Engineer

Data lake

≋ et lager, der gemmer rå data i deres oprindelige form

Hvad det er

En data lake, på dansk også kaldet en datasø, er et lager, der gemmer rådata, som de kommer fra kilden, på det mest detaljerede niveau, kilden leverer, så også spørgsmål, som ingen har stillet på forhånd, kan besvares ud fra dem senere. I praksis ligger data i filer, ofte i et kolonneorienteret format som Apache Parquet, i et distribueret filsystem eller et objektlager, og forespørgselsmotorer læser filerne direkte.

Udtrykket stammer fra James Dixon fra Pentaho. I et blogindlæg dateret 14. oktober 2010, skrevet i den uge, hvor Pentaho annoncerede sin første Hadoop-udgivelse, samlede han op på, hvad han havde hørt fra virksomheder, der brugte Hadoop. Otte eller ni ud af ti af dem arbejdede med strukturerede eller semistrukturerede data, ikke ustrukturerede, og dataene kom som regel fra en enkelt applikation eller et enkelt system. Nogle få spørgsmål til dataene var kendt på forhånd, mange flere ville dukke op senere, flere grupper af brugere havde spørgsmål til dem, og mængden var for stor til en relationel database, det være sig teknisk eller økonomisk. Det sædvanlige svar havde været at udvælge de mest interessante attributter og aggregere dem i et data mart. Det besvarer kun de spørgsmål, der er valgt på forhånd, og aggregeringen skjuler det laveste detaljeniveau for altid.

Dixon tilbød i stedet et billede. Et data mart er som vand på flaske: renset, pakket og formet, så det er let at indtage. En data lake er med hans ord »a large body of water in a more natural state«. En kilde strømmer ind og fylder søen, og alle med et spørgsmål kan komme og kigge, dykke ned i den eller tage en prøve.

Fire år senere vendte Dixon tilbage til begrebet, fordi mange brugte det, og definitionerne var ret forskellige. Han gentog sin egen: En sø rummer data fra én kilde, ikke præaggregerede, og en organisation kan have flere søer. Gemmer man

data fra mange systemer og kobler dem sammen på tværs, skrev han, har man en vandhave (water garden), ikke en sø. Han var lige så klar om, at en data lake ikke erstattede en database, et data mart eller et data warehouse i 2010. Flere af de definitioner, han svarede på, beskrev ét fælles lager til rådata fra mange systemer. Det er den løsere betydning af ordet, og det er værd at vide, hvilken af dem en kollega mener.

Brug begrebet i en beslutning

Brug »Hvad er en data lake?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Tal fra salg, drift og økonomi samlet ét sted, så to afdelinger ikke længere kommer til det samme møde med hver sit svar.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har gjort en geodatapipeline 50 gange hurtigere — 50x hurtigere geografiske datapipelines

Har gjort en geodatapipeline 50 gange hurtigere Kontakt os →

Hvorfor det betyder noget

Enhver sammentælling er en beslutning om, hvilke spørgsmål der betyder noget. En daglig salgstotal pr. butik fortæller, hvor meget hver butik solgte. Den kan ikke fortælle, hvilken time der var travlest, hvilke varer folk kiggede på uden at købe dem, eller om den samme kunde kom tilbage tre gange, før hun bestilte. Når kun totalen gemmes, er de spørgsmål væk for altid, for de rækker, der kunne besvare dem, blev smidt ud, da totalen blev beregnet. Dixons pointe var, at spørgsmålene kommer efter dataene, så rådataene bør gemmes på deres mest detaljerede niveau og først aggregeres, når spørgsmålet er kendt.

Det gør ikke datavarehuset overflødigt. Dixon beskrev, hvordan organisationens faste data marts og datavarehuse fyldes med aggregeringer hentet fra søen, mens spørgsmål ad hoc går direkte til søen. Datavarehuset er stadig stedet for de tal, organisationen er enig om. Søen er der, man går hen, når et nyt spørgsmål kræver detaljer, som datavarehuset aldrig har gemt, og der, man genopbygger en tabel i datavarehuset, når dens definition ændres.

Filformatet betyder lige så meget som idéen. Parquet gemmer hver kolonne i en rækkegruppe (row group) som sin egen blok, og metadataene i slutningen af filen registrerer, hvor hver blok begynder. En læser kigger først på disse metadata og henter derefter kun de kolonneblokke, den har brug for, så en forespørgsel, der rører tre kolonner ud af fyrre, ikke læser de øvrige

syvogtredive. Fordi Parquet er et åbent format, og hver fil bærer sine egne metadata, kan enhver motor, der læser Parquet, forespørge den samme mappe, også DuckDB på en bærbar, uden et eksporttrin.

Sådan ser det ud i praksis

En webshop registrerer hver produktvisning, hver gang en vare lægges i kurven, og hvert køb. I stedet for at lægge dem sammen til daglige totaler gemmer den hver eneste hændelse og skriver hændelserne ud som Parquet-filer, én mappe pr. dag. I DuckDB 2.0 er begge trin almindelig 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` accepterer kun kolonnenavne, ikke udtryk, så dagen beregnes i `SELECT`. DuckDB skriver én mappe pr. værdi, navngivet på formen key=value, som dokumentationen kalder Hive partitioning:

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

Et spørgsmål om én dag filtrerer på den kolonne:

```
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 skubber et filter på en partitionskolonne ned til mappenavnene og springer de filer over, der ikke kan matche. `EXPLAIN ANALYZE` på forespørgslen viser, at én af de to filer blev scannet. Med et års daglige mapper ville samme spørgsmål åbne én mappe ud af 365. Kolonnen `day` kommer tilbage som `DATE`, uden at nogen har bedt om det, fordi DuckDB som standard genkender datoer, tidsstempler og store heltal i mappenavne. Den slår også selv Hive partitioning til, når mapperne følger key=value-mønsteret, så flaget i forespørgslerne gør blot hensigten synlig.

Nu et spørgsmål, ingen havde planlagt: Hvilke varer blev der kigget 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 tabel med daglige købstotaler kunne aldrig besvare det. De rå hændelser besvarer det med endnu en forespørgsel mod den samme mappe, fordi hver eneste hændelse blev gemt på vejen ind.

Virksomhedens ydelse inden for design af data warehouse beskriver målet som ét sted, hvor alle tallene stemmer overens, så to afdelinger ikke længere kommer til det samme møde med hver sit svar. En sø som denne ligger opstrøms for sådan et datavarehus: Datavarehuset rummer de tal, man er enige om, og rådataene ligger klar til det næste spørgsmål.

Hvor det går galt

Filer, ingen kan forklare. En mappe med Parquet-filer uden oplysninger om, hvor de kom fra, hvem der ejer dem, eller hvad hver kolonne betyder, er en sø, som kun den, der byggede den, kan bruge. Da kritikere i 2014 hævdede, at en data lake ikke har metadata, svarede Dixon, at intet i idéen siger det, og tilføjede senere, at den, der kobler data fra forskellige kilder eller sporer deres herkomst (lineage), får brug for metadata. Hav én post i et datakatalog pr. datasæt med kilde, ejer, kolonner, indlæsningstidspunkt og opbevaringsperiode, og gør det at skrive posten til en fast del af indlæsningsjobbet.

Tusindvis af små filer. Hver Parquet-fil har sin egen metadatablok, og en læser skal læse den, før den når frem til nogen data. En strøm, der skriver én lille fil i minuttet, efterlader over 40.000 filer på en måned med 30 dage, og hver af dem åbnes og fortolkes ved hver fuld scanning. Saml skrivningerne i færre, større filer, og læg gamle partitioner sammen efter en fast plan (compaction). Tabelformatet Apache Iceberg registrerer sådan en sammenlægning som et snapshot af typen `replace`: flere filer skiftet ud med én, der rummer de samme rækker.

Læsere, der ser en halv indlæsning. Et job, der skriver ti filer og stopper efter seks, efterlader seks filer i mappen, og alt, hvad der lister mappen, læser dem. En almindelig mappe kender ikke til commit. Det løser et tabelformat. Iceberg holder styr på de enkelte datafiler i metadata i stedet for på hele mapper og gennemfører en ændring ved atomart at udskifte én metadatafil med den næste. Læsere bruger altid et gennemført snapshot, skrivninger er aldrig delvist synlige, og læsere tager ingen låse.

Omdøbte kolonner, der splitter historikken. Gamle filer beholder det gamle kolonnenavn, og hver forespørgsel skal kende begge. Iceberg vælger kolonner i datafiler ud fra et felt-id i stedet for navnet, så en kolonne kan omdøbes, tilføjes, fjernes eller flyttes, også inde i indlejrede strukturer, uden at de filer, der allerede er skrevet, skal skrives om.

Én sø til alle systemer. At hælde alle organisationens databaser ned ét sted og koble det hele sammen på tværs er det, Dixon kaldte en vandhave, og han var enig med en kritiker i, at det ikke var en farbar vej, i hvert fald dengang, at lægge alle virksomhedens data i Hadoop eller et andet enkelt lager. Hans råd var at bruge det bedste værktøj til hver opgave: en transaktionel database til transaktionelle formål og en analytisk database til analytiske. Start en sø fra én kilde med reelle spørgsmål, der venter på svar, og byg de samlede tal, man er enige om, i et datavarehus, der fyldes fra søen.

Personoplysninger, der gemmes for evigt. Et lager, der er bygget til at gemme alt, støder sammen med anmodninger om sletning og grænser for opbevaring. Iceberg version 2 kan slette enkelte rækker ved at skrive sletfiler (delete files) uden at omskrive datafilerne, hvilket betyder, at de oprindelige bytes bliver liggende på lageret. De forsvinder først, når datafilerne skrives om, og de ældre snapshots, der stadig peger på dem, udløber, hvorefter filerne må slettes fysisk. Hold direkte identifikatorer ude af rådataene, hvor det kan lade sig gøre, partitionér, så udløbne data kan fjernes som hele mapper, og planlæg sammenlægning og udløb af snapshots efter sletninger.

Læs mere

- Pentaho, Hadoop, and Data Lakes — James Dixons indlæg fra oktober 2010, der introducerede begrebet og sammenligningen med vand på flaske.
- Data Lakes Revisited — Dixons opfølgning fra 2014: én kilde pr. sø, vandhaven og hans svar til kritikerne.
- File Format — Parquets opbygning med rækkegrupper, kolonneblokke og filmetadata, der skrives efter dataene.
- Iceberg Table Spec — Apache Icebergs specifikation: snapshots, atomare commits, skemaændringer via felt-id, sletfiler og udløb af snapshots.
- Hive Partitioning — DuckDB's dokumentation af key=value-mapper, filter pushdown og genkendelse af partitionstyper.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring et repræsentativt datasæt, de spørgsmål det skal besvare, og den aktuelle flaskehals. Vi kan aftale målinger af kvalitet, gennemløb og gendannelse.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er en data lake?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

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

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/data-lake/