UNIVERSIDAD DE SONSONATE

USO

SCIENTIAE ET BONIS ARTIBUS

INGENIERÍA EN SISTEMAS COMPUTACIONALES

LABORATORIO DE BASES DE DATOS I

Guía 3

DDL (Data Definition Language)

INTRODUCCIÓN

Lenguaje SQL

El **Lenguaje Estructurado de Consultas**, o **SQL (Structured Query Language)**, es un lenguaje diseñado para almacenar, manipular y recuperar información almacenada en una base de datos relacional. El alcance de SQL incluye la inserción de datos, consultas, actualizaciones y borrado, la creación y modificación de esquemas y el control de acceso a los datos. Está constituido principalmente por el Lenguaje de Definición de Datos (DDL), el Lenguaje de Manipulación de Datos (DML) y Lenguaje de Control de Datos (DCL).

DDL

El **Lenguaje de Definición de Datos o DDL (Data Definition Language)** se utiliza para definir la estructura de la base de datos y los objetos que se encuentran dentro de ella, como tablas, vistas, índices, procedimientos almacenados, etc. DDL incluye comandos como CREATE, ALTER y DROP, que se utilizan para crear, modificar y eliminar objetos de la base de datos.

MySQL

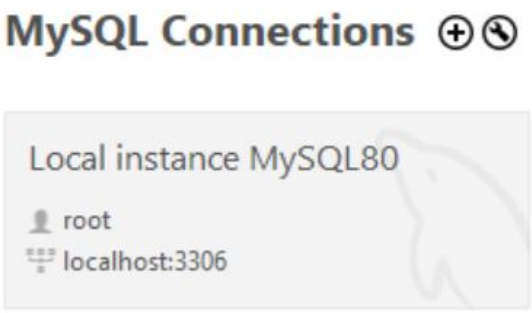
MySQL es un **sistema de administración de bases de datos relacionales o RDBMS (Relational DataBase Management System)**. Se trata de un programa capaz de almacenar una enorme cantidad de datos de gran variedad y de distribuirlos para cubrir las necesidades de cualquier tipo de organización, desde pequeños establecimientos comerciales a grandes empresas y organismos administrativos. MySQL compite con sistemas RDBMS conocidos, como Oracle y SQL Server.

Interfaz gráfica MySQL Workbench

MySQL Workbench es una herramienta visual y un entorno de desarrollo integrado (IDE) para trabajar con la base de datos de MySQL, con una interfaz que puede ser utilizada para manejar la base de datos MySQL de forma gráfica, tanto como la creación de bases de datos, tablas, campos, así como de proveer un diagramador, aunque de igual manera se puede trabajar completamente en archivos SQL con código a mano.

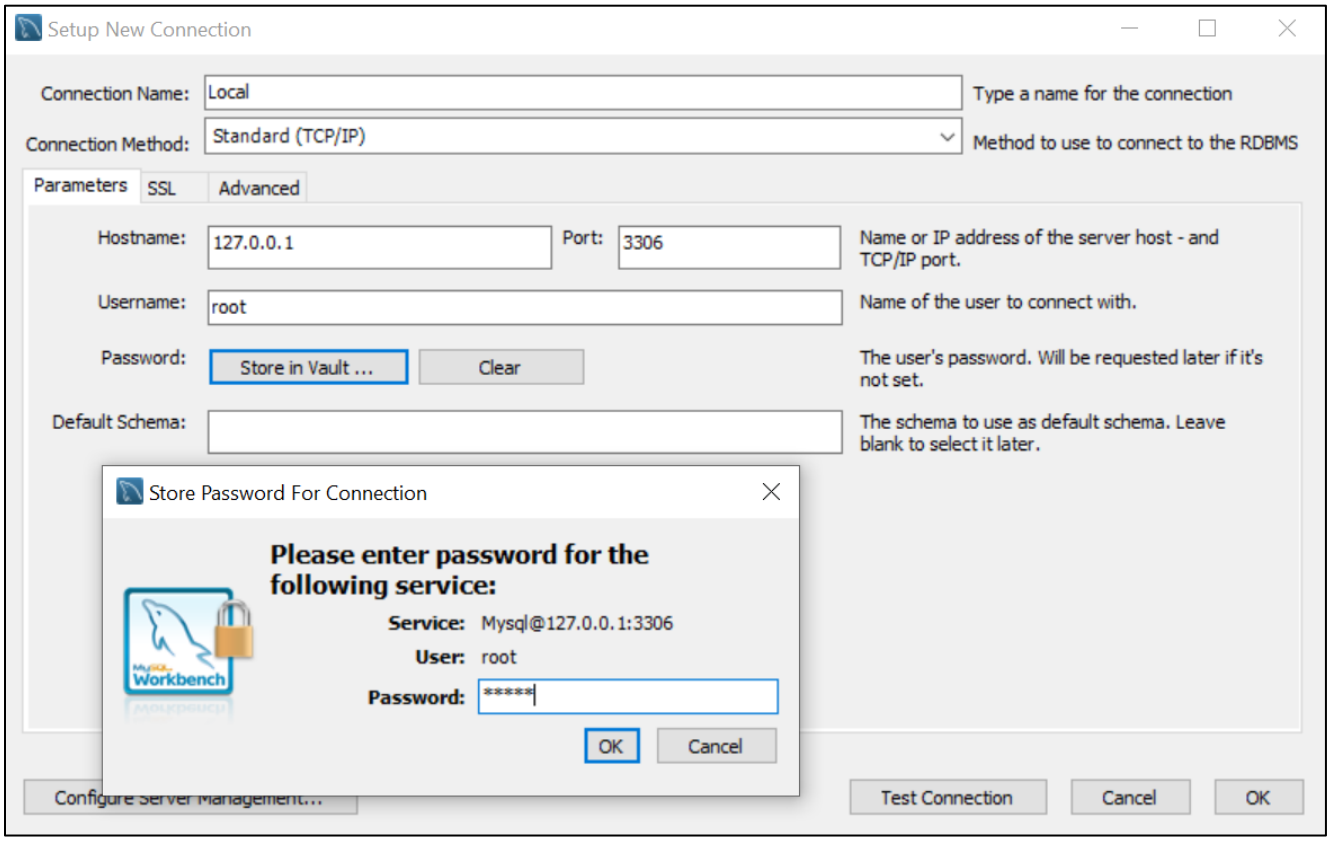
Paso 1:

Damos clic en el botón con el signo "+", para agregar una nueva conexión. La conexión por defecto también funciona.



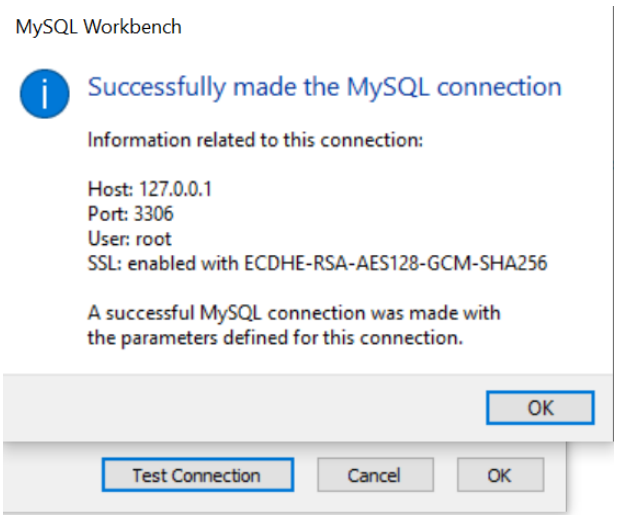
Paso 2:

Definimos un nombre para la conexión, el host, el puerto (por defecto 3306) y el usuario. Al dar clic en "Store in Vault…" nos saldrá una ventana pidiéndonos la contraseña para el usuario, la definimos y damos en "OK".



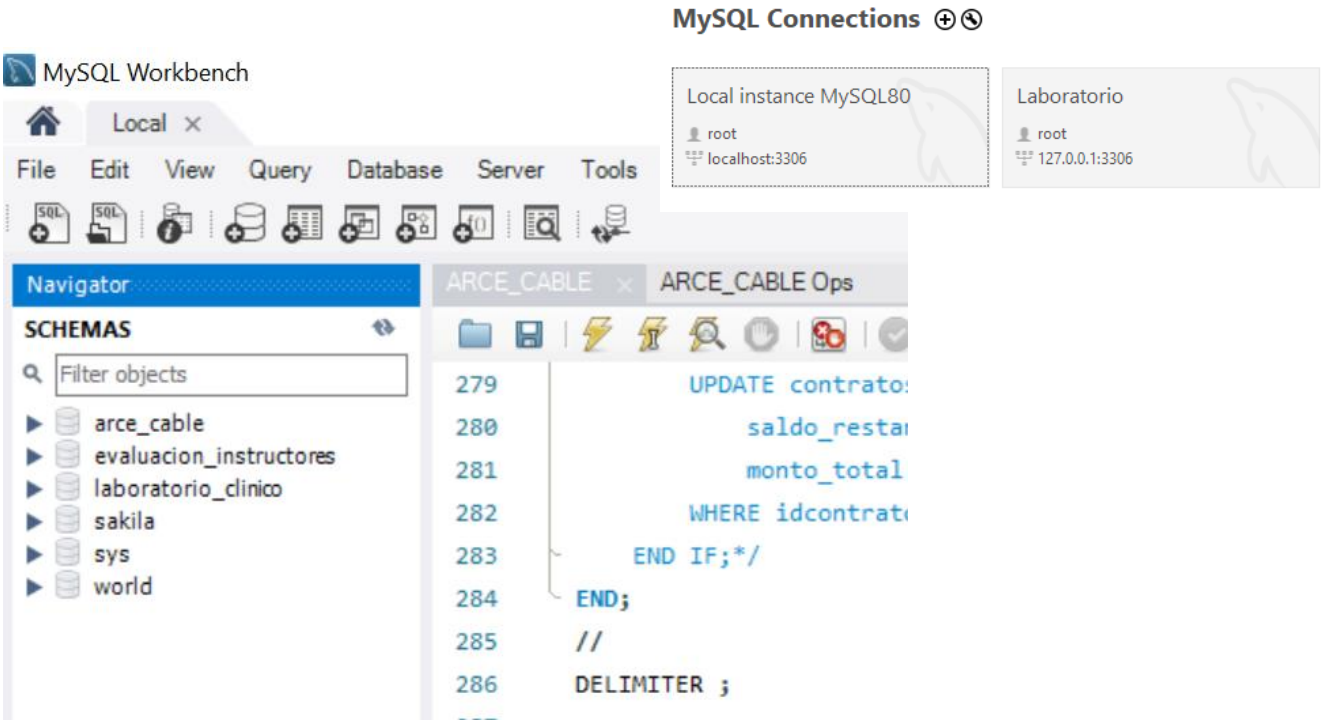
Paso 3:

Para verificar que los datos que ingresamos están correctos, damos clic en “Test Connection”, y nos saldrá un mensaje diciéndonos si la conexión fue exitosa o si no se pudo conectar. Damos clic en “OK”, y luego “OK” para crear la nueva conexión.



Paso 4:

Cuando la conexión esté creada, nos aparecerá en la lista de conexiones, le damos clic y ya estaremos conectados al servidor especificado, con todas las bases de datos MySQL que en él se encuentren y a las que tenga acceso el usuario con el que accedimos.



Comandos básicos DDL

- **CREATE DATABASE dbejemplo:** crea una base de datos con nombre "dbejemplo".
- **USE dbejemplo:** pone en uso la base de datos "dbejemplo".
- **CREATE TABLE tablaejemplo:** crea una tabla/entidad dentro de la base de datos en uso de nombre "tablaejemplo".
- **CREATE USER 'ejempo'@'localhost' IDENTIFIED BY 'admin':** crea un usuario de nombre "ejemplo", que sólo funcionará en el host "localhost" y que su contraseña será "admin".
- **SHOW DATABASES:** muestra todas las bases de datos.
- **SHOW TABLES:** muestra todas las tablas de la base de datos en uso.

Restricciones en DDL

- **PRIMARY KEY:** indica que el campo es llave primaria.
- **AUTO_INCREMENT:** indica que el campo se autoincrementará.
- **NOT NULL:** el campo debe llenarse obligadamente.
- **DEFAULT:** asigna un valor por defecto, en caso de que no se llene el campo.
- **UNIQUE:** el valor del campo debe ser único.
- **REFERENCES:** indica que el campo es llave foránea.
- **ON [DELETE/UPDATE] [CASCADE/SET NULL/RESTRICT/NO ACTION]:** permite definir la acción que se realizará al momento de borrar o modificar una tabla a la que se está haciendo referencia, o sea una tabla principal. Se puede definir si será en el DELETE o en el UPDATE, y se especifica si se modificarán o eliminarán los datos de la tabla secundaria al borrar la tabla primaria (CASCADE), si se quiere que se hagan nulos

(SET NULL), o si no se permitirá borrar o actualizar la tabla primaria debido a que está siendo referenciada (RESTRICT/NO ACTION).

Tipos de datos

- **TINYINT:** es un entero con o sin signo; los valores aceptados son desde 0 a 255 cuando no tiene signo y -127 a 255 cuando lo tiene.

- **BIT:** almacena valores que contienen un bit.

- **BOOL:** sólo permite cero o uno. En algunas versiones se toma como TINYINT.

- **SMALLINT:** representa un entero corto o pequeño; la capacidad de almacenamiento es de 65535.

- **MEDIUMINT:** representa un entero. La diferencia con el SMALLINT es la capacidad, ya que este tipo de campo abarca una mayor cantidad de caracteres para guardar, siendo de 8388607.

- **INTEGER o INT:** representa a un número entero; es más utilizado normalmente para hacer referencia a este tipo de datos; su capacidad de almacenamiento es de 4294967295.

- **BIGINT:** es utilizado cuando la capacidad del INT es poca para representar a un entero; su capacidad de almacenamiento es de 18446744073709551615.

- **DECIMAL:** permite números decimales dentro de sus valores. La sintaxis del decimal es DECIMAL(M,D), donde M tiene un rango de 1 a 64, y D tiene un rango de 0 a 30. Aunque en la declaración se utiliza la coma para separar el entero del decimal, cuando se trabaja con los números el punto es el separador del entero y decimal, como 100.50, por ejemplo.

- **FLOAT:** se utiliza normalmente para el trabajo con números decimales, ya que este permite una especificación opcional de la precisión; la escala de este tipo de datos va desde 0 a 23, que resulta en una precisión de 4 Bytes.

- **DOUBLE:** al igual que el FLOAT, trabaja con decimales. El alcance de este tipo es el doble del FLOAT. Este tipo de datos tiene una precisión de 8 Bytes, el doble de FLOAT y va desde el 24 a 53.

- **REAL:** al igual que DOUBLE, es tomado con dato de doble precisión, es decir de 8 Bytes.

- **DATE:** almacena las fechas. El formato que tiene MySQL para guardar este tipo de datos es año-mes-día (YYYY-MM-DD).

- **TIME:** almacena las horas. El formato es **hora:minutos:**segundos (00:00:00), normalmente guardado en hora militar; si se desea convertir para mostrarse en AM y PM se puede hacer uso de la función DATE_FORMAT().

- **DATETIME:** Como su nombre lo indica es una combinación de los dos tipo de datos anteriores, el formato es el siguiente **año-mes-día hora:minuto:segundo** (0000-00-00 00:00:00).

- **TIMESTAMP:** Se muestra en el mismo formato que los campos de tipo DATETIME, pero la zona horaria actual se convierte a UTC al guardar, y de UTC a la actual al consultar.

- **YEAR:** Como su nombre lo indica se utiliza para almacenar años, por defecto la cantidad de dígitos es de 4 aunque puede establecerse que se utilicen dos si se desea almacenar los años en este formato.

- **CHAR:** almacena una cadena de caracteres indicando al lado la cantidad que contendrá (CHAR(10) indicaría una cadena de 10 caracteres), la cantidad de caracteres que se puede almacenar en un campo de este tipo va desde 0 hasta 255.

- **VARCHAR:** al igual que CHAR, almacena cadenas de caracteres; la cantidad se especifica al lado (VARCHAR(10) indicaría una cadena de 10 caracteres), y se puede almacenar una cantidad de caracteres de 0 a 255.

- **ENUM:** almacena cadenas ya especificadas al momento de definir el campo. Al momento de insertar, se puede usar el texto o el índice de la opción elegida.

DESARROLLO

Ejemplo de creación de tabla:

```
-- Crea una tabla con sus campos, y el campo “id” como llave
primaria

CREATE TABLE usuario(
    id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(20)
);


-- Borra la tabla por completo

DROP TABLE usuario;

-- Borra los datos de la tabla, pero no la tabla

TRUNCATE TABLE usuario;

-- Añade una columna a la tabla

ALTER TABLE usuario ADD apellido VARCHAR(20);

-- Borra una columna de la tabla

ALTER TABLE usuario DROP COLUMN apellido;

-- Cambia el tipo de dato de una columna en la tabla

ALTER TABLE usuario MODIFY COLUMN id INT(12);
```

Ejemplo de creación de llave foránea

```
CREATE TABLE rol(
    idrol INT PRIMARY KEY,
    rol VARCHAR(20)
);

-- El campo “id” como llave primaria y el campo “idrol” como
llave foránea
-- Llave foránea agregada al momento de crear la tabla

CREATE TABLE usuario(
    id INT PRIMARY KEY AUTO_INCREMENT,
    idrol INT NOT NULL,
    nombre VARCHAR(20),
    FOREIGN KEY (idrol) REFERENCES rol(idrol)
);

-- Llave foránea agregada después de la creación de la tabla
-- (Para agregar la llave foránea, el campo de destino ya debe
existir)
CREATE TABLE usuario(
    id INT PRIMARY KEY AUTO_INCREMENT,
    idrol INT NOT NULL,
    nombre VARCHAR(20)
);

ALTER TABLE usuario ADD FOREIGN KEY (idrol) REFERENCES
rol(idrol);
```

[El archivo SQL con la creación de la base de datos escolar se encuentra en la carpeta Recursos]

EJERCICIOS

En base al diagrama creado en la Guía 2, realizar el respectivo DDL para la base de datos seleccionada, siguiendo los siguientes parámetros:

- Se realizará un archivo .sql por cada grupo.
- Cada estudiante debe tener una copia del archivo realizado en su repositorio remoto independiente (Guía 3)