Nombre de la práctica	Operaciones CRUD			No.	3
Asignatura:	Taller de Bases de Datos	Carrera:	ISIC	Duración de la práctica (Hrs)	2hrs

Nombre: Jocelin Reyes Rodriguez
Grupo: 3501
I. Competencia(s) específica(s):

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

III. Material empleado: Computadora

IV. Desarrollo de la práctica:

operacionesCRUD.md > # Operaciones CRUD en MySQL

```
# Operaciones CRUD en MySQL

## Create
La operacion *crear* es responsable de crear nueos datos en la base de datos en lenguaje SQL, esto se realiza con la sentencia `INSERT INTO` y en el caso de MySQL `INSERT` también funciona. El propósito de la operación es añadir el nuevo resgistro a una tabla
```sql

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

-- Ejemplo de una insersión valida usando el comando DEFAULT
INSERT INTO Usuarios VALUES (DEFAULT, "ejemplo2@gmail.com", "abcdefgh");

-- Ejemplo de una insersión sin incluir un ** id_usuario **
INSERT Usuario(email, password) VALUES ("email3@hotmail.com", "12345678");
```

Ejercicios
- Identifica los tipos de errores que pueden salir en esta tabla
```sql
    -- Violación de la unicidad del email:
-- Asumiendo que 'usuario1@mail.com' ya existe en la tabla
INSERT INTO Usuarios (email, password)
VALUES ('usuario1@mail.com', 'nuevaPassword');

    -- Violación de la longitud mínima del password
INSERT INTO Usuarios (email, password)
VALUES ('usuario5@mail.com', 'short');

    -- Violación del formato del email
INSERT INTO Usuarios (email, password)
VALUES ('usuario6mail.com', 'validpassword');

    -- Inserción de valores NULL en campos NOT NULL
INSERT INTO Usuarios (email, password)
VALUES (NULL, 'validpassword');


```
- Inserta 4 registros nuevos en un solo INSERT
```sql
INSERT INTO Usuarios (email, password)
```

Create

La operacion *crear* es responsable de crear nueos datos en la base de datos en lenguaje SQL, esto se realiza con la sentencia `INSERT INTO` y en el caso de MySQL `INSERT` también funciona. El propósito de la operación es añadir el nuevo resgistro a una tabla

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

-- Ejemplo de una insersión valida usando el comando DEFAULT
INSERT INTO Usuarios VALUES (DEFAULT, "ejemplo2@gmail.com", "abcdefgh");

-- Ejemplo de una insersión sin incluir un ** id_usuario **
INSERT Usuario(email, password) VALUES ("email3@hotmail.com", "12345678");
```

Ejercicios

- Identifica los tipos de errores que pueden salir en esta tabla

```sql
    -- Violación de la unicidad del email:
-- Asumiendo que 'usuario1@mail.com' ya existe en la tabla
INSERT INTO Usuarios (email, password)
VALUES ('usuario1@mail.com', 'nuevaPassword');

    -- Violación de la longitud mínima del password
INSERT INTO Usuarios (email, password)
VALUES ('usuario5@mail.com', 'short');

    -- Violación del formato del email
INSERT INTO Usuarios (email, password)
VALUES ('usuario6mail.com', 'validpassword');

    -- Inserción de valores NULL en campos NOT NULL
INSERT INTO Usuarios (email, password)
VALUES (NULL, 'validpassword');
```
```

```
### Ejercicios
 ```
 - Inserta 4 registros nuevos en un solo INSERT
 ```sql
 INSERT INTO Usuarios (email, password)
 VALUES
     ('usuario1@mail.com', 'contraseña1'),
     ('usuario2@mail.com', 'password1234'),
     ('usuario3@mail.com', 'abcdefghi'),
     ('usuario4@mail.com', 'mypassword');

 ```

 ## Read
 La operacion *leer* es utilizada para
 consultar o recuperar datos de la base de
 datos. Esto no modifica los datos, simplemente
 los extrae. En MySQL est operación se realiza
 con la sentencia select

 ```sql

 -- Ejmeplo de una consulta para todos lo datos
 de una tabla
 SELECT * FROM Usuarios;

  -- Ejemplo de consulta para un registro en
  especifico a través del id_usuario
 SELECT * FROM Usuarios WHERE id_usuario=1;
```

- Inserta 4 registros nuevos en un solo INSERT

```sql
INSERT INTO Usuarios (email, password)
VALUES
    ('usuario1@mail.com', 'contraseña1'),
    ('usuario2@mail.com', 'password1234'),
    ('usuario3@mail.com', 'abcdefghi'),
    ('usuario4@mail.com', 'mypassword');
```

Read

La operacion *leer* es utilizada para consultar o recuperar datos de la base de datos. Esto no modifica los datos, simplemente los extrae. En MySQL est operación se realiza con la sentencia select

```sql
-- Ejmeplo de una consulta para todos lo datos de una tabla
SELECT * FROM Usuarios;

 -- Ejemplo de consulta para un registro en especifico a través del id_usuario
SELECT * FROM Usuarios WHERE id_usuario=1;
```

```
 ```sql

 -- Ejmeplo de una consulta para todos lo datos
 de una tabla
 SELECT * FROM Usuarios;

  -- Ejemplo de consulta para un registro en
  especifico a través del id_usuario
 SELECT * FROM Usuarios WHERE id_usuario=1;

 -- Ejemplo de consulta para un rgeistro con un
 email en específico
 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
 lógico
 SELECT * FROM Usuarios WHERE LENGTH(password)
 >9;

 ```
 ### Ejercicio
 - Realiza una consulta que solo muestre el id,
 pero que coincida con una constraseña de mas
 de 8 caracteres
 ```sql
```

```sql
-- Ejmeplo de una consulta para todos lo datos de una tabla
SELECT * FROM Usuarios;

 -- Ejemplo de consulta para un registro en especifico a través del id_usuario
SELECT * FROM Usuarios WHERE id_usuario=1;

-- Ejemplo de consulta para un rgeistro con un email en específico
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 lógico
SELECT * FROM Usuarios WHERE
LENGTH(password)>9;
```

Ejercicio

- Realiza una consulta que solo muestre el id, pero que coincida con una constraseña de mas de 8 caracteres
```

```
## Read

### Ejercicio
- Realiza una consulta que solo muestre el id, pero que coincida con una constraseña de mas de 8 caracteres
```sql
SELECT id_usuario
FROM Usuarios
WHERE LENGTH(password) > 8;
```
 - Otra que realice una consulta a los id´s pares
 ```sql
SELECT id_usuario
FROM Usuarios
WHERE id_usuario % 2 = 0; -- o MOD

 ```

Update
La operación *actualizar* se utiliza para modificar registros existentes en la base de datos. Esto se hace con la sentencia `UPDATE`
```sql
-- Ejmeplo para actualizar la contraseña por su id
UPDATE Usuario SET password = "a1b2c3d4" WHERE id_usuario = 1;

-- Ejemplo para actualizar el email y password
```

Ejercicio

- Realiza una consulta que solo muestre el id, pero que coincida con una constraseña de mas de 8 caracteres

```sql
SELECT id_usuario
FROM Usuarios
WHERE LENGTH(password) > 8;
```

- Otra que realice una consulta a los id´s pares

```sql
SELECT id_usuario
FROM Usuarios
WHERE id_usuario % 2 = 0; -- o MOD
```

Update

La operación *actualizar* se utiliza para modificar registros existentes en la base de datos. Esto se hace con la sentencia `UPDATE`

```sql
-- Ejmeplo para actualizar la contraseña por su id
```

```
## Update
La operación *actualizar* se utiliza para modificar registros existentes en la base de datos. Esto se hace con la sentencia `UPDATE`
```sql
-- Ejmeplo para actualizar la contraseña por su id
UPDATE Usuario SET password = "a1b2c3d4" WHERE id_usuario = 1;

-- Ejemplo para actualizar el email y password de un usaurio en específico
UPDATE Usuario SET password = "a1b2c3d4", email = "luciohdz3012@gmail.com" WHERE id_usuario = 1;
```
Ejercicio
- Intenta actualizar registros con valores que violen las restricciones (minimo 3)
```sql
    -- Actualización que viola la unicidad del email:
-- Asumiendo que los emails 'usuario1@mail.com' y 'usuario2@mail.com' ya existen
UPDATE Usuarios
SET email = 'usuario2@mail.com'
WHERE id_usuario = 1;

    -- Actualización que viola la longitud mínima del password
```

La operación *actualizar* se utiliza para modificar registros existentes en la base de datos. Esto se hace con la sentencia `UPDATE`

```sql
-- Ejmeplo para actualizar la contraseña por su id
UPDATE Usuario SET password = "a1b2c3d4" WHERE id_usuario = 1;

-- Ejemplo para actualizar el email y password de un usaurio en específico
UPDATE Usuario SET password = "a1b2c3d4", email = "luciohdz3012@gmail.com" WHERE id_usuario = 1;
```

Ejercicio

- Intenta actualizar registros con valores que violen las restricciones (minimo 3)

```sql
    -- Actualización que viola la unicidad del email:
-- Asumiendo que los emails 'usuario1@mail.com' y 'usuario2@mail.com' ya existen
UPDATE Usuarios
```

```
-- Asumiendo que los emails 'usuario1@mail.
com' y 'usuario2@mail.com' ya existen
UPDATE Usuarios
SET email = 'usuario2@mail.com'
WHERE id_usuario = 1;

    -- Actualización que viola la longitud
    mínima del password
UPDATE Usuarios
SET password = 'short'
WHERE id_usuario = 2;

    -- Actualización que viola el formato del
    email:
UPDATE Usuarios
SET email = 'invalidemail.com'
WHERE id_usuario = 3;


```

```
## Delete
La operacón *elimianr* se usa para borrar
registros de la base de datos. Esto se realiza
con la sentEncia `DELETE`.  **Debemos ser muy
cuidadosos con esta operación, ya que una vez
que los datos son eliminados, NO pueden ser
RECUPERADOS**
```

```
ya existen
UPDATE Usuarios
SET email = 'usuario2@mail.com'
WHERE id_usuario = 1;

    -- Actualización que viola la longitud
mínima del password
UPDATE Usuarios
SET password = 'short'
WHERE id_usuario = 2;

    -- Actualización que viola el formato
del email:
UPDATE Usuarios
SET email = 'invalidemail.com'
WHERE id_usuario = 3;
```

Delete

La operacón *elimianr* se usa para borrar registros de la base de datos. Esto se realiza con la sentEncia `DELETE`. **Debemos ser muy cuidadosos con esta operación, ya que una vez que los datos son eliminados, NO pueden**

```
registros de la base de datos. Esto se realiza
con la sentEncia `DELETE`.  **Debemos ser muy
cuidadosos con esta operación, ya que una vez
que los datos son eliminados, NO pueden ser
RECUPERADOS**

```sql
-- Eliminar el usuario por el id
DELETE FROM Usuarios WHERE id_usuario = 4;
-- Eliminar los usuarios con el email
específico
DELETE FROM Usuarios WHERE
email="luciohdz3012@gmail.com" ;
```
Eercicios
- Eliminar usuarios cuyo email contenga 1 o
mas "5"
- Eliminar usuarios que tengan una contraseña
que contengan letras mayusculas usando
expresiones regulares (REGEX)
- Elminar usuarios con contraseña que
contengan solo numeros
- Eliminar usuarios con correos que no
contengan el dominio "gmail"

```sql
 -- Eliminar usuarios cuyo email contenga uno
 o más "5"
 DELETE FROM Usuarios
```
```

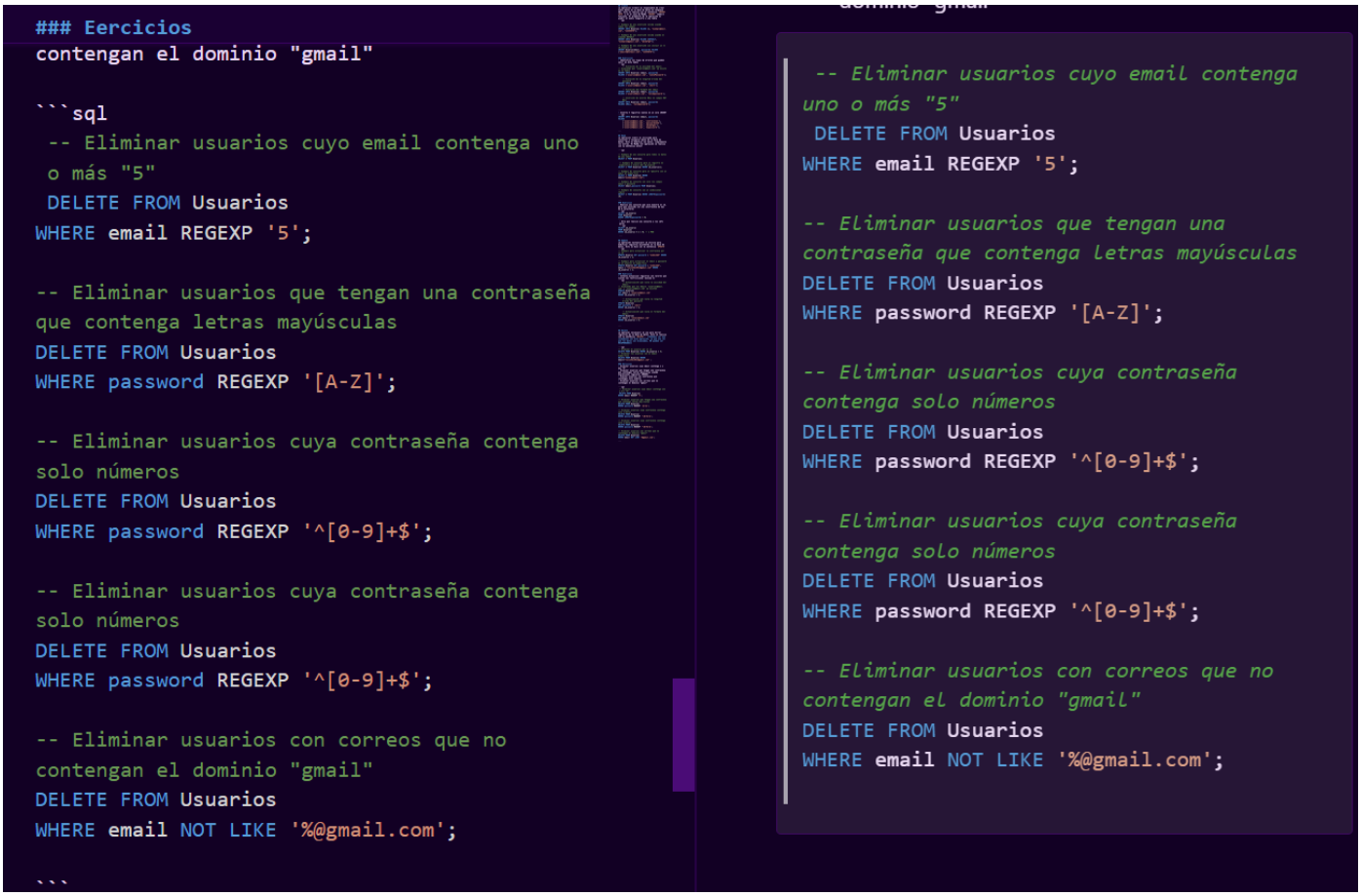
La operación *eliminar* se usa para borrar registros de la base de datos. Esto se realiza con la sentEncia `DELETE`. **Debemos ser muy cuidadosos con esta operación, ya que una vez que los datos son eliminados, NO pueden ser RECUPERADOS**

```sql
-- Eliminar el usuario por el id
DELETE FROM Usuarios WHERE id_usuario = 4;
-- Eliminar los usuarios con el email
específico
DELETE FROM Usuarios WHERE
email="luciohdz3012@gmail.com" ;
```

Eercicios

- Eliminar usuarios cuyo email contenga 1 o mas "5"
- Eliminar usuarios que tengan una contraseña que contengan letras mayusculas usando expresiones regulares (REGEX)
- Elminar usuarios con contraseña que contengan solo numeros
- Eliminar usuarios con correos que no contengan el dominio "gmail"

```sql
 -- Eliminar usuarios cuyo email contenga
uno o más "5"
```



```
### Eercicios
contengan el dominio "gmail"

```sql
 -- Eliminar usuarios cuyo email contenga uno
 o más "5"
 DELETE FROM Usuarios
WHERE email REGEXP '5';

-- Eliminar usuarios que tengan una contraseña
que contenga letras mayúsculas
DELETE FROM Usuarios
WHERE password REGEXP '[A-Z]';

-- Eliminar usuarios cuya contraseña contenga
solo números
DELETE FROM Usuarios
WHERE password REGEXP '^[0-9]+$';

-- Eliminar usuarios cuya contraseña contenga
solo números
DELETE FROM Usuarios
WHERE password REGEXP '^[0-9]+$';

-- Eliminar usuarios con correos que no
contengan el dominio "gmail"
DELETE FROM Usuarios
WHERE email NOT LIKE '%@gmail.com';

```
```

```
 -- Eliminar usuarios cuyo email contenga
uno o más "5"
 DELETE FROM Usuarios
WHERE email REGEXP '5';

-- Eliminar usuarios que tengan una
contraseña que contenga letras mayúsculas
DELETE FROM Usuarios
WHERE password REGEXP '[A-Z]';

-- Eliminar usuarios cuya contraseña
contenga solo números
DELETE FROM Usuarios
WHERE password REGEXP '^[0-9]+$';

-- Eliminar usuarios cuya contraseña
contenga solo números
DELETE FROM Usuarios
WHERE password REGEXP '^[0-9]+$';

-- Eliminar usuarios con correos que no
contengan el dominio "gmail"
DELETE FROM Usuarios
WHERE email NOT LIKE '%@gmail.com';
```

V. Conclusiones:

Las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) son fundamentales en la gestión de bases de datos SQL. Las operaciones CRUD permiten estructurar la interacción con la base de datos de manera clara y organizada, facilitando el desarrollo y mantenimiento de aplicaciones.

Al realizar estas operaciones, es crucial aplicar restricciones y validaciones para mantener la integridad de los datos. Esto incluye el uso de claves primarias y foráneas, así como restricciones de unicidad y no nulidad.Las operaciones de lectura pueden optimizarse mediante el uso de índices, lo que mejora el rendimiento de las consultas y hace que la recuperación de datos sea más eficiente.

En resumen, las operaciones CRUD son la base para interactuar con bases de datos SQL, y su correcta implementación y gestión son cruciales para el éxito de cualquier aplicación que dependa de la persistencia de datos.