Engineer

Base de données

stockage structuré de données que des programmes interrogent

Ce que c’est

Une base de données est un ensemble de données conservé dans une structure définie, avec des règles sur ce qui peut y entrer, pour que des logiciels trouvent, modifient et vérifient son contenu. Le même mot, souvent abrégé en DB, désigne aussi, par extension, le programme qui les conserve, nommé précisément système de gestion de bases de données, ou SGBD (DBMS en anglais). Le glossaire du NIST donne deux définitions ; la plus large décrit un dépôt d’informations ou de données « qui n’est pas nécessairement un système de base de données relationnelle traditionnel ».

Le mot a trois usages, et une conversation tourne en rond si deux personnes ne l’emploient pas dans le même sens. Il nomme les données : les clients, les commandes et les relevés de capteurs qu’une organisation conserve. Il nomme leur moteur : PostgreSQL, SQL Server, SQLite. Dans un serveur, il nomme aussi un conteneur parmi plusieurs. Le tutoriel PostgreSQL décrit les niveaux : des lignes aux colonnes nommées et typées forment des tables, regroupées en bases, et les bases gérées par une instance serveur forment un cluster de bases de données. SQL Server distingue aussi l’instance et les bases hébergées. « La base est lente » peut vouloir dire une table, un conteneur ou toute une machine.

La plupart des moteurs courants sont relationnels : ils stockent des relations, terme mathématique désignant les tables, et reçoivent les questions en SQL. Le tutoriel PostgreSQL cite d’autres organisations. Les fichiers et répertoires d’un système de type Unix forment une base hiérarchique ; les bases orientées objets sont venues ensuite. Redis se présente comme un serveur de structures de données : au lieu de tables, il propose chaînes, hachages, listes, ensembles et autres types natifs, pour le cache, les files d’attente et le traitement d’événements.

Les moteurs diffèrent aussi par leur lieu d'exécution. Un moteur client/serveur est un processus durable accessible par réseau à de nombreux programmes. Un moteur embarqué est une bibliothèque intégrée à un programme, et SQLite garde toute une base, tables, index, déclencheurs et vues compris, dans un fichier ordinaire. Le projet dit concurrencer `fopen()` plutôt que les bases serveur. La différence apparaît dans la façon dont un programme se connecte :

```
# client/server: connect to a service that is already running
psql "host=db.internal dbname=shop user=app"

# embedded: the program opens a file and the engine runs inside it
sqlite3 shop.db
```

Une troisième famille vise l'analyse. BigQuery stocke séparément chaque colonne, là où de nombreux systèmes traditionnels gardent ensemble sur disque les champs d'une même ligne. Le stockage par lignes convient à la recherche d'un enregistrement ; celui par colonnes à la somme d'un champ sur des millions de lignes, car la requête ne lit que cette colonne et laisse les autres de côté.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'une base de données ? » 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

Les fichiers gardent aussi des données. Une base ajoute des garanties lorsque plusieurs programmes utilisent ces données simultanément.

La première est une vue cohérente lors d'accès concurrents. PostgreSQL donne à chaque instruction un instantané des données à un moment donné : le contrôle de concurrence multiversion, MVCC. Une lecture n'attend donc jamais une écriture, ni une écriture une lecture. SQLite autorise autant de lecteurs

simultanés que voulu, mais un seul processus d'écriture à un instant donné, les autres attendant leur tour. Les deux conviennent à des nombres différents de processus d'écriture, et ce nombre détermine une grande partie du choix.

La deuxième est de garder la forme des données. Le schéma définit les colonnes de chaque table et leurs types ; clés et contraintes définissent les lignes permises. Si la base impose ces règles, tous les programmes qui écrivent les respectent. Si elles ne vivent que dans le code applicatif, le premier programme qui les ignore laisse des lignes qui exigeront des vérifications défensives dans toutes les requêtes ultérieures.

La troisième est une langue commune, avec des accents. SQL est la norme ISO/IEC 9075, révisée pour la dernière fois en SQL:2023. L'annexe PostgreSQL sur la conformité note pourtant qu'aucun système actuel ne revendique une conformité totale à son noyau (Core). Un schéma et ses requêtes appartiennent à leur moteur ; les déplacer vers un autre est une migration avec ses propres tests.

La documentation de SQLite transforme ce choix en courte liste de contrôle : des données auxquelles le code qui émet le SQL doit accéder par réseau, de nombreux processus écrivant au même instant, ou des contenus approchant le téraoctet orientent chacun vers client/serveur. Sinon, SQLite est la réponse la plus simple. Les rapports à grande échelle ajoutent une question : les lectures lourdes doivent-elles aller sur une copie analytique distincte alimentée par ETL, pour qu'un long rapport ne concurrence pas les requêtes attendues par les clients ?

À quoi cela ressemble en pratique

À grande échelle, le travail d'ingénierie de bases de données mené dans une entreprise technologique en forte croissance a consisté à créer, gérer et optimiser 100 bases de données ou plus sur PostgreSQL, Microsoft SQL Server et Google BigQuery, contenant pour la plupart des données SIG et des séries temporelles, comme des relevés de capteurs, des journaux et des métriques en temps réel. Le travail a produit des schémas normalisés et dénormalisés pour PostgreSQL et SQL Server, avec indexation spatiale, PostGIS côté PostgreSQL, et partitionnement temporel. BigQuery stockait les grandes séries temporelles dans des tables partitionnées dans le temps et organisées par clustering. Avec indexation, vues matérialisées et cache en place, les temps de réponse des requêtes sur les données SIG et les séries temporelles ont diminué de 40–60 %, et la supervision automatique a réduit la durée d'indisponibilité de 50 %.

Dans un système plus petit, à un stade de développement documenté, un seul fichier SQLite conservait les faits d'un CV, d'une fiche de références, d'un portfolio et d'une lettre de motivation en trois langues. Cette base comptait vingt-trois tables, et une commande les construisait en dix-neuf étapes ordonnées, chacune pouvant être réexécutée sans risque. Six formats de sortie étaient produits depuis la base par 41 modules Python totalisant 10 580 lignes. Un document destiné à un type de lecteur était « une requête plutôt qu'une copie » : un profil sélectionnait les réalisations à afficher et leur ordre. En contrepartie, un document ne pouvait alors provenir que de cette base. Ces chiffres décrivent ce stade de développement documenté, pas le dépôt actuel ni les langues qu'il prend en charge.

Entre ces deux échelles se pose la question que rencontre toute base très sollicitée, et la note sur l'optimisation avant répartition y répond par un ordre de travail : observer, optimiser, déporter les lectures du primaire, partitionner autour du point chaud, puis ne répartir que lorsqu'une machine est saturée de travail utile. Observer signifie interroger PostgreSQL : `pg_stat_statements` pour les instructions au coût total le plus élevé, puis `EXPLAIN (ANALYZE, BUFFERS)` pour le travail de la plus coûteuse.

Où les problèmes surviennent

La plupart des difficultés trouvent leur origine dans une décision prise tôt, quand le volume de données était encore faible et que la décision semblait gratuite.

Deux copies d'un fait divergent. Le générateur de CV existe parce que des faits conservés dans quatre documents doivent être corrigés quatre fois, jusqu'à ce que deux de ces documents divergent sans que rien n'indique lequel est à jour. Il en va de même pour une table exportée et modifiée dans un tableur ou un second service avec sa liste de clients. Attribuez à chaque fait un stockage de référence unique et indiquez pour chaque autre copie sa provenance et sa date. Le cache de la note suit strictement la règle : il peut dire à l'acheteur qu'un article est entièrement réservé, mais chaque réponse positive vient toujours de la base.

On ajoute des machines avant de lire les plans. Répartir les données entre serveurs étale les instructions lentes sans les rendre moins coûteuses et ajoute une facture durable : la note précise que chaque migration, sauvegarde, restauration et rapport s'exécute une fois par shard, et qu'une jointure entre shards devient du code applicatif. Lisez d'abord `pg_stat_statements` et les plans, puis gardez leurs résultats comme référence de départ.

Une lecture arrive sur une copie en retard. Une réplique en streaming déporte les lectures du primaire, mais ses données ont un léger retard. Un utilisateur relisant ce qu'il vient d'écrire peut donc y trouver l'ancienne valeur. Lisez sur le primaire les données que l'utilisateur vient lui-même d'écrire.

Les rapports et les clients partagent un moteur. Un rapport parcourant deux ans d'historique occupe mémoire et disque nécessaires aux requêtes de la boutique, et les acheteurs le ressentent. Déplacez l'analyse lourde vers une copie dans un entrepôt, et gardez la base opérationnelle pour ses nombreuses petites lectures et écritures.

Pour aller plus loin

- Glossaire NIST : base de données — deux courtes définitions de publications NIST, dont une admet les bases non relationnelles.
- Tutoriel PostgreSQL : concepts — tables, lignes, colonnes, bases et cluster de bases, sur une page.
- PostgreSQL : introduction à MVCC — des instantanés pour faire travailler processus de lecture et d'écriture sans blocage mutuel.
- PostgreSQL : conformité SQL — parties et versions d'ISO/IEC 9075 et sens de la conformité au noyau (Core).
- Usages adaptés à SQLite — où convient un moteur embarqué, où client/serveur est préférable et une liste de critères de choix.
- Présentation du stockage BigQuery — stockage par lignes et par colonnes, et raisons pour lesquelles l'analyse privilégie le second.

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 qu'une base de données ? » 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/database/