**Engineer**

# BigQuery

Googles Data Warehouse für Abfragen auf sehr großen Tabellen

## Was es ist

BigQuery ist ein verwalteter SQL-Analysedienst auf Google Cloud: Niemand stellt dafür einen Server bereit, Tabellen werden spaltenweise gespeichert, und jede Abfrage wird entweder nach den gelesenen Bytes oder über vorab reservierte Rechenkapazität abgerechnet. Der Dienst wird auch als Google BigQuery verkauft und unter diesem Namen gesucht. Seine SQL-Sprache heißt GoogleSQL, ein Dialekt, der dem ANSI-Standard folgt und eigene Funktionen ergänzt.

Der Aufbau beruht auf der Trennung von Speicherung und Berechnung. Tabellen liegen in Googles Speicherschicht, und eine Abfrage belegt Rechenleistung für die Dauer ihrer Ausführung. Weil jede Spalte getrennt gespeichert wird, liest eine Abfrage, die drei Spalten einer Tabelle mit fünfzig Spalten nennt, nur diese drei und lässt den Rest auf der Platte. Die Optimierung sieht deshalb anders aus als bei einem PostgreSQL-Server. Statt B-Baum-Indizes zu entwerfen und den Arbeitsspeicher zu dimensionieren, wird an der Tabellenstruktur gearbeitet: Sie lässt sich anhand einer Datums- oder Zeitstempelspalte, nach Eingangszeit oder nach einem Ganzzahlbereich partitionieren und anhand von bis zu vier Spalten clustern, damit zusammengehörige Zeilen dieselben Speicherblöcke belegen.

BigQuery gehört zur Familie der Data Warehouses. Üblicherweise wird es mit Snowflake und Amazon Redshift verglichen, daneben mit der Möglichkeit, die Analyse auf der bereits betriebenen Transaktionsdatenbank des Unternehmens zu belassen. Es ist für Abfragen gebaut, die Millionen oder Milliarden Zeilen durchlaufen und eine Handvoll Summen zurückgeben. Die Einzelzeilenabfragen und Aktualisierungen, die eine Anwendung bei jeder Anfrage ausführt, gehören in eine Datenbank wie PostgreSQL oder SQL Server. Häufig werden beide Systeme betrieben und die Daten über eine Pipeline von einem zum anderen übertragen.

### Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist BigQuery?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Vertriebs-, Betriebs- und Finanzdaten werden zusammengeführt, damit zwei Abteilungen nicht länger mit zwei Antworten in dieselbe Besprechung kommen.

**Unsere Leistungen entdecken** Kontakt aufnehmen →

### Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. Geografische Datenpipeline 50-mal schneller — 50x schnellere Geodaten-Pipelines

**Geografische Datenpipeline 50-mal schneller** Kontakt aufnehmen →

## Warum es wichtig ist

BigQuery stellt ein Team vor die Entscheidung, wie es für Analysen bezahlen möchte. Die Rechnung richtet sich nach den geschriebenen Abfragen statt nach der Größe eines irgendwann ausgewählten Rechners. Das wirkt in beide Richtungen.

Bei der nutzungsabhängigen Abrechnung wird jede Abfrage nach den verarbeiteten Bytes berechnet. Das passt zu unregelmäßiger Arbeit: Ein Team mit wenigen aufwendigen Berichten pro Woche bezahlt diese Berichte und nichts für die Zeit dazwischen; nachts steht kein Server untätig herum. Damit bekommt aber auch eine unbedachte Abfrage einen Preis. Ein `SELECT *` über ein Jahr Sensormesswerte kostet gleich viel, ob die analysierende Person eine einzige Zeile oder alle haben wollte. Das Kapazitätsmodell ersetzt dies durch Slots, über Reservierungen gekaufte Recheneinheiten. Die Rechnung wird so zu einer vorab bekannten Größe, und eine hohe, gleichmäßige Nutzung lässt sich kostengünstiger planen. Ein Projekt kann zwischen beiden Modellen wechseln. Die erste Wahl lässt sich deshalb überdenken, sobald die tatsächliche Nutzung sichtbar ist.

Für Speicher gelten eigene Regeln. Eine Tabelle oder Partition, die 90 aufeinanderfolgende Tage unverändert bleibt, wechselt ohne Zutun zum Langzeittarif, der 50 Prozent niedriger liegt. Jeder Datenbestand wird entweder anhand seiner logischen, unkomprimierten Größe oder seiner physischen, komprimierten Größe abgerechnet. Bei gut komprimierbaren Daten kann das zweite Modell günstiger sein.

Drei Dinge verdienen vor dem Einstieg besondere Aufmerksamkeit. Der Standort eines Datenbestands, eine einzelne Region oder die Multiregion US beziehungsweise EU, wird bei seiner Anlage festgelegt und bleibt

unveränderlich. Damit fällt am ersten Tag eine Compliance-Entscheidung. Der Dialekt, das Kommandozeilenwerkzeug `bq` und die Clientbibliotheken stammen von Google. Abfragen ziehen deshalb mit Anpassungen in ein anderes Warehouse um, statt unverändert zu bleiben. Die Daten selbst lassen sich leicht herausbringen, weil Tabellen als CSV, Avro und Parquet exportiert werden können. Die Personalfrage ist weniger schwierig, als sie aussieht: Wer SQL beherrscht, wird als Analyst innerhalb von Tagen produktiv. Seltener ist die Fähigkeit, eine Abfrage nicht nur in Zeilen, sondern auch in Bytes zu lesen.

Wann ist BigQuery also die passende Wahl? Wenn große, überwiegend durch Anhängen wachsende Datenmengen eintreffen, etwa Ereignisse, Protokolle oder Zählerstände, wenn sich die Fragen daran von Woche zu Woche ändern und wenn das Team keine Datenbankserver für die Analyse betreiben möchte. Für Daten, die bequem auf einen einzelnen PostgreSQL-Server passen, bringt ein Warehouse ein zweites System, eine zweite Rechnung und eine Pipeline zwischen beiden mit. Mit dem Datenvolumen und der Zahl der Fragenden wächst der Nutzen.

## Wie es in der Praxis aussieht

Ein Bestand aus mehr als 100 Datenbanken, verteilt auf PostgreSQL, Microsoft SQL Server und Google BigQuery, enthielt vorwiegend geografische Daten und Zeitreihen: Koordinaten, Sensormesswerte, Protokolle und Echtzeitmetriken. BigQuery übernahm die umfangreichen Zeitreihen. Seine Tabellen waren zeitlich partitioniert und geclustert, mit Komprimierung und spaltenweiser Speicherung für niedrige Speicherkosten und schnelle Scans. Partitionierte und geclusterte Scans in BigQuery senkten zusammen mit Indizes, materialisierten Sichten und Caching im gesamten Bestand die Antwortzeiten von GIS- und Zeitreihenabfragen um 40–60 %.

Bei einer langsam gewordenen Pipeline für geografische Daten wurden zuerst SQL und Datenmodelle in PostgreSQL, MS SQL und Google Cloud BigQuery untersucht. Dabei kamen schwache Indizierung, ineffiziente Abfragen und fehlende Constraints zum Vorschein. Jede Engine erhielt dann das zu ihr passende Werkzeug: B-Baum- und GiST-Indizes in PostgreSQL, gefilterte Indizes in MS SQL und Clustering in BigQuery, ergänzt durch normalisierte Modelle, Partitionierung und neu geschriebene Abfragen. Der Leistungsnachweis der Gründerin berichtet, dass die Pipeline als Ganzes 50-fach schneller lief; Benchmarkdatensatz, Hardware und Vergleichsverfahren werden nicht angegeben.

Für Smart-Meter-Daten aus mehreren Regionen wurden mehr als 500 Analyseberichte in reinem SQL geschrieben, gegen PostgreSQL und BigQuery. Sie verwendeten CTEs, Fensterfunktionen, Unterabfragen und räumliche Joins über Zählerkennungen, Zeitstempel, Koordinaten und Umweltmetadaten. Jeder Bericht wurde mit kommentiertem SQL ausgeliefert, damit Kollegen ihn erneut ausführen und prüfen konnten, sowie mit Hinweisen auf bekannte Datenqualitätsprobleme wie fehlende GPS-Koordinaten und beschädigte Zählerwerte. Die Berichte wurden abteilungsübergreifend zur Referenz und trugen zu einer Verbesserung der Effizienz in der Betriebsplanung um 10–15 % bei.

In allen drei Fällen arbeitete BigQuery neben PostgreSQL statt an seiner Stelle. Jede Engine erhielt die Arbeit, die sie am besten erledigt.

## Wo es schiefgeht

**Eine Abfrage mit `LIMIT 10` wird für die ganze Tabelle berechnet.** Bei einer Tabelle ohne Clustering beschränkt `LIMIT` das Ergebnis, nicht die gelesenen Bytes. `SELECT *` liest sämtliche Spalten. Ein kurzer Blick auf die Daten wird zum vollständigen Scan. Nennen Sie die Spalten, die für die Frage nötig sind. Für einen Blick auf Beispielzeilen eignen sich die Tabellenvorschau der Konsole oder `bq head`, die nichts kosten.

**Eine partitionierte Tabelle wird trotzdem vollständig gescannt.** Partitionen werden nur übersprungen, wenn der Filter auf der Partitionierungsspalte eine Konstante ist, die die Engine vor dem Lesen auswerten kann. Der Vergleich mit dem Ergebnis einer Unterabfrage, das Einbetten der Spalte in eine Funktion oder das Rechnen mit ihr schickt die Abfrage durch jede Partition. Zwei Schutzmaßnahmen helfen. Eine Tabelle kann einen Partitionsfilter verlangen. Eine Abfrage ohne brauchbaren Filter stoppt dann mit einem Fehler, statt zu laufen. Außerdem meldet ein Probelauf die Bytes, die eine Abfrage verarbeiten würde, ohne sie auszuführen oder dafür Gebühren zu berechnen:

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

**Clustering wird ergänzt, die gelesenen Bytes ändern sich kaum.** Clustering sortiert Speicherblöcke anhand von bis zu vier Spalten in einer festgelegten Reihenfolge. Die Reihenfolge zählt: Eine nach Bestelldatum, Land und Status geclusterte Tabelle ist nicht für einen Filter nur auf Land und Status organisiert. Tabellen unter 64 MB gewinnen wenig dadurch. Bei einer geclusterten Tabelle stehen die tatsächlich gescannten Bytes erst nach Abschluss der Abfrage fest. Ordnen Sie die Clusterspalten so an, wie die Abfragen filtern. Müssen die Kosten schon vor der Ausführung begrenzt sein, stützen Sie sich auf Partitionierung: Ein Probelauf berücksichtigt bereits, welche Partitionen dabei übersprungen werden. Legen Sie außerdem eine Höchstzahl abrechenbarer Bytes fest, damit eine darüberliegende Abfrage stoppt, bevor sie in Rechnung gestellt wird.

**Eine Abfrage, die zwei Datenbestände verknüpft, wird abgewiesen.** Jeder Datenbestand liegt an einem Standort, und eine Abfrage läuft an einem Standort. Alle von ihr berührten Datenbestände müssen deshalb dort liegen. Teams, die jedem Projekt seine eigene Voreinstellung überlassen, erhalten Tabellen, die erst nach dem Kopieren einer davon an den anderen Standort verbunden werden können. Wählen Sie den Standort einmal anhand der Compliance-Anforderungen und der Datenherkunft. Legen Sie dann jeden Datenbestand bewusst dort an.

## Weiterführende Lektüre

- Überblick über BigQuery — Googles eigene Beschreibung des Dienstes, seiner getrennten Speicher- und Rechenschichten und der Aufgaben, die der Anbieter übernimmt.
- Einführung in partitionierte Tabellen — Partitionierungsarten, ihre Granularität und ihre Grenzen.
- Partitionierte Tabellen abfragen — welche Filter Partitionen überspringen und welche eine Abfrage durch alle Partitionen schicken.
- Einführung in geclusterte Tabellen — wie die Spaltenreihenfolge das Überspringen von Blöcken bestimmt und was das für Kostenschätzungen bedeutet.
- Kosten schätzen und begrenzen — Probeläufe, die Höchstzahl abrechenbarer Bytes und die kostenlosen Möglichkeiten zur Tabellenvorschau.
- BigQuery-Standorte — Regionen, Multiregionen und die Regeln für Abfragen über mehrere Datenbestände.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Bringen Sie einen repräsentativen Datensatz, die Fragen, die er beantworten muss, und den aktuellen Engpass mit. Wir können vereinbaren, wie Qualität, Durchsatz und Wiederherstellung gemessen werden.

**Preise ansehen** Kontakt aufnehmen →

# Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist BigQuery?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

**Unsere Leistungen entdecken**

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/bigquery/