Архитектура SQLite и его драйвера

Что такое “драйвер” в контексте SQLite

SQLite — это не клиент-серверная база как PostgreSQL или MySQL. Это **встраиваемая библиотека на C**, которая *сама* выполняет SQL-операции и управляет хранением данных в файле. То есть драйвер это на самом деле библиотека sqlite3, которая реализует весь процесс от запроса до работы с физической памятью.

Как это работает

- Пускай есть запрос “SELECT name FROM users WHERE id=1” Драйвер вызывает функцию sqlite3_prepare_v2() из C-библиотеки. Она передает SQL текст во внутренний компилятор SQLite.
- Происходит парсинг. Запрос разбивается на лексемы и затем строится дерево синтаксиса(AST - Abstract syntax tree) - структура описывающая смысл запроса.
- На основе AST генерируется SQLite bytecode, который исполняется виртуальной машиной SQLite (VDBE - Virtual Database engine). На самом деле это похоже на интерпретатор SQL запросов.
- VDBE выполняет инструкции байткода. Каждая инструкция вызывает функции нижележащего слоя(Storage engine или B-tree layer)
- Таблицы и индексы в SQLite это B-древья. Каждый узел этого дерева хранится на отдельной странице(как правило 4кб). Операции идут через btree.c.
- Pager layer управляет чтением и записью страниц. Поддерживает транзакции и журналирование (через WAL или rollback journal). Следит, чтобы изменения были атомарны и не потерялись при сбое.
- OS interface. Внизу драйвер использует virtual file system. Реализует функции xOpen, xRead, xWrite, xLock, xSync.

Почему это круто

- Весь движок 300-400кб
- Все в одном файле, то есть легкость встраивания в любые приложения
- Без сети, все происходит локально
- Один и тот же драйвер работает на разных ОС.

Различия row и column based хранения данных.

При хранении данных по колонкам мы храним данные о каждом столбце в отдельном файле, в отличие от строчного хранения, где все строки лежат подряд в одном файле. Колоночное хранение позволяет более эффективно сжимать данные, так как в каждом отдельном файле лежат однородные данные. В строчном же хранении в файле лежат неоднородные данные, которые требуют более сложных и изощренных алгоритмов

сжатия.
Также колоночное хранение позволяет более быстрым способом работать с SELECTами, т.к. при считывании одного файла мы сразу получаем доступ ко всей колонке. Т.е. для аналитической работы удобнее хранить данные в колонках. Но, не все так радужно с колоночным типом хранения, ведь к примеру процедура вставки строк уже будет не так удобна. И в целом частые изменения удобнее совершать с построчным хранением данных.

Формат хранения данных на первой итерации

JSON