Engineer

SQLite

Base de données complète contenue dans un seul fichier

Ce que c’est

SQLite est un moteur SQL qui s’exécute dans le programme qui l’utilise, comme une bibliothèque, sans processus serveur, port réseau ni administrateur. Il conserve tables, index, déclencheurs et vues ensemble dans un fichier ordinaire sur disque. Son nom apparaît aussi sous la forme `sqlite3` : c’est le nom du shell en ligne de commande et le préfixe de toutes les fonctions de son interface C. L’unique fichier source C livré par le projet s’appelle `sqlite3.c` . Le projet se décrit comme autonome, sans serveur, sans configuration et transactionnel, et demande à être comparé à `fopen()` plutôt qu’à Oracle. Il remplace le fichier de données fait maison, pas le serveur de base de données.

Le projet a commencé le 9 mai 2000, et son code est dans le domaine public : aucune licence n’est donc à obtenir avant de l’intégrer à un produit. Le format de fichier fait aussi partie de ses atouts. Une base écrite sur une machine à 32 bits peut être copiée sur une machine à 64 bits, ou entre des processeurs big-endian et little-endian, puis ouverte sans modification. Les développeurs déclarent vouloir prendre en charge SQLite jusqu’en 2050, et la Bibliothèque du Congrès des États-Unis classe ses fichiers parmi les formats recommandés pour conserver des données. Même avec toutes les fonctionnalités activées, la bibliothèque peut peser moins de 900 Kio. Elle se trouve ainsi dans les téléphones, voitures, montres et applications de bureau, ainsi que sur les serveurs.

Sous le capot, c’est un moteur SQL complet : jointures, sous-requêtes, vues, déclencheurs et transactions qui restent atomiques, cohérentes, isolées et durables même en cas de coupure de courant au milieu d’une écriture. Deux choix de conception le distinguent des bases serveur. Le premier concerne la concurrence. Un nombre quelconque de connexions peut lire en même temps, mais une seule peut écrire à un instant donné, et le verrou couvre tout le fichier. Le second est le typage. Le type déclaré d’une colonne est une préférence,

appelée affinité par SQLite, tandis que le type appartient en réalité à chaque valeur. Une colonne déclarée `INTEGER` accepte donc un texte qui ne ressemble pas à un nombre.

La frontière avec les alternatives est nette. PostgreSQL, MySQL et SQL Server sont des services partagés auxquels de nombreuses machines accèdent par le réseau. SQLite est un stockage privé pour une application sur un hôte. Face aux fichiers CSV ou JSON, il apporte un langage de requête, des index, des contraintes et des transactions, au prix d'une dépendance que la plupart des langages fournissent déjà.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que SQLite ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Une structure fidèle au fonctionnement de l'entreprise, qui permet encore de poser demain les questions dont vous aurez besoin.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 6 pipelines ETL/ELT conçus et exploités — 6 pipelines ETL sur trois moteurs de bases de données

6 pipelines ETL/ELT conçus et exploités Contactez-nous →

Pourquoi c'est important

Pour une grande part des logiciels, la question difficile n'est pas la vitesse de la base, mais la personne qui l'exploite. Un serveur exige installation, mises à niveau, sécurisation, supervision et sauvegardes ; quelqu'un doit être payé pour ce travail. SQLite supprime toute cette catégorie de tâches. Ouvrir une base revient à ouvrir un fichier. Une sauvegarde est une copie de ce fichier, réalisée avec l'API de sauvegarde lorsqu'il est utilisé, et une suite de tests peut donner à chaque test sa propre base vierge.

Cela change aussi ce que peut être un fichier. Un programme qui inventerait autrement son format, écrirait son parseur et espérerait ne jamais planter au milieu d'un enregistrement peut stocker son état dans SQLite et obtenir sans effort supplémentaire des enregistrements atomiques. Le projet nomme directement ces usages : format de fichier d'application, conteneur de transfert entre systèmes où le destinataire exécute du SQL au lieu d'écrire un parseur, et base temporaire pour trier et résumer des données arrivées sous une autre forme.

La décision essentielle est précise et vérifiable avant la première ligne de code : combien de processus écrivent, sur combien de machines ? Un programme sur un hôte, écrivant par courtes rafales, est le terrain de prédilection de SQLite. De nombreux sites à trafic faible ou moyen l'utilisent sans difficulté. Plusieurs serveurs écrivant les mêmes données, ou une charge où de nombreux utilisateurs écrivent en permanence, relèvent d'une base client-serveur. La taille tranche rarement, puisqu'un fichier peut contenir jusqu'à 281 téraoctets. C'est le nombre de processus d'écriture qui décide.

Même sur ce terrain, certains coûts sont à peser. `ALTER TABLE` ne permet que de renommer une table ou une colonne, ou d'ajouter ou supprimer une colonne. La plupart des autres changements de structure exigent donc de construire une nouvelle table, d'y copier les lignes et d'échanger les noms. Les clés étrangères sont désactivées jusqu'à leur activation par chaque connexion. Enfin, le typage souple laisse entrer discrètement des données incorrectes, sauf si la table est déclarée pour les refuser.

À quoi cela ressemble en pratique

Au stade de développement documenté, vingt-trois tables dans un fichier SQLite, des réalisations et entreprises aux profils et références, constituaient la source de vérité d'un générateur de CV, fiches de références, portfolios et lettres de motivation. Une commande transformait un fichier SQLite vide en cette base en dix-neuf étapes ordonnées. Chacune pouvant être répétée sans risque, reconstruire une base existante n'ajoutait aucun doublon. Les six formats de sortie provenaient de 41 modules Python totalisant 10 580 lignes qui lisaient ce seul fichier, avec exactement deux dépendances d'exécution. Un document destiné à un type de lecteur était une requête sur les réalisations sélectionnées par un profil, plutôt qu'une copie modifiée. Le coût était clair : le système était devenu le seul moyen de produire un document.

Au stade d'export documenté, cette même base alimentait aussi la construction du site, puisque son portfolio était généré à partir d'elle. Un exporteur ouvrait le fichier SQLite en lecture seule et écrivait 495 pages dans trois langues, en n'utilisant que la bibliothèque standard Python. La sortie était versionnée pour permettre au site de se construire sans l'environnement du générateur. Les adresses de pages étaient enregistrées comme des lignes, une par adresse et par réalisation : reformuler une phrase ne déplaçait donc plus la page. Ce travail a aussi révélé le piège décrit dans la section suivante : une comparaison de période sur une colonne de texte correspondait aux trente-deux lignes.

Son rôle le plus modeste, permettre l'inspection de résultats, est aussi le plus typique. Un pipeline transformant des relevés électriques bruts en consommation horaire par zone écrivait ses résultats en CSV ou dans une base SQLite légère pour les examiner. La version Python traitait une journée de données en 34 ms et trente jours en 180 ms. Une base utilisée ainsi ne nécessite aucune préparation : toute personne disposant du shell `sqlite3` peut ouvrir le fichier et l'interroger.

Où les problèmes surviennent

La plupart des difficultés avec SQLite viennent de son utilisation comme un petit PostgreSQL. C'est un mécanisme différent, avec d'autres limites.

Un filtre sur l'année sélectionne toutes les lignes. SQLite ne convertit une valeur avant une comparaison que si l'un des opérandes possède une affinité. Le résultat d'une fonction comme `substr()` n'en possède pas, pas plus qu'un nombre isolé. SQLite compare donc le texte `'2019'` à l'entier `2025` tels quels, et son ordre de tri place tout nombre avant tout texte :

```
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
```

Convertissez explicitement dès qu'un texte rencontre un nombre. Pour les nouvelles tables, déclarez de vrais types et ajoutez le mot-clé `STRICT`, disponible depuis la version 3.37.0. Une colonne `INTEGER` stricte accepte encore `'42'` en le stockant comme un nombre, mais refuse `'ten'` avec une erreur plutôt que de le conserver.

Les écritures échouent avec « database is locked ». Une connexion détient le verrou d'écriture, et une seconde connexion souhaitant écrire attend ou abandonne immédiatement. La cause est généralement une transaction longue, ou un lecteur qui bloque une écriture dans le mode de journal de retour arrière utilisé sans autre réglage. Passez à la journalisation à écriture anticipée, où les

lectures et l'écriture ne se bloquent plus mutuellement, gardez les transactions d'écriture courtes et fixez un délai d'attente lorsque la base est occupée. Une connexion souhaitant écrire attendra ainsi brièvement au lieu de renvoyer une erreur. Si les demandes d'écriture sont réellement nombreuses et constantes, il est temps de passer à un serveur.

Le fichier se corrompt sur un disque partagé. SQLite dépend des verrous de fichiers du système d'exploitation. De nombreux systèmes de fichiers réseau les implémentent avec des défauts : deux machines peuvent alors écrire simultanément la même partie du fichier. La journalisation à écriture anticipée ne fonctionne pas du tout sur un système de fichiers réseau, car tous les processus qui utilisent la base doivent partager de la mémoire sur un même hôte. Conservez la base sur un disque local au seul programme qui l'écrit et placez un serveur devant si d'autres machines ont besoin des données.

Une ligne survit au parent qu'elle référence. Les contraintes de clé étrangère sont déclarées dans le schéma, mais restent inactives jusqu'à ce que chaque connexion exécute `PRAGMA foreign_keys = ON`. Une suppression dans une connexion qui a omis ce pragma laisse donc des orphelins sans protestation. Définissez-le à l'endroit unique où l'application ouvre ses connexions et testez son activation.

Pour aller plus loin

- À propos de SQLite — la description du projet : dans le processus, sans serveur, dans le domaine public et dans un fichier multiplateforme.
- Types de données dans SQLite — classes de stockage, affinité des types et règles de comparaison à l'origine du piège ci-dessus.
- Tables STRICT — faire respecter par une table les types déclarés de ses colonnes.
- Journalisation à écriture anticipée — des lecteurs et une connexion d'écriture en parallèle, et la raison de leur présence sur un seul hôte.
- Usages adaptés à SQLite — ses domaines de pertinence, ceux d'une base client-serveur et une liste de critères.
- Prise en charge des clés étrangères dans SQLite — pourquoi les contraintes sont désactivées par défaut et comment les activer.



Glossaire

Choisissez selon vos contraintes réelles

Présentez un jeu de données représentatif, les questions auxquelles il doit répondre et le goulot d'étranglement actuel. Nous pouvons convenir de la manière de mesurer la qualité, le débit et la reprise.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que SQLite ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/sqlite/