Creación de bases de datos en SQLite

1. Características de SQLite

SQLite es un sistema gestor de bases de datos relacionales que está liberado bajo una **licencia de dominio público**, esto quiere decir que no tiene derechos de autor y que no existe ninguna restricción a la hora de utilizarlo.

Algunas de las características que hacen que SQLite sea un proyecto muy interesante son las siguientes:

•**Serverless**. **No necesita** una arquitectura cliente/servidor para funcionar. Tampoco necesita de un proceso específico para ejecutarse como un servicio.

•**Single file database**. Cada base de datos se almacena en un único archivo.

•**Zero Configuration**. Al no ser necesario un servidor, no es necesario realizar ninguna configuración adicional. Crear una instancia de una base de datos SQLite es tan sencillo como crear un archivo.

•**Cross-Platform**. El archivo que contiene la base de datos puede ser utilizado en cualquier plataforma (Linux, Windows, macOS).

•**Self-Contained**. La biblioteca SQLite contiene todo el sistema gestor de bases de datos, de modo que se puede integrar fácilmente con la aplicación que haga uso de este sistema.

•**Small Runtime Footprint**. El ejecutable de SQLite ocupa menos de 1 MByte y necesita pocos Megabytes de memoria para ejecutarse.

•**Transactional**. Permite transacciones ACID y permite el acceso seguro a la base de datos desde múltiples procesos y hilos.

•**Full-Featured**. Tiene soporte para la mayoría de características del estándar SQL92 (SQL2).

2 Usos de SQLite

•SQLite está diseñado para **manipular datos en entornos pequeños**, donde la facilidad de uso es más importante que la capacidad y la concurrencia.

•Es muy útil para **aplicaciones de escritorio y aplicaciones móviles**. Por ejemplo, no sería adecuado instalar MySQL en una aplicación de escritorio o en una aplicación para dispositivos móviles donde lo único que se quiere almacenar es una agenda de contactos.

•Es ideal para **sistemas embebidos y aplicaciones relacionadas con Internet of Things (IoT)**. Es una buena opción para usar en aplicaciones móviles, televisiones, consolas, cámaras, relojes, drones, etc.

•También es posible utilizar SQLite para **sitios web que no tengan un tráfico elevado**. La web oficial del proyecto (https://www.sqlite.org) utiliza una base de datos en SQLite.

•SQLite nos permite crear **bases de datos que son almacenadas exclusivamente en memoria**. Esta característica es útil para crear bases de datos temporales con poca información que no requieren un almacenamiento persistente.

•También son usadas con **fines educativos**, debido a que no es necesario realizar ninguna configuración especial para trabajar con SQLite y el proceso de instalación es muy sencillo.

Importante:

Se recomienda leer la sección Appropiate uses for SQLite en la web oficial del proyecto, para conocer en qué situaciones es apropiado el uso de SQLite.

3 Instalación de `sqlite3`

La utilidad `sqlite3`, que es una aplicación de línea de comandos que nos permite interactuar con bases de datos SQLite.

3.1 Ubuntu

Para instalar `sqlite3` en Ubuntu haremos lo siguiente:

```
sudo apt-get update
sudo apt-get install sqlite3
```

Una vez instalada la utilidad sólo tenemos que escribir `sqlite3` en un terminal para iniciarla. Deberíamos obtener una salida similar a la que se muestra a continuación.

```
$ sqlite3
SQLite version 3.22.0 2018-01-22 18:45:57
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

3.2 Otras plataformas

Para instalar `sqlite3` en otras plataformas sólo habrá que descargar el archivo binario de la web oficial.

4 Comandos de `sqlite3` (dot-commands)

Los `dot-commands` son comandos que nos permiten configurar la utilidad `sqlite3`. Estos comandos se diferencian de las sentencias SQL porque siempre empiezan con un punto. Además los `dot-commands` no terminan en punto y coma, sin embargo las sentencias SQL siempre tienen que terminar en punto y coma.
Algunos de los `dot-commands` que vamos a utilizar son los siguientes:

Comando	Descripción
.help	Muestra la lista completa de todos los `dot-commands` disponibles.
.quit	Se usa para salir de `sqlite3`.
.show	Muestra algunos de la configuración actual de `sqlite3`.
.databases	Muestra una tabla con todas las bases de datos adjuntas.
.tables	Muestra las tablas y vistas de las bases de datos `main` y `temp`.
.schema	Muestra el comando SQL que se ha utilizado para crear una tabla, vista, índice, etc.
.headers	Muestra/Oculta los nombres de las columnas. Por defecto está a `off`.
.mode	Configura cómo será formateada la salida de los datos. Por defecto está en `list`.
.dump	Genera los comandos SQL necesarios para recrear una o más tablas de la base de datos.

4.1 `.databases`

4.1.1 Uso

```
.databases
```

4.1.2 Descripción

Muestra una tabla con todas las bases de datos adjuntas. La tabla que se muestra tiene el siguiente formato:

Nombre de la columna	Tipo	Descripción
`seq`	Integer	Número de base de datos (0: main y 1: temp)
`name`	Text	Nombre lógico de la base de datos
`file`	Text	Ruta donde se encuentra el archivo de la base de datos

La primera base de datos será `main` y tiene como número de secuencia el 0. La segunda base de datos será `temp` y tendrá el número de secuenca 1. Esta base de datos es donde se crean los objetos temporales y no siempre aparecerá en el listado. Si existen más bases de datos adjuntas aparecerán a continuación.

4.2 `.tables`

4.2.1 Uso

```
.tables [table-pattern]
```

4.2.2 Descripción

Muestra las tablas y vistas de las bases de datos `main` y `temp`.

4.3 `.schema`

4.3.1 Uso

```
.schema [table-pattern]
```

4.3.2 Descripción

Muestra el comando SQL que se ha utilizado para crear una tabla, vista, índice, etc.

Si no especifica ningún parámetro se muestran los comandos SQL para crear cada uno de los objetos de las bases de datos `main` y `temp`.

4.4 `.headers`

4.4.1 Uso

```
.headers [on | off]
```

4.4.2 Descripción

Muestra/Oculta los nombres de las columnas. Por defecto está a `off`.

4.5 `.mode`

4.5.1 Uso

```
.mode (colum[s]|csv|html|insert|line[s]|list|tabs|tcl) [table-name]
```

4.6 Descripción

Configura cómo será formateada la salida de los datos. Por defecto está en `list`.
Las opciones que podemos escoger son las siguientes:

- **`column`**. La salida se formatea en forma de tabla con una fila por línea. El ancho de cada columna utilizado será el especificado con el comando `.width`.
- **`csv`**. La salida se formatea separada por comas, con una fila por línea.
- **`html`**. La salida es una tabla en formato HTML.
- **`insert`**. La salida es una lista de sentencias SQL (`INSERT`).

2 Manipulación de Bases de Datos

2.1 Crear una base de datos

2.1.1 Desde la utilidad `sqlite3`

Podemos crear una base de datos SQLite con la utilidad `sqlite3` desde la línea de comandos. Por ejemplo, para crear una base de datos con el nombre `agenda.db` ejecutaremos el siguiente comando.

```
sqlite3 agenda.db
```

Este comando creará un archivo donde se almacenará toda la información de nuestra base de datos.

El encoding que se utilizará por defecto al crear la base de datos será `utf8`.

2.1.2 `PRAGMA encoding;`

Las sentencias `PRAGMA` son una extensión de SQL que se han añadido de forma específica en SQLite y nos permiten modificar el funcionamiento de la librería SQLite.

La sentencia `PRAGMA encoding` nos permite consultar y modificar el encoding utilizado en la base de datos.

Ejemplo:

```
sqlite3> PRAGMA encoding;
UTF-8
```

Los diferentes tipos de encoding que podemos seleccionar son los siguientes:

- `PRAGMA encoding = "UTF-8";`
- `PRAGMA encoding = "UTF-16";`
- `PRAGMA encoding = "UTF-16le";`
- `PRAGMA encoding = "UTF-16be";`