Engineer

Datasjø

≋ et lager som tar vare på rådata i sin opprinnelige form

Hva det er

En datasjø, også kjent under det engelske navnet data lake, er et lager som tar vare på rådata slik de kommer fra kilden, på det mest detaljerte nivået kilden gir, slik at også spørsmål ingen har stilt på forhånd, kan besvares fra dem senere. I praksis ligger dataene i filer, ofte i et kolonneorientert format som Apache Parquet, i et distribuert filsystem eller et objektlager, og spørremotorer leser filene direkte.

Uttrykket kommer fra James Dixon i Pentaho. I et blogginnlegg datert 14. oktober 2010, skrevet den uken Pentaho kunngjorde sin første Hadoop-utgivelse, oppsummerte han det han hadde hørt fra selskaper som brukte Hadoop. Åtte eller ni av ti av dem arbeidet med strukturerte eller semistrukturerte data, ikke ustrukturerte, og dataene kom som regel fra én enkelt applikasjon eller ett enkelt system. Noen få spørsmål til dataene var kjent på forhånd, mange flere ville dukke opp senere, flere grupper av brukere hadde spørsmål til dem, og mengden fikk ikke plass i en relasjonsdatabase, verken teknisk eller til en fornuftig pris. Det vanlige svaret hadde vært å plukke ut de mest interessante attributtene og aggregere dem i et data mart. Det besvarer bare spørsmålene som er valgt på forhånd, og aggregeringen skjuler det laveste detaljnivået for godt.

Dixon tilbød et bilde i stedet. Et data mart er som vann på flaske: renset, pakket og formet slik at det er lett å konsumere. En datasjø er med hans ord «a large body of water in a more natural state». En kilde strømmer inn og fyller sjøen, og alle som har et spørsmål, kan komme og se, dykke ned i den eller ta en prøve.

Fire år senere tok Dixon opp begrepet igjen, fordi mange brukte det og definisjonene varierte. Han gjentok sin egen: En sjø inneholder data fra én kilde, ikke forhåndsaggregert, og en organisasjon kan ha flere sjøer. Lagrer du data fra

mange systemer og kobler dem sammen på tvers, skrev han, har du en vannhage (water garden), ikke en sjø. Han var like tydelig på at en datasjø ikke erstattet en database, et data mart eller et datavarehus i 2010. Flere av definisjonene han svarte på, beskrev ett felles lager for rådata fra mange systemer. Det er den løsere betydningen av ordet, og det er verdt å vite hvilken av dem en kollega mener.

Bruk begrepet i en beslutning

Bruk «Hva er en datasjø?» 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

Hver oppsummering er et valg av hvilke spørsmål som betyr noe. En daglig salgssum per butikk forteller hvor mye hver butikk solgte. Den kan ikke fortelle hvilken time som hadde mest trafikk, hvilke varer folk så på uten å kjøpe dem, eller om den samme kunden kom tilbake tre ganger før hun bestilte. Når bare summen tas vare på, er de spørsmålene borte for godt, for radene som kunne besvart dem, ble kastet da summen ble regnet ut. Dixons poeng var at spørsmålene kommer etter dataene, så rådataene bør lagres på sitt mest detaljerte nivå og aggregeres først når spørsmålet er kjent.

Det gjør ikke datavarehuset overflødig. Dixon beskrev hvordan organisasjonens faste data marts og datavarehus fylles med aggregeringer hentet fra sjøen, mens spørsmål ad hoc går rett til sjøen. Datavarehuset er fortsatt stedet for tallene organisasjonen er enig om. Sjøen er dit man går når et nytt spørsmål trenger detaljer datavarehuset aldri tok vare på, og der man bygger opp igjen en tabell i datavarehuset når definisjonen av den endres.

Filformatet betyr like mye som ideen. Parquet lagrer hver kolonne i en radgruppe (row group) som en egen blokk, og metadataene på slutten av filen registrerer hvor hver blokk begynner. En leser ser først på disse metadataene og henter deretter bare kolonneblokkene den trenger, så en spørring som berører tre kolonner av førti, ikke leser de andre trettisju. Fordi Parquet er et åpent

format og hver fil bærer sine egne metadata, kan enhver motor som leser Parquet, spørre mot den samme mappen, også DuckDB på en bærbar, uten noe eksportsteg.

Slik ser det ut i praksis

En nettbutikk registrerer hver produktvisning, hver gang en vare legges i handlekurven, og hvert kjøp. I stedet for å legge dem sammen til daglige summer tar den vare på hver eneste hendelse og skriver hendelsene ut som Parquet-filer, én mappe per dag. I DuckDB 2.0 er begge stegene vanlig SQL:

```
CREATE TABLE shop_events AS
SELECT * FROM (VALUES
  (TIMESTAMP '2026-10-26 09:14:03', 'view',     'boots-41', NULL::INTEGER),
  (TIMESTAMP '2026-10-26 09:15:40', 'cart',     'boots-41', NULL),
  (TIMESTAMP '2026-10-26 09:17:12', 'purchase', 'boots-41', 1200),
  (TIMESTAMP '2026-10-27 11:02:55', 'view',     'jacket-m', NULL),
  (TIMESTAMP '2026-10-27 11:03:30', 'view',     'boots-41', NULL),
  (TIMESTAMP '2026-10-27 11:05:01', 'cart',     'jacket-m', NULL)
) t(occurred_at, event, product, amount);

COPY (SELECT *, CAST(occurred_at AS DATE) AS day FROM shop_events)
TO 'events' (FORMAT parquet, PARTITION_BY (day));
```

`PARTITION_BY` godtar bare kolonnenavn, ikke uttrykk, så dagen regnes ut i `SELECT`. DuckDB skriver én mappe per verdi, navngitt på formen key=value, som dokumentasjonen kaller Hive partitioning:

```
events/day=2026-10-26/data_0.parquet
events/day=2026-10-27/data_0.parquet
```

Et spørsmål om én dag filtrerer på den kolonnen:

```
SELECT event, count(*) AS n
FROM   read_parquet('events/*/*.parquet', hive_partitioning = true)
WHERE  day = DATE '2026-10-27'
GROUP  BY event
ORDER  BY n DESC;
-- view 2, cart 1
```

DuckDB skyver et filter på en partisjonskolonne ned til mappenavnene og hopper over filene som ikke kan treffe. `EXPLAIN ANALYZE` på spørringen viser at én av de to filene ble skannet. Med et års daglige mapper ville det samme spørsmålet åpne én mappe av 365. Kolonnen `day` kommer tilbake som `DATE` uten at noen har bedt om det, fordi DuckDB som standard gjenkjenner datoer, tidsstempler og store heltall i mappenavn. Den slår også på Hive partitioning av seg selv når mappene følger key=value-mønsteret, så flagget i spørringene gjør bare hensikten synlig.

Nå et spørsmål ingen hadde planlagt: Hvilke varer ble sett mest på?

```
SELECT product, count(*) AS views
FROM   read_parquet('events/*/*.parquet', hive_partitioning = true)
WHERE  event = 'view'
GROUP  BY product
ORDER  BY views DESC, product;
-- boots-41 2, jacket-m 1
```

En tabell med daglige kjøpssummer kunne aldri besvart det. De rå hendelsene besvarer det med enda en spørring mot den samme mappen, fordi hver eneste hendelse ble lagret på veien inn.

Selskapets tjeneste for design av datavarehus beskriver målet som ett sted der alle tallene stemmer overens, slik at to avdelinger slutter å komme med to ulike svar til det samme møtet. En sjø som denne ligger oppstrøms for et slikt datavarehus: Datavarehuset har tallene man er enige om, og rådataene ligger klare for det neste spørsmålet.

Hvor det går galt

Filer ingen kan forklare. En mappe med Parquet-filer uten opplysninger om hvor de kom fra, hvem som eier dem, eller hva hver kolonne betyr, er en sjø bare den som bygde den, kan bruke. Da kritikere i 2014 hevdet at en datasjø ikke har metadata, svarte Dixon at ingenting i ideen sier det, og la senere til at den som kobler data fra ulike kilder eller sporer opphavet deres (lineage), vil trenge metadata. Hold én oppføring i en datakatalog per datasett med kilde, eier, kolonner, innlastingstidspunkt og lagringstid, og gjør det å skrive oppføringen til en fast del av innlastingsjobben.

Tusenvis av små filer. Hver Parquet-fil har sin egen metadatablokk, og en leser må lese den før den kommer til noen data. En strøm som skriver én liten fil i minuttet, etterlater over førti tusen filer i løpet av en måned på 30 dager, og hver av dem åpnes og tolkes ved hver full skanning. Samle skrivingene i færre, større filer, og slå sammen gamle partisjoner etter en fast plan (compaction). Tabellformatet Apache Iceberg registrerer en slik sammenslåing som et snapshot av typen `replace` : flere filer byttet ut med én som har de samme radene.

Lesere som ser en halv innlasting. En jobb som skriver ti filer og stopper etter seks, etterlater seks filer i mappen, og alt som lister mappen, vil lese dem. En vanlig mappe kjenner ikke til commit. Det løser et tabellformat. Iceberg holder oversikt over de enkelte datafilene i metadata i stedet for over hele mapper, og gjennomfører en endring ved å bytte ut én metadatafil med den neste atomisk. Lesere bruker alltid et gjennomført snapshot, skrivinger er aldri delvis synlige, og lesere tar ingen låser.

Omdøpte kolonner som splitter historikken. Gamle filer beholder det gamle kolonnenavnet, og hver spørring må kjenne begge. Iceberg velger kolonner i datafiler etter en felt-ID i stedet for etter navn, så en kolonne kan få nytt navn, legges til, fjernes eller flyttes, også inne i nestede strukturer, uten at filene som allerede er skrevet, må skrives om.

Én sjø for alle systemer. Å helle alle organisasjonens databaser ut på ett sted og koble alt sammen på tvers er det Dixon kalte en vannhage, og han var enig med en kritiker i at det ikke var noen farbar vei, i hvert fall ikke den gang, å legge alle virksomhetens data i Hadoop eller et annet enkelt lager. Rådet hans var å bruke det beste verktøyet til hver jobb: en transaksjonsdatabase til transaksjonelle formål og en analysedatabase til analytiske. Start en sjø fra én kilde med reelle spørsmål som venter på svar, og bygg de samlede tallene man er enige om, i et datavarehus som fylles fra sjøen.

Personopplysninger som lagres for alltid. Et lager som er bygd for å ta vare på alt, kolliderer med krav om sletting og grenser for lagringstid. Iceberg versjon 2 kan slette enkeltrader ved å skrive slettefiler (delete files) uten å skrive om datafilene, og det betyr at de opprinnelige bytene blir liggende i lageret. De forsvinner først når datafilene skrives om og de eldre snapshotene som fortsatt viser til dem, utløper, og etter det kan filene slettes fysisk. Hold direkte identifikatorer utenfor rådataene der det lar seg gjøre, partisjoner slik at utløpte data kan fjernes som hele mapper, og planlegg sammenslåing og utløp av snapshots etter slettinger.

Les mer

- Pentaho, Hadoop, and Data Lakes — James Dixons innlegg fra oktober 2010, som introduserte begrepet og sammenligningen med vann på flaske.
- Data Lakes Revisited — Dixons oppfølging fra 2014: én kilde per sjø, vannhagen og svarene hans til kritikerne.
- File Format — Parquets oppbygning med radgrupper, kolonneblokker og filmetadata som skrives etter dataene.
- Iceberg Table Spec — Apache Icebergs spesifikasjon: snapshots, atomiske commits, skjemaendringer via felt-ID, slettefiler og utløp av snapshots.
- Hive Partitioning — DuckDBs dokumentasjon av key=value-mapper, filter pushdown og gjenkjenning av partisjonstyper.

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 en datasjø?» 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/data-lake/