Base de Datos Embebida

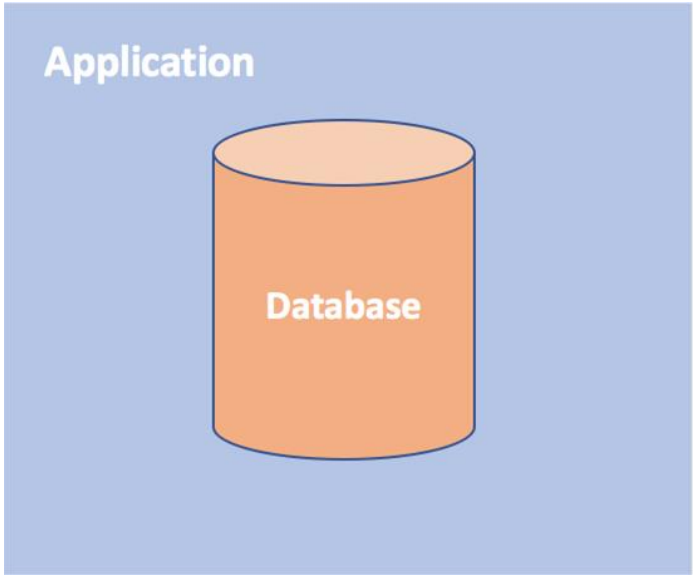
Laboratorio IV

Integrantes

Andrés Isaac Biso - Legajo 0125044

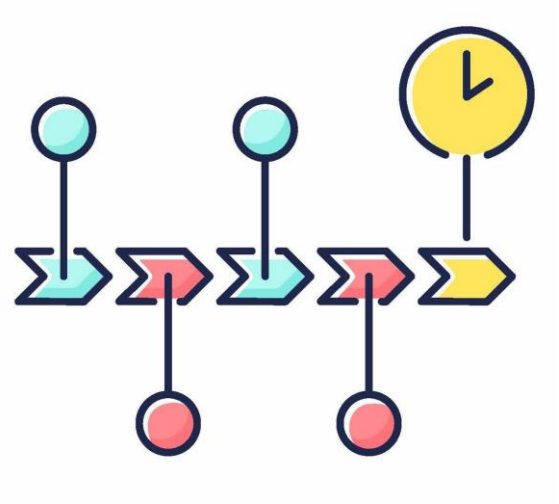
Matias Hernan Coria - Legajo 0127111

Introducción



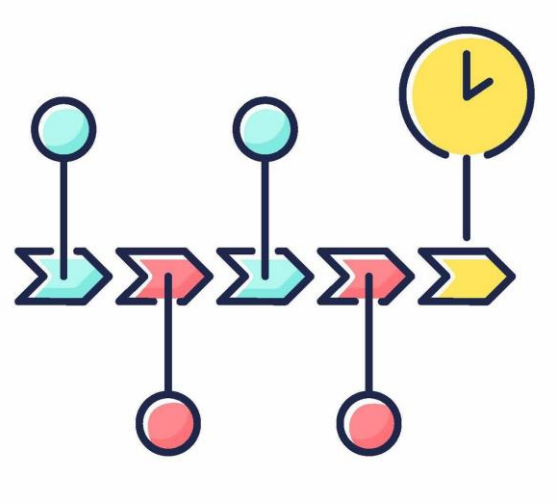
- Un sistema de base de datos embebido es un sistema de administración de base de datos (DBMS) que está estrechamente integrado con un software de aplicación; está incrustado en la aplicación.
- Son sistemas de bases de datos con diferentes interfaces de programación de aplicaciones (SQL y API nativas propietarias).

Historia



- Los sistemas de bases de datos embebidos han existido desde finales de los años 70.
- Podemos mencionar: Btrieve, C-tree, Empress, db_VISTA y dBASE.
- En el pasado, las bases de datos integradas se usaban para aplicaciones de línea de negocios de computación departamental, principalmente en PC y pequeños sistemas Unix. Estos sistemas eran silos: no compartían ni necesitaban compartir sus datos.

Historia



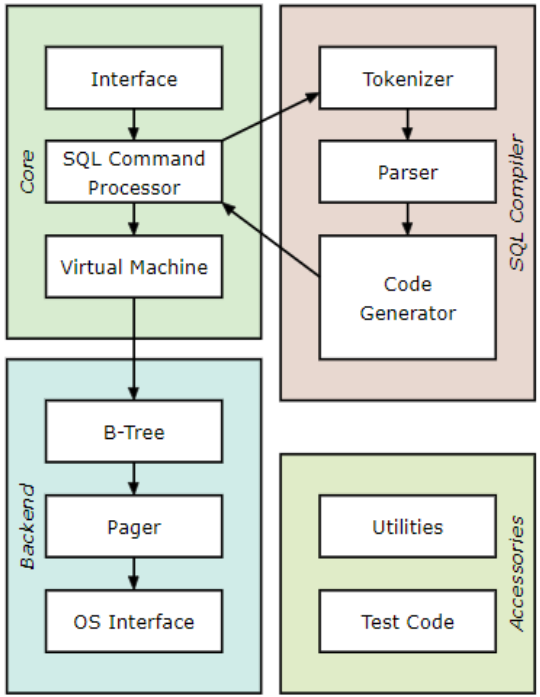
En la medida en que se compartieron los datos, era en forma de informes generados por el sistema. Por ejemplo, en un proveedor de sistemas de bases de datos, desarrollaron un sistema para rastrear el movimiento de motores de misiles nucleares (motores de cohetes), que se utilizó para cumplir con el tratado START. Era un sistema basado en PC que, podría generar un informe sobre cualquier motor específico: dónde está, dónde había estado, si se había gastado en una prueba, etc.

*Tratado START: https://en.wikipedia.org/wiki/START_I

Acerca de SQLite

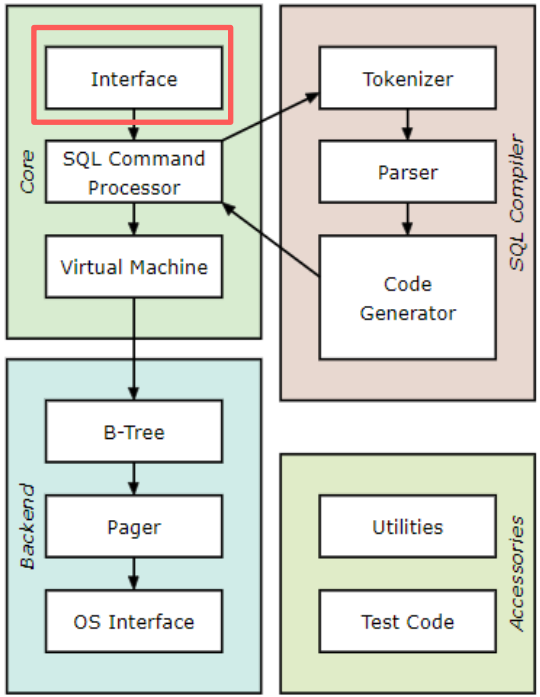
- La base de datos embebida elegida para el desarrollo de este informe es SQLite ya que es la base de datos embebida más popular del mercado.
- SQLite es una biblioteca y un motor de base de datos SQL embebido. A diferencia de la mayoría de las otras bases de datos SQL, SQLite no tiene un proceso de servidor separado. SQLite lee y escribe directamente en archivos de disco ordinarios. Una base de datos SQL completa con varias tablas, índices, disparadores y vistas está contenida en un solo archivo de disco. El formato de archivo de la base de datos es multiplataforma.

Descripción de la Arquitectura



- SQLite funciona compilando texto SQL en bytecode y luego ejecutando ese bytecode usando una máquina virtual.
- El mismo cuenta con las siguientes interfaces:
  - El sqlite3_prepare_v2() y las interfaces relacionadas actúan como un compilador para convertir texto SQL en bytecode.
  - El objeto sqlite3_stmt es un contenedor para un único programa en bytecode que implementa una sola instrucción SQL.
  - La interfaz sqlite3_step() pasa un programa en bytecode a la máquina virtual y ejecuta el programa hasta que este finaliza o forma una cola de resultados para devolver, se produce un error fatal o se interrumpe.
- La arquitectura se compone de cuatro módulos: Core, Backend, SQL Compiler y Accessories.

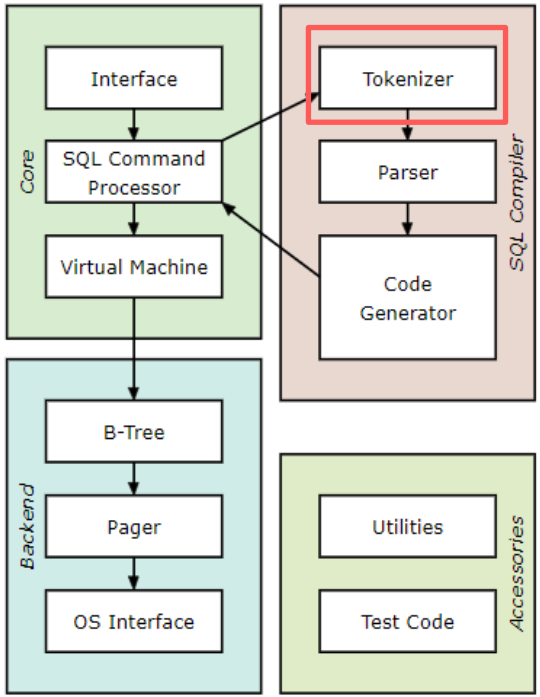
Descripción de la Arquitectura



Interfáz (Interface)

- Gran parte de la interfaz en lenguaje C se encuentra en los archivos fuente main.c , legacy.c y vdbeapi.c aunque algunas rutinas están dispersas en otros archivos donde pueden tener acceso a estructuras de datos con alcance de archivo.
- Para evitar colisiones de nombres, todos los símbolos externos en la biblioteca SQLite comienzan con el prefijo sqlite3 . Aquellos símbolos que están destinados para uso externo (en otras palabras, aquellos símbolos que forman la API para SQLite) agregan un guión bajo y, por lo tanto, comienzan con sqlite3.

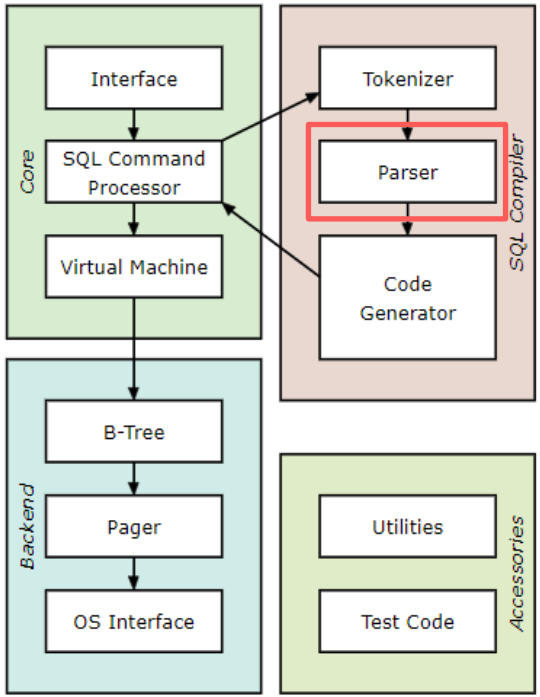
Descripción de la Arquitectura



Tokenizador (Tokenizer)

- Cuando se va a evaluar una cadena que contiene instrucciones SQL, primero se envía al tokenizador. El tokenizador divide el texto SQL en tokens y entrega esos tokens uno por uno al analizador. El tokenizador está codificado a mano en el archivo tokenize.c.
- Tenga en cuenta que en este diseño, el tokenizador llama al analizador sintáctico (parser).

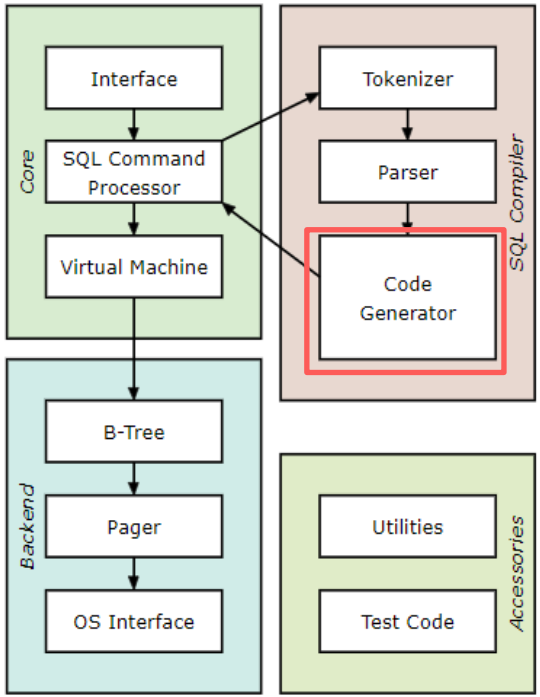
Descripción de la Arquitectura



Analizador Sintáctico (Parser)

- El analizador asigna significado a los tokens en función de su contexto. El analizador para SQLite se genera utilizando el generador de analizador Lemon (lemon parser generator).

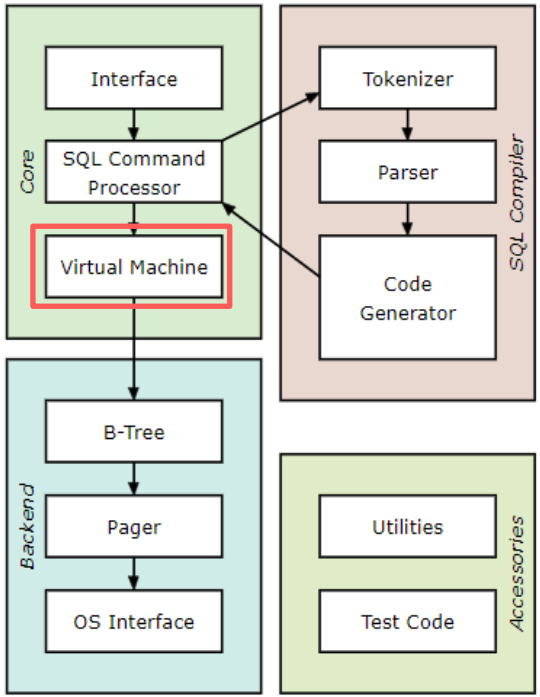
Descripción de la Arquitectura



Generador de Código (Code Generator)

- Después de que el analizador ensambla los tokens en un árbol de análisis (parse tree), el generador de código se ejecuta para analizar el árbol de análisis y generar un bytecode que realiza el trabajo de la instrucción SQL. El objeto de declaración preparada (prepared statement object) es un contenedor para este bytecode.

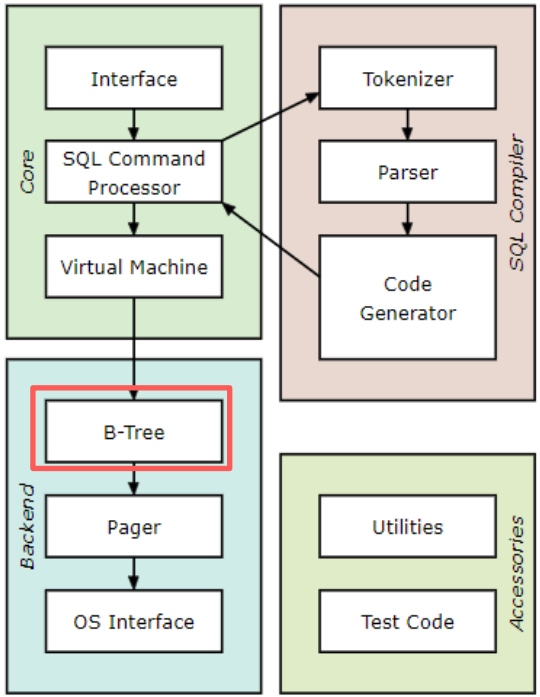
Descripción de la Arquitectura



Motor de Bytecode (Bytecode Engine)

- Después de que el analizador ensambla los tokens en un árbol de análisis (parse tree), el generador de código se ejecuta para analizar el árbol de análisis y generar un bytecode que realiza el trabajo de la instrucción SQL. El objeto de declaración preparada (prepared statement object) es un contenedor para este bytecode.

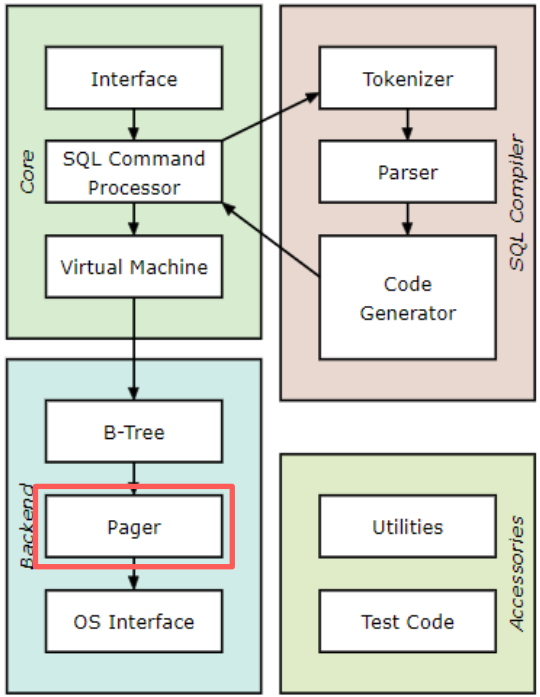
Descripción de la Arquitectura



Árbol-B (B-Tree)

- Una base de datos SQLite se mantiene en el disco utilizando una implementación de árbol B que se encuentra en el archivo fuente btree.c . Se utilizan árboles B independientes para cada tabla y cada índice de la base de datos. Todos los árboles B se almacenan en el mismo archivo de disco.

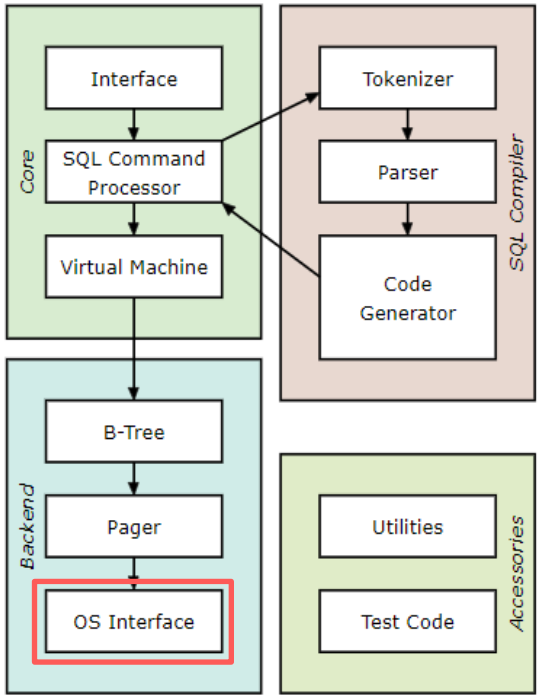
Descripción de la Arquitectura



Caché de Página (Page Cache)

- El módulo B-Tree solicita información del disco en páginas de tamaño fijo. El tamaño de página predeterminado es 4096 bytes, pero puede ser cualquier potencia de dos entre 512 y 65536 bytes. El caché de página es responsable de leer, escribir y almacenar en caché estas páginas. La memoria caché de la página también proporciona la abstracción de rollback y commit atómico y se encarga de bloquear (locking) el archivo de la base de datos. El controlador (driver) de árbol B solicita páginas particulares de la page cache y notifica a la page cache cuando desea modificar páginas o confirmar o revertir cambios. El page cache maneja todos los detalles desordenados para asegurarse de que las solicitudes se manejen de manera rápida, segura y eficiente.

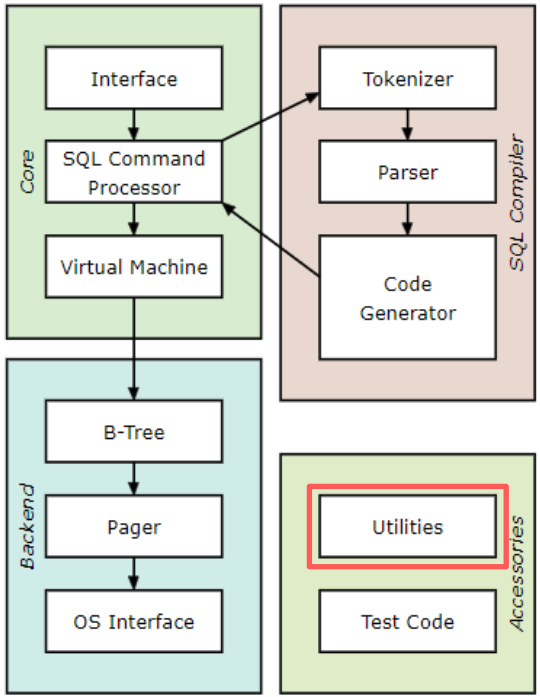
Descripción de la Arquitectura



Interfaz con el Sistema Operativo (OS Interface)

- Para proporcionar portabilidad entre sistemas operativos, SQLite usa un objeto abstracto llamado VFS. Cada VFS proporciona métodos para abrir, leer, escribir y cerrar archivos en el disco, y para otras tareas específicas del sistema operativo, como encontrar la hora actual u obtener aleatoriedad para inicializar el generador de números pseudoaleatorios incorporado. SQLite actualmente proporciona VFS para Unix (en el archivo os_unix.c) y Windows (en el archivo os_win.c).

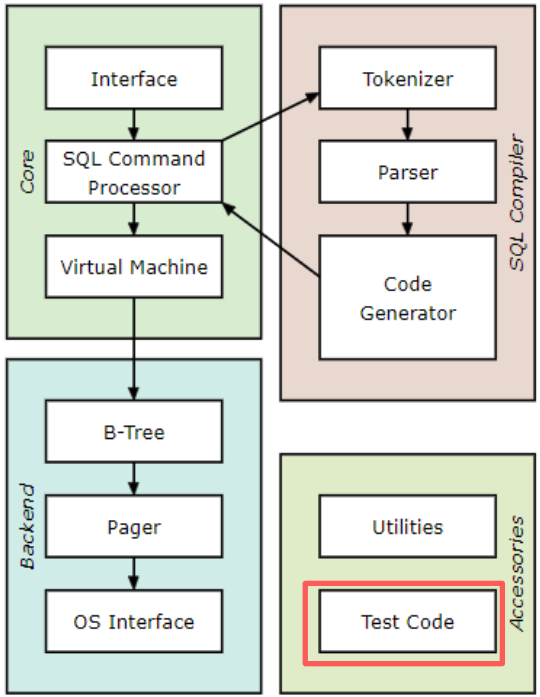
Descripción de la Arquitectura



Utilidades (Utilities)

- La asignación de memoria, las rutinas de comparación de cadenas sin mayúsculas y minúsculas, las rutinas portables de conversión de texto a número y otras utilidades se encuentran en util.c.

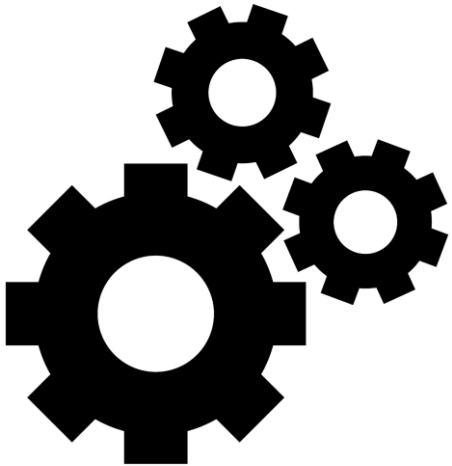
Descripción de la Arquitectura



Código de Prueba (Test Code)

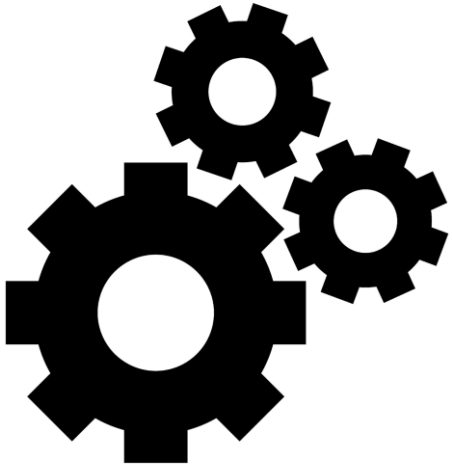
- Los archivos en el directorio "src/" cuyos nombres comienzan con “test” son únicamente para testing y no están incluídos en el build estándar de la biblioteca.

Características y Funcionalidades



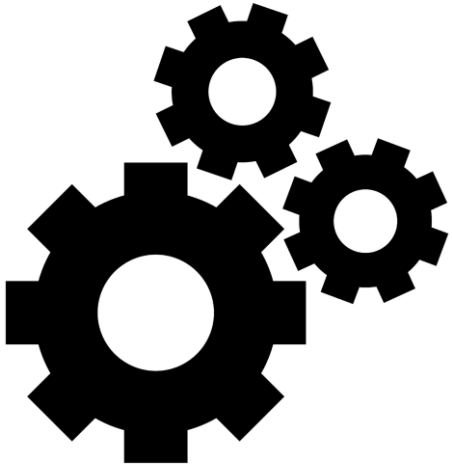
- Las transacciones son atómicas, consistentes, aisladas y duraderas (ACID) incluso después de fallas del sistema y fallas de energía.
- Configuración cero: no se necesita configuración ni administración.
- Implementación completa de SQL con capacidades avanzadas como: índices parciales, índices en expresiones, JSON, expresiones comunes de tablas y funciones de ventana.
    - Similar a una función de agregación, una función de ventana realiza un cálculo (ej: SUM()) en un conjunto de filas. Sin embargo, una función de ventana no hace que las filas se agrupen en una sola fila de salida.
- Una base de datos completa se almacena en un único archivo de disco multiplataforma. Ideal para usar como formato de archivo de aplicación.

Características y Funcionalidades



- Admite bases de datos del tamaño de un terabyte y cadenas y blobs del tamaño de un gigabyte.
- Huella de código pequeña: menos de 750 KiB completamente configurado o mucho menos con funciones opcionales omitidas.
- API simple y fácil de usar.
- Rápido: en algunos casos, SQLite es más rápido que la E/S directa del sistema de archivos
- Escrito en ANSI-C. TCL Bindings incluidas. Bindings para docenas de otros idiomas disponibles por separado.
- Código fuente bien comentado con una cobertura de pruebas del 100%.
- Disponible como un solo archivo de código fuente ANSI-C que es fácil de compilar y, por lo tanto, fácil de agregar a un proyecto más grande.

Características y Funcionalidades



- Autónomo: sin dependencias externas.
- Multiplataforma: Android, *BSD, iOS, Linux, Mac, Solaris, VxWorks y Windows (Win32, WinCE, WinRT) son compatibles desde el primer momento. Fácil de portar a otros sistemas.
- Los archivos fuente son de dominio público. Se puede utilizar para cualquier propósito.
- Viene con un cliente de interfaz de línea de comandos (CLI) independiente que se puede usar para administrar bases de datos SQLite.

Usos sugeridos de SQLite

Base de datos para Internet de las cosas (IoT)

SQLite es una opción popular para el motor de base de datos en teléfonos celulares, PDA, reproductores de MP3, decodificadores y otros dispositivos electrónicos. SQLite tiene una huella de código pequeña, hace un uso eficiente de la memoria, el espacio en disco y el ancho de banda del disco, es altamente confiable y no requiere mantenimiento por parte de un administrador de base de datos.

Usos sugeridos de SQLite

Formato de Archivo de la Aplicación

En lugar de utilizar fopen() para escribir XML, JSON, CSV o algún formato propietario en archivos de disco utilizados por su aplicación, utilice una base de datos SQLite. Evitará tener que escribir y solucionar problemas de un analizador, sus datos serán más fácilmente accesibles y multiplataforma, y sus actualizaciones serán transaccionales.

Usos sugeridos de SQLite

Base de Datos de Sitio Web

Debido a que no requiere configuración y almacena información en archivos de disco ordinarios, SQLite es una opción popular como base de datos para respaldar sitios web pequeños y medianos.

Usos sugeridos de SQLite

Sustituto de una RDBMS Empresarial

SQLite se utiliza a menudo como sustituto de un RDBMS empresarial con fines de demostración o de prueba. SQLite es rápido y no requiere configuración, lo que elimina muchas molestias de las pruebas y hace que las demostraciones sean alegres y fáciles de iniciar.

Ventajas de SQLite

Es fácil de usar

SQLite es muy sencillo de utilizar, ya que no utiliza una comunicación cliente-servidor para las consultas, ya que se comunica con un archivo que es la base de datos y que puede ser autogenerado por la propia aplicación.

Ideal para el desarrollo de apps móviles

Sus características lo convierten en una alternativa ideal para el desarrollo de aplicaciones para celulares. Se puede utilizar fácilmente para gestionar bases de datos en app que usen motores como Java, o en proyectos desarrollados con Flutter.

Como la base es un archivo, si se apaga el celular o no hay conexión a internet, el almacenamiento de datos no se ve afectado.

Ventajas de SQLite

Utiliza SQL

Las consultas a la base de datos se realizan en SQL, reduciendo la complejidad del código de la app. SQLite es una versión reducida de SQL que sigue utilizando este estándar, aunque con pequeñas modificaciones, a la hora de realizar consultas a las bases de datos.

Ocupa poco espacio

El almacenamiento de una base de datos SQLite se realiza en un solo archivo y tiene una huella de código pequeña (ocupa poco espacio). En comparación con MySQL, SQLite es una alternativa mucho más ligera, por lo que puede ser utilizada como software integrado en dispositivos como celulares, Smart TV, cámaras…

Desventajas de SQLite

No es fácilmente escalable

No se adapta bien a grandes bases de datos, por lo que si una app comienza a crecer se complica su gestión utilizando SQLite.

Problemas de seguridad

Al no contar con funciones de seguridad y administración de usuarios puede presentar problemas en cuanto a seguridad.

Monousuario para escrituras

No permite que un usuario modifique datos, si otro se encuentra conectado y realizando acciones sobre la base de datos.

Desventajas de SQLite

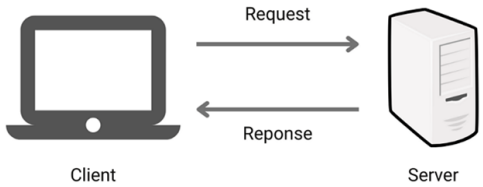
Limitación de almacenamiento

El tamaño de la base de datos se encuentra normalmente restringido a un tamaño chico en comparación con otras bases de datos (no es ideal para grandes bases de datos).

Comparación con otras bases de datos

- SQLite no es directamente comparable con los motores de base de datos SQL de cliente/servidor como MySQL, Oracle, PostgreSQL o SQL Server, ya que SQLite está tratando de resolver un problema diferente.
- Los motores de base de datos SQL cliente/servidor se esfuerzan por implementar un repositorio compartido de datos empresariales. Destacan la escalabilidad, la concurrencia, la centralización y el control. SQLite se esfuerza por proporcionar almacenamiento de datos local para aplicaciones y dispositivos individuales. SQLite enfatiza la economía, la eficiencia, la confiabilidad, la independencia y la simplicidad.

Situaciones en las que un RDBMS cliente/servidor funciona mejor

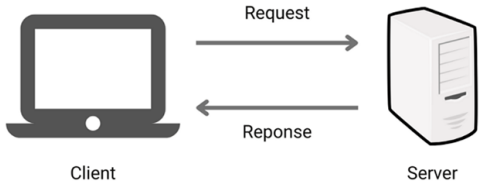


Una buena regla general es evitar el uso de SQLite en situaciones en las que se accede a la misma base de datos directamente (sin un servidor de aplicaciones intermedio) y simultáneamente desde muchas computadoras en una red.

Aplicaciones cliente/servidor

Si hay muchos programas cliente que envían SQL a la misma base de datos a través de una red, es mejor utilizar un motor de base de datos cliente/servidor en lugar de SQLite. SQLite funcionará en un sistema de archivos de red, pero debido a la latencia asociada con la mayoría de los sistemas de archivos de red, el rendimiento no será muy bueno. Además, la lógica de bloqueo de archivos tiene errores en muchas implementaciones de sistemas de archivos de red (tanto en Unix como en Windows). Si el bloqueo de archivos no funciona correctamente, es posible que dos o más clientes intenten modificar la misma parte de la misma base de datos al mismo tiempo, lo que provocaría daños. Debido a que este problema se debe a errores en la implementación del sistema de archivos subyacente, SQLite no puede hacer nada para evitarlo.

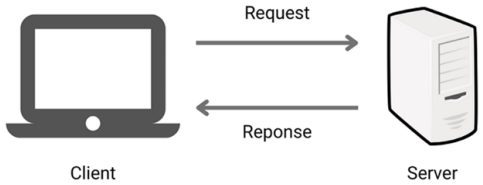
Situaciones en las que un RDBMS cliente/servidor funciona mejor



Sitios web de alto volumen

SQLite normalmente funcionará bien como base de datos de un sitio web. Pero si el sitio web requiere mucha escritura o está tan ocupado que requiere varios servidores, hay que considerar el uso de un motor de base de datos cliente/servidor de clase empresarial en lugar de SQLite.

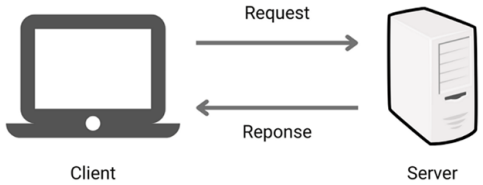
Situaciones en las que un RDBMS cliente/servidor funciona mejor



Conjuntos de datos muy grandes

Una base de datos SQLite tiene un tamaño limitado. Incluso si pudiera manejar bases de datos más grandes, SQLite almacena la base de datos completa en un solo archivo de disco y muchos sistemas de archivos limitan el tamaño máximo de los archivos a algo menos que esto. Entonces, si está contemplando bases de datos de esta magnitud, haría bien en considerar el uso de un motor de base de datos de cliente/servidor que distribuye su contenido en varios archivos de disco y quizás en varios volúmenes.

Situaciones en las que un RDBMS cliente/servidor funciona mejor



Alta concurrencia

SQLite admite una cantidad ilimitada de lectores simultáneos, pero solo permitirá un escritor en cualquier momento. Para muchas situaciones, esto no es un problema. Los escritores hacen cola. Cada aplicación hace su trabajo de base de datos rápidamente y continúa, y ningún bloqueo dura más de unos pocos milisegundos. Pero hay algunas aplicaciones que requieren más simultaneidad, y es posible que esas aplicaciones deban buscar una solución diferente.

Lista de verificación para elegir el motor de base de datos adecuado

¿Los datos están separados de la aplicación por una red?

→ elegir cliente/servidor

Los motores de bases de datos relacionales actúan como filtros de datos que reducen el ancho de banda. Por lo tanto, es mejor mantener el motor de base de datos y los datos en el mismo dispositivo físico para que el enlace de motor a disco de ancho de banda alto no tenga que atravesar la red, solo el enlace de aplicación a motor de ancho de banda más bajo.

Lista de verificación para elegir el motor de base de datos adecuado

¿Muchos escritores concurrentes?

→ elegir cliente/servidor

Si muchos subprocesos y/o procesos necesitan escribir la base de datos en el mismo instante (y no pueden hacer cola y tomar turnos), entonces es mejor seleccionar un motor de base de datos que admita esa capacidad, lo que siempre significa un motor de base de datos cliente/servidor.

Lista de verificación para elegir el motor de base de datos adecuado

¿Grandes datos?

→ elegir cliente/servidor

Si sus datos crecerán a un tamaño que no le resulte cómodo o que no pueda caber en un solo archivo de disco, entonces debe seleccionar una solución que no sea SQLite. SQLite admite bases de datos de hasta 281 terabytes de tamaño, suponiendo que pueda encontrar una unidad de disco y un sistema de archivos que admita archivos de 281 terabytes. Aun así, cuando parece que el tamaño del contenido podría llegar al rango de terabytes, sería bueno considerar una base de datos cliente/servidor centralizada.

Lista de verificación para elegir el motor de base de datos adecuado

En caso de que la respuesta a estas tres preguntas sea No, entonces la utilización de una base de datos embebida como SQLite es una muy buena opción.

¿Cuándo usar una base de datos embebida?

- **Portabilidad**: La base de datos embebida se encuentra incluida en la aplicación
- **Rapidez**: La base de datos embebida se encuentra en la misma máquina que la aplicación
- **Recursos de hardware bajos**: La base de datos embebida generalmente requiere menos recursos de hardware, como CPU y RAM

Proveedores

- Advantage Database Server de Sybase Inc
- Berkeley DB de Oracle Corporation
- CSQL de csqlcache.com
- Extensible Storage Engine de Microsoft
- Firebird Embedded
- HSQLDB de HSQLDB.ORG
- HarperDB
- Informix Dynamic Server (IDS) de IBM
- InfinityDB de Boiler Bay Inc
- InnoDB de Oracle Corporation
- InterBase de Embarcadero Technologies
- RDM Database Manager de Raima
- solidDB
- SQLite
- SQL Server Compact de Microsoft Corporation
- Sophia Embeddable

Licenciamiento

SQLite: Open-Source - not Open-Contribution sin embargo algunas compañías necesitar prueba legal del uso. Precio: USD 6000

Berkeley DB: Se pueden apreciar los precios en la imagen.

Firebird: Los módulos de Firebird se publican bajo la licencia pública de desarrollador. InterBase fue lanzado por Inprise bajo la licencia Pública de InterBase.

Berkeley Database

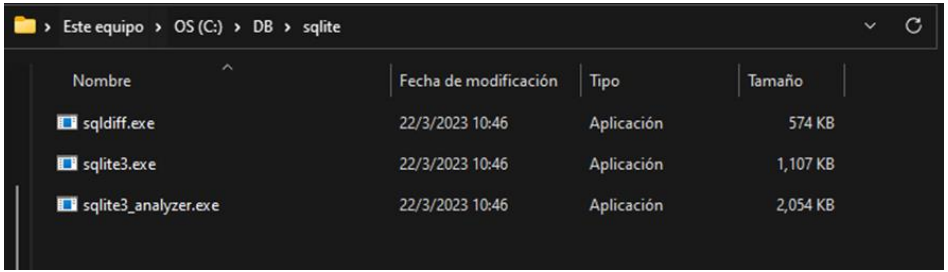
Berkeley DB - High Availability	-	-	9,800	2,156.00

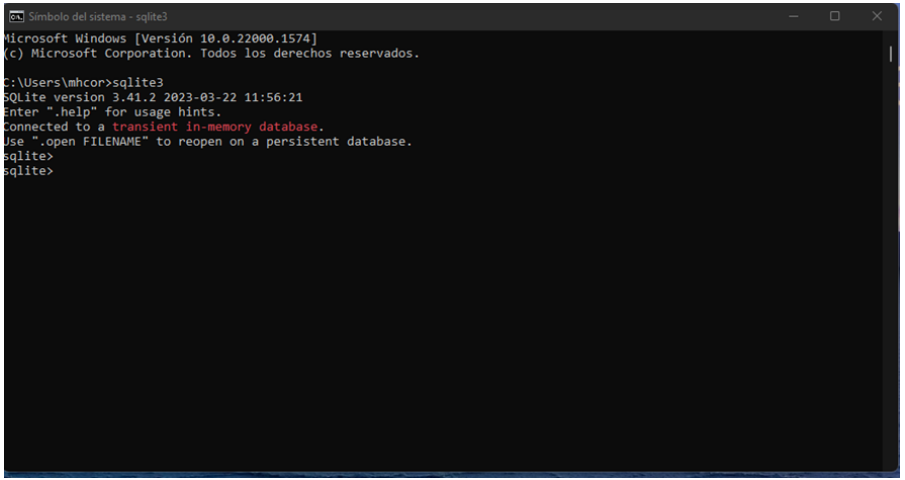
	Per Wireless Handset	Software Update License & Support	Processor License	Software Update License & Support
Berkeley DB - Transactional Data Store	6	1.32	5,800	1,276.00
Berkeley DB - Concurrent Data Store	6	1.32	1,800	396.00
Berkeley DB - Data Store	6	1.32	900	198.00

	Named User Plus	Software Update License & Support	Processor License	Software Update License & Support
Berkeley DB - Transactional Data Store	-	-	5,800	1,276.00
Berkeley DB - Concurrent Data Store	-	-	1,800	396.00
Berkeley DB - Data Store	-	-	900	198.00
Berkeley DB Java Edition - High Availability	-	-	9,800	2,156.00
Berkeley DB Java Edition - Transactional Data Store	-	-	5,800	1,276.00
Berkeley DB Java Edition - Concurrent Data Store	-	-	1,800	396.00
Berkeley DB XML - High Availability	-	-	13,800	3,036.00
Berkeley DB XML - Transactional Data Store	-	-	8,100	1,782.00
Berkeley DB XML - Concurrent Data Store	-	-	2,600	572.00
Berkeley DB XML - Data Store	-	-	1,800	396.00

Instalación del Motor

- Estos tipos de DB no poseen una instalación en sí.
- SQLite crea un archivo y dentro de este se guardan los datos

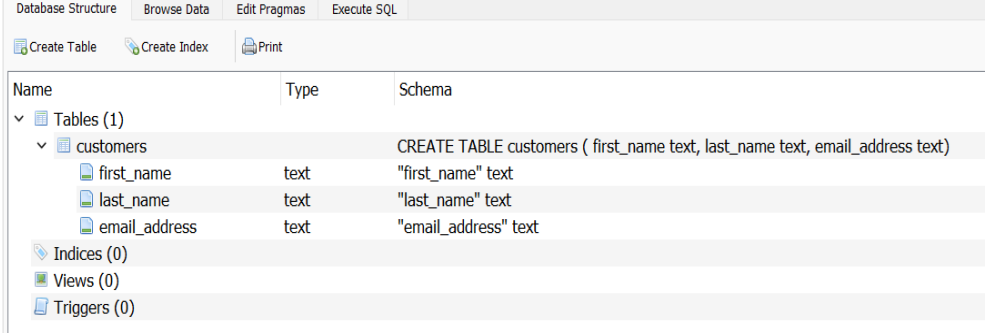


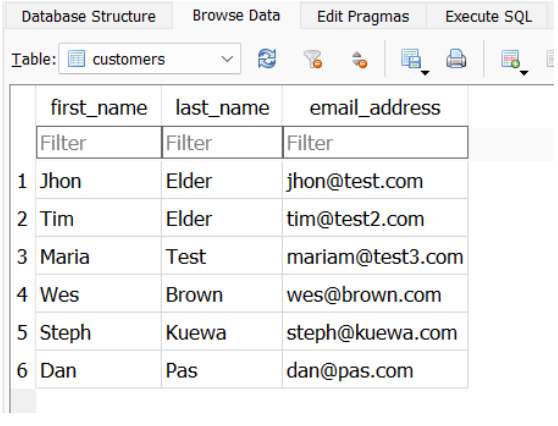


Instalación del Motor - Cont.

Código ejemplo.

```python
import sqlite3

# Crear conexion
conn = sqlite3.connect('customer.db')

# Crear una instancia de cursor
c = conn.cursor()

# Crear una tabla con el metodo execute
# Hay 5 tipos de DATATYPE en sqlite: NULL, INTEGER, REAL, TEXT y BLOB
c.execute(""" CREATE TABLE customers (
          first_name text,
          last_name text,
          email_address text)
          """)

many_customers = [('Wes', 'Brown', 'wes@brown.com'),
                  ('Steph', 'Kuewa', 'steph@kuewa.com'),
                  ('Dan', 'Pas', 'dan@pas.com')]

# Se pone un marcador donde se van a enviar los valores
c.executemany("INSERT INTO customers VALUES (?,?,?)", many_customers)

# Realizar un commit
conn.commit()

# Cerrar la conexion
conn.close()
```





Años de antigüedad

- SQLite: Lanzamiento: 17 de agosto de 2000
- Barkeley DB: Relase inicial 1994
- Firebird: Lanzamiento inicial 2000

Empresas

Demo

Repositorio - Demo

- https://github.com/andresbiso/laboratorio_IV_bd_embebida

¿Preguntas?

Cuestionario

Pregunta 1

¿Las bases de datos embebidas aparecieron en el mercado a principios del año 2000?

A. Verdadero
B. Falso

Pregunta 1

Las bases de datos embebidas aparecieron en el mercado a principios del año 2000. ¿Verdadero o Falso?

A. Verdadero
B. Falso

Opción Correcta: B

Pregunta 2

Un sistema de base de datos embebido…

A. Es un sistema de administración de base de datos (DBMS) que está estrechamente integrado con un software de aplicación.
B. Está incrustado en la aplicación.
C. Todas las anteriores.

Pregunta 2

Un sistema de base de datos embebido…

A. Es un sistema de administración de base de datos (DBMS) que está estrechamente integrado con un software de aplicación.
B. Está incrustado en la aplicación.
C. Todas las anteriores.

Opción Correcta: C

Pregunta 3

Una base de datos SQL completa con varias tablas, índices, disparadores y vistas está contenida en un solo archivo de disco. ¿Verdadero o Falso?

A. Verdadero
B. Falso

Pregunta 3

Una base de datos SQL completa con varias tablas, índices, disparadores y vistas está contenida en un solo archivo de disco. ¿Verdadero o Falso?

A. Verdadero

B. Falso

Opción Correcta: A

Pregunta 4

El módulo core de la arquitectura de SQLite está compuesto por:

A. Interface, Tokenizer, Parser

B. Interface, Sql Command Processor, Virtual Machine

C. B-Tree, Utilities, Test Code

Pregunta 4

El módulo core de la arquitectura de SQLite está compuesto por:

A. Interface, Tokenizer, Parser

B. Interface, Sql Command Processor, Virtual Machine

C. B-Tree, Utilities, Test Code

Opción Correcta: B

Pregunta 5

¿En cuál de las siguientes situaciones conviene usar SQLite?

A. Base de datos para internet de las cosas
B. Base de datos de Sitios Web pequeños
C. Sustituto de RDBMS para pruebas
D. Todas las anteriores.

Pregunta 5

¿En cuál de las siguientes situaciones conviene usar SQLite?

A. Base de datos para internet de las cosas
B. Base de datos de Sitios Web pequeños
C. Sustituto de RDBMS para pruebas
D. Todas las anteriores.

Opción Correcta: D

Pregunta 6

¿Cuál de las siguientes opciones NO es una desventaja de SQLite?

A. Ocupa poco espacio
B. No es fácilmente escalable
C. Problemas de Seguridad

Pregunta 6

¿Cuál de las siguientes opciones NO es una desventaja de SQLite?

A. Ocupa poco espacio
B. No es fácilmente escalable
C. Problemas de Seguridad

Opción Correcta: A

Pregunta 7

Si tenemos una situación dónde los datos están separados de la aplicación por una red. Es mejor hacer uso de:

A. RDBMS Cliente/Servidor
B. Base de datos embebida

Pregunta 7

Si tenemos una situación dónde los datos están separados de la aplicación por una red. Es mejor hacer uso de:

A. RDBMS Cliente/Servidor
B. Base de datos embebida

Opción Correcta: A

Pregunta 8

¿Cuáles de las siguientes organizaciones ofrecen base de datos embebidas?

A. Advantage Database Server de Sybase Inc
B. Berkeley DB de Oracle Corporation
C. SQLite
D. Firebird Embedded
E. Todas las anteriores

Pregunta 8

¿Cuáles de las siguientes organizaciones ofrecen base de datos embebidas?

A. Advantage Database Server de Sybase Inc
B. Berkeley DB de Oracle Corporation
C. SQLite
D. Firebird Embedded
E. Todas las anteriores

Opción Correcta: E

Pregunta 9

Mozilla Firefox usa SQLite para almacenar, entre otros, las cookies, los favoritos, el historial y las direcciones de red válidas. ¿Verdadero o Falso?

A. Verdadero
B. Falso

Pregunta 9

Mozilla Firefox usa SQLite para almacenar, entre otros, las cookies, los favoritos, el historial y las direcciones de red válidas. ¿Verdadero o Falso?

A. Verdadero

B. Falso

Opción Correcta: A

Pregunta 10

SQLite NO hace uso del lenguaje SQL. ¿Verdadero o Falso?

A. Verdadero

B. Falso

Pregunta 10

SQLite NO hace uso del lenguaje SQL. ¿Verdadero o Falso?

A. Verdadero

B. Falso

Opción Correcta: B

Gracias!