Nombre de la práctica	OPERACIONES CRUD			No.	01
Asignatura:	Taller de bases de datos	Carrera:	Ingeniería en sistemas	Duración de la práctica (Hrs)	

Nombre del alumno: Jesús Navarrete Martínez
Grupo: 3501

I. Competencia(s) específica(s):

II. Lugar de realización de la práctica (laboratorio, taller, aula u otro): Aula

III. Material empleado:

- **Equipo de computo**

IV. Desarrollo de la práctica:

```
CREATE TABLE usuarios(
    id_usuario INT PRIMARY KEY AUTO_INCREMENT,
    email VARCHAR(100) UNIQUE NOT NULL CHECK (email LIKE "%_@_%._%"),
    password VARCHAR(15) NOT NULL CHECK(LENGTH(password)>=8)

);

            --  EJEMPLOS DE OPERACIONES CREATE --

-- Ejemplo de una inserción valida usando todos los campos --
INSERT INTO usuarios VALUES(1,"ejemplo@mail.com","12345678");

-- Ejemplo de una insercion valida usando DEFAULT --
INSERT INTO usuarios VALUES (DEFAULT,"ejemplo2@gmail.com","abcdefgh");


-- Ejemplo de una insercion sin incluir el id_usuario --

INSERT usuarios(email,password) VALUES ("email3@hotmail.com","12345678");


                -- EJERCICIOS CREATE --

/*
 IDENTIFICACION DE 4 ERRORES QUE PUEDEN SURGIR EN ESTA TABLA:

1. Violación de la restricción de longitud del password: Si se intenta insertar una
contraseña con menos de 8 caracteres, se generará un error debido al
CHECK(LENGTH(password) >= 8).

Ejemplo de error:

INSERT INTO usuarios(email, password) VALUES ("usuario@mail.com", "12345");
-- Error: CHECK constraint failed: usuarios (la longitud de password es menor a 8)
```

2.Violación de la restricción de formato del email: El campo email tiene una restricción que obliga a que cumpla con un formato específico (_@_._). Si no cumple con este formato, generará un error.

Ejemplo de error:

```
INSERT INTO usuarios(email, password) VALUES ("usuario_invalido", "12345678");
-- Error: CHECK constraint failed: usuarios (el formato del email no es válido)
```

3.Violación de la unicidad del email: Dado que el campo email es UNIQUE, no se pueden insertar dos usuarios con el mismo correo electrónico. Si se intenta duplicar el correo, se lanzará un error.

Ejemplo de error:

```
INSERT INTO usuarios(email, password) VALUES ("ejemplo@mail.com", "12345678");
-- Error: Duplicate entry 'ejemplo@mail.com' for key 'usuarios.email'
```

4.Violación de valores nulos en campos NOT NULL: Si se intenta insertar un valor NULL en cualquiera de los campos que no lo permiten (en este caso, tanto email como password son NOT NULL), se generará un error.

Ejemplo de error:

```
INSERT INTO usuarios(email, password) VALUES (NULL, "abcdefgh");
-- Error: Column 'email' cannot be null */

-- INSERCION DE 4 REGISTROS NUEVOS CON UN SOLO INSERT:
INSERT INTO usuarios(email, password)
VALUES
("usuario4@mail.com", "claveSecreta1"),
("usuario5@mail.com", "passwordABC123"),
("usuario6@mail.com", "contraseña789"),
("usuario7@mail.com", "claveSegura876");



                -- EJEMPLOS DE OPERACIONES READ --

-- Ejemplo de una consulta para todos los datos de una tabla --
SELECT *FROM usuarios; -- selecciona todo de la tabla usuarios --

-- Ejemplo de consulta para un registro en especifico a traves del id --
SELECT *FROM usuarios WHERE id_usuario=1;

-- Ejemplo de consulta con un email en especifico --
SELECT *FROM usuarios WHERE email="ejemplo@mail.com";

-- Ejemplo de consulta con solo los campos email y password --
SELECT email,password FROM usuarios;
-- Ejemplo de consulta con un condicional logico --
SELECT *FROM usuarios WHERE LENGTH(password)>9;

                -- EJERCICIOS READ --
/*
Realiza una consulta que muestre solo el email pero que coincida con una contraseña de mas de 8 caracteres y otra que realice una consulta a los id pares.
*/
-- 1
SELECT email FROM usuarios WHERE LENGTH(password) > 8;

-- 2
SELECT * FROM usuarios WHERE id_usuario % 2 = 0;
```

```
-- EJEMPLOS DE OPERACIONES UPDATE --

 -- Ejemplo para actualizar la contraseña por su id --
UPDATE usuarios SET password="a1b2c3d4e5" WHERE id_usuario=1;

-- Ejemplo para actualizar el email y password de un usuario en especifico --
UPDATE usuarios SET password="a1b2c3d4e5",email="luciohdz3012@gmail.com" WHERE
id_usuario=1;

-- EJERCICIO UPDATE --

/*
1. Actualización con un email en un formato incorrecto (violación del CHECK de formato
del email):

UPDATE usuarios SET email = "emailinvalido" WHERE id_usuario = 1;
Error esperado: El email no sigue el formato especificado en la restricción (email
LIKE "%_@_%._%"). Esto generará un error de violación de la restricción.

2. Actualización con una contraseña demasiado corta (violación del CHECK de longitud
de la contraseña):

UPDATE usuarios SET password = "12345" WHERE id_usuario = 2;
Error esperado: La longitud de la contraseña es menor a 8 caracteres, lo que viola la
restricción CHECK(LENGTH(password) >= 8).

3. Actualización con un email que ya existe (violación de la restricción UNIQUE):

UPDATE usuarios SET email = "ejemplo@mail.com" WHERE id_usuario = 3;
Error esperado: El correo "ejemplo@mail.com" ya existe en la base de datos, por lo que
se violaría la restricción UNIQUE, que impide la duplicación de valores en el campo
email.
*/



-- EJEMPLOS DE OPERACIONES DELETE --

-- Eliminar el usuario por el id --
DELETE FROM usuarios WHERE id_usuario=4;

-- Ejemplo de eliminar el usuario con un email especifico --
DELETE FROM usuarios WHERE email="luciohdz3012@gmail.com";

-- EJERCICIOS DELETE --

-- eliminar usuarios cuyo email contenga 1 o mas 5
DELETE FROM usuarios WHERE email LIKE '%5%';
```

```sql
-- Eliminar usuarios que tengan una contraseña que contenga letras mayusculas usando expresiones regulares
DELETE FROM usuarios WHERE password REGEXP '[A-Z]';

-- Eliminar usuarios con contraseñas que contengan solo numeros usa expresion regular
DELETE FROM usuarios WHERE password  REGEXP '^[0-9]+$';

--  Eliminar usuarios con correos que no tengan el dominio gmail
DELETE FROM usuarios WHERE email     'gmail';
```

V. Conclusiones:

La implementación de las operaciones CRUD es fundamental para la gestión eficiente de bases de datos en aplicaciones modernas. A lo largo de la práctica, se demostró cómo cada una de estas operaciones permite la interacción con los datos de manera estructurada y organizada. Crear registros, leer información, actualizar datos existentes y eliminar entradas son acciones clave para el ciclo de vida de la información. Además, se evidenció la importancia de validar los datos y asegurar que las operaciones sean realizadas de manera segura y precisa, minimizando el riesgo de errores y asegurando la integridad de la base de datos. En resumen, el dominio de CRUD es esencial para el desarrollo de aplicaciones sólidas y escalables.