Engineer

Datasjö

≋ ett lager som sparar rådata i sin ursprungliga form

Vad det är

En datasjö, även känd under det engelska namnet data lake, är ett lager som sparar rådata precis som den kommer från källan, på den mest detaljerade nivå källan ger, så att även frågor som ingen har ställt i förväg går att besvara ur den senare. I praktiken ligger datan i filer, ofta i ett kolumnorienterat format som Apache Parquet, i ett distribuerat filsystem eller ett objektlager, och frågemotorer läser filerna direkt.

Uttrycket kommer från James Dixon på Pentaho. I ett blogginlägg daterat den 14 oktober 2010, skrivet samma vecka som Pentaho tillkännagav sin första Hadoop-version, sammanfattade han vad han hade hört från företag som använde Hadoop. Åtta eller nio av tio av dem arbetade med strukturerad eller semistrukturerad data, inte ostrukturerad, och datan kom oftast från en enda applikation eller ett enda system. Några frågor till datan var kända i förväg, betydligt fler skulle dyka upp senare, flera grupper av användare ville ställa dem, och mängden fick inte plats i en relationsdatabas, varken tekniskt eller till en rimlig kostnad. Det vanliga svaret hade varit att plocka ut de mest intressanta attributen och aggregera dem i ett data mart. Det besvarar bara de frågor som valts i förväg, och aggregeringen döljer den lägsta detaljnivån för gott.

Dixon erbjöd en bild i stället. Ett data mart är som vatten på flaska: renat, förpackat och format så att det är lätt att konsumera. En datasjö är med hans ord ”a large body of water in a more natural state”. En källa strömmar in och fyller sjön, och alla som har en fråga kan komma och titta, dyka ner i den eller ta ett prov.

Fyra år senare tog Dixon upp begreppet igen, eftersom många använde det och definitionerna gick isär. Han upprepade sin egen: en sjö innehåller data från en enda källa, inte föraggregerad, och en organisation kan ha flera sjöar. Sparar man

data från många system och kopplar ihop dem tvärs över, skrev han, har man en vattenträdgård (water garden), inte en sjö. Han var lika tydlig med att en datasjö inte ersatte en databas, ett data mart eller ett data warehouse 2010. Flera av de definitioner han svarade på beskrev ett gemensamt lager för rådata från många system. Det är den lösare betydelsen av ordet, och det lönar sig att veta vilken av dem en kollega menar.

Använd begreppet i ett beslut

Använd ”Vad är en datasjö?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Siffror från försäljning, drift och ekonomi samlade på ett ställe, så att två avdelningar slutar komma med två olika svar till samma möte.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Gjorde pipelinen för geodata 50 gånger snabbare — 50x snabbare datapipelines för geodata

Gjorde pipelinen för geodata 50 gånger snabbare Hör av er →

Varför det spelar roll

Varje sammanställning är ett val av vilka frågor som räknas. En daglig försäljningssumma per butik visar hur mycket varje butik sålde. Den kan inte visa vilken timme som hade mest trafik, vilka varor folk tittade på utan att köpa, eller om samma kund kom tillbaka tre gånger innan hon beställde. När bara summan sparas är de frågorna borta för gott, eftersom raderna som kunde ha besvarat dem kastades när summan räknades ut. Dixons poäng var att frågorna kommer efter datan, så rådatan bör sparas på sin mest detaljerade nivå och aggregeras först när frågan är känd.

Det gör inte analysdatalagret överflödigt. Dixon beskrev hur organisationens fasta data marts och datalager fylls med aggregeringar hämtade ur sjön, medan frågor ad hoc går direkt till sjön. Datalagret är fortfarande platsen för de siffror organisationen är överens om. Sjön är dit man går när en ny fråga behöver detaljer som datalagret aldrig sparade, och där man bygger om en tabell i datalagret när dess definition ändras.

Filformatet spelar lika stor roll som idén. Parquet lagrar varje kolumn i en radgrupp (row group) som ett eget block, och metadatan i slutet av filen anger var varje block börjar. En läsare tittar först på den metadatan och hämtar sedan bara de kolumnblock den behöver, så en fråga som rör tre kolumner av fyrtio

läser inte de andra trettiosju. Eftersom Parquet är ett öppet format och varje fil bär sin egen metadata, kan varje motor som läser Parquet ställa frågor mot samma mapp, även DuckDB på en bärbar dator, utan något exportsteg.

Så ser det ut i praktiken

En webbutik registrerar varje produktvisning, varje gång en vara läggs i varukorgen och varje köp. I stället för att summera dem till dagliga summor sparar den varje enskild händelse och skriver ut händelserna som Parquet-filer, en mapp per dag. I DuckDB 2.0 är båda stegen 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` tar bara kolumnnamn, inte uttryck, så dagen räknas ut i `SELECT`. DuckDB skriver en mapp per värde, namngiven på formen key=value, som dokumentationen kallar Hive partitioning:

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

En fråga om en enda dag filtrerar på den kolumnen:

```
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 trycker ner ett filter på en partitionskolumn till mappnamnen och hoppar över de filer som inte kan matcha. `EXPLAIN ANALYZE` på frågan visar att en av de två filerna lästes. Med ett års dagliga mappar skulle samma fråga öppna en mapp av 365. Kolumnen `day` kommer tillbaka som `DATE` utan att någon har bett om det, eftersom DuckDB som standard känner igen datum, tidsstämplar och stora heltal i mappnamn. Den slår också på Hive partitioning av sig själv när mapparna följer mönstret key=value, så flaggan i frågorna gör bara avsikten synlig.

Nu en fråga som ingen hade planerat: vilka varor tittade folk 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 dagliga köpsummor hade aldrig kunnat besvara det. Rådatan besvarar det med ytterligare en fråga mot samma mapp, eftersom varje händelse sparades på vägen in.

Företagets tjänst för design av datalager beskriver målet som ett ställe där alla siffror stämmer överens, så att två avdelningar slutar komma med två olika svar till samma möte. En sjö som den här ligger uppströms om ett sådant datalager: datalagret har de siffror man är överens om, och rådatan finns till hands för nästa fråga.

Var det går fel

Filer som ingen kan förklara. En mapp med Parquet-filer utan uppgifter om var de kommer ifrån, vem som äger dem eller vad varje kolumn betyder är en sjö som bara den som byggde den kan använda. När kritiker 2014 hävdade att en datasjö saknar metadata svarade Dixon att ingenting i idén säger så, och lade senare till att den som kopplar ihop data från olika källor eller spårar dess ursprung (lineage) kommer att behöva metadata. Håll en post i en datakatalog per dataset med källa, ägare, kolumner, inläsningstid och lagringstid, och gör det till en fast del av inläsningsjobbet att skriva posten.

Tusentals små filer. Varje Parquet-fil har sitt eget metadatablock, och en läsare måste läsa det innan den kommer åt någon data. En ström som skriver en liten fil i minuten lämnar efter sig över fyrtio tusen filer under en månad på 30 dagar, och var och en öppnas och tolkas vid varje fullständig genomläsning. Samla skrivningarna i färre och större filer, och slå ihop gamla partitioner enligt ett fast schema (compaction). Tabellformatet Apache Iceberg registrerar en sådan sammanslagning som en snapshot av typen `replace` : flera filer utbytta mot en som innehåller samma rader.

Läsare som ser en halv inläsning. Ett jobb som skriver tio filer och stannar efter sex lämnar sex filer i mappen, och allt som listar mappen kommer att läsa dem. En vanlig katalog känner inte till någon commit. Det löser ett tabellformat. Iceberg håller reda på de enskilda datafilerna i metadata i stället för på hela kataloger, och genomför en ändring genom att atomärt byta ut en metadatafil mot nästa. Läsare använder alltid en genomförd snapshot, skrivningar syns aldrig till hälften, och läsare tar inga lås.

Omdöpta kolumner som delar historiken. Gamla filer behåller det gamla kolumnnamnet, och varje fråga måste känna till båda. Iceberg väljer kolumner i datafiler efter ett fält-id i stället för efter namn, så en kolumn kan döpas om, läggas till, tas bort eller flyttas, även inne i nästlade strukturer, utan att de filer som redan är skrivna behöver skrivas om.

En sjö för alla system. Att hälla ut alla organisationens databaser på ett ställe och koppla ihop allt tvärs över är det Dixon kallade en vattenträdgård, och han höll med en kritiker om att det inte var någon framkomlig väg, åtminstone inte då, att lägga all företagets data i Hadoop eller något annat enskilt lager. Hans råd var att använda det bästa verktyget för varje uppgift: en transaktionsdatabas för transaktionella ändamål och en analysdatabas för analytiska. Börja en sjö från en enda källa med verkliga frågor som väntar på svar, och bygg de samlade siffror man är överens om i ett datalager som fylls från sjön.

Personuppgifter som sparas för alltid. Ett lager som är byggt för att spara allt krockar med begäranden om radering och gränser för lagringstid. Iceberg version 2 kan radera enskilda rader genom att skriva raderingsfiler (delete files) utan att skriva om datafilerna, vilket betyder att de ursprungliga byten ligger kvar i lagret. De försvinner först när datafilerna skrivs om och de äldre snapshots som fortfarande pekar på dem löper ut, varefter filerna får raderas fysiskt. Håll

direkta identifierare utanför rådatan där det går, partitionera så att data vars lagringstid har gått ut kan tas bort som hela mappar, och schemalägg sammanslagning och utgång av snapshots efter raderingar.

Läs mer

- Pentaho, Hadoop, and Data Lakes — James Dixons inlägg från oktober 2010, som introducerade begreppet och jämförelsen med vatten på flaska.
- Data Lakes Revisited — Dixons uppföljning från 2014: en källa per sjö, vattenträdgården och hans svar till kritikerna.
- File Format — Parquets uppbyggnad med radgrupper, kolumnblock och filmetadata som skrivs efter datan.
- Iceberg Table Spec — Apache Icebergs specifikation: snapshots, atomära commits, schemaändringar via fält-id, raderingsfiler och utgång av snapshots.
- Hive Partitioning — DuckDB:s dokumentation av key=value-mappar, filter pushdown och igenkänning av partitionstyper.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med ett representativt dataset, frågorna det ska besvara och dagens flaskhals. Vi kan avtala mått på kvalitet, genomströmning och återställning.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är en datasjö?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

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

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/data-lake/