Engineer

SQLite

Eine vollständige Datenbank in einer einzigen Datei

Was es ist

SQLite ist eine SQL-Datenbankengine, die als Bibliothek innerhalb des Programms läuft, das sie nutzt: ohne Serverprozess, Netzwerkport oder Administrator. Tabellen, Indizes, Trigger und Sichten speichert sie gemeinsam in einer gewöhnlichen Datei auf dem Datenträger. Der Name erscheint auch als `sqlite3` : So heißt die Kommandozeilenshell, und dieses Präfix trägt jede Funktion der C-Schnittstelle. Die eine vom Projekt ausgelieferte C-Quelldatei heißt `sqlite3.c` . Das Projekt beschreibt sich als eigenständig, serverlos, konfigurationsfrei und transaktional und bittet um einen Vergleich mit `fopen()` statt mit Oracle. Es ersetzt die selbst entwickelte Datendatei, nicht den Datenbankserver.

Das Projekt begann am 9. Mai 2000. Sein Code ist gemeinfrei, sodass vor der Auslieferung innerhalb eines Produkts keine Lizenzfreigabe einzuholen ist. Zum Konzept gehört auch das Dateiformat: Eine auf einem 32-Bit-Rechner geschriebene Datenbank lässt sich auf einen 64-Bit-Rechner oder zwischen Prozessoren mit Big-Endian- und Little-Endian-Byteordnung kopieren und unverändert öffnen. Die Entwickler erklären, SQLite bis zum Jahr 2050 unterstützen zu wollen. Die US Library of Congress führt seine Dateien als empfohlenes Format zur Datenaufbewahrung auf. Selbst mit allen aktivierten Funktionen kann die Bibliothek kleiner als 900 KiB sein. Deshalb findet sie sich nicht nur auf Servern, sondern auch in Telefonen, Autos, Uhren und Desktopanwendungen.

Im Inneren steckt eine vollständige SQL-Engine: Joins, Unterabfragen, Sichten, Trigger und Transaktionen, die selbst bei einem Stromausfall mitten in einem Schreibvorgang atomar, konsistent, isoliert und dauerhaft bleiben. Zwei Entwurfsentscheidungen unterscheiden sie von Serverdatenbanken. Die erste betrifft gleichzeitige Zugriffe. Beliebig viele Verbindungen dürfen gleichzeitig lesen, aber zu jedem Zeitpunkt darf nur eine schreiben, und die Sperre gilt für die

gesamte Datei. Die zweite betrifft Typen. Der deklarierte Typ einer Spalte ist eine Präferenz, die SQLite Affinität nennt. Der tatsächliche Typ gehört zu jedem einzelnen Wert; deshalb nimmt eine als `INTEGER` deklarierte Spalte auch Text an, der nicht wie eine Zahl aussieht.

Die Abgrenzung zu den Alternativen ist klar. PostgreSQL, MySQL und SQL Server sind gemeinsam genutzte Dienste, mit denen viele Rechner über ein Netzwerk sprechen. SQLite ist ein privater Datenspeicher für eine Anwendung auf einem Rechner. Gegenüber CSV- oder JSON-Dateien ergänzt es eine Abfragesprache, Indizes, Constraints und Transaktionen. Der Preis dafür ist eine Abhängigkeit, die die meisten Sprachen bereits mitliefern.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist SQLite?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Eine Struktur, die zur tatsächlichen Arbeitsweise Ihres Unternehmens passt, damit sich auch spätere Fragen noch beantworten lassen.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 6 ETL/ELT-Pipelines entworfen und betrieben — 6 ETL-Pipelines über drei Datenbankengines hinweg

6 ETL/ELT-Pipelines entworfen und betrieben Kontakt aufnehmen →

Warum es wichtig ist

Für einen großen Teil der Software ist die schwierige Datenbankfrage nicht die Geschwindigkeit, sondern wer sie betreibt. Ein Server muss installiert, aktualisiert, abgesichert, überwacht und gesichert werden, und jemand muss für all das bezahlt werden. SQLite beseitigt diese ganze Aufgabenklasse. Eine Datenbank zu öffnen bedeutet, eine Datei zu öffnen. Eine Sicherung ist eine Kopie dieser Datei, die während der Nutzung über die Sicherungs-API erstellt wird. Eine Testsuite kann jedem Test eine eigene frische Datenbank geben.

Es verändert auch, was eine Datei leisten kann. Ein Programm, das sonst ein Format erfinden, einen Parser dafür schreiben und hoffen müsste, dass kein Absturz mitten im Speichern auftritt, kann seinen Zustand in SQLite ablegen und erhält atomare Speichervorgänge dazu. Das Projekt nennt diese Einsatzfälle ausdrücklich: als Anwendungsdateiformat, als Container für den Datenaustausch zwischen Systemen, bei dem der Empfänger SQL ausführt statt einen Parser zu schreiben, und als temporäre Arbeitsdatenbank zum Sortieren und Zusammenfassen anders angelieferter Daten.

Die entscheidende Frage ist eng umrissen und lässt sich vor der ersten Codezeile prüfen: Wie viele Prozesse schreiben, und auf wie vielen Rechnern? Ein Programm auf einem Rechner, das in kurzen Schüben schreibt, ist der ideale Einsatzbereich von SQLite. Auch viele Websites mit geringem bis mittlerem Datenverkehr laufen problemlos damit. Mehrere Server, die dieselben Daten beschreiben, oder eine Last, bei der ständig viele Benutzer schreiben, gehören auf eine Client-Server-Datenbank. Die Größe entscheidet selten darüber, denn eine Datei kann bis zu 281 Terabyte aufnehmen. Entscheidend ist die Zahl der Schreiber.

Auch in diesem Einsatzbereich gibt es Nachteile abzuwägen. `ALTER TABLE` unterstützt nur das Umbenennen einer Tabelle oder Spalte sowie das Hinzufügen oder Entfernen einer Spalte. Die meisten anderen Strukturänderungen erfordern deshalb eine neue Tabelle, das Kopieren der Zeilen und das Vertauschen der Namen. Fremdschlüssel sind deaktiviert, bis jede Verbindung sie einschaltet. Die lockere Typisierung lässt fehlerhafte Daten unbemerkt zu, sofern die Tabelle nicht ausdrücklich so deklariert wird, dass sie diese ablehnt.

Wie es in der Praxis aussieht

Zum dokumentierten Entwicklungsstand bildeten dreiundzwanzig Tabellen in einer SQLite-Datei, von Leistungen und Unternehmen bis zu Profilen und Referenzen, die maßgebliche Datenquelle für einen Generator von Lebensläufen, Referenzblättern, Portfolios und Anschreiben. Ein Befehl machte aus einer leeren SQLite-Datei diese Datenbank, in neunzehn geordneten Schritten. Da jeder Schritt gefahrlos zweimal ausgeführt werden konnte, fügte ein Neuaufbau über einer bestehenden Datenbank keine doppelten Zeilen hinzu. Die sechs Ausgabeformate entstanden aus 41 Python-Modulen mit 10.580 Zeilen, die diese eine Datei lasen, mit genau zwei Laufzeitabhängigkeiten. Ein auf eine bestimmte Leserschaft zugeschnittenes Dokument war eine durch ein Profil ausgewählte Abfrage über die Leistungen statt einer bearbeiteten Kopie. Der Preis war klar: Das System war zum einzigen Weg geworden, ein Dokument zu erzeugen.

Zum dokumentierten Exportstand diente dieselbe Datenbank auch als Eingabe für den Build, denn das Portfolio dieser Website wurde daraus erzeugt. Ein Exporter öffnete die SQLite-Datei schreibgeschützt und schrieb 495 Seiten in drei Sprachen, ausschließlich mit Pythons Standardbibliothek. Die Ausgabe wurde versioniert, damit die Website ohne die Umgebung des Generators gebaut werden konnte. Seitenadressen wurden als Zeilen gespeichert, eine pro Adresse und Leistung, sodass das Umformulieren eines Satzes eine Seite nicht

mehr verschob. Bei dieser Arbeit zeigte sich auch die im nächsten Abschnitt beschriebene Falle: Ein Zeitraumvergleich gegen eine Textspalte traf auf alle zweiunddreißig Zeilen zu.

Die kleinste Rolle, nämlich Ergebnisse einsehen zu können, ist zugleich die typischste. Eine Pipeline, die rohe Strommesswerte in stündlichen Verbrauch pro Zone umwandelte, schrieb ihre Ergebnisse entweder als CSV oder zur Untersuchung in eine schlanke SQLite-Datenbank. Die Python-Version verarbeitete einen Tag Daten in 34 ms und dreißig Tage in 180 ms. Eine so verwendete Datenbank benötigt keinerlei Einrichtung: Wer über die Shell `sqlite3` verfügt, kann die Datei öffnen und abfragen.

Wo es schiefgeht

Die meisten SQLite-Probleme entstehen, wenn es wie ein kleines PostgreSQL behandelt wird. Es ist ein anderes System mit anderen Grenzen.

Ein Jahresfilter trifft auf jede Zeile zu. SQLite wandelt einen Wert vor einem Vergleich nur dann um, wenn eine Seite eine Affinität besitzt. Ein Funktionsergebnis wie `substr()` hat keine, ebenso wenig eine alleinstehende Zahl. Deshalb vergleicht SQLite den Text `'2019'` mit der Ganzzahl `2025` unverändert, und in seiner Sortierreihenfolge steht jede Zahl vor jedem Text:

```
CREATE TABLE stint (period TEXT, hours TEXT);
INSERT INTO stint VALUES
  ('2019-03', '9'), ('2024-11', '10'), ('2026-01', '42');

SELECT count(*) FROM stint
WHERE substr(period, 1, 4) >= 2025;                    -- 3

SELECT count(*) FROM stint
WHERE CAST(substr(period, 1, 4) AS INTEGER) >= 2025;   -- 1
```

Konvertieren Sie Typen ausdrücklich, wo Text auf eine Zahl trifft. Deklarieren Sie für neue Tabellen konkrete Datentypen und ergänzen Sie das seit Version 3.37.0 verfügbare Schlüsselwort `STRICT`: Eine strikte `INTEGER`-Spalte akzeptiert weiterhin `'42'` und speichert es als Zahl. `'ten'` lehnt sie dagegen mit einem Fehler ab, statt es zu speichern.

Schreibvorgänge scheitern mit „database is locked“. Eine Verbindung hält die Schreibsperre, und ein zweiter Schreiber wartet oder gibt sofort auf. Ursache ist meist eine lange Transaktion oder ein Leser, der im standardmäßigen Rollback-Journal-Modus einen Schreiber aufhält. Stellen Sie die Datenbank auf Write-Ahead Logging um, bei dem Leser und Schreiber einander nicht mehr blockieren, halten Sie Schreibtransaktionen kurz und legen Sie ein Busy-Timeout fest, damit ein Schreiber kurz wartet, statt einen Fehler auszulösen. Sind die Schreiber tatsächlich zahlreich und ständig aktiv, ist das das Signal für den Wechsel zu einem Server.

Die Datei wird auf einem Netzlaufwerk beschädigt. SQLite ist auf die Dateisperren des Betriebssystems angewiesen. Viele Netzwerkdateisysteme implementieren sie fehlerhaft, sodass zwei Rechner gleichzeitig denselben Teil der Datei beschreiben können. Write-Ahead Logging funktioniert über ein Netzwerkdateisystem überhaupt nicht, weil alle Prozesse, die die Datenbank verwenden, gemeinsamen Speicher auf einem Rechner benötigen. Halten Sie die Datenbank auf einem lokalen Datenträger des einen schreibenden Programms und schalten Sie einen Server davor, wenn andere Rechner die Daten benötigen.

Eine Zeile überlebt den übergeordneten Datensatz, auf den sie verweist. Fremdschlüsselbedingungen sind im Schema deklariert, bleiben aber inaktiv, bis jede Verbindung `PRAGMA foreign_keys = ON` ausführt. Ein Löschvorgang in einer Verbindung, die dieses Pragma ausgelassen hat, hinterlässt deshalb ohne Fehlermeldung verwaiste Datensätze. Setzen Sie das Pragma an der zentralen Stelle, an der die Anwendung Verbindungen öffnet, und testen Sie, dass es aktiviert ist.

Weiterführende Lektüre

- Über SQLite — die Beschreibung des Projekts: im Prozess, serverlos, gemeinfrei, eine plattformübergreifende Datei.
- Datentypen in SQLite — Speicherklassen, Typaffinität und die Vergleichsregeln hinter der obigen Falle.
- STRICT-Tabellen — wie eine Tabelle ihre deklarierten Spaltentypen durchsetzt.
- Write-Ahead Logging — parallele Leser und ein Schreiber und warum dies auf einen Rechner beschränkt bleibt.
- Geeignete Einsatzfälle für SQLite — wo es passt, wo eine Client-Server-Datenbank besser passt, und eine Checkliste.
- SQLite-Fremdschlüsselunterstützung — warum Bedingungen standardmäßig deaktiviert sind und wie sie aktiviert werden.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Bringen Sie einen repräsentativen Datensatz, die Fragen, die er beantworten muss, und den aktuellen Engpass mit. Wir können vereinbaren, wie Qualität, Durchsatz und Wiederherstellung gemessen werden.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist SQLite?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/sqlite/