MANUAL DE INSTRUCCIONES

Administración de Base de Datos

Alumno: Medrano Rojas Adherly Jesus

Profesor: Perez Zanabria Mesias William

Semestre: IV

INDICE

Índice..... 1
¿Qué es la Administración de base de datos? 2
Introducción a la Administración de base de datos..... 3
Funciones de un DBA..... 4
Creación de una base de datos y tabla..... 5
Backup 6
Gestión de almacenamiento de datos 7
Consultas básicas con filtros con Where 8
Usuarios y Permisos 9
Control de usuario y Gestión 10
Monyog – Herramienta de monitoreo 11
Creación de vistas Mysql 12
Lenguaje DML y DDL..... 13
Procedimientos Almacenados 14
Programación de Triggers..... 15

¿Qué es la Administración de base de datos?

La administración de bases de datos es el conjunto de actividades, técnicas y herramientas utilizadas para gestionar y mantener bases de datos de manera eficiente, segura y confiable. Su objetivo principal es garantizar que las bases de datos funcionen correctamente y cumplan con los requisitos de las aplicaciones y usuarios que las utilizan.

Funciones principales de la administración de bases de datos:

Diseño de bases de datos:

• Modelar y estructurar la base de datos según las necesidades del sistema.
• Diseñar esquemas lógicos y físicos, definiendo tablas, relaciones, claves primarias y foráneas.

Gestión de datos:

• Insertar, actualizar y eliminar datos en la base de datos. Asegurar la consistencia, integridad y validez de los datos.

Seguridad:

• Configurar permisos y roles de usuario para proteger los datos sensibles.
• Implementar controles de acceso y cifrado.

Respaldo y recuperación:

- **Crear copias de seguridad regulares para proteger los datos contra pérdidas.**
- **Diseñar estrategias de recuperación ante desastres.**

Optimización del rendimiento:

- **Monitorear y ajustar el rendimiento de la base de datos.**
- **Realizar tareas como indexación, análisis de consultas y particionamiento de tablas.**

Mantenimiento:

- **Actualizar software de bases de datos (DBMS).**
- **Corregir errores y eliminar datos obsoletos.**

Monitoreo:

- **Supervisar la actividad de la base de datos y su uso.**
- **Detectar y resolver problemas como bloqueos, cuellos de botella o errores.**

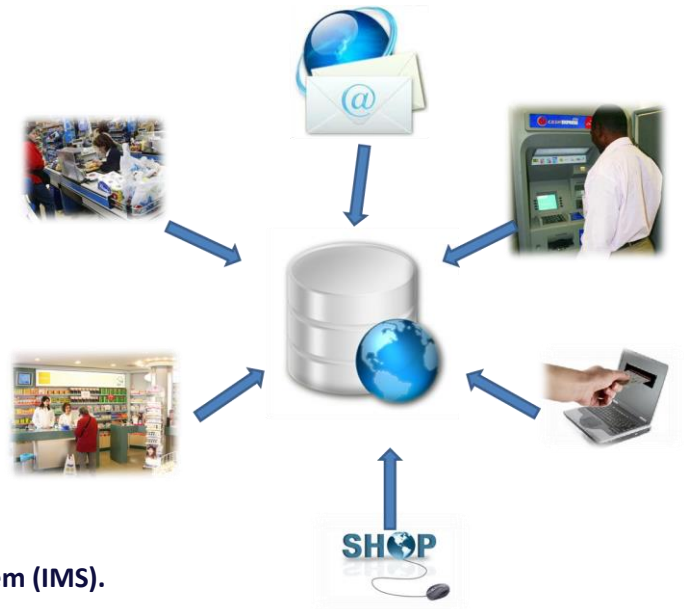
Introducción a la Administración a la base de datos

La Administración de Bases de Datos (DBA) es un campo que se enfoca en la gestión, mantenimiento y optimización de los sistemas de bases de datos para garantizar su funcionamiento eficiente, seguro y confiable. Las bases de datos son fundamentales para almacenar y gestionar grandes volúmenes de información, siendo una herramienta esencial en todos los sectores, desde empresas hasta aplicaciones web.

Tipos de Bases de Datos:

Relacionales (RDBMS):

- **Organizan datos en tablas.**
- **Utilizan SQL (Structured Query Language).**
- **Ejemplo: MySQL, PostgreSQL, MariaDB.**



No Relacionales (NoSQL):

- **Diseñadas para datos no estructurados o semiestructurados.**
- **Ejemplo: MongoDB, Redis, Cassandra.**

Jerárquicas:

- **Estructura en forma de árbol.**
- **Ejemplo: IBM Information Management System (IMS).**

Bases de Datos Orientadas a Objetos:

- **Manejan datos como objetos.**
- **Ejemplo: ObjectDB**

Roles del Administrador de Bases de Datos (DBA):

Diseño y Configuración:

• Definir esquemas y estructuras de las bases de datos.
• Configurar servidores y herramientas relacionadas.

Seguridad:

• Implementar controles de acceso y permisos.
• Realizar copias de seguridad y planes de recuperación ante desastres.

Funciones de un DBA

¿Qué es un DBA?

Un DBA Administrador de base de datos, es un profesional encargado de gestionar, mantener y proteger los sistemas de las bases de datos dentro de una organización. Su trabajo es crucial para asegurar que los datos sean accesibles, seguros y estén bien organizados.

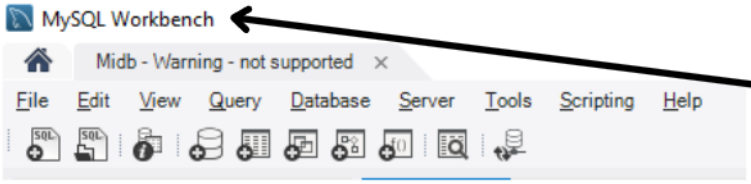
Funciones:

- **Diseño de base de datos: Crear y estructurar base de datos según las necesidades de la organización.**
- **Migración de datos: Facilitar la migración de datos entre diferentes sistemas o versiones**
- **Gestión de usuarios: Administrar cuentas de usuarios y controlar el acceso a la base de datos.**
- **Auditoria y cumplimiento: Realizar auditorías para asegurar el cumplimiento de normativas y políticas de datos.**
- **Planificación de capacidad: Anticipar las necesidades futuras de almacenamiento y rendimiento.**
- **Documentación: Mantener documentación, configuraciones, procedimientos y políticas.**

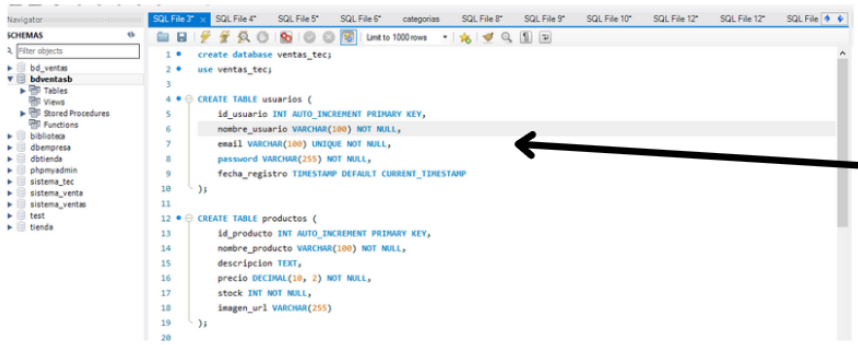
Responsabilidades de un DBA:

- **Instalación y configuración: Instalar y configurar sistema de gestión de base de datos(DBMS(Sistema de gestión de base de datos)).**
- **Mantenimiento: Realizar tareas de mantenimiento regular, como actualizaciones y parches.**
- **Copia de seguridad y recuperación: Implementar estrategias de copia de seguridad y recuperación de datos para evitar pérdidas.**
- **Seguridad: Configurar permisos y roles para proteger información sensible.**
- **Monitoreo y Optimización: Monitorizar el rendimiento de la base de datos y realizar ajustes para mejorar la eficiencia.**
- **Soporte Técnico: Proporcionar soporte de desarrolladores y usuarios en temas relacionados con la base de datos.**

Creación de una Base de Datos y Tablas



Para poder crear una base de datos usamos el aplicativo de MysSQL Worcbench



```
create database ventas_tec;
use ventas_tec;

CREATE TABLE usuarios (
    id_usuario INT AUTO_INCREMENT PRIMARY KEY,
    nombre_usuario VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE productos (
    id_producto INT AUTO_INCREMENT PRIMARY KEY,
    nombre_producto VARCHAR(100) NOT NULL,
    descripcion TEXT,
    precio DECIMAL(10, 2) NOT NULL,
    stock INT NOT NULL,
    imagen_url VARCHAR(255)
);
```

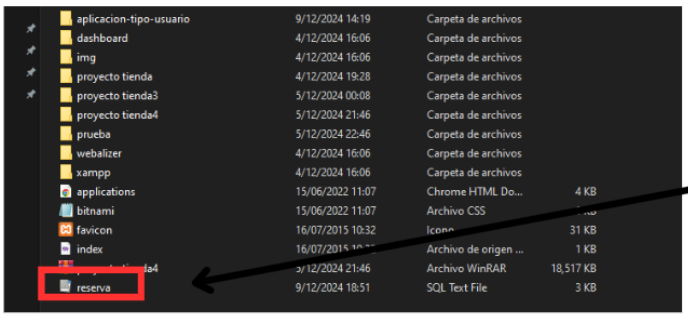
Entonces ahora creamos nuestra base de datos y tablas con los siguientes codigos

```
create database ventas_tec;
use ventas_tec;
```

Usamos el create database para poder crear una base datos.

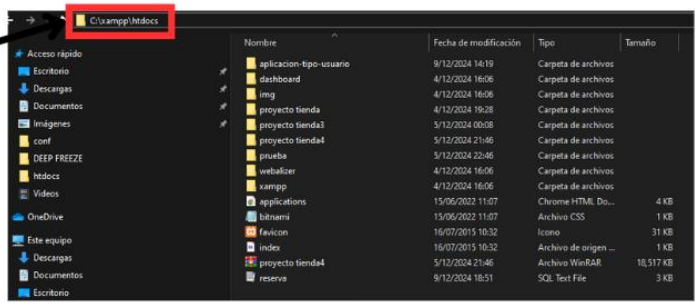
Para poder crear una tabla hacemos uso de la base de datos

Backup



Al final de la codificación en el CMD, si hemos realizado los pasos correctamente, nos mostrara nuestra base de datos ya como respaldo.

En la siguiente imagen muestra la ruta donde se almacenara nuestro respaldo




```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON reserva_hotel.* TO 'jose_luis'@'localhost';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> EXIT;
Bye

Adherly@DESKTOP-4H0LSS7 c:\xampp
# mysqldump -u jose_luis -p reserva_hotel > C:\xampp\htdocs\reserva.sql
Enter password: ******
```

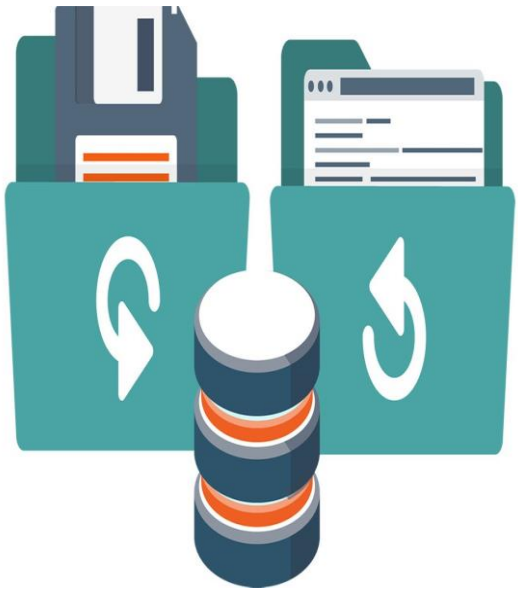
Para poder realizar un Backup de nuestra base de datos procedemos a realizar el siguiente código

En la imagen se muestra el usuario , la base de datos, la dirección donde se almacenada nuestro respaldo y al final nos pide la contraseña. Y con esos pasos se realiza nuestro respaldo

Tipos de almacenamiento

Al abrir nuestro respaldo que hicimos, se mostrara de la siguiente manera.

Gestión de almacenamiento de datos



Bases de datos relacionales (SQL):

- **Organizan datos en tablas con filas y columnas.**
- **Ejemplos: MySQL, MariaDB, PostgreSQL, Oracle, SQL Server.**
- **Usan lenguaje SQL para manipular y consultar los datos.**

Bases de datos no relacionales (NoSQL):

- **Diseñadas para trabajar con grandes volúmenes de datos no estructurados o semiestructurados.**
- **Ejemplos: MongoDB, Cassandra, Redis, CouchDB.**
- **Usan estructuras como documentos, gráficos, pares clave – valor.**

Almacenamiento en la nube:

- **Servicios escalables para almacenar datos en la nube.**
- **Ejemplos: Amazon S3, Google Cloud Storage, Microsoft Azure, Blob Storage.**
- **Proporcionan alta disponibilidad, redundancia, y accesibilidad global.**

Sistemas de archivos:

- **Usados para almacenar datos en la nube.**
- **Ejemplo: NTFS, FAT32, ext4.**
- **Ideal para almacenamiento local o en servidores.**

Data Warehousing:

- **Reuné grandes volúmenes de datos para análisis y reportes.**
- **Ejemplos: Snowflake, Amazon Redshift, Google BigQuery.**

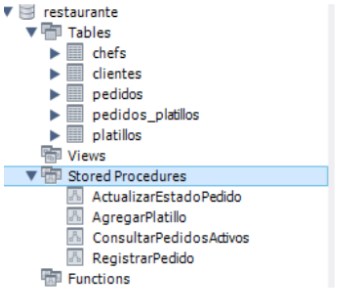
Elementos clave para la gestión de almacenamiento:

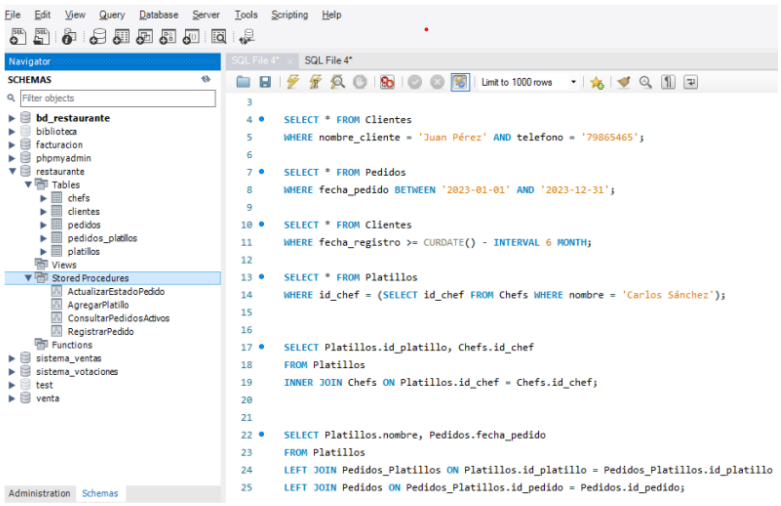
- **Seguridad de datos**
- **Optimización del rendimiento**
- **Escalabilidad**
- **Cumplimiento y Gobernanza**
- **Disponibilidad**

Consultas básicas de filtros de los datos con (WHERE).

Practica calificada:

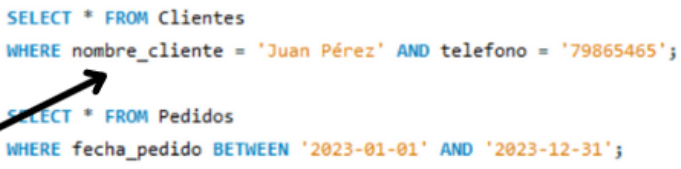
1. **Crear una base de datos con las respectivas tablas.**
2. **Hacer un llamado a las tablas con Select * From y con el Where solo llamar al cliente seleccionado con su teléfono.**
3. **También se tenía que hacer uso del INNER JOIN y LEFT JOIN.**





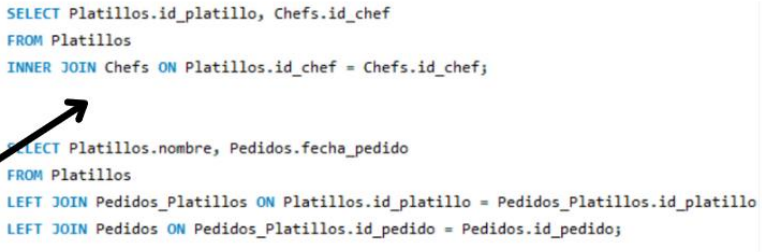
1. Crear una base de datos con sus respectivas tablas .

2. Hacer un llamado a las tablas con el Select * From y con el Where solo llamar al cliente seleccionado con su teléfono.



```
SELECT * FROM Clientes
WHERE nombre_cliente = 'Juan Pérez' AND telefono = '79865465';

SELECT * FROM Pedidos
WHERE fecha_pedido BETWEEN '2023-01-01' AND '2023-12-31';
```

3. Tambien se tenia que hacer uso del INNER JOIN y LEFT JOIN



```
SELECT Platillos.id_platillo, Chefs.id_chef
FROM Platillos
INNER JOIN Chefs ON Platillos.id_chef = Chefs.id_chef;

SELECT Platillos.nombre, Pedidos.fecha_pedido
FROM Platillos
LEFT JOIN Pedidos_Platillos ON Platillos.id_platillo = Pedidos_Platillos.id_platillo
LEFT JOIN Pedidos ON Pedidos_Platillos.id_pedido = Pedidos.id_pedido;
```

Usuarios y Permisos

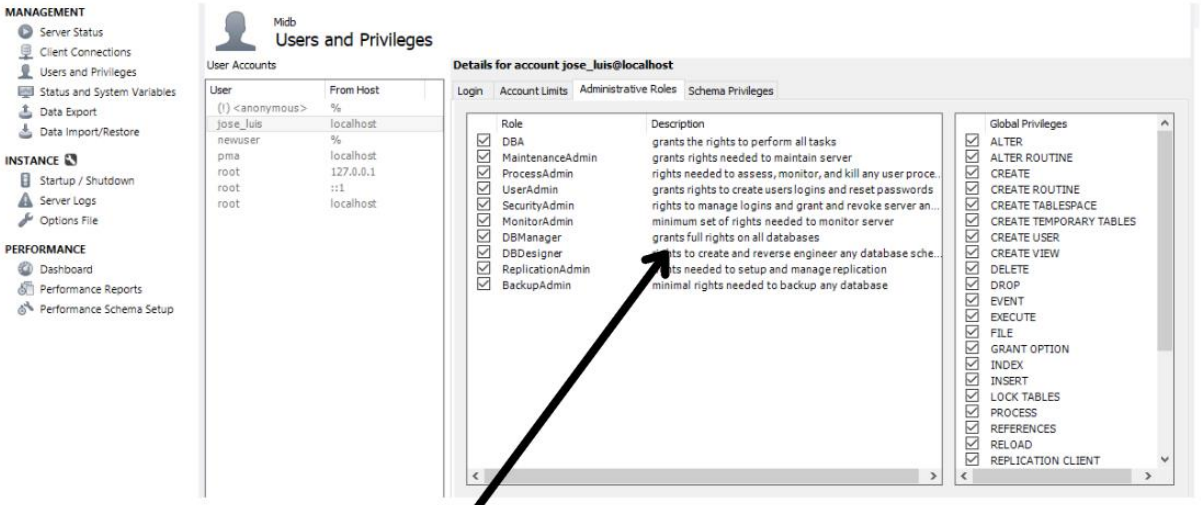
Para crear un usuario y con respectiva contraseña se realiza el siguiente código

```
1     -- Crear un usuario llamado 'nuevo_usuario' con contraseña 'mi_contraseña'
2 •   CREATE USER 'jose_luis'@'localhost' IDENTIFIED BY '123456';
3
4     -- Otorgar permisos al usuario sobre la base de datos 'ventas_tec'
5 •   GRANT ALL PRIVILEGES ON ventas_tec.* TO 'jose_luis'@'localhost';
6
7     -- Aplicar los cambios
8 •   FLUSH PRIVILEGES;
9
```

Para poder dar privilegios al usuario se realiza el siguiente codigo

```
-- Otorgar permisos al usuario sobre la base de datos 'ventas_tec'
GRANT ALL PRIVILEGES ON ventas_tec.* TO 'jose_luis'@'localhost';

-- Revocar permisos específicos al usuario 'usuario_ejemplo' en la base de datos 'ventas_tec'
REVOKE SELECT, INSERT, UPDATE ON ventas_tec.* FROM 'jose_luis'@'localhost';

-- Aplicar los cambios
FLUSH PRIVILEGES;
```

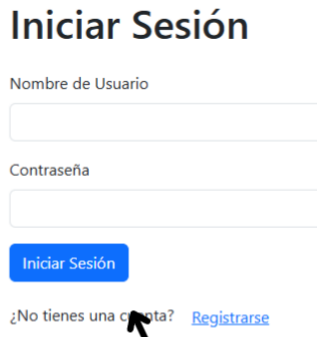
Para quitarle el privilegio al usuario se realiza el siguiente código



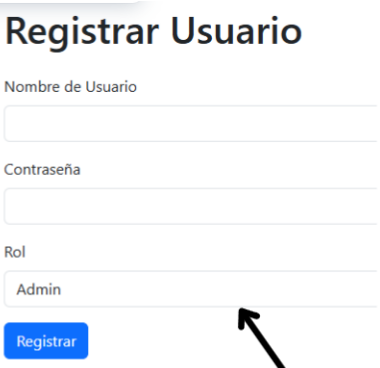
Privilegios otorgados al usuario jose_luis

Control y Gestión de usuarios

En el sistema de Instituto CRUD , cuando el usuario ingresa como ADMINISTRADOR tiene acceso a todas las paginas para poder editarlas.



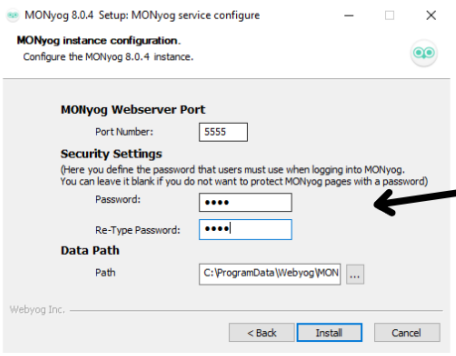
El sistema Instituto CRUD cuenta con un login donde se ingresa el usuario según su rol previo a su registro.



En el sistema de Instituto CRUD, también cuenta con un registro para poder registrarse en el sistema según un rol establecido.

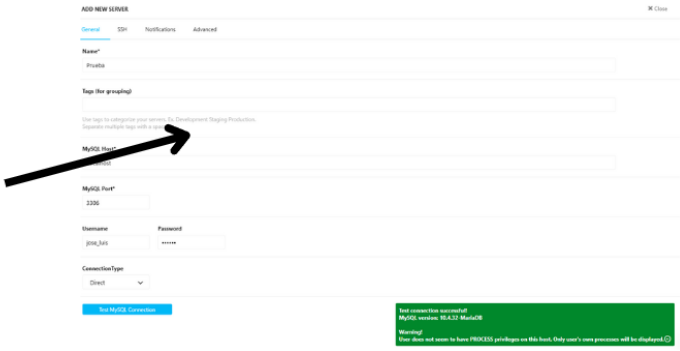
El sistema cuenta con dos tipos de roles, uno es de USUARIO normal y el otro de ADMINISTRADOR

Monyog-Herramienta de monitoreo



Para poder ingresar al entorno de Monyug primero necesitamos instalar y usar un usuario y una contraseña para acceder a Monyug

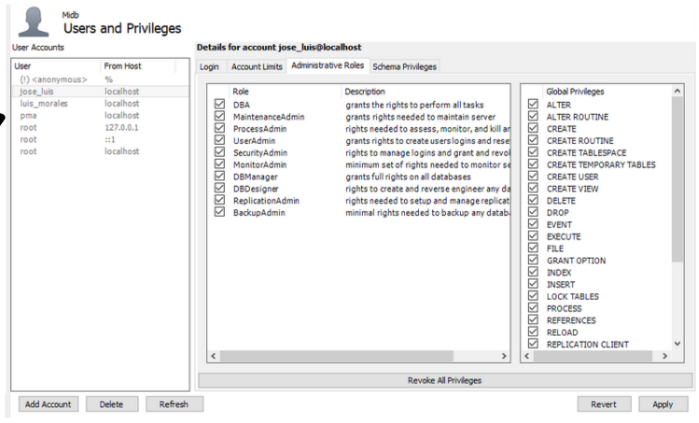
Una vez dentro del entorno, necesitamos de un usuario y su contraseña para ello lo creamos dentro de Worcbench.

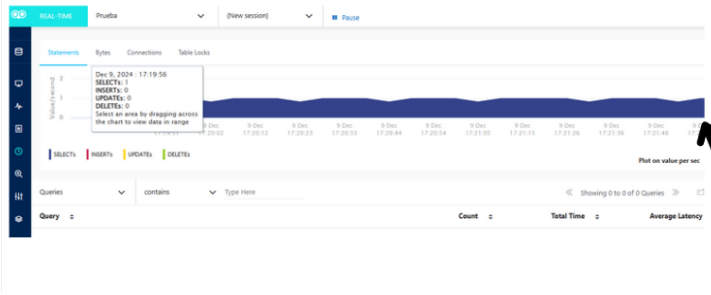


```
1    -- Crear un usuario llamado 'nuevo_usuario' con contraseña 'mi_contraseña'
2    CREATE USER 'jose_luis'@'localhost' IDENTIFIED BY '123456';
3
4    -- Otorgar permisos al usuario sobre la base de datos 'ventas_tec'
5    GRANT ALL PRIVILEGES ON ventas_tec.* TO 'jose_luis'@'localhost';
6
7    -- Revocar permisos específicos al usuario 'usuario_ejemplo' en la base de datos 'ventas_tec'
8    REVOKE SELECT, INSERT, UPDATE ON ventas_tec.* FROM 'jose_luis'@'localhost';
```

En esta ocasión se creo un usuario llamado jose_luis con la contraseña de “123456” y tambien se le dio privilegios

En la siguiente imagen se muestra los privilegios brindados al usuario jose_luis



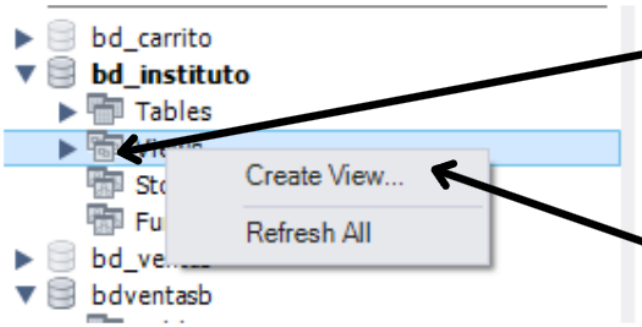


En esta imagen podemos apreciar dentro del entorno de Monyug como va siendo el monitoreo del usuario.

Creación de vistas Mysql

```sql
CREATE DATABASE bd_instituto;

USE bd_instituto;

CREATE TABLE Usuarios (
    idUsuario INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    role ENUM('admin', 'user') NOT NULL
);

CREATE TABLE Carreras (
    idCarrera INT AUTO_INCREMENT PRIMARY KEY,
    nombreCarrera VARCHAR(100) NOT NULL
);
INSERT INTO Carreras (nombreCarrera) VALUES ('Ingeniería de Sistemas');
INSERT INTO Carreras (nombreCarrera) VALUES ('Medicina');
INSERT INTO Carreras (nombreCarrera) VALUES ('Arquitectura');
```

Para la creación de la vista necesitamos una base de datos con la cual se va a trabajar, en este caso la base de datos que se uso para realizar la vista fue bd_instituto;

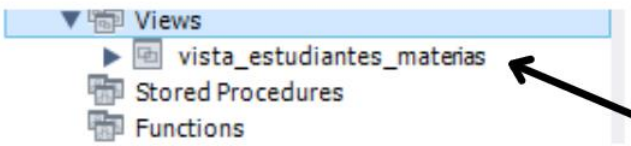


Para crear una vista accedemos en Views y se hace un anticlick

Se selecciona el create view y luego se crea la vista correspondiente.

```sql
CREATE VIEW vista_estudiantes_materias AS
SELECT
    e.nombre AS nombreEstudiante,
    e.apellido AS apellidoEstudiante,
    c.nombreCarrera,
    m.nombreMateria
FROM Estudiantes e
JOIN Carreras c ON e.idCarrera = c.idCarrera
JOIN Materias m ON c.idCarrera = m.idCarrera;
```

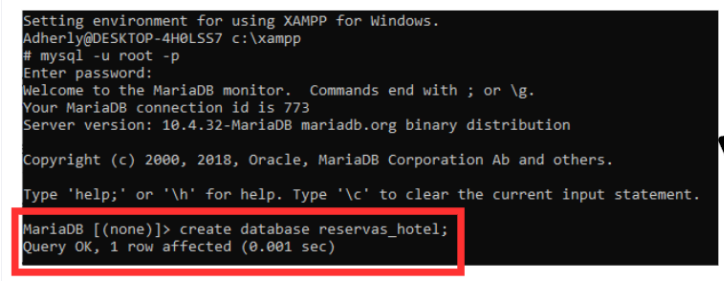
En la imagen se muestra una vista de la base de datos bd_instituto



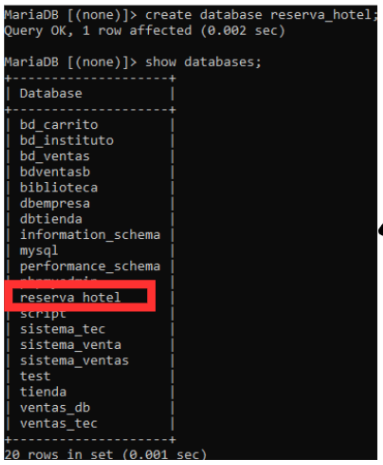
Una vez creada la vista, se podrá observa de la siguiente forma

Lenguaje DML y DDL

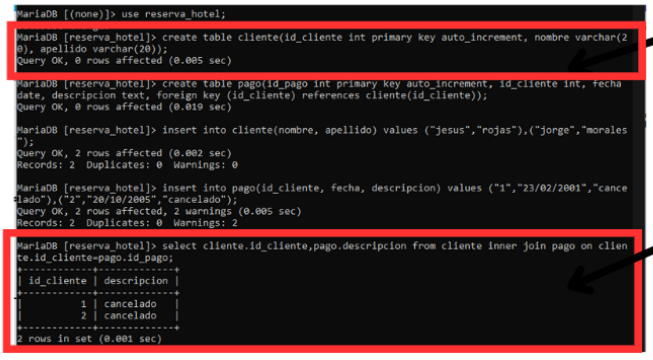
Diferencia	DML (Lenguaje de Manipulación de Datos)	DDL (Lenguaje de Definición de Datos)
Objetivo	Manipula los **datos** de las tablas.	Manipula la **estructura** de la base de datos.
Operaciones comunes	`INSERT`, `SELECT`, `UPDATE`, `DELETE`.	`CREATE`, `ALTER`, `DROP`, `TRUNCATE`.
Afecta	Los registros de las tablas.	La estructura de la base de datos (tablas, vistas, etc.).
Transacciones	Acepta transacciones (puedes hacer `ROLLBACK`).	No acepta transacciones (una vez ejecutado, es irreversible).
Ejemplo	Insertar un nuevo registro en una tabla.	Crear o eliminar una tabla.



En la siguiente imagen se muestra la creación de una base de datos



En la siguiente imagen se muestra todas las base de datos y también se muestra la base de datos creada



muestra la creación de las tablas y también la inserción de datos mediante el INSERT INTO

En la siguiente imagen se muestra el INNER JOIN de las tablas cliente y pago

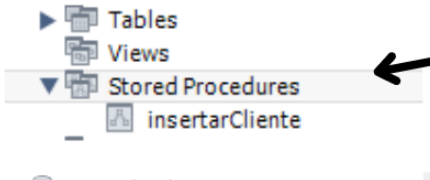
Procedimientos Almacenados

```
Setting environment for using XAMPP for Windows.
Adherly@DESKTOP-4H0LSS7 c:\xampp
# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 773
Server version: 10.4.32-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> create database reservas_hotel;
Query OK, 1 row affected (0.001 sec)
```

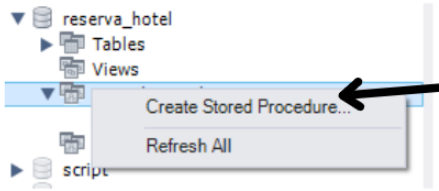
En la siguiente imagen se muestra la creación de una base de datos llamado reservas_hotel.

En la siguiente imagen se muestra todas las base de datos y también se muestra la base de datos creada y en esta ocasión usaremos la base de datos reservas_hotel.

```
MariaDB [(none)]> create database reserva_hotel;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| bd_carrito         |
| bd_instituto       |
| bd_ventas          |
| bdventasb          |
| biblioteca         |
| dbempresa          |
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
| reserva_hotel      |
| script             |
| sistema_tec        |
| sistema_venta      |
| sistema_ventas     |
| test               |
| tienda             |
| ventas_db          |
| ventas_tec         |
+--------------------+
20 rows in set (0.001 sec)
```



Seleccionamos el Stored Procedures



Para crear un procedimiento almacenado hacemos un anticlick en Create Stored Procedure

Luego procedemos a realizar el código correspondiente para nuestro procedimiento almacenado

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `insertarCliente`(
    IN nombreCliente VARCHAR(20),
    IN apellidoCliente VARCHAR(20)
)
BEGIN
    INSERT INTO cliente (nombre, apellido)
    VALUES (nombreCliente, apellidoCliente);
END
```

```
MariaDB [reserva_hotel]> call insertarCliente('Ana', 'Gómez');
Query OK, 1 row affected, 1 warning (0.002 sec)

MariaDB [reserva_hotel]> call insertarCliente('juan','lopez');
Query OK, 1 row affected (0.002 sec)

MariaDB [reserva_hotel]>
```

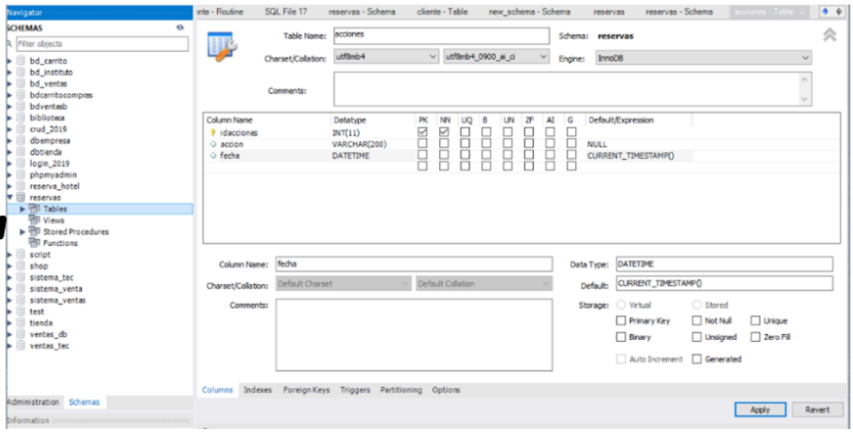
Dentro del entorno del CMD hacemos un llamado con el CALL insertarCliente para agregar datos en nuestra tabla

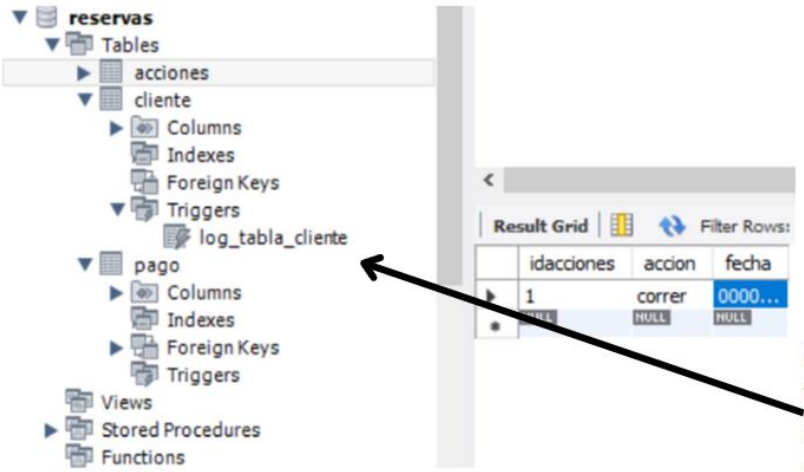
Programación de Triggers

```
1  USE reservas;
2  -- crear un triggers
3  DELIMITER //
4  CREATE TRIGGER  log_tabla_cliente AFTER INSERT ON cliente
5  FOR EACH ROW BEGIN
6     INSERT INTO acciones (accion) value ('Se creo un registro en cliente');
7  END //
8  DELIMITER ;
9  -- eliminar triggers
10 DROP TRIGGER IF EXISTS log_tabla_cliente;
11 -- ver los registros triggers
12 SHOW TRIGGERS;
```

Hacemos uso de la base de datos de reservas para poder crear nuestro trigger

En la siguiente imagen creamos una nueva tabla llamada acciones





Después de haber creado nuestra tabla acciones podemos observar que nuestro trigger se ha creado dentro de la tabla clientes