



Evidencia de desempeño: GA6-220501096-AA2-EV01 destrezas y conocimientos en el manejo de sentencias DDL y DML de SQL (Prácticas de creación de base de datos y consultas) - scripts de base de datos

Integrantes:

Jenny Maria Meza Forero

Ficha: 2834921

Gustavo Adolfo Rodríguez Quinayas

Instructor

Análisis y Desarrollo de Software

Centro de Formación Comercio y Servicios

Servicio Nacional de Aprendizaje SENA

2024



Tabla de contenido

Introducción	3
Objetivo.....	3
• Creación de la Tabla "libreta"	3
• Visualice las tablas existentes para verificar la creación de "libreta"	4
• Visualice la estructura de la tabla "libreta".	5
• Ingrese los siguientes registros: ('Alberto Mores','Colon 123','4234567'); ('Juan Torres','Avellaneda 135','4458787');	6
• Seleccione y muestre todos los registros de la tabla. SELECT *	7
• Construya las sentencias que actualicen los datos que acaba de insertar.	8
• Insertar 5 registros más.	9
• Cuente cuántos registros se ingresan.....	9
Conclusiones	11



Introducción

El presente informe tiene como objetivo documentar de manera detallada el proceso llevado a cabo para resolver los problemas relacionados con la manipulación de datos en una base de datos relacional. En un entorno donde la gestión eficiente de la información es crucial, se ha decidido centrar el análisis en la tabla "Libreta". A través de esta investigación, se abordarán las diversas dificultades encontradas durante el manejo de los datos, así como las soluciones implementadas para superarlas.

Para ello, se han utilizado una serie de sentencias SQL que permiten no solo la creación de la tabla, sino también la inserción de datos, la actualización de registros y la visualización de información relevante. Cada uno de estos pasos será analizado en profundidad, destacando las mejores prácticas y las estrategias adoptadas para asegurar la integridad y accesibilidad de los datos. Además, se reflexionará sobre la importancia de un diseño adecuado en las bases de datos relacionales, así como su impacto en la eficiencia de los procesos de manejo de datos.

A lo largo del informe, se espera proporcionar una visión integral que no solo exponga los resultados obtenidos, sino que también sirva como referencia para futuras implementaciones y mejoras en la gestión de bases de datos.

Objetivo:

Documentar el proceso de trabajo y las sentencias SQL utilizadas para abordar los problemas planteados en relación con la manipulación de datos en una base de datos relacional. Se empleó SQLite3 DB Browser como herramienta para ejecutar las sentencias SQL y resolver los desafíos propuestos.

- **Creación de la Tabla "libreta"**

Para llevar a cabo la creación de la tabla "libreta", que contendrá los campos necesarios para almacenar la información de contacto, se ejecutó la siguiente sentencia SQL. Esta tabla incluye tres columnas específicas: "nombre", "domicilio" y "teléfono", cada una diseñada para almacenar datos de diferentes tipos y longitudes.

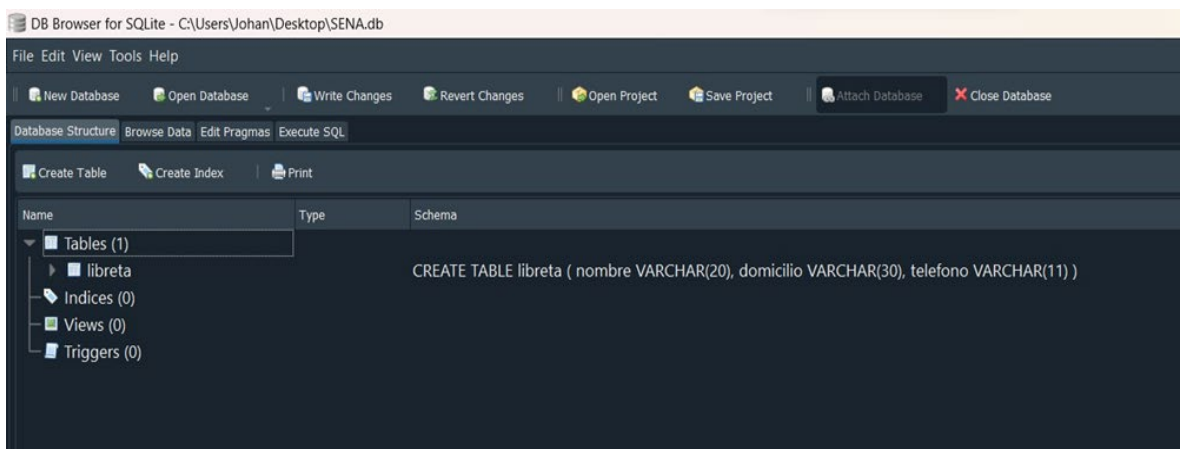
La sentencia SQL utilizada es la siguiente:



```
CREATE TABLE libreta (  
  
    nombre VARCHAR(20),  
  
    domicilio VARCHAR(30),  
  
    teléfono VARCHAR(11)  
  
);
```

En esta declaración, se ha definido el campo "nombre" como un tipo de dato VARCHAR con una

- **Visualice las tablas existentes para verificar la creación de "libreta"**



En esta declaración SQL, se ha definido el campo "nombre" como un tipo de dato VARCHAR con una longitud máxima de 20 caracteres. Esta elección permite almacenar nombres que, en su mayoría, son relativamente breves, como nombres propios o apellidos. Sin embargo, el límite de 20 caracteres también proporciona una flexibilidad adecuada para incluir nombres compuestos o nombres de dos partes, lo que es común en muchas culturas. Al optar por un tipo VARCHAR, se garantiza que el almacenamiento sea eficiente, ya que este tipo de dato se adapta dinámicamente a la longitud real de los nombres ingresados, evitando el desperdicio de espacio que ocurriría con un tipo de dato de longitud fija.

El campo "domicilio", por su parte, también se ha definido como VARCHAR, pero con una longitud mayor de 30 caracteres. Este tamaño se ha establecido con la intención de proporcionar suficiente espacio para incluir direcciones completas, que a menudo requieren más caracteres debido a elementos como números de casa, nombres de calles, barrios y, en algunos casos, detalles adicionales como referencias o puntos de interés cercanos. Esta flexibilidad es fundamental para asegurar que los usuarios puedan registrar información de manera precisa y sin restricciones innecesarias.

Por último, el campo "teléfono" ha sido configurado como VARCHAR con un límite de 11 caracteres. Esta longitud se considera apropiada para almacenar números telefónicos que pueden incluir códigos de área, así como diferentes formatos de presentación, como guiones o espacios. La elección de utilizar VARCHAR en lugar de un tipo de dato numérico también permite la inclusión de caracteres



no numéricos, lo que puede ser útil para los números de teléfono internacionales o para aquellos que incluyen un prefijo especial.

En conjunto, esta estructura asegura que la tabla "libreta" sea capaz de manejar la información de contacto de manera organizada y accesible, facilitando la entrada y recuperación de datos en un formato que se alinea con las necesidades prácticas de los usuarios. De esta forma, se optimiza la gestión de la información, garantizando que se puedan almacenar y consultar los datos de contacto de manera eficiente y efectiva.

- **Visualice la estructura de la tabla "libreta".**

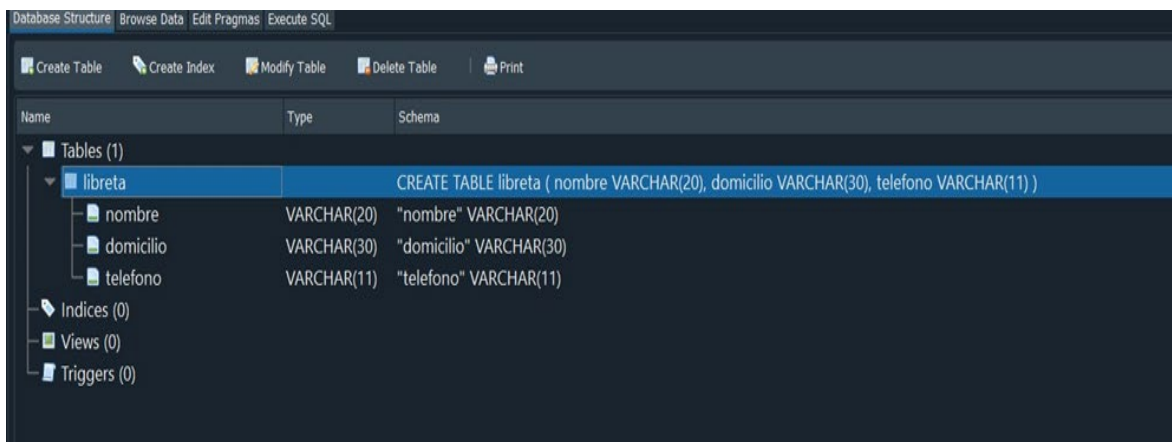
Para examinar la organización de la tabla "libreta", se utilizó el comando DESCRIBE libreta;. Este comando proporciona un resumen de los campos de la tabla, mostrando los tipos de datos y sus longitudes.

La estructura de la tabla es la siguiente:

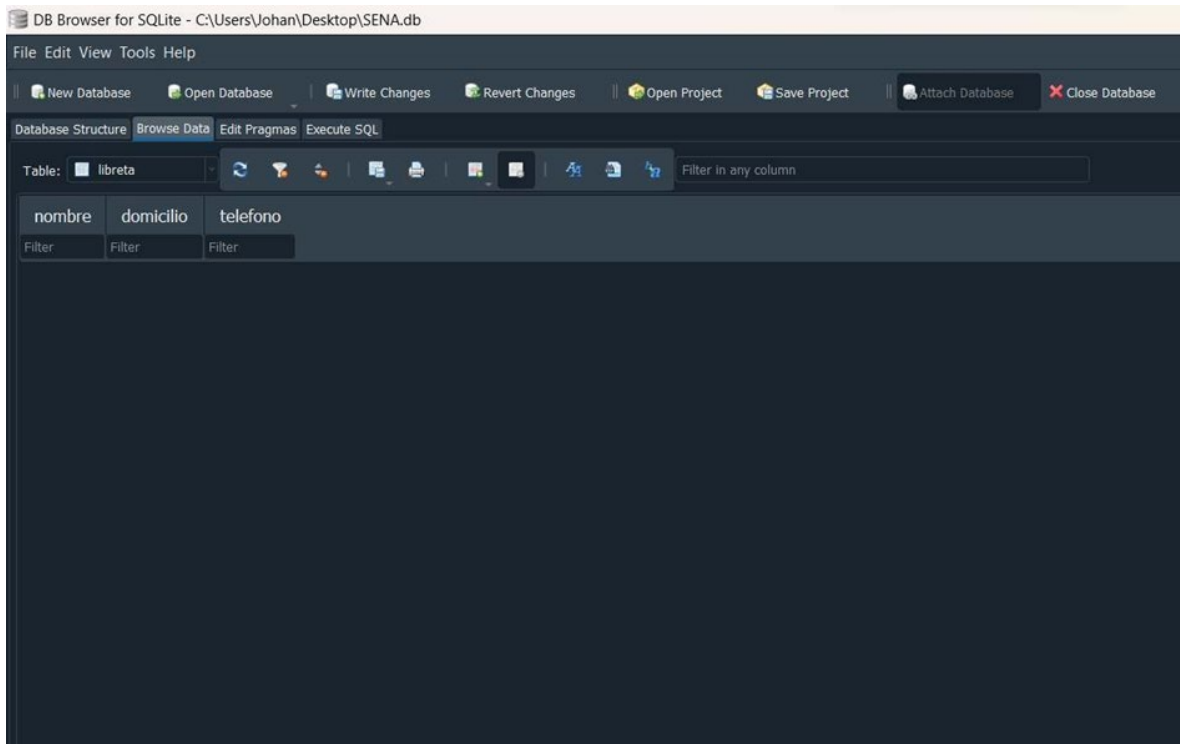
nombre: VARCHAR(20) – Almacena nombres de hasta 20 caracteres.

domicilio: VARCHAR(30) – Permite registrar direcciones de hasta 30 caracteres.

teléfono: VARCHAR(11) – Capaz de almacenar números de teléfono de hasta 11 caracteres.



Name	Type	Schema
Tables (1)		
libreta	CREATE TABLE libreta (nombre VARCHAR(20), domicilio VARCHAR(30), telefono VARCHAR(11))	
nombre	VARCHAR(20)	"nombre" VARCHAR(20)
domicilio	VARCHAR(30)	"domicilio" VARCHAR(30)
telefono	VARCHAR(11)	"telefono" VARCHAR(11)
Indices (0)		
Views (0)		
Triggers (0)		



- **Ingrese los siguientes registros: ('Alberto Mores','Colon 123','4234567'); ('Juan Torres','Avellaneda 135','4458787');**

Para completar la tabla "libreta" con información de contacto, se ejecutaron las siguientes sentencias SQL para insertar dos registros:

```
INSERT INTO libreta (nombre, domicilio, teléfono) VALUES ('Alberto Mores', 'Colon 123', '4234567');
```

```
INSERT INTO libreta (nombre, domicilio, teléfono) VALUES ('Juan Torres', 'Avellaneda 135', '4458787');
```

Estas sentencias SQL permiten insertar dos nuevos registros en la tabla "libreta", almacenando de manera precisa los datos de contacto de Alberto Mores y Juan Torres. Cada instrucción INSERT INTO especifica el nombre de la tabla y los campos correspondientes en los que se guardarán los datos: "nombre", "domicilio" y "teléfono".

Al ejecutar estos comandos, la tabla "libreta" se actualiza con la información ingresada, lo que significa que ahora cuenta con registros específicos que pueden ser fácilmente recuperados y manipulados en el futuro. Esta actualización no solo facilita la gestión de datos, sino que también asegura que la información esté organizada y accesible para realizar consultas posteriores, como buscar contactos, modificar registros existentes o eliminar datos innecesarios.



The screenshot shows a SQL IDE interface with a menu bar (New Database, Open Database, Write Changes, Revert Changes, Open Project, Save Project, Attach Database) and a toolbar. The 'Execute SQL' tab is active. The SQL editor contains the following code:

```
1 INSERT INTO libreta (nombre, domicilio, telefono)
2 VALUES ('Alberto Mores', 'Colon 123', '4234567'),
3         ('Juan Torres', 'Avellaneda 135', '4458787');
4
```

Below the editor, the execution results are displayed:

Execution finished without errors.
Result: query executed successfully. Took 3ms, 2 rows affected
At line 1:
INSERT INTO libreta (nombre, domicilio, telefono)
VALUES ('Alberto Mores', 'Colon 123', '4234567'),
('Juan Torres', 'Avellaneda 135', '4458787');

- Seleccione y muestre todos los registros de la tabla. **SELECT ***

The screenshot shows the same SQL IDE interface. The SQL editor contains the following code:

```
1 SELECT * FROM libreta;
2
```

Below the editor, the 'Browse Data' tab is active, showing the results of the query for the 'libreta' table. The results are displayed in a table with columns 'nombre', 'domicilio', and 'telefono'.

	nombre	domicilio	telefono
1	Alberto ...	Colon 123	4234567
2	Juan ...	Avellaneda...	4458787



Estas sentencias permiten insertar dos nuevos registros en la tabla, almacenando los datos de contacto de Alberto Mores y Juan Torres. Al ejecutar estos comandos, la tabla "libreta" se actualiza con la información correspondiente, facilitando futuras consultas y gestiones de datos.

FROM libreta;

- **Construya las sentencias que actualicen los datos que acaba de insertar.**

Para actualizar los datos recién insertados en la tabla "libreta", se construyeron las siguientes sentencias de actualización:

-- Actualizar el domicilio de Alberto Mores UPDATE libreta

SET domicilio = 'Calle Nueva 456' WHERE nombre = 'Alberto Mores';

-- Actualizar el teléfono de Juan Torres UPDATE libreta

SET telefono = '5555555' WHERE nombre = 'Juan Torres';

```
1  -- Actualizar el domicilio de Alberto Mores
2  UPDATE libreta
3  SET domicilio = 'Calle Nueva 456'
4  WHERE nombre = 'Alberto Mores';
5
6  -- Actualizar el teléfono de Juan Torres
7  UPDATE libreta
8  SET telefono = '5555555'
9  WHERE nombre = 'Juan Torres';
10
11
```

	nombre	domicilio	telefono
	Filter	Filter	Filter
1	Alberto ...	Calle Nue...	4234567
2	Juan ...	Avellaned...	5555555



- Insertar 5 registros más.

```
INSERT INTO libreta (nombre, domicilio, telefono) VALUES ('Carlos López', 'Bogotá123',  
'1234567'),  
('María García', 'Medellín246', '9876543'), ('Pedro Martínez', 'Juárez 369', '4561238'),  
('Ana Rodríguez', 'Cali 852', '7894561'),  
('Luis González', 'Cartagena 753', '3216549');
```

The screenshot shows the SQL Developer interface with the 'Execute SQL' tab selected. The SQL editor contains the following statement:

```
1 INSERT INTO libreta (nombre, domicilio, telefono) VALUES  
2 ('Carlos López', 'Bogotá123', '1234567'),  
3 ('María García', 'Medellín246', '9876543'),  
4 ('Pedro Martínez', 'Juárez 369', '4561238'),  
5 ('Ana Rodríguez', 'Cali852', '7894561'),  
6 ('Luis González', 'Cartagena753', '3216549');  
7  
8  
9
```

The screenshot shows the 'libreta' table in the 'Browse Data' tab. The table has three columns: nombre, domicilio, and telefono. The data is as follows:

	nombre	domicilio	telefono
	Filter	Filter	Filter
1	Alberto ...	Calle Nue...	4234567
2	Juan ...	Avellaned...	5555555
3	Carlos ...	Bogotá123	1234567
4	María ...	Medellín246	9876543
5	Pedro ...	Juárez 369	4561238
6	Ana ...	Cali852	7894561
7	Luis ...	Cartagen...	3216549

- Cuento cuántos registros se ingresan

Para contar cuántos registros se han ingresado en la tabla "libreta", se utilizó la siguiente sentencia:



- **SELECT COUNT(*) AS total_registros FROM libreta;**

The screenshot shows a SQL IDE window titled 'SQL 1'. The query editor contains the following SQL statement:

```
1 SELECT COUNT(*) AS total_registros FROM libreta;
```

Below the query editor, the results pane displays a single row with the column header 'total_registros' and the value '7'.

	total_registros
1	7

El comando SQL `SELECT COUNT(*) AS total_registros FROM libreta;` se utiliza para contar el número total de registros en la tabla "libreta". Al ejecutar esta consulta, se obtuvo un total de 7 registros.

Desglose del comando:

`SELECT COUNT(*)`: Indica que se cuentan todas las filas en la tabla.

`AS total_registros`: Asigna el alias "total_registros" al resultado del conteo.

`FROM libreta`: Especifica que el conteo se realiza en la tabla "libreta".

Resultados y su significado

Con un total de 7 registros, este resultado permite:

Monitorear el crecimiento de datos: Proporciona una referencia sobre el volumen actual de información.

Validar inserciones: Confirma que los registros se han agregado correctamente.

Planificar operaciones futuras: Facilita la gestión de consultas y el rendimiento de la base de datos.

El conteo de 7 registros en la tabla "libreta" ofrece una visión clara del estado actual de los datos, ayudando en la gestión y en la toma de decisiones para futuras operaciones.



Conclusiones

- 1) Este informe ha evidenciado que el uso de sentencias SQL es una estrategia altamente eficaz para interactuar con bases de datos relacionales, específicamente con SQLite3. Estas sentencias ofrecen un método preciso y eficiente para llevar a cabo diversas tareas, como la creación de tablas, la inserción de datos, la actualización de registros y la realización de consultas de información. La implementación de SQL no solo optimiza la gestión de datos, sino que también asegura la integridad y la accesibilidad de la información almacenada.
- 2) Asimismo, la herramienta SQLite3 DB Browser se ha destacado por su facilidad de uso y su interfaz intuitiva. Esta aplicación permite ejecutar sentencias SQL de manera sencilla, al mismo tiempo que proporciona una visualización clara y organizada de los resultados obtenidos. Esta combinación de características facilita significativamente el proceso de desarrollo y la depuración de consultas, haciendo que el manejo de bases de datos sea más accesible, incluso para aquellos que se inician en el ámbito de la programación SQL.