Engineer

BigQuery

🏛 Googles varehus for spørringer mot svært store tabeller

Hva det er

BigQuery er en administrert SQL-tjeneste for analyse på Google Cloud. Det er ingen server å sette opp, tabellene lagres kolonne for kolonne, og hver spørring betales enten etter hvor mange byte den leser, eller av regnekraft som er reservert på forhånd. Tjenesten selges og søkes også etter som Google BigQuery, og SQL-dialekten den snakker, heter GoogleSQL, som følger ANSI-standarden og legger til egne funksjoner.

Designet hviler på at lagring og beregning holdes atskilt. Tabellene ligger i Googles lagringslag, og en spørring låner regnekraft så lenge den kjører. Fordi hver kolonne lagres for seg, leser en spørring som nevner tre kolonner i en tabell med femti, bare de tre og lar resten ligge på disken. Finjustering ser derfor annerledes ut enn på en PostgreSQL-server. I stedet for å utforme B-tree-indekser og dimensjonere minne handler arbeidet om formen på tabellen: den kan partisjoneres etter en dato- eller tidsstempelkolonne, etter innlastingstid eller etter et heltallsintervall, og klynges (clustering) på opptil fire kolonner, slik at rader som hører sammen, deler lagringsblokker.

BigQuery hører til familien av datavarehus, og produktene det som regel veies mot, er Snowflake og Amazon Redshift, pluss muligheten til å holde analysen på transaksjonsdatabasen et selskap allerede kjører. Det er bygd for spørringer som går gjennom millioner eller milliarder av rader og returnerer en håndfull summer. Enkeltoppslagene og oppdateringene en applikasjon gjør ved hver forespørsel, hører hjemme i en database som PostgreSQL eller SQL Server, og et vanlig oppsett kjører begge, med en pipeline som fører data fra den ene til den andre.

Bruk begrepet i en beslutning

Bruk «Hva er BigQuery?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Tall fra salg, drift og økonomi ført sammen, slik at to avdelinger slutter å komme med to ulike svar til det samme møtet.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Gjorde en geodatapipeline 50 ganger raskere — 50x raskere pipelines for geografiske data

Gjorde en geodatapipeline 50 ganger raskere Ta kontakt →

Hvorfor det er viktig

Beslutningen BigQuery legger foran et team, er hvordan det vil betale for analyse. Regningen følger spørringene folk skriver, og ikke størrelsen på en maskin noen en gang valgte, og det virker begge veier.

Med prising etter bruk (on-demand) belastes hver spørring for de bytene den behandler. Det passer arbeid som kommer ujevnt: et team som kjører noen tunge rapporter i uken, betaler for de rapportene og ikke for noe imellom, uten en server som står og går om natten. Det setter også en pris på en uforsiktig spørring. En `SELECT *` over et år med sensoravlesninger koster det samme enten analytikeren ville ha én rad eller alle. Kapasitetsprising erstatter dette med slots, enheter med regnekraft kjøpt gjennom reservasjoner. Da blir regningen et tall som er kjent på forhånd, og tung, jevn bruk blir billigere å planlegge for. Et prosjekt kan bytte mellom de to modellene, så det første valget kan tas opp igjen når den faktiske bruken er synlig.

Lagring har egne regler. En tabell eller partisjon som ikke er endret på 90 dager på rad, går over til langtidsprisen, som er 50 % lavere, uten at noen flytter den. Hvert datasett faktureres enten etter den logiske, ukomprimerte størrelsen på dataene eller etter den fysiske, komprimerte, og for data som lar seg komprimere godt, kan den andre modellen bli den billigste.

Tre ting fortjener vekt før du velger. Plasseringen til et datasett, én region eller multiregionen US eller EU, settes når datasettet opprettes, og ligger fast etter det, så den er en beslutning om regeletterlevelse som tas den første dagen. Dialekten, kommandolinjeverktøyet `bq` og klientbibliotekene er Googles egne, så spørringer flytter til et annet datavarehus med endringer og ikke uendret; selve dataene kommer seg lett ut, siden tabeller kan eksporteres til CSV, Avro og

Parquet. Og spørsmålet om å finne folk er enklere enn det ser ut: analytikere som kan SQL, er produktive i løpet av dager, mens den sjeldnere ferdigheten er å lese en spørring i byte like mye som i rader.

Så når er BigQuery det riktige valget? Når data kommer i store mengder som for det meste bare legges til, som hendelser, logger eller måleravlesninger, når spørsmålene som stilles til dem, skifter fra uke til uke, og når teamet helst slipper å drive databaseservere for analyse. For data en enkelt PostgreSQL-server rommer uten problemer, legger et datavarehus til et system nummer to, en regning nummer to og en pipeline mellom dem. Argumentet for det vokser med datamengden og med hvor mange som stiller spørsmål.

Slik ser det ut i praksis

BigQuery kom inn ved siden av PostgreSQL og ikke i stedet for det i alle de tre prosjektene nedenfor, og hver motor fikk det arbeidet den gjør best.

I en samling på mer enn 100 databaser fordelt på PostgreSQL, Microsoft SQL Server og Google BigQuery var de fleste dataene geografiske eller tidsserier: koordinater, sensoravlesninger, logger og metrikker i sanntid. På BigQuery-siden ble de store tidsserietabellene tidspartisjonert og klynget, og komprimering og kolonnebasert lagring holdt lagringskostnadene nede og skanningene raske. Sammen med indeksering, materialiserte visninger og caching i hele samlingen kuttet arbeidet svartidene for spørringer mot GIS- og tidsseriedata med 40–60 %.

I en geografisk datapipeline som hadde blitt treg, ble SQL-en og datamodellene i PostgreSQL, MS SQL og Google Cloud BigQuery først analysert. Det avdekket svak indeksering, ineffektive spørringer og manglende integritetsregler. Deretter fikk hver motor sitt verktøy: B-tree- og GiST-indekser i PostgreSQL, filtrerte indekser i MS SQL og clustering i BigQuery, sammen med normaliserte modeller, partisjonering og omskrevne spørringer. Grunnleggerens resultatoppføring rapporterer at datapipelinen som helhet gikk 50 ganger raskere; den spesifiserer ikke datasettet, maskinvaren eller sammenligningsprosedyren for ytelsesmålingen.

Og for data fra smartmålere i flere regioner ble mer enn 500 analyserapporter skrevet i ren SQL mot PostgreSQL og BigQuery, med CTE-er, vindusfunksjoner, delspørringer og geospatiale joins over måler-ID-er, tidsstempler, koordinater og miljømetadata. Hver rapport ble levert med kommentert SQL, slik at en kollega kunne kjøre den på nytt og etterprøve den, og med notater om kjente problemer

med datakvaliteten, som manglende GPS-koordinater og ødelagte målerverdier. Rapportene ble et oppslagsverk på tvers av avdelinger og bidro til at effektiviteten i den operative planleggingen ble 10–15 % bedre.

Hvor det går galt

En spørring med `LIMIT 10` blir fakturert for hele tabellen. På en tabell uten clustering begrenser `LIMIT` resultatet og ikke bytene som leses, og `SELECT *` leser hver kolonne som finnes. En rask titt på dataene blir en full skanning. Nevn bare kolonnene spørsmålet trenger, og bruk forhåndsvisningen av tabellen i konsollen eller `bq head` for å se noen eksempelrader, siden de ikke koster noe.

En partisjonert tabell blir likevel skannet fra ende til annen. Partisjoner hoppes bare over når filteret på partisjoneringskolonnen er en konstant motoren kan regne ut før den leser. Sammenligner du kolonnen med resultatet av en delspørring, pakker den inn i en funksjon eller regner på den, går spørringen gjennom hver partisjon. To sperrer hjelper. En tabell kan kreve et partisjonsfilter, så en spørring uten et brukbart filter stopper med en feil i stedet for å kjøre. Og en tørrkjøring (dry run) rapporterer hvor mange byte en spørring ville behandlet, uten å kjøre den eller ta betalt for den:

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

Clustering blir lagt til, og antallet leste byte flytter seg knapt. Clustering sorterer lagringsblokker etter opptil fire kolonner i rekkefølge, og rekkefølgen betyr noe: en tabell klynget etter ordredato, land og status er ikke organisert for et filter på bare land og status. Tabeller under 64 MB har lite å hente, og på en klynget tabell er antallet byte som faktisk skannes, kjent først etter at spørringen er ferdig. Sett klyngekolonnene i den rekkefølgen spørringene filtrerer i. Der kostnaden må ha et tak før noe kjører, lener du deg på partisjonering, som en tørrkjøring allerede tar med i beregningen, og setter et maksimum for fakturerte byte (maximum bytes billed), slik at en spørring over grensen stopper før den blir belastet.

En spørring som joiner to datasett, blir avvist. Hvert datasett ligger på én plassering, og en spørring kjører på én plassering, så hvert datasett den rører, må dele den. Team som lar hvert prosjekt ta sin egen standard, ender med tabeller som bare kan joines etter at én av dem er kopiert over. Velg plasseringen én gang, ut fra regeletterlevelse og hvor dataene kommer fra, og opprett hvert datasett der med vilje.

Les mer

- BigQuery overview — Googles egen beskrivelse av tjenesten, de atskilte lagene for lagring og beregning og hva som blir administrert for deg.
- Introduction to partitioned tables — typene partisjonering, hvor finmasket de deler opp, og grensene deres.
- Query partitioned tables — hvilke filtre som beskjærer partisjoner, og hvilke som sender en spørring gjennom alle.
- Introduction to clustered tables — hvordan kolonnerekkefølgen avgjør hvilke blokker som hoppes over, og hva det betyr for kostnadsoverslag.
- Estimate and control costs — tørrkjøringer, et maksimum for fakturerte byte og de gratis måtene å forhåndsvise en tabell på.
- BigQuery locations — regioner, multiregioner og reglene for spørringer som rører mer enn ett datasett.

Ordliste

Velg ut fra deres faktiske rammer

Ta med et representativt datasett, spørsmålene det skal besvare og dagens flaskehals. Vi kan avtale målinger av kvalitet, gjennomstrømning og gjenoppretting.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er BigQuery?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

welcome@engineer.company +45 50 10 61 84 København, Danmark

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/bigquery/