Engineer

BigQuery

l'entrepôt de Google pour interroger de très grandes tables

Ce que c’est

BigQuery est un service géré d’analyse SQL sur Google Cloud : personne ne provisionne de serveur pour lui, les tables sont stockées colonne par colonne et chaque requête est facturée soit selon les octets lus, soit au moyen d’une capacité de calcul réservée à l’avance. Il est aussi commercialisé et recherché sous le nom Google BigQuery. Son langage SQL est GoogleSQL, un dialecte qui suit la norme ANSI et y ajoute ses propres fonctions.

Sa conception repose sur la séparation du stockage et du calcul. Les tables résident dans la couche de stockage de Google, et une requête emprunte des ressources de calcul pendant son exécution. Chaque colonne étant stockée séparément, une requête qui nomme trois colonnes d’une table de cinquante colonnes ne lit que ces trois-là et laisse les autres sur disque. L’optimisation diffère donc de celle d’un serveur PostgreSQL. Plutôt que de concevoir des index B-tree et de dimensionner la mémoire, on travaille sur l’organisation de la table : elle peut être partitionnée selon une colonne de date ou d’horodatage, l’heure d’ingestion ou une plage d’entiers, et regroupée par clustering sur un maximum de quatre colonnes, pour que les lignes apparentées partagent les mêmes blocs de stockage.

BigQuery appartient à la famille des entrepôts de données. On le compare généralement à Snowflake et Amazon Redshift, ainsi qu’à la possibilité de garder l’analyse dans la base transactionnelle que l’entreprise exploite déjà. Il est conçu pour les requêtes qui parcourent des millions ou des milliards de lignes et renvoient quelques totaux. Les recherches et mises à jour d’une seule ligne qu’une application effectue à chaque requête relèvent d’une base comme PostgreSQL ou SQL Server. Une architecture courante associe les deux, avec un pipeline de traitement transportant les données de l’une à l’autre.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que BigQuery ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Réunir les chiffres des ventes, des opérations et des finances pour que deux services ne présentent plus deux réponses à la même réunion.

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. Pipeline géographique accéléré de 50 fois — 50x d'accélération des pipelines de données géographiques

Pipeline géographique accéléré de 50 fois Contactez-nous →

Pourquoi c'est important

BigQuery amène une équipe à décider comment elle souhaite payer l'analyse. La facture suit les requêtes écrites, plutôt que la taille d'une machine choisie autrefois, et cela joue dans les deux sens.

Avec la tarification à la demande, chaque requête est facturée selon les octets qu'elle traite. Cela convient à un travail irrégulier : une équipe exécutant quelques rapports lourds par semaine paie ces rapports et rien entre les deux, sans serveur inutilisé pendant la nuit. Une requête négligente a toutefois elle aussi un prix. Un `SELECT *` sur une année de relevés de capteurs coûte autant, que l'analyste ait voulu une ligne ou toutes. La tarification à la capacité remplace ce modèle par des slots, unités de calcul achetées au moyen de réservations. La facture devient alors un montant connu à l'avance, et il est plus facile de prévoir à moindre coût un usage intensif et régulier. Un projet peut passer d'un modèle à l'autre : le choix initial peut donc être revu une fois l'usage réel connu.

Le stockage a ses propres règles. Une table ou une partition non modifiée pendant 90 jours consécutifs passe au tarif de longue durée, inférieur de 50 %, sans que personne ne la déplace. Chaque jeu de données est facturé soit selon la taille logique, non compressée, de ses données, soit selon leur taille physique, compressée. Pour des données qui se compressent bien, le second modèle peut être moins cher.

Trois points méritent réflexion avant de s'engager. L'emplacement d'un jeu de données, une région unique ou la multirégion US ou EU, est fixé à sa création et reste immuable : c'est donc une décision de conformité prise dès le premier jour. Le dialecte, l'outil en ligne de commande `bq` et les bibliothèques clientes sont propres à Google. Les requêtes nécessitent donc des modifications pour passer à un autre entrepôt, plutôt que d'y être transférées telles quelles ; les données,

elles, sortent facilement, puisque les tables s'exportent en CSV, Avro et Parquet. Enfin, la question du recrutement est moins difficile qu'elle n'y paraît : des analystes connaissant SQL sont opérationnels en quelques jours, tandis que la compétence plus rare consiste à lire une requête en octets autant qu'en lignes.

Quand BigQuery est-il donc le bon choix ? Lorsque les données arrivent en grands volumes, principalement par ajout, comme les événements, les journaux ou les relevés de compteurs ; lorsque les questions changent d'une semaine à l'autre ; et lorsque l'équipe préfère éviter d'exploiter des serveurs de base de données pour l'analyse. Pour des données qu'un seul serveur PostgreSQL contient sans difficulté, un entrepôt ajoute un deuxième système, une deuxième facture et un pipeline de traitement entre les deux. Son intérêt grandit avec le volume et le nombre de personnes posant des questions.

À quoi cela ressemble en pratique

Un parc de plus de 100 bases de données, réparties entre PostgreSQL, Microsoft SQL Server et Google BigQuery, contenait surtout des données géographiques ou des séries temporelles : coordonnées, relevés de capteurs, journaux et métriques en temps réel. BigQuery gérait les séries temporelles à grande échelle, avec des tables partitionnées dans le temps et organisées par clustering, la compression et le stockage en colonnes servant à limiter le coût du stockage et à accélérer les lectures. Les parcours de tables partitionnées et organisées par clustering dans BigQuery, associés à l'indexation, aux vues matérialisées et à la mise en cache dans l'ensemble du parc, ont réduit de 40–60 % les temps de réponse des requêtes GIS et de séries temporelles.

Sur un pipeline de données géographiques devenu lent, le SQL et les modèles de données de PostgreSQL, MS SQL et Google Cloud BigQuery ont d'abord été analysés. Cela a révélé une indexation insuffisante, des requêtes inefficaces et des contraintes manquantes. Chaque moteur a ensuite reçu les outils dont il dispose : index B-tree et GiST dans PostgreSQL, index filtrés dans MS SQL et clustering dans BigQuery, en parallèle de modèles normalisés, du partitionnement et de requêtes réécrites. Le dossier de réalisations de la fondatrice indique que le pipeline dans son ensemble s'est exécuté 50 fois plus vite ; il ne précise pas le jeu de données du benchmark, le matériel ni la procédure de comparaison.

Sur des données de compteurs intelligents provenant de plusieurs régions, plus de 500 rapports d'analyse ont été écrits entièrement en SQL sur PostgreSQL et BigQuery, avec des CTE, des fonctions de fenêtre, des sous-requêtes et des jointures géospatiales portant sur les identifiants de compteurs, horodatages, coordonnées et métadonnées environnementales. Chaque rapport était livré avec son SQL annoté, pour qu'un collègue puisse le réexécuter et l'auditer, ainsi qu'avec des notes sur les problèmes connus de qualité des données, comme les coordonnées GPS manquantes et les valeurs de compteurs corrompues. Les rapports sont devenus une référence entre services et ont contribué à améliorer de 10–15 % l'efficacité de la planification opérationnelle.

Dans les trois cas, BigQuery fonctionnait à côté de PostgreSQL plutôt qu'à sa place, et chaque moteur recevait le travail qu'il accomplit le mieux.

Où les problèmes surviennent

Une requête `LIMIT 10` est facturée pour toute la table. Sur une table sans clustering, `LIMIT` réduit le résultat, pas les octets lus, et `SELECT *` lit toutes les colonnes. Un coup d'œil aux données devient un parcours complet de la table. Nommez les colonnes nécessaires à la question et, pour consulter quelques lignes d'exemple, utilisez l'aperçu de table de la console ou `bq head`, qui ne coûtent rien.

Une table partitionnée est tout de même parcourue de bout en bout. Des partitions ne sont écartées que si le filtre sur la colonne de partitionnement est une constante que le moteur peut évaluer avant la lecture. Comparer cette colonne au résultat d'une sous-requête, l'envelopper dans une fonction ou effectuer un calcul sur elle fait parcourir toutes les partitions à la requête. Deux protections aident. Une table peut exiger un filtre de partition : une requête dépourvue de filtre exploitable s'arrête alors sur une erreur au lieu de s'exécuter. Une simulation indique aussi les octets qu'une requête traiterait, sans l'exécuter ni la facturer :

```
bq query --use_legacy_sql=false --dry_run \
  'SELECT meter_id, kwh
   FROM energy.readings
   WHERE reading_date = DATE "2026-10-01"'
```

Le clustering est ajouté, mais les octets lus diminuent à peine. Le clustering trie les blocs de stockage selon un maximum de quatre colonnes, dans l'ordre indiqué, et cet ordre compte : une table organisée par date de commande, pays et statut n'est pas structurée pour un filtre portant uniquement sur pays et statut. Les tables de moins de 64 Mo en tirent peu de bénéfice. Sur une table organisée par clustering, les octets réellement parcourus ne sont connus qu'une fois la requête terminée. Ordonnez les colonnes de clustering comme les filtres des requêtes. Si le coût doit être plafonné avant toute exécution, appuyez-vous sur le partitionnement, dont l'élimination des partitions est déjà prise en compte dans une simulation, et fixez un maximum d'octets facturables pour qu'une requête dépassant la limite s'arrête avant d'être facturée.

Une requête joignant deux jeux de données est refusée. Chaque jeu de données réside dans un emplacement et une requête s'exécute dans un emplacement : tous les jeux de données qu'elle touche doivent donc le partager. Les équipes laissant chaque projet utiliser l'emplacement prévu en l'absence de choix explicite se retrouvent avec des tables qui ne peuvent être jointes qu'après copie de l'une d'elles dans l'autre emplacement. Choisissez l'emplacement une fois, en fonction de la conformité et de la provenance des données, puis créez-y délibérément chaque jeu de données.

Pour aller plus loin

- Présentation de BigQuery — la description du service par Google, ses couches séparées de stockage et de calcul et ce qui est géré pour vous.
- Introduction aux tables partitionnées — les types de partitionnement, leur granularité et leurs limites.
- Interroger des tables partitionnées — les filtres qui éliminent des partitions et ceux qui font toutes les parcourir.
- Introduction aux tables organisées en clusters — l'effet de l'ordre des colonnes sur l'élimination de blocs et ses conséquences pour l'estimation du coût.
- Estimer et maîtriser les coûts — simulations, maximum d'octets facturables et moyens gratuits de prévisualiser une table.
- Emplacements BigQuery — régions, multirégions et règles des requêtes qui touchent plusieurs jeux de données.

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 BigQuery ? » 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/bigquery/