



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



**TECNOLÓGICO
NACIONAL DE MÉXICO®**
INSTITUTO TECNOLÓGICO DE TIJUANA



INSTITUTO TECNOLÓGICO DE TIJUANA



TECNOLOGIAS DE BASE DE DATOS

Unidad 1
Sistemas Gestores de Base de Datos

Practica 1
Introducción a MySQL

PRESENTA:
Camacho Lopez Valeria

C22211475

DOCENTE:
Fortunato Ramírez Arzate

TIJUANA, B. C, A 14 DE SEPTIEMBRE DEL 2024

INTRODUCCIÓN

El objetivo de la primera practica es empezar a familiarizarse con los distintos comandos (query) básicos para interactuar en MySQL a través de la consola, comprendiendo desde la instalación y configuración hasta la creación y manipulación de bases de datos, principalmente tablas y datos (registros) para la realización de nuevos conceptos.

CONCEPTOS BASICOS DE MySQL

Una base de datos relacional es un sistema que organiza la información en tablas relacionadas entre sí. Cada tabla almacena datos en filas y columnas, donde las filas son registros y las columnas son atributos o campos.

SQL (Structured Query Language) es el lenguaje estándar utilizado para interactuar con estas bases de datos. Permite realizar consultas, insertar, actualizar y eliminar datos, además de administrar la estructura de las tablas.

Las tablas son esenciales porque organizan los datos de manera estructurada. Un registro es una fila dentro de una tabla que representa una entrada específica de datos.

La clave primaria es un identificador único para cada registro en una tabla, lo que garantiza que cada fila sea distinta y facilita la búsqueda y la relación de datos entre tablas.

OBJETIVO GENERAL:

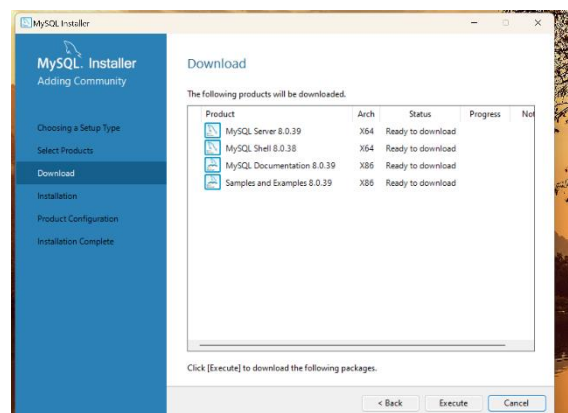
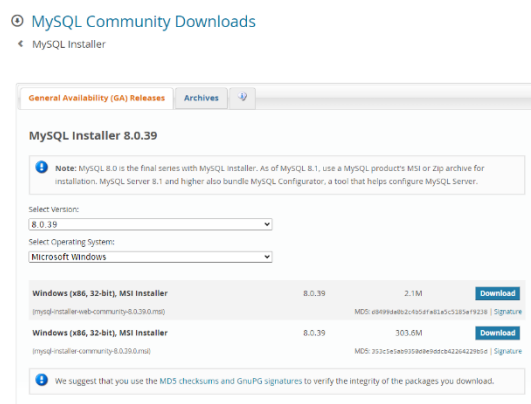
Familiarizar al estudiante con los conceptos básicos y las operaciones fundamentales en MySQL, desde la instalación y configuración hasta la creación y manipulación de bases de datos, tablas, y datos.

Sección 1: Instalación y Configuración

Objetivo: Asegurarse de que MySQL esté instalado y configurado correctamente en el sistema del estudiante.

1. Instalación de MySQL:

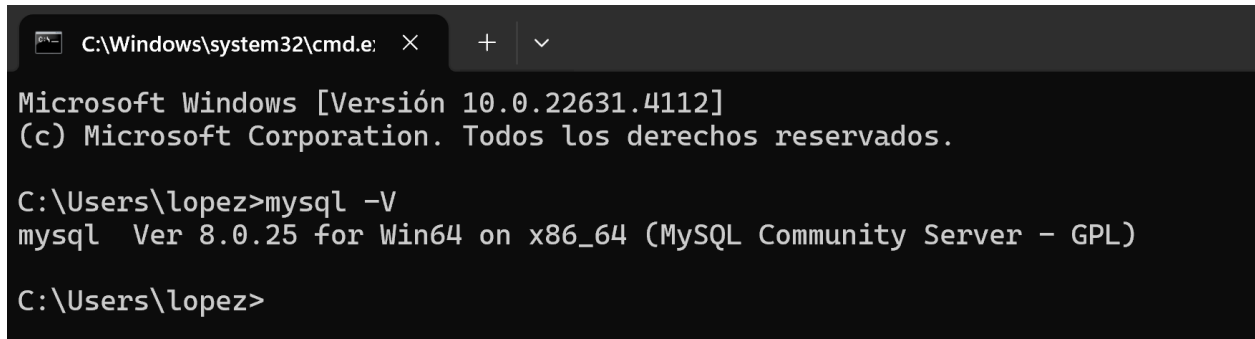
- **Windows:** El estudiante debe descargar el instalador de MySQL desde la página oficial. Debe seguir las instrucciones del instalador para la configuración básica.



2. Verificación de la instalación:

- El estudiante debe verificar que MySQL está instalado ejecutando el siguiente comando en la terminal:

mysql -V



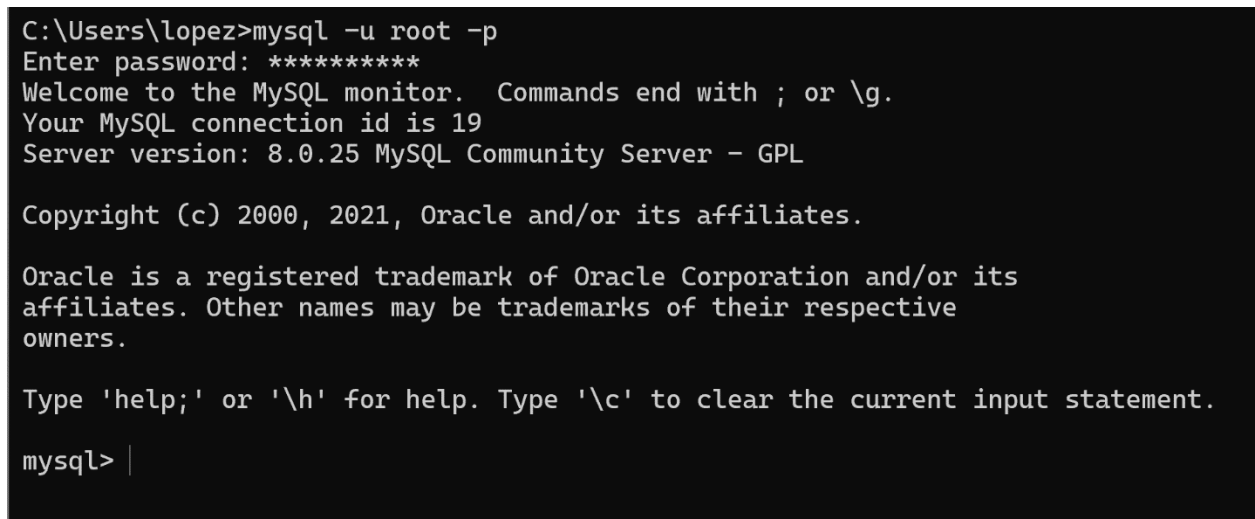
```
C:\Windows\system32\cmd.e: X + v
Microsoft Windows [Versión 10.0.22631.4112]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\lopez>mysql -V
mysql Ver 8.0.25 for Win64 on x86_64 (MySQL Community Server - GPL)

C:\Users\lopez>
```

- Luego, deberá iniciar sesión en MySQL como root para confirmar que todo funciona correctamente:

mysql -u root -p



```
C:\Users\lopez>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.0.25 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

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> |
```

Sección 2: Conceptos Básicos de MySQL

Objetivo: Introducir al estudiante en los conceptos fundamentales de bases de datos relacionales y el lenguaje SQL.

1. Bases de Datos Relacionales:

- Una **base de datos** es una colección organizada de datos.
- Una **tabla** es una estructura dentro de una base de datos que almacena datos en filas y columnas.
- **SQL (Structured Query Language)** es el lenguaje estándar para interactuar con bases de datos relacionales.

2. Componentes Clave:

- **Filas (Registros):** Cada fila representa una entrada única en la tabla.
- **Columnas (Campos):** Cada columna representa un atributo de los datos, como nombre, edad, etc.
- **Claves Primarias:** Un campo (o conjunto de campos) que identifica de manera única cada registro en la tabla.

Sección 3: Iniciando Sesión y Gestión de Usuarios

Objetivo: Aprender a iniciar sesión en MySQL y manejar usuarios y permisos.

1. Iniciar sesión en MySQL:

- El estudiante debe iniciar sesión como el usuario root con el siguiente comando:

```
mysql -u root -p
```

2. Crear un nuevo usuario:

- El estudiante creará un usuario llamado nuevo_usuario utilizando la siguiente instrucción SQL:

```
CREATE USER 'nuevo_usuario'@'localhost' IDENTIFIED BY 'contraseña_segura';
```

```
mysql> CREATE USER 'nuevo_usuario'@'local_host' IDENTIFIED BY 'contraseña_segura';  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql>
```

3. Asignar permisos:

- Se deben otorgar todos los privilegios al nuevo usuario sobre todas las bases de datos:

```
GRANT ALL PRIVILEGES ON *.* TO 'nuevo_usuario'@'localhost' WITH GRANT OPTION;
```

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'nuevo_usuario'@'local_host' WITH GRANT OPTION;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

4. Aplicar cambios:

- El estudiante debe asegurarse de que los cambios en los permisos se apliquen correctamente:

```
FLUSH PRIVILEGES;
```

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

Sección 4: Creación y Gestión de Bases de Datos

Objetivo: Aprender a crear, modificar y eliminar bases de datos.

1. Crear una base de datos:

- El estudiante debe crear una base de datos llamada `mi_base_de_datos` con el siguiente comando:

```
CREATE DATABASE mi_base_de_datos;
```

```
mysql> CREATE DATABASE mi_base_de_datos;
Query OK, 1 row affected (0.01 sec)

mysql> S
```

2. Mostrar bases de datos existentes:

- Se deben listar todas las bases de datos disponibles en el sistema:

```
SHOW DATABASES;
```

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| empresa  |
| information_schema |
| mi_base_de_datos |
| mysql    |
| performance_schema |
| sakila   |
| sys      |
| world    |
+-----+
8 rows in set (0.00 sec)
```

3. Seleccionar una base de datos:

- El estudiante debe usar la base de datos recién creada con el siguiente comando:

USE mi_base_de_datos;

```
mysql> USE mi_base_de_datos;  
Database changed  
mysql>
```

4. Eliminar una base de datos:

- Finalmente, se eliminará la base de datos usando el siguiente comando:

DROP DATABASE mi_base_de_datos;

```
mysql> DROP DATABASE mi_base_de_datos;  
Query OK, 0 rows affected (0.01 sec)  
  
mysql>
```

Sección 5: Creación y Gestión de Tablas

Objetivo: Crear, modificar y eliminar tablas en MySQL.

1. Crear una tabla:

- Dentro de la base de datos mi_base_de_datos, el estudiante debe crear una tabla llamada empleados con la siguiente estructura:

```
CREATE TABLE empleados (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  nombre VARCHAR(50),  
  apellido VARCHAR(50),  
  salario DECIMAL(10, 2)  
);
```

```
mysql> CREATE TABLE empleadoss(  
-> id INT AUTO_INCREMENT PRIMARY KEY,  
-> nombre VARCHAR(50) NOT NULL,  
-> apellido VARCHAR(50) NOT NULL,  
-> salario DECIMAL (10,2),  
-> departamento_id INT,  
-> FOREIGN KEY(departamento_id) REFERENCES departamentos(id)  
-> );  
Query OK, 0 rows affected (0.17 sec)
```

2. Mostrar tablas existentes:

- Se deben listar todas las tablas en la base de datos actual con el siguiente comando:

SHOW TABLES;

```
mysql> SHOW TABLES;
+-----+
| Tables_in_mi_base_de_datos |
+-----+
| empleados                  |
+-----+
1 row in set (0.00 sec)

mysql>
```

3. Modificar la estructura de la tabla:

- El estudiante debe agregar una columna fecha_contratacion a la tabla empleados con el siguiente comando:

ALTER TABLE empleados ADD fecha_contratacion DATE;

```
mysql> ALTER TABLE empleados ADD fecha_contratacion DATE;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

4. Eliminar una tabla:

- Finalmente, se eliminará la tabla empleados con el siguiente comando:

DROP TABLE empleados;

```
mysql> DROP TABLE empleados;
Query OK, 1 row affected (0.10 sec)
```

Sección 6: Manipulación de Datos

Objetivo: Insertar, actualizar, consultar y eliminar datos en las tablas.

1. Insertar datos en una tabla:

- El estudiante debe insertar registros en la tabla empleados utilizando las siguientes instrucciones SQL:

```
INSERT INTO empleados (nombre, apellido, salario, fecha_contratacion) VALUES ('Juan', 'Pérez', 3000.00, '2023-01-15');
```

```
INSERT INTO empleados (nombre, apellido, salario, fecha_contratacion) VALUES ('Ana', 'Gómez', 3200.00, '2023-03-22');
```

```
INSERT INTO empleados (nombre, apellido, salario, fecha_contratacion) VALUES ('Luis', 'Martínez', 2800.00, '2023-02-10');
```

```
mysql> INSERT INTO empleados (nombre, apellido, salario, fecha_contratacion) VALUES ('juan', 'perez', 3000.00, '2023-01-15');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO empleados (nombre, apellido, salario, fecha_contratacion) VALUES ('ana', 'gomez', 3200.00, '2023-03-22');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO empleados (nombre, apellido, salario, fecha_contratacion) VALUES ('luis', 'martinez', 2800.00, '2023-02-10');
Query OK, 1 row affected (0.01 sec)

mysql> |
```

- Se deben recuperar todos los registros de la tabla empleados con el siguiente comando:

```
SELECT * FROM empleados;
```

```
mysql> SELECT * FROM empleados;
+----+-----+-----+-----+-----+
| id | nombre | apellido | salario | fecha_contratacion |
+----+-----+-----+-----+-----+
| 1  | juan   | perez   | 3000.00 | 2023-01-15         |
| 2  | ana    | gomez   | 3200.00 | 2023-03-22         |
| 3  | luis   | martinez | 2800.00 | 2023-02-10         |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

- El estudiante también debe filtrar los registros donde el salario es mayor a 3000:

```
SELECT * FROM empleados WHERE salario > 3000;
```

```
mysql> SELECT * FROM empleados WHERE salario > 3000;
+----+-----+-----+-----+-----+
| id | nombre | apellido | salario | fecha_contratacion |
+----+-----+-----+-----+-----+
| 2  | ana    | gomez   | 3200.00 | 2023-03-22         |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> |
```


3. Actualizar registros:

- El estudiante debe aumentar el salario de Ana Gómez a 3500 con el siguiente comando:

```
UPDATE empleados SET salario = 3500.00 WHERE nombre = 'Ana' AND apellido = 'Gómez';
```

(volver a ejecutar query `SELECT * FROM empleados` para ver el registro en la tabla actualizada)

```
mysql> UPDATE empleados SET salario = 3500.00 WHERE nombre = 'ana' AND apellido = 'gomez';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM empleados;
+----+-----+-----+-----+-----+
| id | nombre | apellido | salario | fecha_contratacion |
+----+-----+-----+-----+-----+
| 1 | juan | perez | 3000.00 | 2023-01-15 |
| 2 | ana | gomez | 3500.00 | 2023-03-22 |
| 3 | luis | martinez | 2800.00 | 2023-02-10 |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> |
```

4. Eliminar registros:

- Finalmente, se eliminará el registro de Luis Martínez con el siguiente comando:

```
DELETE FROM empleados WHERE nombre = 'Luis' AND apellido = 'Martínez';
```

(volver a ejecutar query `SELECT * FROM empleados` para verificar que el registro fue eliminado de la tabla correctamente)

```
mysql> DELETE FROM empleados WHERE nombre= 'luis' AND apellido= 'martinez';
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM empleados;
+----+-----+-----+-----+-----+
| id | nombre | apellido | salario | fecha_contratacion |
+----+-----+-----+-----+-----+
| 1 | juan | perez | 3000.00 | 2023-01-15 |
| 2 | ana | gomez | 3500.00 | 2023-03-22 |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> |
```

CONCLUSION

Es interesante empezar a navegar a través de SGBD ya que facilitara a futuro en multiples tareas en las que se requiera tener el control de datos o conjuntos.

El creo que será empezar a familiarizarse con los query o comandos que utilizamos en cada práctica, además de que estamos empezando de lo más básico a un nivel más avanza, por eso es importante no atrasarse y evitar quedarse con alguna duda, solo es cuestión de comprensión, práctica y dedicación.

Un desafío, o mas un problema que se presento en algunas computadoras fue que al momento de ejecutar MySQL desde la terminal cmd ya que se tenia que agregar al pad que posteriormente lo reconociera.

Se soluciono de la siguiente manera:

Abrir el explorador de archivos, dirigirse al disco local (C) y abrir la carpeta de l programa MySQL.



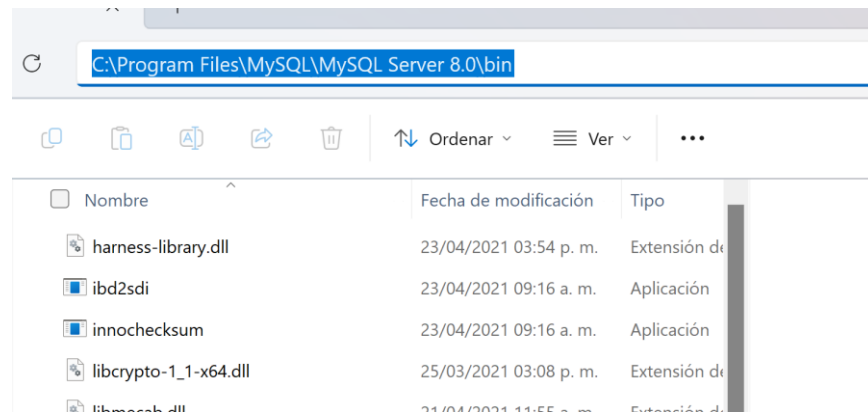
Se observa lo instalado dentro de MySQL y abrimos la carpeta MySQL Server

<input type="checkbox"/> Nombre	Fecha de modificación	Tipo
<input checked="" type="checkbox"/> MySQL Server 8.0	14/09/2024 03:13 p. m.	Carpeta de arch
MySQL Shell 8.0	14/09/2024 03:14 p. m.	Carpeta de arch

Seleccionamos la carpeta de binarios “bin”

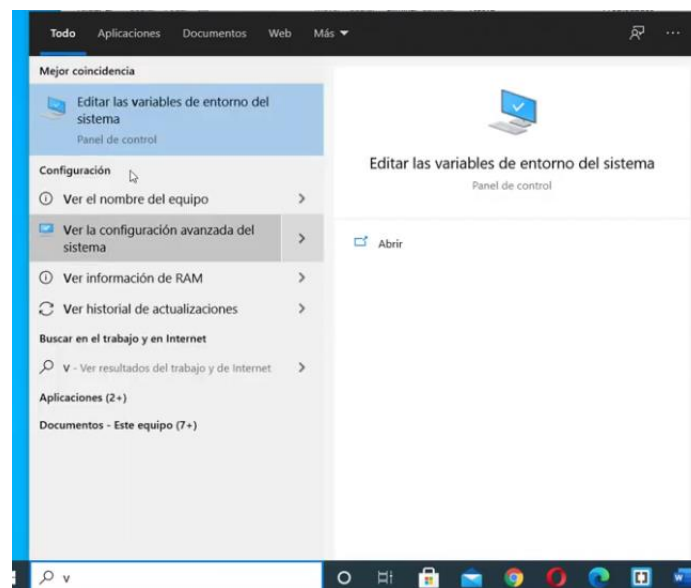
<input type="checkbox"/> Nombre	Fecha de modificación	Tipo
<input checked="" type="checkbox"/> bin	14/09/2024 03:13 p. m.	Carpeta de arch
<input type="checkbox"/> docs	14/09/2024 03:13 p. m.	Carpeta de arch

Abrimos la carpeta y bin y seleccionamos y copiamos la ruta que se ha seguido

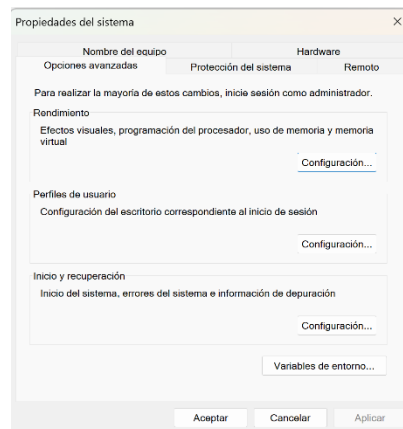


C:\Program Files\MySQL\MySQL Server 8.0\bin

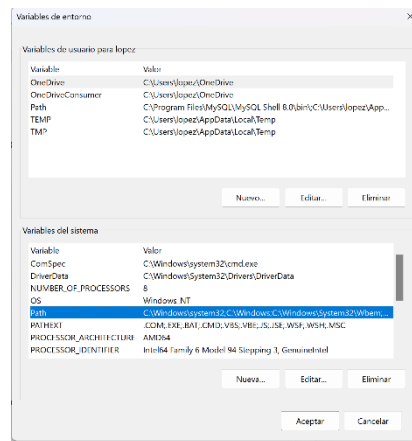
Buscar en la computadora la variable de autenticación, se coloca una V en el buscador y aparece como “Editar las variables de entorno del sistema”, doble clic para ejecutar.



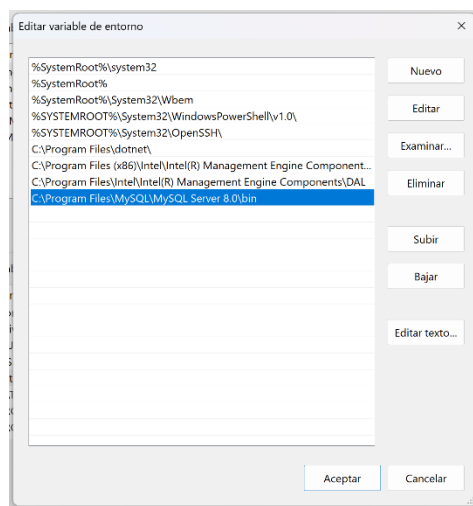
Se ejecuta y aparece una ventana llamada “propiedades del sistema”



Seleccionamos la opción “variables de entorno” que se encuentra en la parte inferior derecha.



Se abrirá otra ventana y seleccionamos la opción de “nuevo” y pegamos la ruta que copiamos anteriormente, una vez pegada pulsa “aceptar” y “aceptar”.



CODIGO COMPLETO

```
C:\Windows\system32\cmd.e: X + v

Microsoft Windows [Versión 10.0.22631.4112]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\loper>mysql -V
mysql Ver 8.0.25 for Win64 on x86_64 (MySQL Community Server - GPL)

C:\Users\loper>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.0.25 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.
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> CREATE USER 'nuevo_usuario'@'local_host' IDENTIFIED BY 'contrasella_segura';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'nuevo_usuario'@'localhost' WITH GRANT OPTION;
ERROR 1410 (42000): You are not allowed to create a user with GRANT
mysql> GRANT ALL PRIVILEGES ON *.* TO 'nuevo_usuario'@'local_host' WITH GRANT OPTION;
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE DATABASE;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1
mysql> CREATE DATA DATABASE mi_base_de_datos;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'DATA DATABASE mi_base_de_datos' at line 1
mysql> CREATE DATABASE mi_base_de_datos;
Query OK, 1 row affected (0.01 sec)

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mi_base_de_datos |
| mysql |
| performance_schema |
| sacola |
| sys |
| world |
+-----+
7 rows in set (0.00 sec)

mysql> USE mi_base_de_datos;
Database changed
mysql> DROP DATABASE mi_base_de_datos;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE empleados (
  -> id INT AUTO_INCREMENT PRIMARY KEY,
  -> nombre VARCHAR(50),
  -> apellido VARCHAR(50),
  -> salario DECIMAL(10, 2)
  -> );
ERROR 1046 (3D000): No database selected
mysql> CREATE TABLE empleados(
  -> id INT AUTO_INCREMENT PRIMARY KEY,
  -> nombre VARCHAR(50),
  -> apellido VARCHAR(50),
  -> salario DECIMAL(10, 2)
  -> );
ERROR 1064 (3D000): No database selected
mysql> CREATE DATABASE mi_base_de_datos;
Query OK, 1 row affected (0.01 sec)

mysql> USE mi_base_de_datos;
Database changed
mysql> CREATE TABLE empleados (
  -> id INT AUTO_INCREMENT PRIMARY KEY,
  -> nombre VARCHAR(50),
  -> apellido VARCHAR(50),
  -> salario DECIMAL(10, 2)
  -> );
Query OK, 0 rows affected (0.02 sec)

mysql> SHOW TABLES;
+-----+
| Tables_in_mi_base_de_datos |
+-----+
| empleados |
+-----+
1 row in set (0.00 sec)

mysql> ALTER TABLE empleados;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE empleados ADD fecha_contratacion DATE;
Query OK, 0 rows affected (0.02 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> INSERT INTO empleados (nombre, apellido, salario, fecha_contratacion) VALUES ('juan', 'perez', 3000.00, '2023-01-15');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO empleados (nombre, apellido, salario, fecha_contratacion) VALUES ('ana', 'gomez', 3200.00, '2023-03-22');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO empleados (nombre, apellido, salario, fecha_contratacion) VALUES ('luis', 'martinez', 2800.00, '2023-02-10');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM empleados;
+----+-----+-----+-----+-----+
| id | nombre | apellido | salario | fecha_contratacion |
+----+-----+-----+-----+-----+
| 1 | juan | perez | 3000.00 | 2023-01-15 |
| 2 | ana | gomez | 3200.00 | 2023-03-22 |
| 3 | luis | martinez | 2800.00 | 2023-02-10 |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> SELECT * FROM empleados WHERE salario > 3000;
+----+-----+-----+-----+-----+
| id | nombre | apellido | salario | fecha_contratacion |
+----+-----+-----+-----+-----+
| 2 | ana | gomez | 3200.00 | 2023-03-22 |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> UPDATE empleados SET salario = 3500.00 WHERE nombre = 'ana' AND apellido = 'gomez';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> SELECT * FROM empleados;
+----+-----+-----+-----+-----+
| id | nombre | apellido | salario | fecha_contratacion |
+----+-----+-----+-----+-----+
| 1 | juan | perez | 3000.00 | 2023-01-15 |
| 2 | ana | gomez | 3500.00 | 2023-03-22 |
| 3 | luis | martinez | 2800.00 | 2023-02-10 |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> DELETE FROM empleados WHERE nombre = 'luis' AND apellido 'martinez';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''martinez'' at line 1
mysql> DELETE FROM empleados WHERE nombre= 'luis' AND apellido= 'martinez';
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM empleados;
+----+-----+-----+-----+-----+
| id | nombre | apellido | salario | fecha_contratacion |
+----+-----+-----+-----+-----+
| 1 | juan | perez | 3000.00 | 2023-01-15 |
| 2 | ana | gomez | 3500.00 | 2023-03-22 |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> |
```



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



**TECNOLÓGICO
NACIONAL DE MÉXICO®**
INSTITUTO TECNOLÓGICO DE TIJUANA



INSTITUTO TECNOLÓGICO DE TIJUANA



TECNOLOGIAS DE BASE DE DATOS

Unidad 1

Sistemas Gestores de Base de Datos

Practica 2

Gestión Avanzada de Tablas y Consultas en MySQL

PRESENTA:

Camacho Lopez Valeria

C22211475

DOCENTE:

Fortunato Ramírez Arzate

TIJUANA, B. C, A 24 DE SEPTIEMBRE DEL 2024

INTRODUCCIÓN

El objetivo de esta práctica es que el estudiante adquiera una comprensión más profunda de la creación y gestión de múltiples tablas en MySQL, así como de la realización de consultas avanzadas utilizando funciones y operaciones SQL.

Sección 1: Creación de Tablas con Relaciones

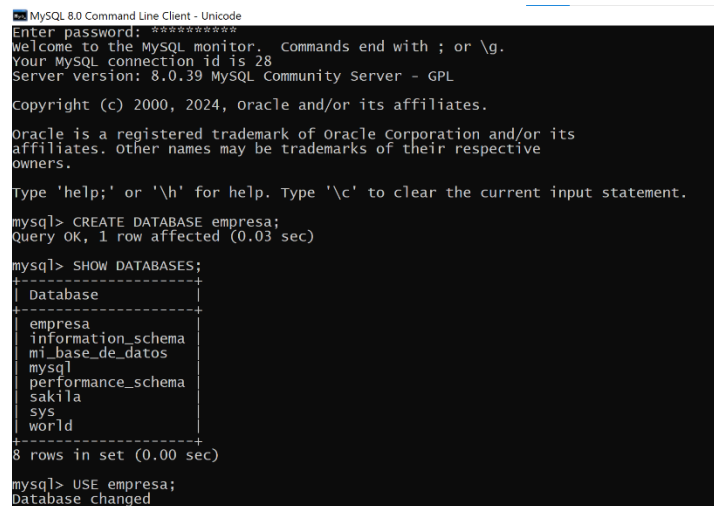
Objetivo de la Sección:

El estudiante aprenderá a crear tablas que contengan relaciones mediante el uso de claves foráneas, y a insertar datos en dichas tablas.

1. Crear una nueva base de datos:

- Crear y seleccionar la base de datos en la que trabajará el estudiante.

```
CREATE DATABASE empresa;  
USE empresa;
```

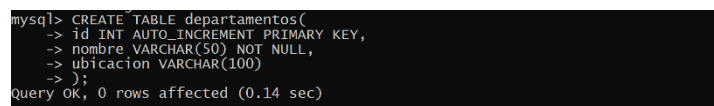


```
MySQL 8.0 Command Line Client - Unicode  
Enter password: *****  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 28  
Server version: 8.0.39 MySQL Community Server - GPL  
Copyright (c) 2000, 2024, Oracle and/or its affiliates.  
Oracle is a registered trademark 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> CREATE DATABASE empresa;  
Query OK, 1 row affected (0.03 sec)  
mysql> SHOW DATABASES;  
+-----+  
| Database |  
+-----+  
| empresa |  
| information_schema |  
| mi_base_de_datos |  
| mysql |  
| performance_schema |  
| sakila |  
| sys |  
| world |  
+-----+  
8 rows in set (0.00 sec)  
mysql> USE empresa;  
Database changed
```

2. Crear una tabla llamada departamentos:

- Diseñar la estructura de una tabla para almacenar los departamentos de una empresa.

```
CREATE TABLE departamentos (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(50) NOT NULL,  
    ubicacion VARCHAR(100)  
);
```



```
mysql> CREATE TABLE departamentos(  
-> id INT AUTO_INCREMENT PRIMARY KEY,  
-> nombre VARCHAR(50) NOT NULL,  
-> ubicacion VARCHAR(100)  
-> );  
Query OK, 0 rows affected (0.14 sec)
```

3. Crear una tabla llamada empleados:

- Crear una tabla para almacenar los empleados, vinculada a la tabla departamentos mediante una clave foránea.

```
CREATE TABLE empleados (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(50) NOT NULL,  
    apellido VARCHAR(50) NOT NULL,  
    salario DECIMAL(10,2),  
    departamento_id INT,  
    FOREIGN KEY (departamento_id) REFERENCES departamentos(id)  
);
```

```
mysql> CREATE TABLE empleadoss(  
-> id INT AUTO_INCREMENT PRIMARY KEY,  
-> nombre VARCHAR(50) NOT NULL,  
-> apellido VARCHAR(50) NOT NULL,  
-> salario DECIMAL(10,2),  
-> departamento_id INT,  
-> FOREIGN KEY(departamento_id) REFERENCES departamentos(id)  
-> );  
Query OK, 0 rows affected (0.17 sec)
```

4. Insertar datos en ambas tablas:

- Insertar registros en las tablas departamentos y empleados.

```
INSERT INTO departamentos (nombre, ubicacion) VALUES ('Recursos Humanos',  
'Planta 1');
```

```
INSERT INTO departamentos (nombre, ubicacion) VALUES ('Ventas', 'Planta  
2');
```

```
INSERT INTO departamentos (nombre, ubicacion) VALUES ('IT', 'Planta 3');
```

```
INSERT INTO empleados (nombre, apellido, salario, departamento_id) VALUES  
('Ana', 'López', 45000, 1);
```

```
INSERT INTO empleados (nombre, apellido, salario, departamento_id) VALUES  
('Juan', 'Pérez', 50000, 2);
```

```
INSERT INTO empleados (nombre, apellido, salario, departamento_id) VALUES  
('Carlos', 'García', 60000, 3);
```

```
mysql> INSERT INTO departamentos (nombre, ubicacion) VALUES('recursos humanos','planta 1');  
Query OK, 1 row affected (0.11 sec)  
  
mysql> INSERT INTO departamentos (nombre, ubicacion) VALUES('ventas','planta 2');  
Query OK, 1 row affected (0.10 sec)  
  
mysql> INSERT INTO departamentos (nombre, ubicacion) VALUES('IT','planta 3');  
Query OK, 1 row affected (0.01 sec)  
  
mysql> INSERT INTO empleadoss(nombre, apellido, salario, departamento_id)VALUES ('ana','lopez',45000, 1);  
Query OK, 1 row affected (0.09 sec)  
  
mysql> INSERT INTO empleadoss(nombre, apellido, salario, departamento_id)VALUES ('juan','perez',50000, 2);  
Query OK, 1 row affected (0.09 sec)  
  
mysql> INSERT INTO empleadoss(nombre, apellido, salario, departamento_id)VALUES ('carlos','garcia',60000, 3);  
Query OK, 1 row affected (0.03 sec)
```


Sección 2: Consultas con Múltiples Tablas (JOINS)

Objetivo de la Sección:

El estudiante aprenderá a realizar consultas que combinen datos de múltiples tablas mediante el uso de JOIN.

1. Mostrar los empleados junto con el nombre de su departamento:

- Realizar una consulta que relacione las tablas empleados y departamentos usando JOIN.

```
SELECT empleados.nombre, empleados.apellido, departamentos.nombre AS departamento FROM empleados JOIN departamentos ON empleados.departamento_id = departamentos.id;
```

```
mysql> SELECT empleados.nombre, empleados.apellido, departamentos.nombre AS departamento FROM empleados JOIN departamentos ON empleados.departamento_id = departamentos.id;
```

nombre	apellido	departamento
ana	lopez	recursos humanos
juan	perez	ventas
carlos	garcia	IT

```
3 rows in set (0.00 sec)
```

2. Consultar empleados de un departamento específico:

- Consultar solo los empleados que pertenecen al departamento de IT.

```
SELECT empleados.nombre, empleados.apellido
FROM empleados
JOIN departamentos ON empleados.departamento_id = departamentos.id
WHERE departamentos.nombre = 'IT';
```

(se realizó la consulta por separado para visualizar los empleados que corresponden a cada departamento)

```
mysql> SELECT empleados.nombre, empleados.apellido, departamentos.nombre AS departamento FROM empleados JOIN departamentos ON empleados.departamento_id = departamentos.id WHERE departamentos.nombre = 'ventas';
```

nombre	apellido	departamento
diego	perez	ventas
valeria	camacho	ventas

```
2 rows in set (0.00 sec)
```

```
mysql> SELECT empleados.nombre, empleados.apellido, departamentos.nombre AS departamento FROM empleados JOIN departamentos ON empleados.departamento_id = departamentos.id WHERE departamentos.nombre = 'recursos humanos';
```

nombre	apellido	departamento
ana	lopez	recursos humanos
emilia	silva	recursos humanos
rafael	herrera	recursos humanos

```
3 rows in set (0.00 sec)
```

```
mysql> SELECT empleados.nombre, empleados.apellido, departamentos.nombre AS departamento FROM empleados JOIN departamentos ON empleados.departamento_id = departamentos.id WHERE departamentos.nombre = 'IT';
```

nombre	apellido	departamento
carlos	garcia	IT
cristina	lopez	IT

```
2 rows in set (0.00 sec)
```

Sección 3: Uso de Funciones de Agregación

Objetivo de la Sección:

El estudiante utilizará funciones de agregación como COUNT, AVG, MAX, y MIN para generar estadísticas sobre los datos almacenados.

1. Contar cuántos empleados hay en cada departamento:

- Usar COUNT para contar registros agrupados por departamento.

```
SELECT departamentos.nombre, COUNT(empleados.id) AS num_empleados
FROM empleados
JOIN departamentos ON empleados.departamento_id = departamentos.id
GROUP BY departamentos.nombre;
```

```
mysql> SELECT departamentos.nombre, COUNT(empleadoss.id) AS num_empleados FROM empleadoss JOIN departamentos ON empleadoss.departamento_id = departamentos.id GROUP BY departamentos.nombre;
+-----+-----+
| nombre | num_empleados |
+-----+-----+
| recursos humanos | 3 |
| ventas | 2 |
| IT | 2 |
+-----+-----+
```

2. Calcular el salario promedio por departamento:

- Usar AVG para calcular el salario promedio por departamento.

```
SELECT departamentos.nombre, AVG(empleados.salario) AS salario_promedio
FROM empleados
JOIN departamentos ON empleados.departamento_id = departamentos.id
GROUP BY departamentos.nombre;
```

```
mysql> SELECT departamentos.nombre, AVG(empleadoss.salario) AS salario_promedio FROM empleadoss JOIN departamentos ON empleadoss.departamento_id = departamentos.id GROUP BY departamentos.nombre;
+-----+-----+
| nombre | salario_promedio |
+-----+-----+
| recursos humanos | 55000.000000 |
| ventas | 60000.000000 |
| IT | 62500.000000 |
+-----+-----+
3 rows in set (0.04 sec)
```

3. Obtener el salario máximo y mínimo de los empleados :

- Usar MAX y MIN para determinar el salario más alto y más bajo.

```
SELECT MAX(salario) AS salario_maximo, MIN(salario) AS salario_minimo
FROM empleados;
```

```
mysql> SELECT MAX(salario) AS salario_maximo, MIN(salario) AS salario_minimo FROM empleadoss;
+-----+-----+
| salario_maximo | salario_minimo |
+-----+-----+
| 70000.00 | 45000.00 |
+-----+-----+
1 row in set (0.00 sec)
```

Sección 4: Modificación de Tablas

Objetivo de la sección:

El estudiante aprenderá a modificar la estructura de las tablas agregando nuevas columnas y actualizando los datos existentes.

1. **Agregar una columna llamada fecha_contratacion a la tabla empleados:**
 - Agregue una nueva columna para almacenar la fecha de contratación de cada empleado.

```
ALTER TABLE empleados ADD COLUMN fecha_contratacion DATE;
```

```
mysql> ALTER TABLE empleados ADD COLUMN fecha_contratacion DATE;  
Query OK, 0 rows affected (0.14 sec)
```

2. **Actualizar datos en la nueva columna fecha_contratacion:**
 - Incluir la fecha de contratación para algunos empleados.

```
UPDATE empleados SET fecha_contratacion = '2020-05-10' WHERE nombre =  
'Ana';  
UPDATE empleados SET fecha_contratacion = '2019-07-22' WHERE nombre =  
'Juan';  
UPDATE empleados SET fecha_contratacion = '2021-01-15' WHERE nombre =  
'Carlos';
```

```
mysql> UPDATE empleados SET fecha_contratacion = '2020-05-10' WHERE nombre = 'Ana';  
Query OK, 0 rows affected (0.00 sec)  
Rows matched: 1  Changed: 0  Warnings: 0  
  
mysql> UPDATE empleados SET fecha_contratacion = '2019-07-22' WHERE nombre = 'Juan';  
Query OK, 0 rows affected (0.00 sec)  
Rows matched: 1  Changed: 0  Warnings: 0  
  
mysql> UPDATE empleados SET fecha_contratacion = '2021-01-15' WHERE nombre = 'Carlos';  
Query OK, 0 rows affected (0.00 sec)  
Rows matched: 1  Changed: 0  Warnings: 0  
mysql> SELECT * FROM empleados;  
+-----+-----+-----+-----+-----+-----+  
| id | nombre | apellido | salario | departamento_id | fecha_contratacion |  
+-----+-----+-----+-----+-----+-----+  
| 2 | ana | lopez | 45000.00 | 1 | 2020-05-10 |  
| 3 | juan | perez | 50000.00 | 1 | 2019-07-22 |  
| 4 | carlos | garcia | 60000.00 | 1 | 2021-01-15 |  
| 5 | valeria | camacho | 65000.00 | 2 | 2020-09-16 |  
| 6 | rafael | herrera | 55000.00 | 2 | 2023-07-14 |  
| 7 | emilia | silva | 57000.00 | 2 | 2021-03-18 |  
| 8 | ximena | solis | 40000.00 | 3 | 2018-11-25 |  
| 9 | anette | vidal | 48000.00 | 3 | 2019-05-23 |  
| 10 | fernanda | almazan | 54000.00 | 3 | 2024-10-07 |  
| 12 | Pedro | Sánchez | 65000.00 | 2 | NULL |  
+-----+-----+-----+-----+-----+-----+  
10 rows in set (0.00 sec)
```

Sección 5: Consultas Condicionales y Ordenación de Datos

Objetivo de la Sección:

El estudiante realizará consultas condicionales utilizando WHERE y aprenderá a ordenar los resultados con ORDER BY.

1. Consultar empleados contratados después del 1 de enero de 2020:

- Utilizar la cláusula WHERE para obtener resultados basados en una condición.

```
SELECT nombre, apellido, fecha_contratacion
FROM empleados
WHERE fecha_contratacion > '2020-01-01';
```

```
mysql> SELECT nombre, apellido, fecha_contratacion FROM empleados WHERE fecha_contratacion > '2020-01-01';
+-----+-----+-----+
| nombre | apellido | fecha_contratacion |
+-----+-----+-----+
| ana     | lopez    | 2020-05-10         |
| carlos  | garcia   | 2021-01-15         |
| valeria | camacho  | 2020-09-16         |
| rafael  | herrera  | 2023-07-14         |
| emilia  | silva    | 2021-03-18         |
| fernanda | almazan  | 2024-10-07         |
+-----+-----+-----+
6 rows in set (0.05 sec)
```

2. Ordenar los empleados por salario en orden descendente:

- Utilizar ORDER BY para ordenar los resultados de mayor a menor.

```
SELECT nombre, apellido, salario
FROM empleados
ORDER BY salario DESC;
```

```
mysql> SELECT nombre, apellido, salario FROM empleados ORDER BY salario DESC;
+-----+-----+-----+
| nombre | apellido | salario |
+-----+-----+-----+
| valeria | camacho  | 65000.00 |
| Pedro  | Sánchez  | 65000.00 |
| carlos  | garcia   | 60000.00 |
| emilia  | silva    | 57000.00 |
| rafael  | herrera  | 55000.00 |
| fernanda | almazan  | 54000.00 |
| juan    | perez    | 50000.00 |
| anette  | vidal    | 48000.00 |
| ana     | lopez    | 45000.00 |
| ximena  | solis    | 40000.00 |
+-----+-----+-----+
10 rows in set (0.00 sec)
```

Sección 6: Eliminar Datos y Tablas

Objetivo de la Sección:

El estudiante aprenderá a eliminar registros específicos y tablas completas en MySQL.

1. Eliminar un empleado de la tabla `empleados`:

- Borrar un registro específico de la tabla.

```
DELETE FROM empleados WHERE nombre = 'Carlos';
```

```
mysql> DELETE FROM empleados WHERE nombre = 'Carlos';
Query OK, 1 row affected (0.09 sec)

mysql> SELECT * FROM empleados;
+-----+-----+-----+-----+-----+-----+
| id | nombre | apellido | salario | departamento_id | fecha_contratacion |
+-----+-----+-----+-----+-----+-----+
| 2 | ana | lopez | 45000.00 | 1 | 2020-05-10 |
| 3 | juan | perez | 50000.00 | 1 | 2019-07-22 |
| 5 | valeria | camacho | 65000.00 | 2 | 2020-09-16 |
| 6 | rafael | herrera | 55000.00 | 2 | 2023-07-14 |
| 7 | emilia | silva | 57000.00 | 2 | 2021-03-18 |
| 8 | ximena | solis | 40000.00 | 3 | 2018-11-25 |
| 9 | anette | vidal | 48000.00 | 3 | 2019-05-23 |
| 10 | fernanda | almazan | 54000.00 | 3 | 2024-10-07 |
| 12 | Pedro | Sánchez | 65000.00 | 2 | NULL |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)
```

2. Eliminar la tabla `empleados`:

- Eliminar toda la tabla `empleados`.

```
DROP TABLE empleados;
```

```
mysql> DROP TABLE empleados;
Query OK, 0 rows affected (0.14 sec)

mysql> SELECT * FROM empleados;
ERROR 1146 (42S02): Table 'empresa.empleados' doesn't exist
mysql> _
```

Se eliminó la tabla de `empleados` y posteriormente verificamos que haya sido eliminada correctamente ejecutando el query `SELECT * FROM empleados;` arrojando un error ya que dicha tabla ya no existe dentro de la base de datos creada.

CONCLUSION

Se realizó con éxito la práctica de gestión avanzada de tablas y consultas en MySQL con ayuda de nuevos comandos que complementaron los query básicos vistos anteriormente, fue interesante ya que nos demostró con la práctica constante que se pueden realizar diversas combinaciones personalizadas de acuerdo a lo que el usuario necesita para realizar alguna tarea o consulta más elaborada.



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



**TECNOLÓGICO
NACIONAL DE MÉXICO®**
INSTITUTO TECNOLÓGICO DE TIJUANA



INSTITUTO TECNOLÓGICO DE TIJUANA



TECNOLOGIAS DE BASE DE DATOS

Unidad 1

Sistemas Gestores de Base de Datos

Practica 3

Consultas Avanzadas y Optimización en MySQL

PRESENTA:

Camacho Lopez Valeria

C22211475

DOCENTE:

Fortunato Ramírez Arzate

TIJUANA, B. C, A 4 DE OCTUBRE DEL 2024

INTRODUCCIÓN

Conceptos generales

1. **Índices:** Un índice en MySQL es una estructura de datos que mejora la velocidad de las operaciones de selección en una tabla a costos de una mayor utilización de espacio y tiempo durante las operaciones de inserción y actualización. Los índices funcionan como un índice de un libro, facilitando la búsqueda rápida de registros en una tabla.

Diagrama de Concepto de Índices:

Tabla sin índice:		Tabla con índice:
+-----+-----+-----+		+-----+-----+-----+-----+
ID Nombre Apellido		ID Nombre Apellido Índice
+-----+-----+-----+		+-----+-----+-----+-----+
1 Juan Pérez	---	1 Juan Pérez P
2 Ana Gómez	>	2 Ana Gómez G
3 Luis Martínez		3 Luis Martínez M
+-----+-----+-----+		+-----+-----+-----+-----+

Los índices permiten encontrar rápidamente valores como `Pérez` al organizar los datos internamente.

2. **Subconsultas:** Las subconsultas son consultas SQL anidadas dentro de otras consultas. Se utilizan para obtener resultados intermedios que luego se utilizan en la consulta externa.

Diagrama de Concepto de Subconsultas:

```
Consulta externa:
SELECT * FROM empleados WHERE salario > (...);
```

```
Subconsulta:
(SELECT AVG(salario) FROM empleados)
```

La subconsulta calcula un valor (el salario promedio en este caso) que luego es utilizado por la consulta externa para obtener los empleados cuyo salario es mayor que el promedio.

3. **Vistas:** Una vista en MySQL es una tabla virtual que resulta de una consulta. No almacena datos básicamente, pero facilita la reutilización de consultas complejas y permite organizar mejor las consultas.

Diagrama de Concepto de Vistas:

```
Vista creada:
CREATE VIEW empleados_vista
departamentos.nombre
AS SELECT ...
...;

Consulta original:
SELECT empleados.nombre,
FROM empleados JOIN departamentos ON

Consulta a la vista:
SELECT * FROM empleados_vista;
```

La vista permite simplificar consultas repetitivas.

4. **Transacciones:** Una transacción en MySQL es un conjunto de operaciones SQL que se ejecutan como una unidad. Si alguna de las operaciones falla, todas las demás se revierten, garantizando que la base de datos quede en un estado coherente.

Diagrama de Concepto de Transacciones:

```
START TRANSACTION;           -- Inicia la transacción
INSERT INTO empleados ...;    -- Operación 1
INSERT INTO empleados ...;    -- Operación 2

Si todo es correcto:
COMMIT;                       -- Confirma los cambios

Si hay un error:
ROLLBACK;                     -- Revierte todos los cambios
```

Las transacciones permiten garantizar la integridad de los datos.

5. **Optimización de Consultas (EXPLAIN):** La instrucción `EXPLAIN` en MySQL se utiliza para analizar cómo se ejecuta una consulta y sugerir mejoras, como el uso de índices.

Diagrama de Concepto de EXPLAIN:

Consulta: `SELECT * FROM empleados WHERE salario > 50000;`

Salida de EXPLAIN:

id	select_type	table	type	possible_keys	key	rows
1	SIMPLE	empleados	ALL	NULL	NULL	10000

`EXPLAIN` muestra cómo MySQL procesa una consulta y ayuda a encontrar áreas de mejora, como la adición de índices.

6. **Manejo de errores:** Los errores en MySQL pueden ocurrir por sintaxis incorrecta o por intentos de ejecutar operaciones no permitidas. Al aprender a gestionar errores, los estudiantes pueden depurar sus consultas y mejorar su código SQL.

Diagrama de errores comunes:

```
Consulta con error:  
SELECT * FORM empleados;
```

```
Error generado:  
Error: You have an error in your SQL syntax...
```

```
Consulta corregida:  
SELECT * FROM empleados;
```

Aprender a identificar y corregir errores en consultas es fundamental para la depuración.

Objetivo General de la Práctica 3:

El objetivo de esta práctica es que el estudiante explore técnicas avanzadas de MySQL, desde el uso de índices y subconsultas hasta la optimización de consultas y el manejo de transacciones, aplicando estos conceptos para gestionar bases de datos de manera eficiente.

Sección 1: Uso de índices

Objetivo: Mejorar el rendimiento de las consultas utilizando índices en MySQL.

1. Crear un índice en la tabla `empleados`:

- El estudiante debe crear un índice en la columna `apellido` para acelerar las consultas que la involucren:

```
CREATE INDEX idx_apellido ON empleados(apellido);
```

```
mysql> CREATE INDEX idx_apellido ON empleados(apellido);  
Query OK, 0 rows affected (0.22 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

2. Consultar utilizando el índice:

- Realizar una consulta que aproveche el índice creado para encontrar empleados por su apellido:

```
SELECT * FROM empleados WHERE apellido = 'Pérez';
```

```
mysql> SELECT * FROM empleados WHERE apellido = 'perez';
+----+-----+-----+-----+-----+-----+
| id | nombre | apellido | salario | departamento_id | fecha_contratacion |
+----+-----+-----+-----+-----+-----+
| 3 | juan | perez | 50000.00 | 1 | 2019-07-22 |
+----+-----+-----+-----+-----+-----+
1 row in set (0.04 sec)
```

3. Mostrar los índices de una tabla:

- El estudiante debe listar los índices disponibles en la tabla `empleados`:

```
SHOW INDEX FROM empleados;
```

```
mysql> SHOW INDEX FROM empleados;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| empleados | 0 | PRIMARY | 1 | id | A | 9 | NULL | NULL | NULL | BTREE | |
| empleados | 1 | YES | 1 | departamento_id | A | 3 | NULL | NULL | YES | BTREE | |
| empleados | 1 | YES | 1 | idx_apellido | A | 9 | NULL | NULL | NULL | BTREE | |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.12 sec)
```

Sección 2: Subconsultas

Objetivo: Aprender a utilizar subconsultas para realizar consultas más complejas y obtener resultados más específicos.

1. Subconsulta simple:

- El estudiante debe encontrar los empleados cuyo salario sea mayor al promedio de todos los empleados:

```
SELECT nombre, apellido, salario FROM empleados
WHERE salario > (SELECT AVG(salario) FROM empleados);
```

```
mysql> SELECT nombre, apellido, salario FROM empleados WHERE salario > (SELECT AVG(salario) FROM empleados);
+-----+-----+-----+
| nombre | apellido | salario |
+-----+-----+-----+
| carlos | garcia | 60000.00 |
| valeria | camacho | 65000.00 |
| rafael | herrera | 55000.00 |
| emilia | silva | 57000.00 |
| fernanda | almazan | 54000.00 |
+-----+-----+-----+
5 rows in set (0.05 sec)
```

2. Subconsulta con IN:

- El estudiante debe listar todos los empleados que pertenecen a los departamentos que están en la planta 2:

```
SELECT nombre, apellido FROM empleados WHERE departamento_id IN  
(SELECT id FROM departamentos WHERE ubicacion = 'Planta 2');
```

```
mysql> SELECT nombre, apellido FROM empleados WHERE departamento_id IN (SELECT id FROM departamentos WHERE ubicacion = 'planta 2');  
+-----+-----+  
| nombre | apellido |  
+-----+-----+  
| valeria | camacho |  
| rafael | herrera |  
| emilia | silva |  
+-----+-----+  
3 rows in set (0.02 sec)
```

Sección 3: Vistas

Objetivo: Facilitar consultas repetitivas y mejorar la organización de consultas complejas mediante la creación de vistas.

1. Crear una vista para mostrar empleados y sus departamentos:

- El estudiante debe crear una vista que combine la información de las tablas `empleados` y `departamentos`:

```
CREATE VIEW vista_empleados_departamentos AS SELECT  
empleados.nombre, empleados.apellido, departamentos.nombre AS  
departamento FROM empleados JOIN departamentos ON  
empleados.departamento_id = departamentos.id;
```

```
mysql> CREATE VIEW vista_empleados_departamentos AS  
-> SELECT empleados.nombre, empleados.apellido, departamentos.nombre AS departamento  
-> FROM empleados  
-> JOIN departamentos ON empleados.departamento_id = departamentos.id;  
Query OK, 0 rows affected (0.09 sec)  
  
mysql> SELECT * FROM vista_empleados_departamentos;  
+-----+-----+-----+  
| nombre | apellido | departamento |  
+-----+-----+-----+  
| ana | lopez | recursos humanos |  
| juan | perez | recursos humanos |  
| carlos | garcia | recursos humanos |  
| valeria | camacho | ventas |  
| rafael | herrera | ventas |  
| emilia | silva | ventas |  
| ximena | solis | it |  
| anette | vidal | it |  
| fernanda | almazan | it |  
+-----+-----+-----+  
9 rows in set (0.05 sec)
```

2. Consultar la vista creada:

- El estudiante debe consultar la vista para obtener los datos combinados:

```
SELECT * FROM vista_empleados_departamentos;
```

```
mysql> SELECT * FROM vista_empleados_departamentos;  
+-----+-----+-----+  
| nombre | apellido | departamento |  
+-----+-----+-----+  
| ana | lopez | recursos humanos |  
| juan | perez | recursos humanos |  
| carlos | garcia | recursos humanos |  
| valeria | camacho | ventas |  
| rafael | herrera | ventas |  
| emilia | silva | ventas |  
| ximena | solis | it |  
| anette | vidal | it |  
| fernanda | almazan | it |  
+-----+-----+-----+  
9 rows in set (0.05 sec)
```

1. Modificar una vista:

- El estudiante debe actualizar la vista para incluir el salario de los empleados:

```
CREATE OR REPLACE VIEW vista_empleados_departamentos AS
SELECT empleados.nombre, empleados.apellido, empleados.salario,
departamentos.nombre AS departamento FROM empleados
JOIN departamentos ON empleados.departamento_id =
departamentos.id;
```

```
mysql> CREATE OR REPLACE VIEW vista_empleados_departamentos AS
-> SELECT empleados.nombre, empleados.apellido, empleados.salario, departamentos.nombre AS departamento
-> FROM empleados
-> JOIN departamentos ON empleados.departamento_id = departamentos.id;
Query OK, 0 rows affected (0.01 sec)
```

Sección 4: Transacciones y Control de Integridad

Objetivo: Comprender el uso de transacciones en MySQL para asegurar la consistencia y la integridad de los datos.

1. Iniciar una transacción:

- El estudiante debe iniciar una transacción, realizar una inserción en la tabla `empleados` y luego revertirla:

```
START TRANSACTION;
INSERT INTO empleados (nombre, apellido, salario,
departamento_id) VALUES ('María', 'Fernández', 70000, 1);
ROLLBACK;
```

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO empleados (nombre, apellido, salario, departamento_id) VALUES ('María', 'Fernández', 70000, 1);
Query OK, 1 row affected (0.05 sec)

mysql> ROLLBACK;
Query OK, 0 rows affected (0.09 sec)
```

2. Confirmar cambios usando `COMMIT`:

- El estudiante debe realizar una inserción y confirmar los cambios con `COMMIT`:

```
START TRANSACTION;
INSERT INTO empleados (nombre, apellido, salario,
departamento_id) VALUES ('Pedro', 'Sánchez', 65000, 2);
COMMIT;
```

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO empleados (nombre, apellido, salario, departamento_id) VALUES ('Pedro', 'Sánchez', 65000, 2);
Query OK, 1 row affected (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)
```

Sección 5: Optimización de Consultas

Objetivo: Mejorar la eficiencia de las consultas mediante la utilización de buenas prácticas y herramientas de MySQL.

1. Uso de **EXPLAIN**:

- El estudiante debe analizar el plan de ejecución de una consulta para optimizarla:

```
EXPLAIN SELECT * FROM empleados WHERE salario > 50000;
```

```
mysql> EXPLAIN SELECT * FROM empleados WHERE salario > 3100;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | empleados | NULL | ALL | NULL | NULL | NULL | NULL | 3 | 33.33 | Using where |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)

mysql>
```

El query EXPLAIN indica que se requiere proporcionar detalles y optimiza la información del índice, por eso al momento de ejecutarlo se muestra una tabla con varias columnas que detallan como MySQL ejecutaría la consulta, en este caso solo muestra el proceso que MySQL haría, no muestra datos reales.

2. Optimización de una consulta con índices:

- Cree un índice en la columna `salario` y ejecute de nuevo la consulta para verificar la mejora:

```
CREATE INDEX idx_salario ON empleados(salario);
SELECT * FROM empleados WHERE salario > 50000;
```

```
mysql> CREATE INDEX idx_salario ON empleados(salario);
Query OK, 0 rows affected (0.07 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM empleados WHERE salario > 3100;
+-----+-----+-----+-----+-----+-----+-----+
| id | nombre | apellido | salario | fecha_contratacion | rfc |
+-----+-----+-----+-----+-----+-----+-----+
| 2 | emilia | SILVA | 3200.00 | 2022-03-17 | SILE84537 |
| 4 | emilia | silva | 3200.00 | 2023-07-25 | NULL |
| 3 | rafael | herrera | 3500.00 | 2022-03-17 | RAFHER6443 |
+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)

mysql>
```

Sección 6: Manejo de Errores y Depuración

Objetivo: Comprender cómo gestionar errores en MySQL y depurar consultas SQL.

1. Captura de errores:

- El estudiante debe ejecutar una consulta incorrecta y observar cómo se maneja el error:

```
SELECT * FORM empleados; -- Error en la palabra FORM
```

```
mysql> SELECT * FORM empleados;  
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'FORM empleados' at line 1
```

2. Revisión y corrección de errores comunes:

- El estudiante debe identificar el error en la consulta anterior y corregirlo:

```
SELECT * FROM empleados;
```

```
mysql> SELECT * FROM empleados;  
+----+-----+-----+-----+-----+-----+  
| id | nombre | apellido | salario | fecha_contratacion | rfc |  
+----+-----+-----+-----+-----+-----+  
| 1 | valeria | camacho | 3000.00 | 2023-01-15 | CALV040916 |  
| 2 | emilia | SILVA | 3200.00 | 2022-03-17 | SILE84537 |  
| 3 | rafael | herrera | 3500.00 | 2022-03-17 | RAFHER6443 |  
| 4 | emilia | silva | 3200.00 | 2023-07-25 | NULL |  
+----+-----+-----+-----+-----+-----+  
4 rows in set (0.00 sec)  
  
mysql> _
```

Es un error común ya que es gramatical y se da en ocasiones que se está escribiendo rápido y no se tiene el cuidado al ejecutar el query, la solución es volver a escribirlo correctamente para que se ejecute adecuadamente y muestre lo que se solicita.

Conclusión

Se realizó una práctica muy didáctica implementando de cierta manera material que no habíamos realizado anteriormente; el uso de los índices nos resultó útil para realizar consultas más rápidas y eficientes ya que nos muestra solo que necesitamos dependiendo la tarea o la necesidad, al igual se implementaron algunos query con los que ya estamos familiarizados, pero en esta ocasión para reforzar y analizar el uso de cada uno de ellos.