Engineer

SQLite

en hel database i én enkelt fil

Hva det er

SQLite er en SQL-databasemotor som kjører inne i programmet som bruker den, som et bibliotek, uten serverprosess, nettverksport eller administrator, og som holder tabellene, indeksene, triggerne og visningene sine samlet i én vanlig fil på disk. Navnet dukker også opp som `sqlite3`, som er navnet på kommandolinjeskallet og prefikset til hver funksjon i C-grensesnittet. Den ene C-kildefilen prosjektet leverer, heter `sqlite3.c`. Prosjektets egen oppsummering er at motoren er selvstendig, serverløs, uten konfigurasjon og transaksjonell, og prosjektet ber om å bli sammenlignet med `fopen()` heller enn med Oracle. Det SQLite erstatter, er den hjemmesnekrede datafilen, og databaseserveren får stå i fred.

Prosjektet startet 9. mai 2000, og koden er allemannseie (public domain), så det finnes ingen lisens å klarere før den bygges inn i et produkt. Filformatet er en del av avtalen. En database skrevet på en 32-biters maskin kan kopieres til en 64-biters, eller mellom prosessorer med ulik byterekkefølge, big-endian og little-endian, og åpnes uendret. Utviklerne sier at de har tenkt å støtte SQLite fram til år 2050, og det amerikanske Library of Congress oppfører filene som et anbefalt format for å bevare data. Med alle funksjoner slått på kan biblioteket være mindre enn 900 KiB, og derfor sitter det i telefoner, biler, klokker og skrivebordsprogrammer så vel som på servere.

Innvendig er det en komplett SQL-motor: sammenkoblinger, underspørringer, visninger, triggere og transaksjoner som forblir atomiske, konsistente, isolerte og varige selv når strømmen går midt i en skriving. To designvalg skiller den fra serverdatabaser. Det første er samtidighet. Et hvilket som helst antall tilkoblinger kan lese på en gang, men bare én kan skrive i et gitt øyeblikk, og låsen dekker hele filen. Det andre er typing. Den deklarerte typen til en kolonne

er en preferanse, som SQLite kaller affinitet, og typen tilhører egentlig hver enkelt verdi, så en kolonne deklarert som `INTEGER` godtar tekst som ikke ligner et tall.

Mot alternativene er grensen tydelig. PostgreSQL, MySQL og SQL Server er delte tjenester som mange maskiner snakker med over et nettverk. SQLite er et privat lager for én applikasjon på én vert. Mot CSV- eller JSON-filer legger den til et spørrespråk, indekser, begrensninger og transaksjoner, til prisen av én avhengighet som de fleste programmeringsspråk allerede har med.

Bruk begrepet i en beslutning

Bruk «Hva er SQLite?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. En struktur som samsvarer med hvordan virksomheten faktisk fungerer, slik at spørsmålene du trenger svar på senere, fortsatt er mulige å stille.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Designet og driftet 6 ETL/ELT-pipelines — 6 ETL-pipelines på tvers av tre databasemotorer

Designet og driftet 6 ETL/ELT-pipelines Ta kontakt →

Hvorfor det er viktig

For en stor del av programvaren er det vanskelige spørsmålet om en database hvem som drifter den, mer enn hvor rask den er. En server må installeres, oppgraderes, sikres, overvåkes og sikkerhetskopieres, og noen må få betalt for å gjøre alt det. SQLite fjerner hele den kategorien. Å åpne en database er å åpne en fil, en sikkerhetskopi er en kopi av den filen, tatt gjennom backup-API-et mens den er i bruk, og en testsuite kan gi hver test sin egen ferske database.

Den endrer også hva en fil kan være. Et program som ellers ville funnet opp et format, skrevet en parser for det og håpet at et krasj aldri traff midt i en lagring, kan holde tilstanden sin i SQLite og få atomiske lagringer gratis. Prosjektet nevner denne bruken direkte: som filformat for applikasjoner, som beholder for å flytte data mellom systemer, der mottakeren kjører SQL i stedet for å skrive en parser, og som kladdedatabase for å sortere og oppsummere data som kom i en annen form.

Beslutningen som teller, er smal, og den kan sjekkes før en eneste kodelinje er skrevet: hvor mange prosesser skriver, og på hvor mange maskiner? Ett program på én vert som skriver i korte støt, er hjemmebanen til SQLite, og mange nettsteder med lav til middels trafikk kjører komfortabelt på den. Flere servere

som skriver til de samme dataene, eller en last der mange brukere skriver hele tiden, hører hjemme i en klient-server-database. Størrelsen avgjør det sjelden, siden én fil kan holde opptil 281 terabyte. Antallet skrivere gjør det.

Selv på hjemmebanen er det kostnader å veie. `ALTER TABLE` støtter bare å gi nytt navn til en tabell eller en kolonne og å legge til eller fjerne en kolonne, så de fleste andre endringer i formen på en tabell betyr å bygge en ny tabell, kopiere radene over og bytte navnene. Fremmednøkler er slått av til hver tilkobling slår dem på. Og løs typing slipper dårlige data inn i det stille, med mindre tabellen er deklarert til å avvise dem.

Slik ser det ut i praksis

SQLite fyller tre ulike roller i tre prosjekter her: som sannhetskilde, som inndata til et bygg og som et sted å se på resultater.

Det første eksemplet er en generator for CV-er, referanselister, porteføljer og søknadsbrev. På det dokumenterte utviklingsstadiet var kilden en SQLite-database med 23 tabeller, fra resultater og selskaper til profiler og referanser, satt sammen av en pipeline i 19 trinn som gikk fra en tom fil til en komplett database med én kommando. Hvert trinn kunne trygt gjentas, så pipelinen kunne kjøres mot en eksisterende database uten å duplisere en rad. Førtien Python-moduler, 10 580 linjer i alt, gjenga seks utdataformater fra den ene filen, med nøyaktig to avhengigheter under kjøring. Et dokument rettet mot én type leser var en spørring over resultatene, valgt av en profil, og aldri en redigert kopi. Siden var ærlig om prisen: systemet var blitt den eneste måten å produsere et dokument på.

Det andre eksemplet er porteføljen på dette nettstedet, som på det dokumenterte eksportstadiet ble generert fra den samme databasen. En eksportør åpnet SQLite-filen skrivebeskyttet og skrev 495 sider på tre språk, bare med Pythons standardbibliotek, og utdataene ble sjekket inn, slik at nettstedet kunne bygges uten generatorens miljø. Sideadressene ble lagret som rader, én per adresse per resultat, så en omskrevet setning ikke lenger flyttet en side. Det samme arbeidet dokumenterte fellen som beskrives i neste del: en sammenligning av perioder mot en tekstkolonne traff alle trettito radene.

Det tredje er det minste og det mest typiske. En pipeline som gjorde rå strømavlesninger om til forbruk per time og sone, skrev resultatene enten som CSV eller inn i en lettvekts SQLite-database for inspeksjon, og Python-versjonen

behandlet ett døgn med data på 34 ms og tretti døgn på 180 ms. En database brukt på den måten trenger ikke noe oppsett i det hele tatt: alle som har `sqlite3` -skallet, kan åpne filen og spørre mot den.

Hvor det går galt

Det meste av trøbbelet med SQLite kommer av å behandle den som en liten PostgreSQL. Den er en annen maskin med andre kanter.

Et filter på årstallet som treffer hver rad. SQLite konverterer en verdi før en sammenligning bare når en av sidene har en affinitet. Et funksjonsresultat som `substr()` har ingen, og det har heller ikke et nakent tall, så SQLite sammenligner teksten `'2019'` med heltallet `2025` slik de står, og i sorteringsrekkefølgen kommer hvert tall før hver tekst:

```
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
```

Bruk `CAST` eksplisitt overalt der tekst møter et tall. I nye tabeller bør du deklarere ekte typer og legge til nøkkelordet `STRICT`, som finnes fra versjon 3.37.0. En streng `INTEGER` -kolonne godtar fortsatt `'42'` og lagrer det som et tall, men avviser `'ten'` med en feil i stedet for å lagre verdien.

Skrivinger som begynner å feile med `database is locked`. Én tilkobling holder skrivelåsen, og en ny skriver enten venter eller gir opp med en gang. Årsaken er som regel en lang transaksjon, eller en leser i standardmodusen med tilbakerullingsjournal som holder igjen en skriver. Sett databasen over på write-ahead logging (WAL), der lesere og skriveren slutter å blokkere hverandre, hold skrivetransaksjonene korte, og sett en ventetid for opptatt database (busy timeout), slik at en skriver venter litt i stedet for å feile. Er skriverne virkelig mange og konstante, er det signalet om å flytte til en server.

En fil som blir korrupt på en delt disk. SQLite er avhengig av fillåsene i operativsystemet, og mange nettverksfilsystemer har feil i hvordan de implementerer disse låsene, så to maskiner kan ende opp med å skrive til den samme delen av filen samtidig. WAL virker overhodet ikke over et nettverksfilsystem, fordi hver prosess som bruker databasen, må dele minne på én vert. Hold databasen på en disk som er lokal for det ene programmet som skriver til den, og sett en server foran den hvis andre maskiner trenger dataene.

En rad som overlever forelderen den peker på. Fremmednøkler deklareres i skjemaet, men forblir inaktive til hver tilkobling kjører `PRAGMA foreign_keys = ON`, så en sletting i en tilkobling som hoppet over pragmaen, etterlater foreldreløse rader uten et ord. Sett pragmaen der applikasjonen åpner en tilkobling, på det ene stedet som gjør det, og test at den er på.

Les mer

- About SQLite — prosjektets egen beskrivelse: i prosessen, serverløs, allemannseie, én fil som virker på tvers av plattformer.
- Datatypes In SQLite — lagringsklasser, typeaffinitet og sammenligningsreglene bak fellen over.
- STRICT Tables — hvordan du får en tabell til å håndheve de deklarerte kolonnetypene.
- Write-Ahead Logging — lesere og en skriver parallelt, og hvorfor det holder seg på én vert.
- Appropriate Uses For SQLite — hvor den passer, hvor en klient-server-database passer bedre, og en sjekkliste.
- SQLite Foreign Key Support — hvorfor begrensningene er av som standard, og hvordan du slår dem på.

Ordliste

Velg ut fra deres faktiske rammer

Ta med et representativt datasett, spørsmålene det skal besvare og dagens flaskehals. Vi kan avtale målinger av kvalitet, gjennomstrømning og gjenoppretting.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er SQLite?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

welcome@engineer.company +45 50 10 61 84 København, Danmark

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/sqlite/