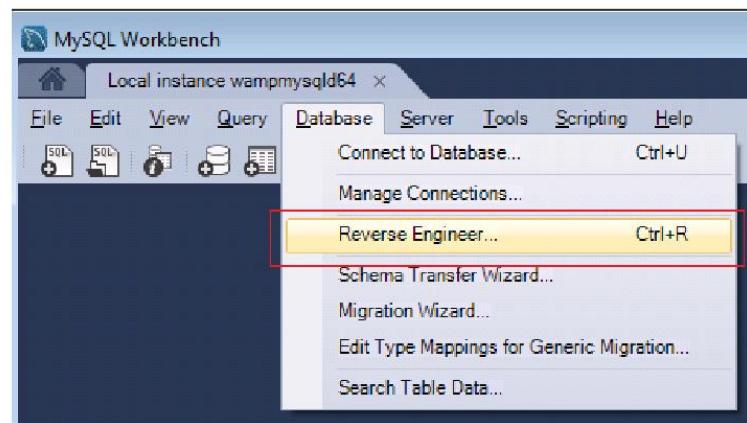
- Restrict: indica que si se intenta eliminar o actualizar un registro en la tabla referencia y hay registros asociados en la tabla donde está la clave foránea, MySQL impide que se haga la eliminación o la actualización en la tabla referencia.
- Cascade: indica que si se intenta eliminar o actualizar un registro en la tabla referencia y hay registros asociados en la tabla donde está la clave foránea, MySQL eliminará o actualizará en cascada en la tabla donde está la clave foránea los registros asociados al registro que se está eliminando o actualizando en la tabla referencia.
- Set null: indica que si se intenta eliminar o actualizar un registro en la tabla referencia y hay registros asociados en la tabla donde está la clave foránea, MySQL asignará el valor NULL en el campo que es clave foránea en los registros asociados al registro que se está eliminando o actualizando en la tabla referencia.
- No action: indica que no sucederá nada en la tabla donde está la clave foránea si se actualiza o se elimina un registro en la tabla referencia.

Cuando se establecen todos los parámetros de la(s) clave(s) foránea(s), al guardar los cambios, MySQL Workbench mostrará las instrucciones SQL que se van a ejecutar para crear la(s) clave(s) foránea(s):

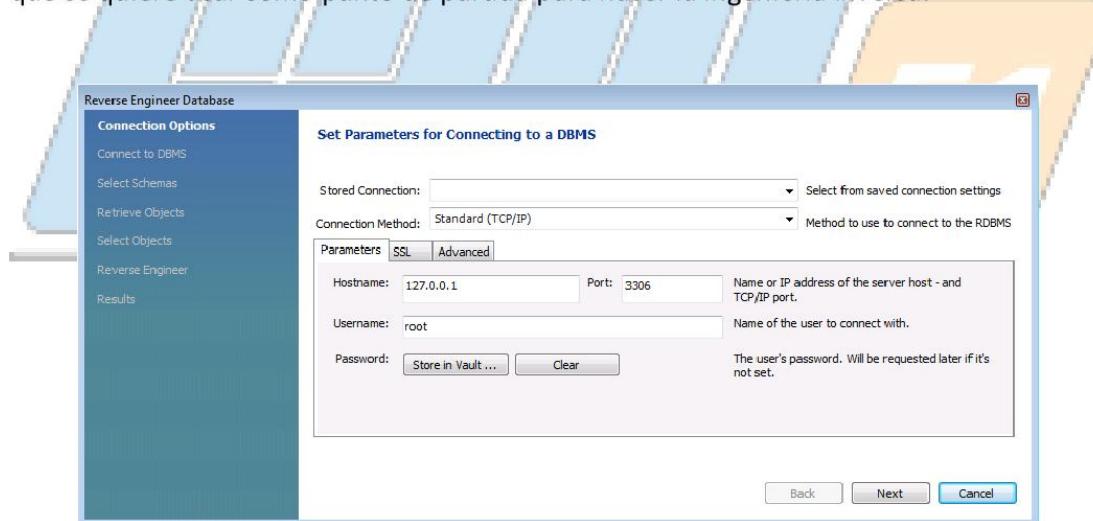


### 10.3.- Ingeniería Inversa

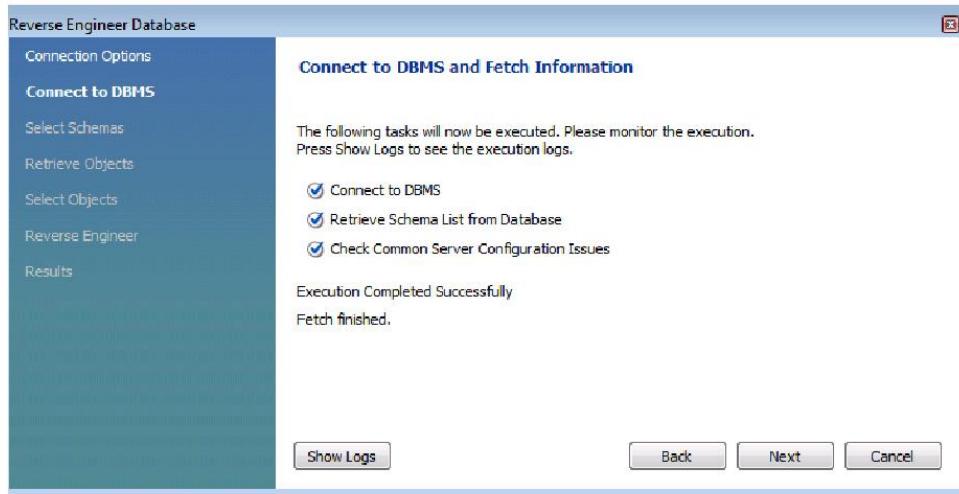
Es una herramienta que posee MySQL Workbench, que permite crear un modelo de base a una base de datos ya existente. Para aprovechar al máximo esta herramienta, deben estar establecidas previamente las claves foráneas. Para iniciar la herramienta, se hace click en la opción "Reverse Engineer":



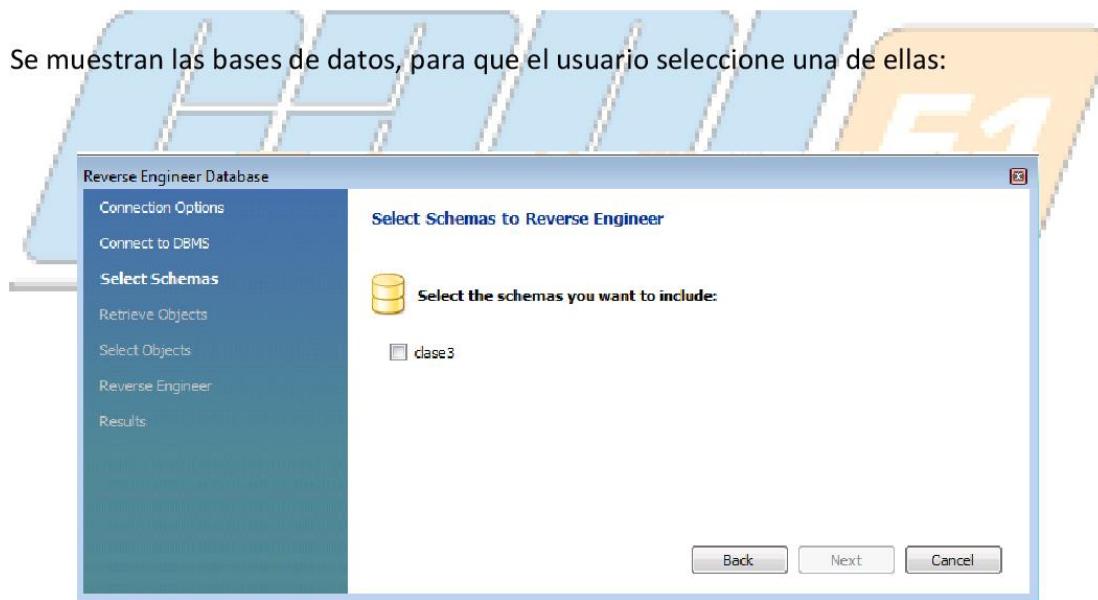
El primer paso que se debe realizar es conectarse al servidor donde esta la base de datos que se quiere usar como punto de partida para hacer la ingeniería inversa:



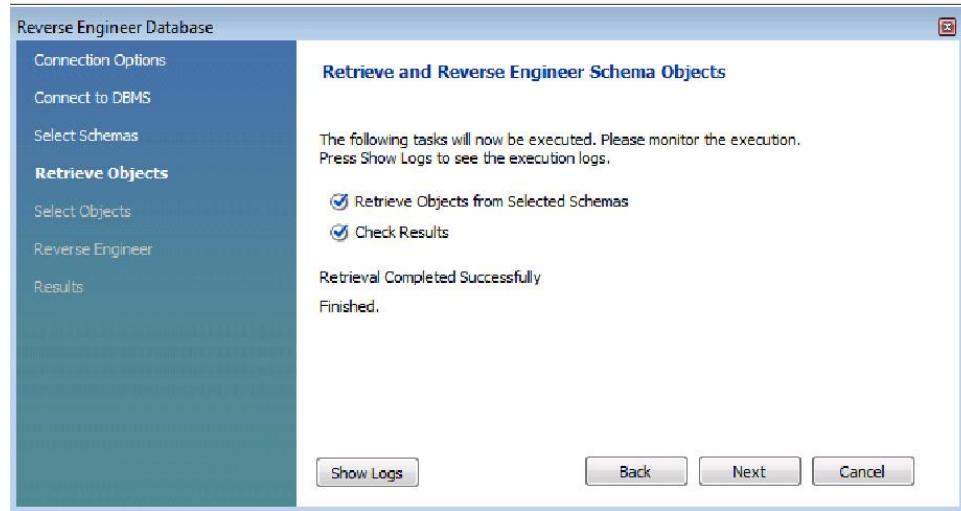
MySQL Workbench se conectará con el servidor para leer las bases de datos que están almacenadas en este:



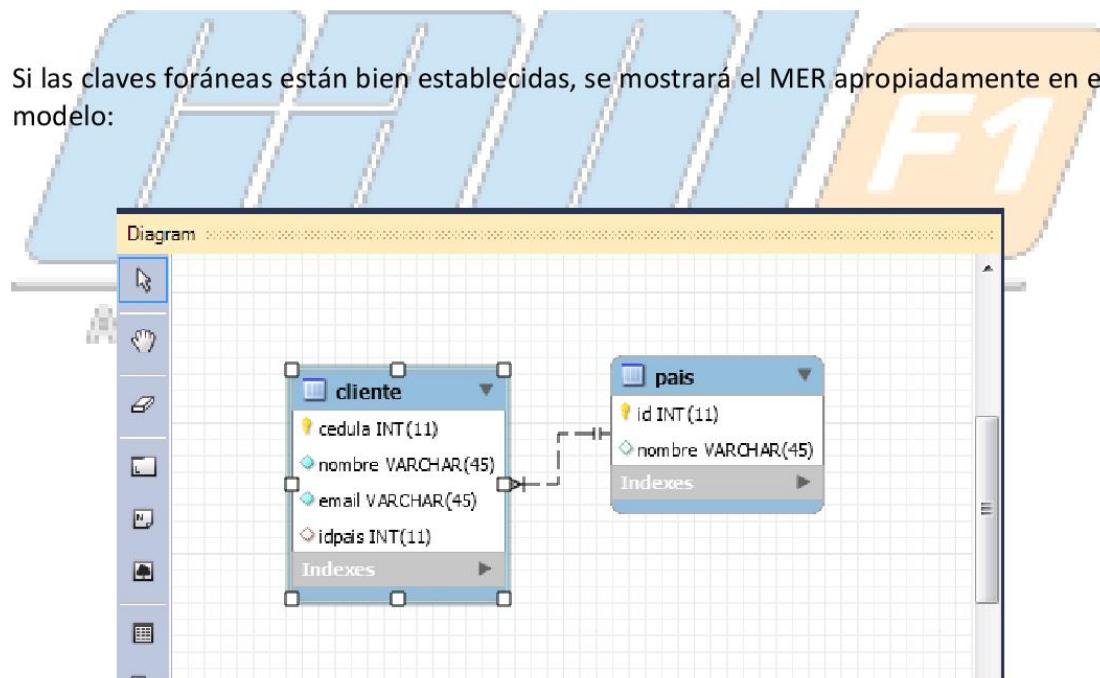
Se muestran las bases de datos, para que el usuario seleccione una de ellas:



Lee el contenido de cada tabla de la base de datos seleccionada y a partir de esa información creará el modelo:



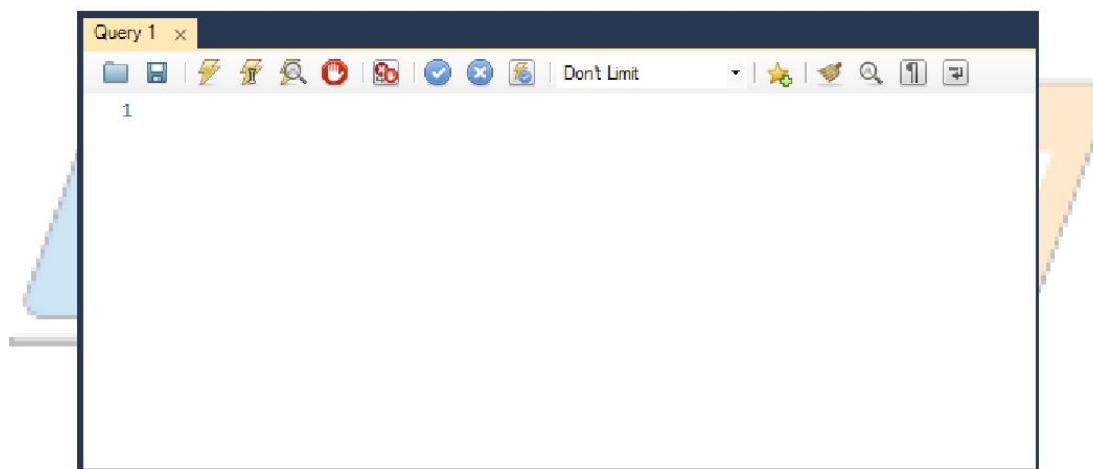
Si las claves foráneas están bien establecidas, se mostrará el MER apropiadamente en el modelo:



## Capítulo 11. TRABAJANDO CON INSTRUCCIONES SQL

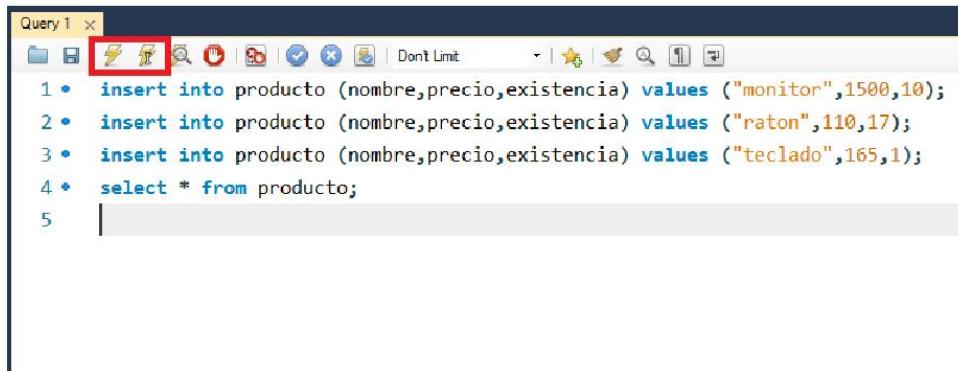
### 11.1.- Ejecutar Instrucciones Sql

En MySQL Workbench el uso de SQL está implícito en casi todas las operaciones. Cada vez que se desean aplicar los cambios en una base de datos, se muestra la ventana con las instrucciones SQL que están a punto de enviarse al servidor. Sin embargo, MySQL Workbench permite escribir instrucciones SQL personalizadas. Cuando se abre una nueva conexión a un servidor, por defecto MySQL Workbench abre una ventana nueva para escribir instrucciones SQL:



Para ejecutar las instrucciones, se deben usar los íconos que tienen un rayo. Como existe la posibilidad de escribir varias instrucciones SQL separadas con un punto y coma (;), existen 2 opciones:

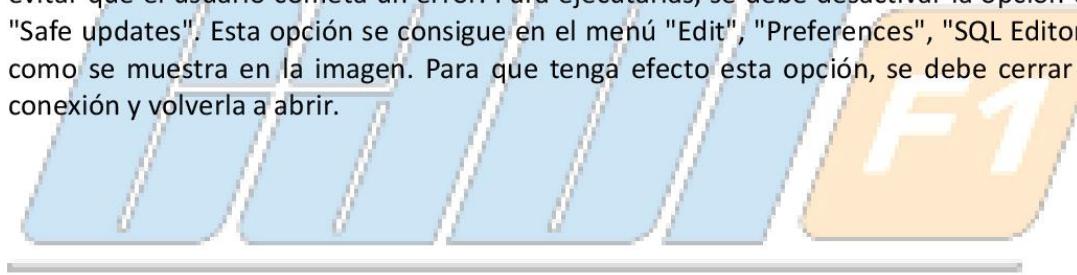
- ejecutar la porción seleccionada o todo si no hay texto seleccionado (la opción más usada).
- ejecutar la instrucción donde esta el cursor.



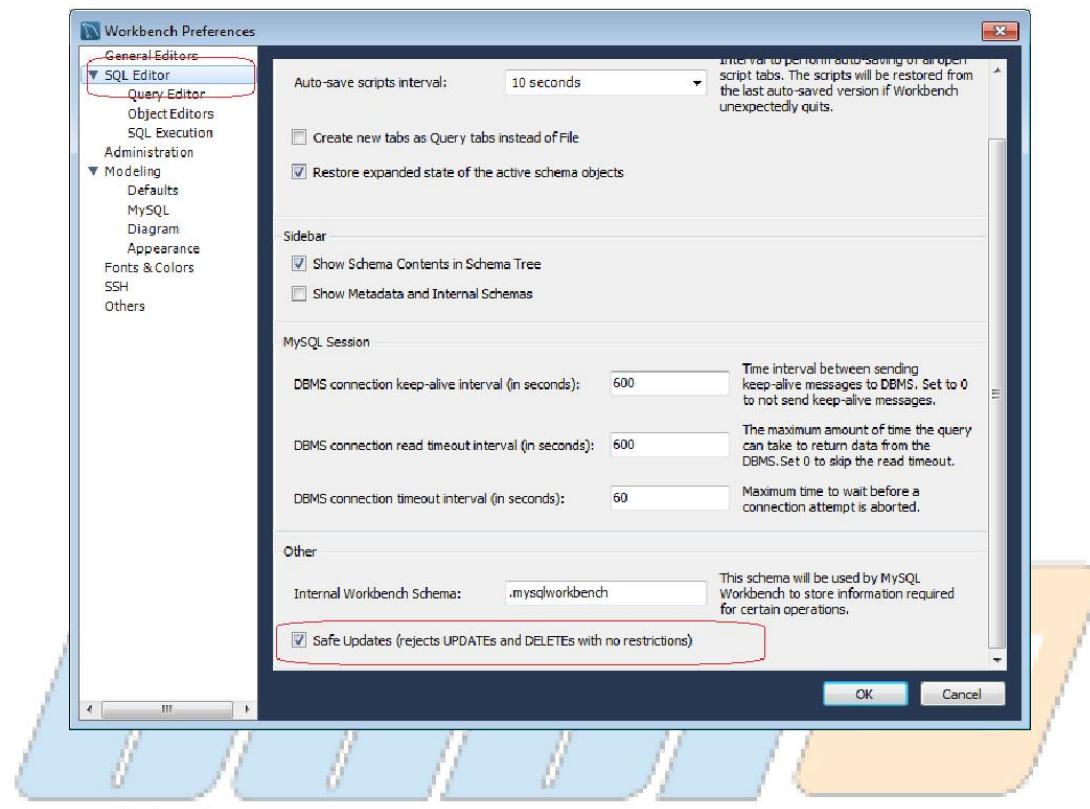
The screenshot shows the MySQL Workbench Query Editor window titled "Query 1". The toolbar at the top has several icons, with the first two (red box) being the "Run" and "Stop" buttons. Below the toolbar, the SQL code is displayed:

```
1 • insert into producto (nombre,precio,existencia) values ("monitor",1500,10);
2 • insert into producto (nombre,precio,existencia) values ("raton",110,17);
3 • insert into producto (nombre,precio,existencia) values ("teclado",165,1);
4 • select * from producto;
5 |
```

Cuando se ejecutan instrucciones SQL de tipo "delete" o "update" sin condición (sin where), MySQL Workbench no permite ejecutar las instrucciones por seguridad para que evitar que el usuario cometa un error. Para ejecutarlas, se debe desactivar la opción de "Safe updates". Esta opción se consigue en el menú "Edit", "Preferences", "SQL Editor", como se muestra en la imagen. Para que tenga efecto esta opción, se debe cerrar la conexión y volverla a abrir.



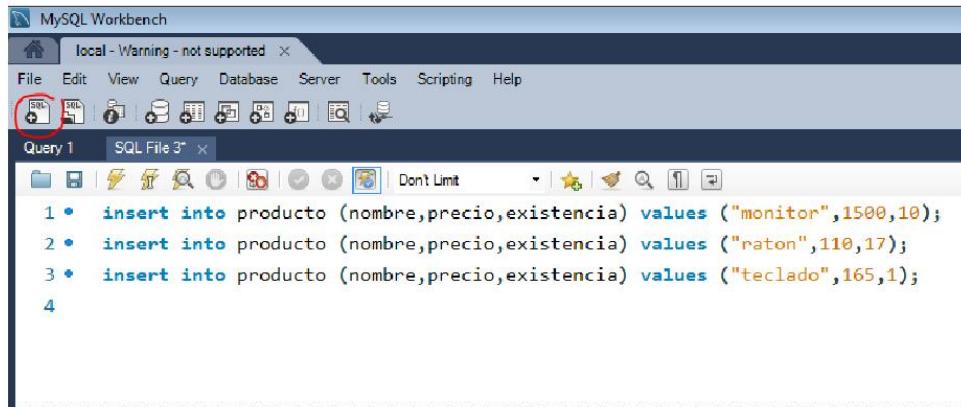
ACADEMIA DE SOFTWARE



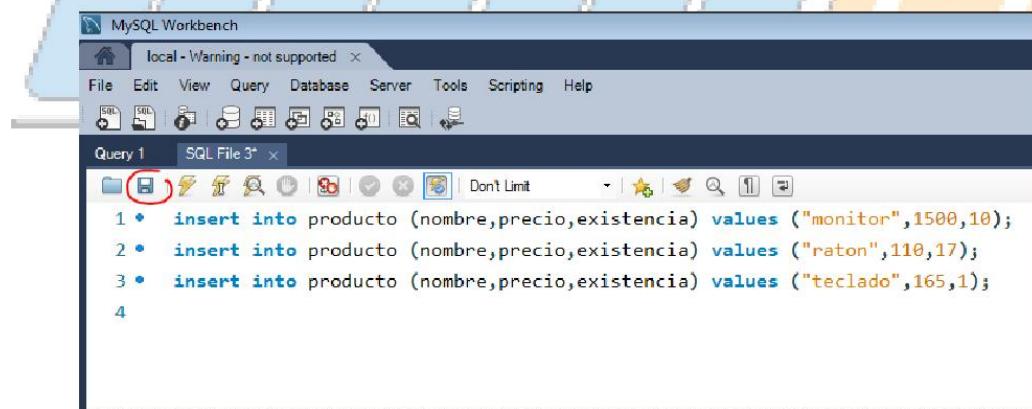
## 11.2.- Trabajando Con Scripts

# ACADEMIA DE SOFTWARE

MySQL Workbench por defecto abre una pestaña para escribir scripts, pero se pueden abrir tantas pestañas nuevas como hagan falta. Haciendo click en el botón "Create a new SQL Tab" se abrirá una nueva pestaña:



Al crear un script de SQL particular, estos scripts pueden ser guardados para posteriores ejecuciones. También se puede abrir un script y ejecutarlo. Haciendo click en el botón "Save Script", se abrirá una ventana de dialogo para indicar el lugar donde se guardará el archivo y el nombre:



MySQL Workbench tiene una opción que permite embellecer el código SQL que se escribe en una pestaña. Esto se logra con el botón "Beautify / Reformat". En la siguiente imagen se muestra un código escrito original y el mismo código luego de aplicar la herramienta:

```

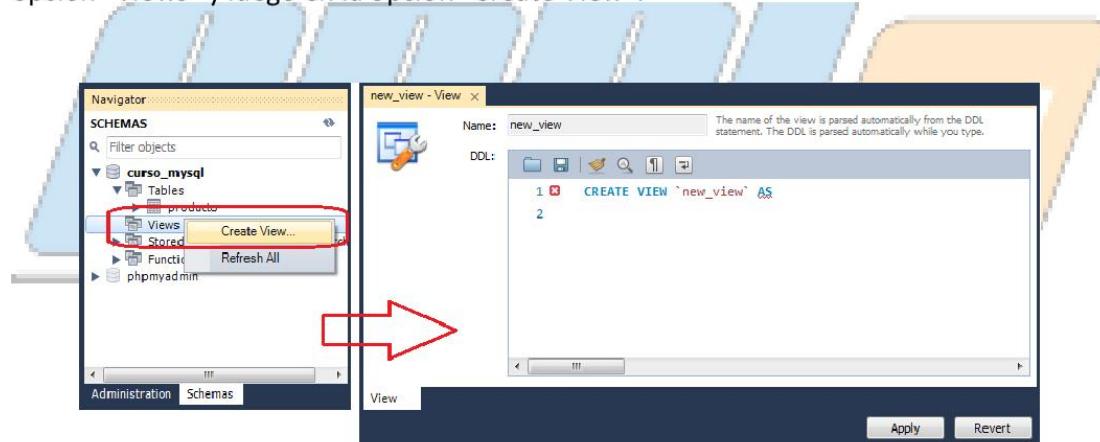
Query 1 x
1 • select nombre,precio from producto where id=50
2

Query 1 x
1 • SELECT
2         nombre, precio
3     FROM
4         producto
5     WHERE
6         id = 50

```

### 11.3.- Trabajando Con Vistas

Las vistas son instrucciones SQL "select" almacenadas en la base de datos, que pueden consultarse como si fuera una tabla más de la base de datos, aunque ésta no almacena nada propiamente, por el contrario, el servidor ejecuta la instrucción guardada y retorna el resultado. Para crear una nueva consulta se hace click con el botón derecho sobre la opción "Views" y luego en la opción "Create View":



Se establece un nombre a la vista y se escribe la instrucción SQL select que va a almacenar:

The screenshot shows the MySQL Workbench interface. A new view is being created with the name 'nuevos\_productos'. The DDL code is:

```
1 • CREATE VIEW `nuevos_productos` AS
2     select * from productos where id > 15
```

En cualquier momento, la vista puede ser ejecutada con una instrucción SELECT, usando el nombre de la vista:

The screenshot shows the MySQL Workbench interface. The 'nuevos\_productos' view is selected in the Navigator pane. A context menu is open over the view, with the 'Select Rows' option highlighted.

The SQL Editor contains the following query:

```
1 • SELECT * FROM curso_mysql.nuevos_productos;
```

The Result Grid displays the following data:

id	nombre	precio	existencia
16	Monitor	150.00	50
17	Teclado	50.00	100
18	Raton	20.00	150
19	Unidad de Estado Sólido	800.00	30
20	Impresora	300.00	40
21	Escáner	250.00	20
22	Lápiz	10.00	1000
23	Bolígrafo	5.00	500
24	Portafolios	10.00	100
25	Alfileres	1.00	1000
26	Cartulinas	5.00	500
27	Cola	2.00	1000
28	Tijeras	3.00	1000
29	Regla	1.00	1000
30	Alfileras	1.00	1000
31	Portapapeles	2.00	1000
32	Portafotos	1.00	1000
33	Alfileras	1.00	1000
34	Alfileras	1.00	1000
35	Alfileras	1.00	1000
36	Alfileras	1.00	1000
37	Alfileras	1.00	1000
38	Alfileras	1.00	1000
39	Alfileras	1.00	1000
40	Alfileras	1.00	1000
41	Alfileras	1.00	1000
42	Alfileras	1.00	1000
43	Alfileras	1.00	1000
44	Alfileras	1.00	1000
45	Alfileras	1.00	1000
46	Alfileras	1.00	1000
47	Alfileras	1.00	1000
48	Alfileras	1.00	1000
49	Alfileras	1.00	1000
50	Alfileras	1.00	1000
51	Alfileras	1.00	1000
52	Alfileras	1.00	1000
53	Alfileras	1.00	1000
54	Alfileras	1.00	1000
55	Alfileras	1.00	1000
56	Alfileras	1.00	1000
57	Alfileras	1.00	1000
58	Alfileras	1.00	1000
59	Alfileras	1.00	1000
60	Alfileras	1.00	1000
61	Alfileras	1.00	1000
62	Alfileras	1.00	1000
63	Alfileras	1.00	1000
64	Alfileras	1.00	1000
65	Alfileras	1.00	1000
66	Alfileras	1.00	1000
67	Alfileras	1.00	1000
68	Alfileras	1.00	1000
69	Alfileras	1.00	1000
70	Alfileras	1.00	1000
71	Alfileras	1.00	1000
72	Alfileras	1.00	1000
73	Alfileras	1.00	1000
74	Alfileras	1.00	1000
75	Alfileras	1.00	1000
76	Alfileras	1.00	1000
77	Alfileras	1.00	1000
78	Alfileras	1.00	1000
79	Alfileras	1.00	1000
80	Alfileras	1.00	1000
81	Alfileras	1.00	1000
82	Alfileras	1.00	1000
83	Alfileras	1.00	1000
84	Alfileras	1.00	1000
85	Alfileras	1.00	1000
86	Alfileras	1.00	1000
87	Alfileras	1.00	1000
88	Alfileras	1.00	1000
89	Alfileras	1.00	1000
90	Alfileras	1.00	1000
91	Alfileras	1.00	1000
92	Alfileras	1.00	1000
93	Alfileras	1.00	1000
94	Alfileras	1.00	1000
95	Alfileras	1.00	1000
96	Alfileras	1.00	1000
97	Alfileras	1.00	1000
98	Alfileras	1.00	1000
99	Alfileras	1.00	1000
100	Alfileras	1.00	1000

## Capítulo 12. FUNCIONES DE MYSQL

### 12.1.- Funciones y Operadores

MySQL permite usar en las instrucciones SQL todos los operadores aritméticos +,-,\*,/ , div y mod, además de los operadores SQL del estándar: like, between, in, is null, entre otros.

Para tener una referencia completa de todos los operadores de MySQL, consultar la siguiente URL:

<https://dev.mysql.com/doc/refman/8.0/en/non-typed-operators.html>

Adicionalmente, MySQL posee una gran cantidad de funciones que pueden ser usadas en las instrucciones SQL. Tiene funciones para muchos fines, entre ellas:

- funciones de cadenas
- funciones de fechas y horas
- funciones de números
- funciones de JSON
- funciones de conversiones de tipos
- funciones de bits
- funciones de encriptamiento
- funciones de control de flujo
- funciones de XML



A continuación, se estudiaran algunas de las funciones más usadas.

### 12.2.- Funciones de Cadena

Las funciones de cadena permiten manipular los valores varchar o char almacenados en los campos de una tabla, ya sea para modificarlos o para mostrarlos en el resultado de una consulta. Las funciones más usadas son:

- ASCII() retorna el valor numérico del carácter que está más a la izquierda.
- BIN() retorna la representación en string de un carácter en binary.
- CHAR() retorna el carácter correspondiente a un número.
- CONCAT() retorna las cadenas concatenadas.

- CONCAT\_WS() retorna las cadenas concatenadas separadas por un separador.
- FORMAT() retorna un número formateado con una cantidad de decimales.
- INSERT() inserta un string en otro string.
- INSTR() retorna el indice de la primera ocurrencia de un string en otro.
- LCASE() sinónimo de LOWER().
- LEFT() retorna un número de caracteres que estan más a la izquierda de una cadena.
- LENGTH() retorna la longitud de un string en bytes.
- LOCATE() retorna la posición de la primera ocurrencia de una cadena.
- LOWER() retorna una cadena en minúsculas.
- LTRIM() remueve los espacios en blanco al inicio de una cadena.

Otras funciones de cadena muy usadas son:

- MID() retorna un string iniciando desde una posición especifica.
- REPEAT() repite un string un número de veces.
- REPLACE() reemplaza todas las ocurrencias de un string por otro.
- REVERSE() revierte los caracteres de un string.
- RIGHT() retorna un número de caracteres que esten a la derecha de una cadena.
- RPAD() agregar un string un número específico de veces.
- RTRIM() remueve los espacios a la derecha de string.
- SPACE() retorna un string con un número específico de espacios.
- STRCMP() compara 2 strings.
- SUBSTR() retornar un sub string contenido en otro string.
- TRIM() remueve los espacios en blanco al comienzo y al final de una cadena.
- UCASE() sinónimo de UPPER().
- UPPER() convierte una cadena a mayúsculas.

### 12.3.- Funciones de Fecha

Las funciones de fecha permiten manipular valores de los tipos: date, time, datetime y timestamp. Son muchas las utilidades que se pueden conseguir, tales como: restar/sumar fechas, determinar fecha y hora actual, darle formato, entre otras. Las funciones son:

- ADDDATE() agrega tiempo a valor date.
- ADDTIME() agrega hora a un valor time.
- CONVERT\_TZ() convierte una hora de una zona horaria a otra.
- CURDATE() retorna la fecha actual del servidor.

- CURRENT_DATE()	sinónimo para CURDATE().
- CURRENT_TIME()	sinónimo para CURTIME().
- CURRENT_TIMESTAMP()	sinónimo para NOW().
- CURTIME()	retorna la hora actual.
- DATE()	extrae el valor date de un valor datetime.
- DATE_ADD()	suma valores time a un valor date
- DATE_FORMAT()	formatea la fecha con un formato específico.
- DATE_SUB()	resta un valor time a un date.
- DATEDIFF()	resta 2 fechas (valores date).
- DAY()	sinónimo para DAYOFMONTH().
- DAYNAME()	retorna el nombre del día de la semana.
- DAYOFMONTH()	retorna el día del mes (0-31).

Otras funciones de fecha y hora son:

- DAYOFWEEK()	retorna el índice del día de la semana.
- DAYOFYEAR()	retorna el día del año (1-366).
- EXTRACT()	extrae una parte de un date.
- FROM_DAYS()	convierte un número de días a un date.
- FROM_UNIXTIME()	formatea un valor Unix timestamp a date.
- GET_FORMAT()	retorna una fecha en formato string.
- HOUR()	extrae la hora de un valor datetime.
- LAST_DAY	retorna el último día del mes de un mes dado.
- MAKEDATE()	crea un date dado un día de año y el año.
- MAKETIME()	crea un valor time dado el valor de hora, minutos y segundos.
- MICROSECOND()	retorna los milisegundos de un valor datetime.
- MINUTE()	retorna los minutos de un valor datetime.
- MONTH()	retorna el mes de un valor date o datetime.
- MONTHNAME()	retorna el nombre del mes dado un valor date.
- NOW()	retorna la fecha y hora actual.
- PERIOD_ADD()	agrega un periodo a un año-mes.
- PERIOD_DIFF()	retorna el número de meses entre 2 períodos.
- STR_TO_DATE()	convierte un string a date.

Otra funciones adicionales son:

- SECOND()	retorna los segundos de un valor time o datetime (0-59).
------------	--

- SUBTIME()	resta 2 valores tipos time.
- TIME()	retorna la porción de hora de un datetime.
- TIME_FORMAT()	formatea un valor time.
- TIME_TO_SEC()	retorna un time convertido a segundos.
- TIMEDIFF()	resta 2 valores tipo time.
- TIMESTAMPADD()	agrega un intervalo datetime a una expresión.
- TIMESTAMPDIFF()	le resta un intervalo a un valor datetime.
- TO_DAYS()	convierte un valor date a dias.
- TO_SECONDS()	convierte un valor data a segundos (desde el año 0).
- UNIX_TIMESTAMP()	retorna un valor Unix timestamp.
- UTC_DATE()	retorna la fecha actual UTC.
- UTC_TIME()	retorna la hora actual UTC.
- UTC_TIMESTAMP()	retorna la fecha hora actual UTC.
- WEEK()	retorna el número de la semana de una fecha.
- WEEKOFYEAR()	retorna la semana calendario de una fecha.
- YEAR()	retorna el año de un valor date.
- YEARWEEK()	retorna el año de una fecha.

#### 12.4.- Funciones de Números

Entre las funciones de números que dispone MySQL estan:

- ABS()	retorna el valor absoluto.
- ACOS()	retorna el arc coseno.
- ASIN()	retorna el arc seno.
- ATAN()	retorna el arc tangente.
- ATAN2()	retorna el arc tangente de 2 argumentos.
- CEIL()	retorna el smallest integer value not less than the argument
- CONV()	Convert numbers between different number bases
- COS()	retorna el coseno.
- COT()	retorna la cotangente.
- DEGREES()	convierte randianes a grados.
- EXP()	eleva un valor a un exponente.
- FLOOR()	retorna el valor del entero más alto no mayor al argumento.

Otras funciones de números son:

- LN() retorna el logaritmo natural.
- LOG() retorna el logaritmo natural.
- LOG10() retorna el logaritmo base-10.
- LOG2() retorna el logaritmo base-2.
- MOD() retorna el resto de la división.
- PI() retorna el valor de pi.
- POW() retorna el argumento elevado a una potencia específica.
- RADIANS() retorna el argumento convertido en radianes.
- RAND() retornar un valor aleatorio.
- ROUND() redondea un argumento.
- SIGN() retorna el signo de un argumento.
- SIN() retorna el seno de un argumento.
- SQRT() retorna la raíz cuadrada de un argumento.
- TAN() retorna la tangente de un argumento.
- TRUNCATE() trunca una cantidad de decimales a un número.



## Capítulo 13. STORED PROCEDURES. PARTE 1

### 13.1.- Crear Store Procedures

A partir de MySQL 5 se agregó soporte para almacenar rutinas en el servidor (procedures and functions). Una rutina almacenada es un conjunto de instrucciones SQL que se guardan en el servidor bajo un nombre. Una vez ahí, un cliente puede hacer referencia al nombre de la rutina y así ejecutar todas las instrucciones que esta contiene.

Las rutinas almacenadas pueden ser muy útiles en situaciones como:

- Múltiples clientes escritos en diferentes lenguajes o diferentes plataformas, que hacen operaciones sobre la misma base de datos.
- Para evitar que ciertos usuarios accedan a ciertas tablas y que sólo puedan acceder al procedimiento almacenado que las manipula.

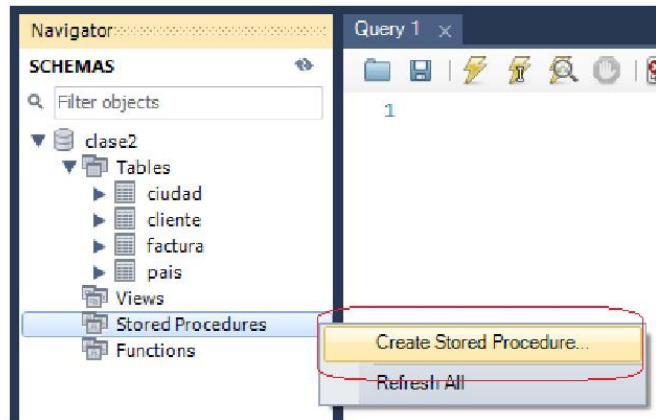
Una ventaja de las rutinas almacenadas es que mejoran el desempeño de las aplicaciones porque se reduce la cantidad de datos que deben enviarse entre cliente y servidor. Lo negativo es que la carga del servidor puede aumentar porque es mayor el trabajo que realiza.

La estructura general de la instrucción SQL para crear una rutina almacenada es la siguiente:

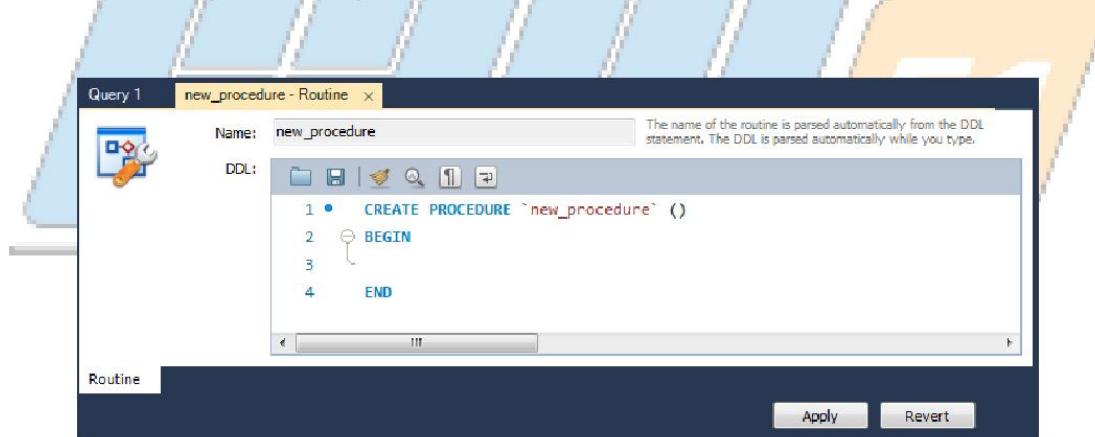
ACADEMIA DE SOFTWARE

```
1 CREATE
2     [DEFINER = user]
3     PROCEDURE sp_name ([proc_parameter[,...]])
4     [characteristic ...] routine_body
5
6 CREATE
7     [DEFINER = user]
8     FUNCTION sp_name ([func_parameter[,...]])
9     RETURNS type
10    [characteristic ...] routine_body
11
12 proc_parameter:
13     [ IN | OUT | INOUT ] param_name type
14
15 func_parameter:
16     param_name type
17
18 type:
19     Any valid MySQL data type
20
21 characteristic:
22     COMMENT 'string'
23     | LANGUAGE SQL
24     | [NOT] DETERMINISTIC
25     | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
26     | SQL SECURITY { DEFINER | INVOKER }
27
28 routine_body:
29     Valid SQL routine statement
```

Para crear un procedimiento almacenado en MySQL Workbench, se hace click con el botón derecho en el navegador de bases de datos, sobre la opción "Stored Procedures" adentro de la base de datos en la cual se desea guardar el procedimiento y seleccionar la opción "Create Stored Procedure..":



Se muestra una ventana para escribir el código que contendrá el procedimiento, como se muestra en la siguiente imagen:



El nombre del procedimiento se modifica directamente en el editor de código. Al cambiarlo, se modifica el nombre en la pestaña y en la propiedad name, como se muestra en el siguiente ejemplo, en el cual se le coloca al procedimiento el nombre "actualizar\_precios":

actualizar\_precios - Routine x

Name: actualizar\_precios

DDL:

```

1 • CREATE PROCEDURE `actualizar_precios`()
2   BEGIN
3   END

```

Apply    Revert

Un procedimiento almacenado puede contener cualquier instrucción SQL, tanto DML como DDL. También puede tener comentarios, variables, instrucciones de control, entre otras. En el siguiente ejemplo, un procedimiento utiliza una instrucción DML para modificar el valor del campo "precio" de todos los registros de la tabla "producto":

Name: actualizar\_precios

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```

1 • CREATE PROCEDURE `actualizar_precios`()
2   BEGIN
3     # AQUI VA EL CUERPO DEL PROCEDIMIENTO
4     # SE AUMENTAN LOS PRECIOS EN UN 10%
5     update producto set precio=precio*1.1;
6   END

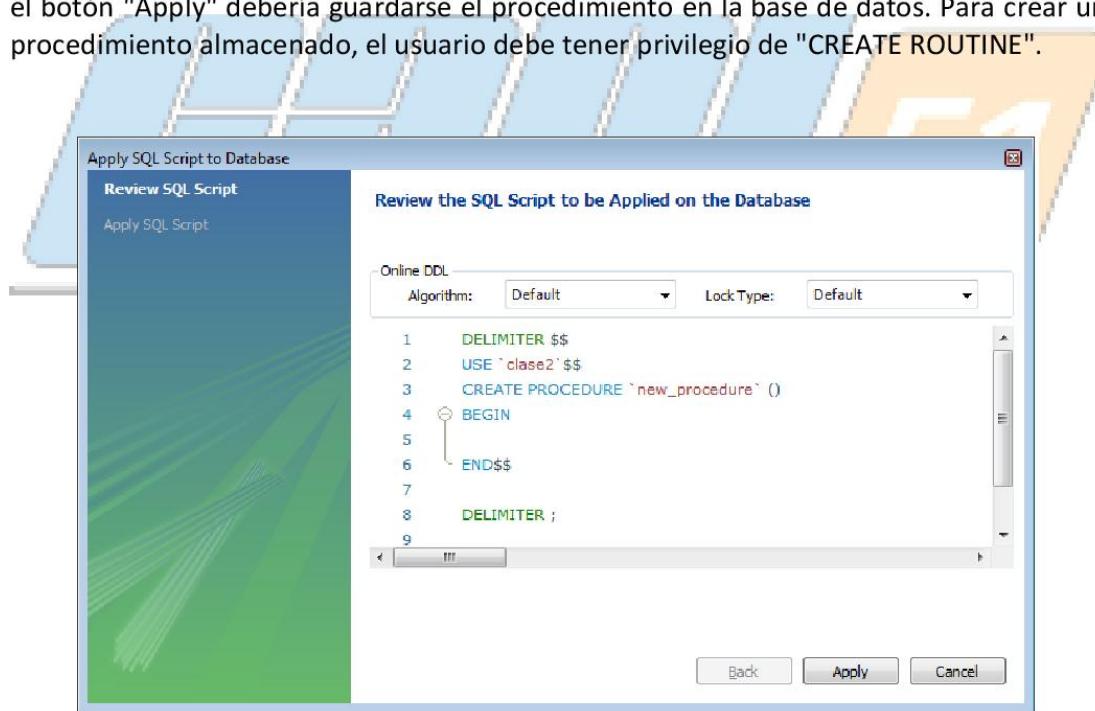
```

Apply    Revert

El siguiente es un ejemplo de un procedimiento que elimina los registros de las tablas "cliente" y "ciudad", para luego establecer el auto increment de cada tabla en 1:

```
1 • CREATE PROCEDURE `eliminar_registros` ()
2   BEGIN
3     # se eliminan los registros de clientes
4     delete from cliente;
5     alter table cliente auto_increment 1;
6     # se eliminan los registros de ciudad
7     delete from ciudad;
8     alter table ciudad auto_increment 1;
9   END
```

Al hacer click en el botón "Apply", se muestra la ventana con la instrucción SQL que está apunto de ejecutarse en el servidor. Si todo está en orden, al hacer click nuevamente en el botón "Apply" debería guardarse el procedimiento en la base de datos. Para crear un procedimiento almacenado, el usuario debe tener privilegio de "CREATE ROUTINE".

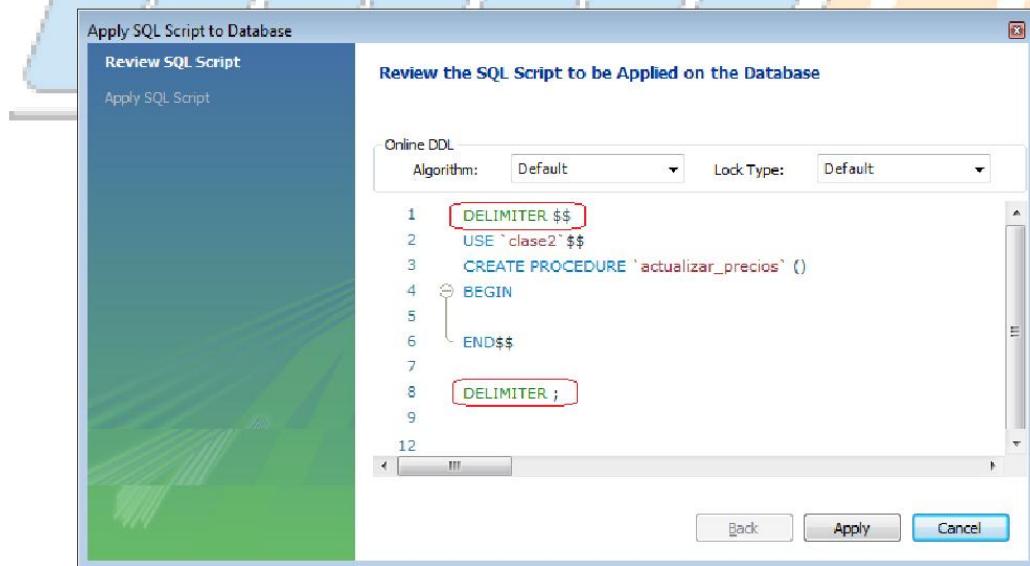


### 13.2.- El Delimitador

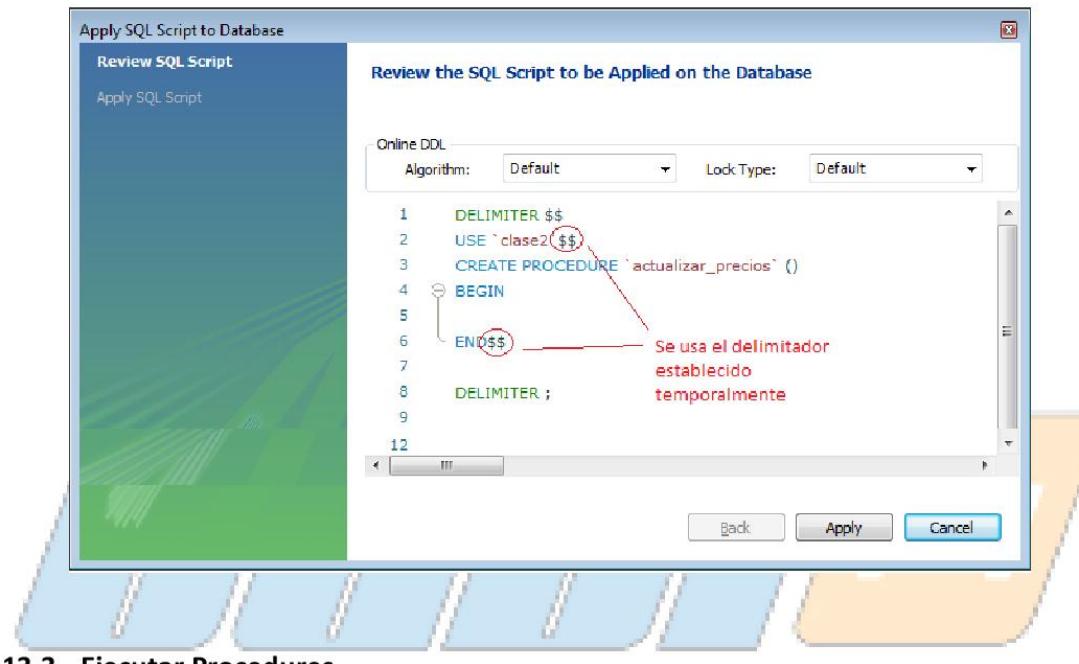
Un procedimiento almacenado contiene un cuerpo que consiste en varias instrucciones separadas por el carácter punto y coma (;). El problema, es que MySQL por defecto asume el punto y coma como separador de las instrucciones SQL al momento de ejecutar varias instrucciones a la vez, lo que genera un problema al momento de crear procedimientos almacenados.

La solución es cambiar momentáneamente el delimitador o separador de instrucciones por otro, para que MySQL no intente ejecutar inmediatamente el procedimiento sino guardararlo en la base de datos. Esto se logra con la instrucción "delimiter" seguido del carácter o conjunto de caracteres que se establecerá como delimitador.

Por ejemplo, el delimitador puede ser " \$\$ " (sin las comillas), como lo hace MySQL Workbench por defecto al crear procedimientos almacenados. En la siguiente imagen, se resalta como debe colocarse antes del procedimiento el cambio a \$\$ y luego del procedimiento se vuelve a establecer el punto y coma:



Luego de ejecutar el comando delimiter antes del procedimiento, se debe usar el nuevo delimitador fuera del procedimiento almacenado:



### 13.3.- Ejecutar Procedures

## ACADEMIA DE SOFTWARE

Para ejecutar un procedimiento almacenado se utiliza la palabra reservada "call" seguido del nombre del procedimiento almacenado. El usuario que intente ejecutar un procedimiento debe tener privilegios de "EXECUTE". Al ejecutar un procedimiento no se obtiene ningún resultado visible, solo es evidenciable su ejecución al verificar los datos que éste afecte. Por ejemplo, al consultar la tabla "productos" de la base de datos que manipula el procedimiento "actualizar\_precios" se visualiza el valor del campo "precio":

Query 1    producto x    actualizar\_precios    producto    actualizar\_precios - Routine

```
1 •  SELECT * FROM clase2.producto;
```

Result Grid    Filter Rows:    Edit:    Export/Imports:    Wrap Cell Content:

	id	nombre	precio
1	1	Harina pan	10000
2	2	Pasta corta	9500
3	3	Queso blanco	17000
*	NULL	NULL	NULL

Al ejecutar el procedimiento se evidencia en el panel de salidas la cantidad de registros que fueron afectados. Cada vez que se hace click en el ícono del rayo se ejecuta el procedimiento:

actualizar\_precios x

```
1 •  call clase2.actualizar_precios();
```

Output

#	Time	Action	Message	Duration / Fetch
1	21:18:05	call clase2.actualizar_precios()	3 row(s) affected	0.000 sec

Luego de ejecutar el procedimiento y hacer una nueva consulta a la tabla, se visualizan los precios aumentados:

The screenshot shows a MySQL Workbench interface with a query editor window titled 'producto'. The query is:

```
1 •  SELECT * FROM clase2.producto;
```

The result grid displays the following data:

	id	nombre	precio
1	1	Harina pan	11000
2	2	Pasta corta	10450
3	3	Queso blanco	18700
*	NULL	NULL	NULL

Un procedimiento almacenado puede ejecutar a otro procedimiento almacenado, por ejemplo:

The screenshot shows a MySQL Workbench interface with a code editor window containing a stored procedure definition:

```
1 •  CREATE PROCEDURE `actualizar_precios`()
2   BEGIN
3     declare porc float default 0.1;
4     # AQUI VA EL CUERPO DEL PROCEDIMIENTO
5     # SE AUMENTAN LOS PRECIOS EN UN % ALEATORIO
6     set porc = rand();
7     update producto set precio=precio*(1+porc);
8     call eliminar_registros();
9   END
10
```

## Capítulo 14. STORED PROCEDURES. PARTE 2

### 14.1.- Trabajando Con Variables

La declaración de variables se hace utilizando la palabra reservada "declare", seguido del nombre de la variable, el tipo de dato (cualquiera de los tipos de datos permitidos por MySQL) y el valor por defecto (si se omite el valor por defecto se asume que es null). Por ejemplo:

```
1 • CREATE PROCEDURE `actualizar_precios`()
2   BEGIN
3     declare porc float default 0.1;
4     # AQUI VA EL CUERPO DEL PROCEDIMIENTO
5     # SE AUMENTAN LOS PRECIOS EN UN 10%
6     update producto set precio=precio*(1+porc);
7   END
```

Si se necesita que una variable tome un valor distinto al valor por defecto, ya sea asignando un valor literal o lo que retorna una función, se utiliza la palabra reservada "set", como se muestra en el siguiente ejemplo donde se le asigna a la variable "porc" el valor que retorna la función "rand" de MySQL:

```
1 • CREATE PROCEDURE `actualizar_precios`()
2   BEGIN
3     declare porc float default 0.1;
4     # AQUI VA EL CUERPO DEL PROCEDIMIENTO
5     # SE AUMENTAN LOS PRECIOS EN UN % ALEATORIO
6     set porc = rand();
7     update producto set precio=precio*(1+porc);
8   END
```

## 14.2.- Select Into

El resultado de una consulta SELECT puede almacenarse en una(s) variable(s), para luego utilizar el valor de las variables en otras operaciones adentro del procedimiento almacenado. Para esto se utiliza la instrucción "SELECT ... INTO". La estructura general de la instrucción es la siguiente:

```
"SELECT col1,col2,...,coln FROM ... INTO var1,var2,...,varn"
```

La cantidad de columnas en la instrucción SELECT debe coincidir con la cantidad de variables. La consulta debería retornar un único valor. Si la consulta no retorna filas, ocurre una advertencia con código 1329 (No data) y el valor de las variables no cambiará. Si la consulta retorna varias filas, ocurrirá un error 1172 (Result consisted of more than one row). Si es posible que la instrucción retorne más de una columna, se debe utilizar LIMIT 1 para limitar a una sola fila el resultado.

En el siguiente ejemplo se determina el precio promedio de los productos y se guarda el valor en la variable "promedio", para luego utilizarlo para actualizar los precios de otros productos:

```

1 • CREATE PROCEDURE `actualizar_precio_al_promedio` ()
2   BEGIN
3     declare promedio float;
4     select avg(precio) from producto into promedio;
5     update producto set precio=precio*1.1 where precio>promedio;
6   END

```

## 14.3.- Paso de Parámetros

Un procedimiento almacenado puede recibir parámetros desde el exterior. Los parámetros se colocan entre los paréntesis en la definición del procedimiento seguido del tipo de dato. Al llamar un procedimiento almacenado que tiene parámetros, se deben pasar los parámetros entre paréntesis (sin colocar el tipo de dato). Por ejemplo:

### Definición del procedimiento almacenado con un parámetro

```
CREATE PROCEDURE `actualizar_precio`(aumento float)
BEGIN
    # se hace un aumento utilizando el valor
    # recibido por parametro
    update producto set precio=precio*aumento;
END
```

### Uso del procedimiento almacenado pasando el parámetro

```
call actualizar_precio(0.15)
```

Los parámetros pueden ser de tres tipos: IN, OUT o INOUT. El tipo de parámetro se especifica antes de su nombre. Cada parámetro, por defecto, se asume de tipo IN a menos que se diga lo contrario. La explicación de para qué sirve cada tipo de parámetro es la siguiente:

- Un parámetro tipo IN se utiliza para enviarle un dato a un procedimiento. Si en el código del procedimiento se modifica el valor del parámetro, la modificación no será conocida cuando termine el llamado.
- Un parámetro An OUT pasa un valor de retorno desde su interior a quien lo llame. Su valor inicial adentro del procedimiento es NULL.
- Un parámetro INOUT se comporta igual que el parámetro OUT, con la diferencia que su valor puede ser inicializado por el llamador.

Los nombres de los parámetros no son sensibles a mayúsculas y minúsculas.

Un ejemplo de cómo se define un procedimiento con 2 parámetros, uno IN y otro OUT:

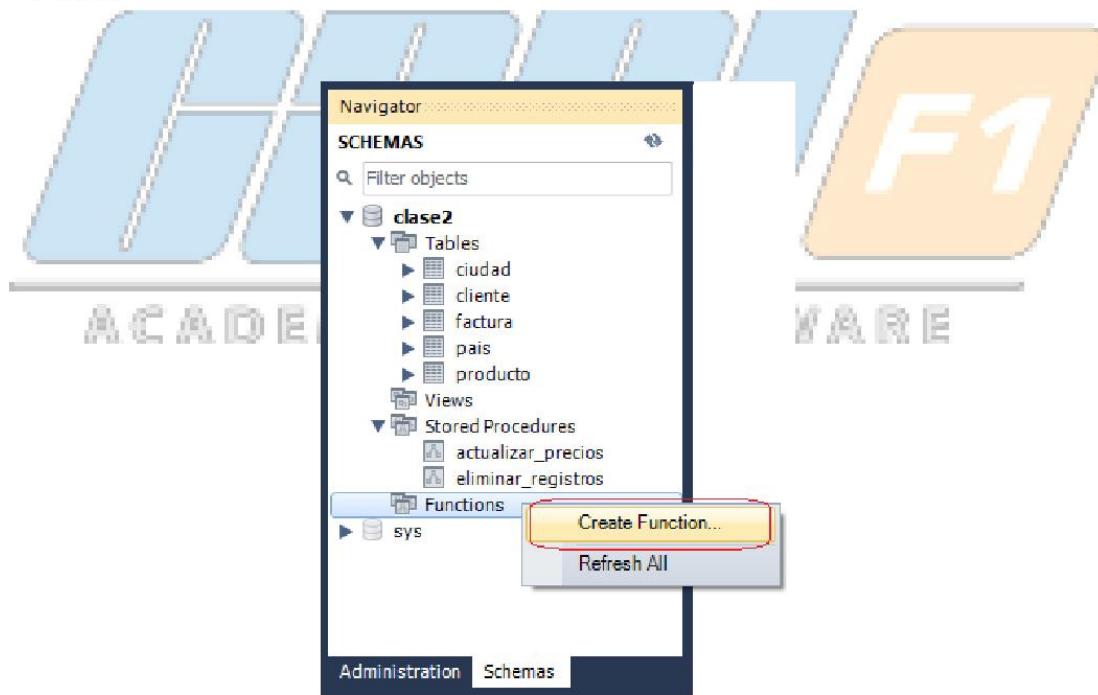
```

1 • CREATE PROCEDURE `determinar_maximo` (IN mes integer,OUT maximo float)
2   BEGIN
3     # se busca el maximo precio de los productos de cierto mes
4     select max(precio) from producto where month(fecha)=mes into maximo;
5   END

```

#### 14.4.- Funciones

Una función es un tipo de procedimiento almacenado que retorna un valor a quien la llame. Tiene la misma estructura de un procedimiento almacenado, con la diferencia de que se utiliza la palabra reservada "FUNCTION" al momento de definirla y se indica el tipo de dato del valor de retorno. Para crear una función nueva, se hace click con el botón derecho en el navegador en la opción "Functions" y luego se selecciona la opción "Create Function":



En el interior de la función, debe existir obligatoriamente una instrucción RETURN junto con el valor que retornará la función. El valor de retorno de la función debe coincidir con

el tipo de dato indicado en el encabezado de la función. Para una función sólo puede haber parámetros IN. En el siguiente ejemplo, se crea una función con el nombre "mayor\_precio", que retorna un valor "integer" y no tiene parámetros:

The screenshot shows the MySQL Workbench interface with a window titled "mayor\_precio - Routine". The "Name:" field contains "mayor\_precio". The "DDL:" pane displays the following SQL code:

```

1 • CREATE FUNCTION `mayor_precio` ()
2     RETURNS INTEGER
3     BEGIN
4
5         RETURN 1;
6     END

```

Below the DDL pane, there are "Routine" and "Script" tabs, and buttons for "Apply" and "Revert".

En el siguiente ejemplo, se crea una función con el nombre "calcular\_promedio", que retorna un valor "float" y que recibe un parámetro con el nombre "monto":

The screenshot shows the MySQL Workbench interface with a window titled "ACADEMIA DE SOFTWARE". The DDL pane displays the following SQL code:

```

1 • CREATE FUNCTION `calcular_promedio` (monto integer)
2     RETURNS float
3     BEGIN
4         declare maximo float;
5         select max(precio) from producto where precio>monto into maximo;
6         RETURN maximo;
7     END

```

El llamado de una función puede hacerse en cualquier instrucción SQL (select, update o delete) o puede llamarse desde otro procedimiento almacenado:

```
select calcular_promedio(4)
```

		Result Grid	Filter Rows:	Export:	Wrap Cell Content:
		calcular_promedio(4)			
▶		NULL			

