Engineer

SQL

Langage des bases de données relationnelles

Ce que c’est

SQL est un langage déclaratif servant à définir, modifier et lire des données conservées dans des tables relationnelles : on décrit le résultat voulu, et la base décide comment le produire. Les lettres sont généralement développées en Structured Query Language et se prononcent soit une par une, soit comme le mot anglais « sequel ». La norme internationale qui le définit, ISO/IEC 9075, l’appelle simplement Database Language SQL.

Déclaratif est le mot essentiel. L’instruction ci-dessous nomme une table, une condition, un regroupement et un ordre. Elle ne dit rien des index, des boucles ou des lignes à lire en premier :

```
SELECT country, count(*) AS signups
FROM accounts
WHERE created_at >= now() - interval '30 days'
GROUP BY country
ORDER BY signups DESC;
```

Dans PostgreSQL, ce texte traverse quatre étapes. Un parseur vérifie la syntaxe et construit un arbre ; une réécriture développe les vues mentionnées dans l’instruction ; un planificateur compare les voies d’exécution possibles, comme un parcours séquentiel ou un accès par index ; enfin, un exécuteur récupère les lignes. La même phrase peut prendre une milliseconde ou une minute selon ce que trouve le planificateur. Lire un plan fait donc partie de l’écriture du SQL.

Le langage comporte des parties nommées séparément. La définition des données crée et modifie les tables, schémas et index. La manipulation des données insère, met à jour, supprime et fusionne les lignes. Les requêtes les lisent avec SELECT. Les autorisations et les transactions déterminent qui peut

faire quoi et quels changements sont appliqués ensemble. Le tout circule sous forme de texte par une seule connexion : SQL constitue ainsi le terrain commun d'une application, d'un outil de reporting et d'une personne devant un terminal.

La norme a été révisée à de nombreuses reprises, la plus récente étant SQL:2023, et chaque moteur en implémente son propre sous-ensemble. L'annexe de conformité de PostgreSQL indique qu'il respecte au moins 170 des 177 fonctionnalités obligatoires du noyau, Core, et qu'au moment de sa rédaction aucun système de base de données ne revendiquait une conformité Core complète. Chaque moteur ajoute aussi son dialecte : Microsoft SQL Server parle Transact-SQL, BigQuery parle GoogleSQL et PostgreSQL ajoute PL/pgSQL pour le code procédural exécuté dans la base.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que SQL ? » 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

Avec SQL, l'équipe doit décider où traiter les données : dans la base, dans le langage qu'elle est conçue pour exécuter, ou dans le code applicatif qui récupère les lignes et les traite lui-même.

Une instruction ensembliste s'exécute près des données. Un filtrage dans la base qui ramène un million de lignes à quarante ne transmet que quarante lignes sur le réseau ; effectué dans l'application, il en transmet un million et en écarte la majorité. Jointures, agrégations et dédoublonnage obéissent à la même arithmétique. Déplacer ce travail vers SQL est souvent l'accélération la moins coûteuse pour un système lent, car elle utilise la base déjà en place.

La responsabilité de maintenance fait contrepoids. La logique écrite en SQL réside dans des migrations et des fonctions qui exigent les mêmes revues, tests et gestion de versions que tout autre code. Moins d'ingénieurs lisent une longue fonction PL/pgSQL avec la même aisance que du Go ou du Python. Une

répartition viable confie à la base ce qui doit examiner de nombreuses lignes et garde dans l'application les règles qui évoluent avec l'activité, où elles sont plus faciles à tester isolément.

La portabilité constitue le deuxième compromis. Le cœur du langage — SELECT avec jointures, GROUP BY et ORDER BY — passe d'un moteur à l'autre avec peu de modifications. Les aspects périphériques se transposent moins bien : calculs de dates, insertions avec mise à jour en cas de conflit, pagination, fonctions JSON et code procédural diffèrent tous. Une migration d'une base à l'autre y consacre l'essentiel de son effort. Choisir un moteur revient aussi à choisir un dialecte ; plus la logique repose sur ce dialecte, plus une migration ultérieure coûte cher.

Le recrutement est le troisième aspect. SQL est l'une des rares compétences communes aux analystes, développeurs backend, ingénieurs de données et administrateurs de bases. Chacun peut donc lire et revoir une couche de données écrite en SQL ordinaire. Un ORM masque le SQL jusqu'à ce qu'une page devienne lente ; quelqu'un doit alors tout de même lire ce qu'il a généré.

À quoi cela ressemble en pratique

Sur une plateforme, des fonctions PostgreSQL assemblent chaque réponse API en un document JSON complet. Le gestionnaire Go la transmet comme `json.RawMessage`, sans décodage ni réencodage. Cette architecture de transmission directe du JSON fait de la mise en forme de la réponse une responsabilité SQL, là où résident déjà les données. Un petit utilitaire QueryJSON a fait de ce modèle le plus facile à suivre, et les DTO, mappers et structures de réponse qu'accumule habituellement une API en couches ont disparu. Le code des gestionnaires est devenu beaucoup plus court, et un changement de ce que renvoie une fonction atteint le client sans modification d'un gestionnaire.

Le même langage prend une autre forme dans un pipeline de données. Dans six pipelines ETL/ELT couvrant BigQuery, MSSQL et PostgreSQL, des fichiers bruts étaient chargés dans des schémas de préparation avec bcp, psql et SSIS. Le nettoyage, la normalisation et l'enrichissement étaient écrits sous forme de procédures PL/pgSQL et T-SQL. Trois moteurs, chacun avec son dialecte, dans un seul traitement : la question de portabilité précédente, vue de l'intérieur. La page consigne le résultat : au moins trois heures de travail manuel ramenées à moins de 20 minutes de bout en bout, des données actualisées chaque jour plutôt que chaque semaine et des API Python consommant ces données 30 % plus rapides.

Où les problèmes surviennent

Assembler des instructions à partir de chaînes. Un client nommé O'Brien casse une page de recherche, ou une saisie conçue à cet effet lit les données d'autrui. La cause est une entrée collée dans le texte SQL, où la base ne peut distinguer donnée et commande. Transmettez plutôt les valeurs comme paramètres, séparément de l'instruction, comme le permettent tous les pilotes courants. Dans la bibliothèque cliente de PostgreSQL, un appel paramétré n'accepte en outre qu'une commande par chaîne : une seconde ligne de défense contre l'injection.

Un ordre qui ne tenait qu'au hasard. Une liste triée pendant des mois arrive dans le désordre après un chargement ou un nouveau plan. Sans ORDER BY, PostgreSQL renvoie les lignes dans l'ordre le plus rapide à produire, et un LIMIT sans ORDER BY peut même renvoyer un sous-ensemble différent à chaque fois. Tout résultat dont l'ordre compte doit l'indiquer dans la requête.

Traiter NULL comme une valeur. Un filtre écrit `WHERE discount = NULL` ne renvoie rien, pas plus que `NOT IN` face à une sous-requête contenant un NULL. Toute comparaison avec NULL produit un résultat inconnu, plutôt que vrai ou faux, et une clause WHERE ne garde que les lignes dont la condition est vraie. Écrivez `IS NULL` et préférez `NOT EXISTS` à `NOT IN` lorsque la colonne interne peut contenir un NULL.

Un dialecte s'invite dans les tests. Une suite passe sous SQLite, puis le même code échoue sous PostgreSQL en production ; ou une migration vers SQL Server bute sur chaque `LIMIT`, que ce moteur exprime avec `TOP` ou `OFFSET ... FETCH`. Chaque moteur accepte des constructions refusées par les autres. Testez avec le moteur déployé et concentrez autant que possible la syntaxe qui lui est propre.

Pour aller plus loin

- Partie II : le langage SQL — le tutoriel et la référence complets de PostgreSQL, de la syntaxe aux conseils de performances.
- Annexe D : conformité SQL — les parties d'ISO/IEC 9075 et les fonctionnalités implémentées par PostgreSQL.
- Référence Transact-SQL — le dialecte SQL Server documenté par Microsoft.
- Introduction à SQL dans BigQuery — GoogleSQL et l'ancien dialecte qu'il a remplacé.

- Langage de requête compris par SQLite — la syntaxe d’un moteur embarqué, instruction par instruction.

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 SQL ? » 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/sql/