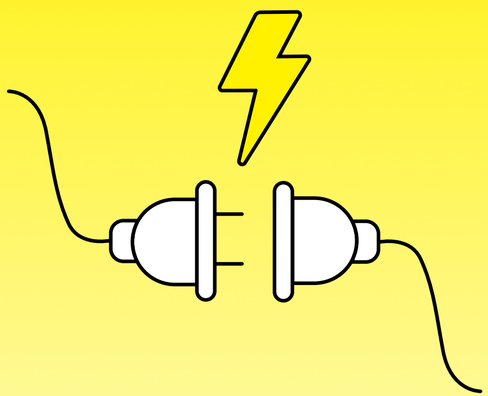
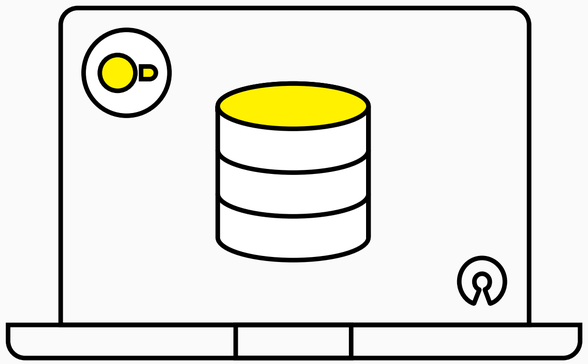


COMPASS AI & DATA SUMMIT, 2025-11-13

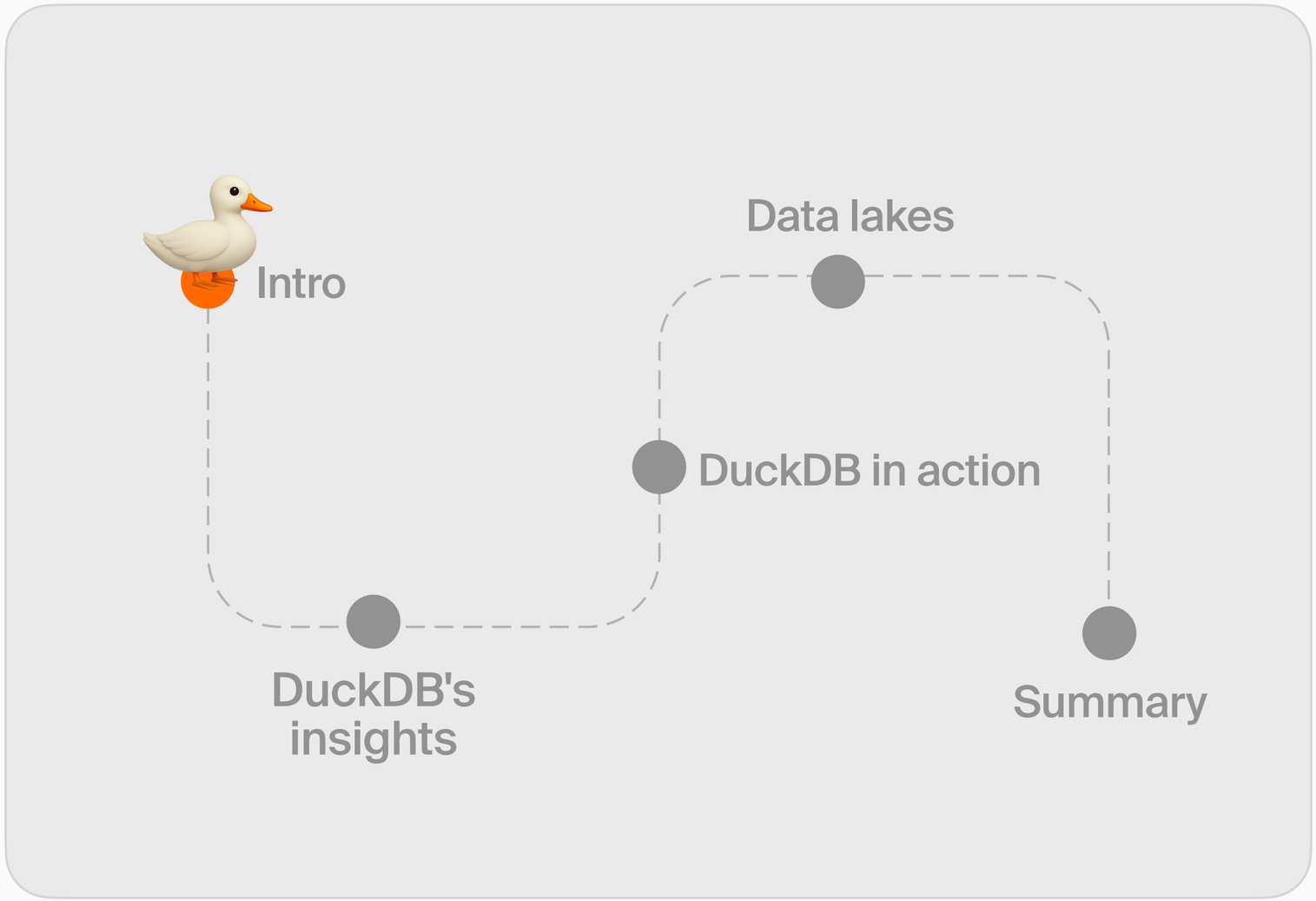
DuckDB

Big Data Unplugged

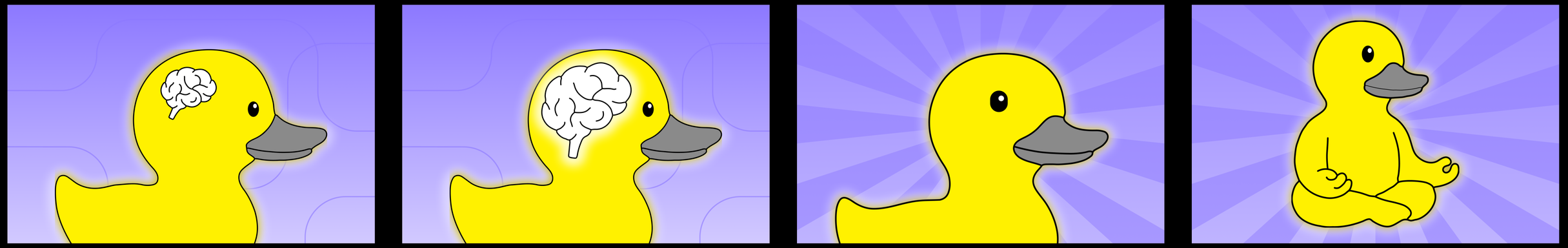
GÁBOR SZÁRNYAS



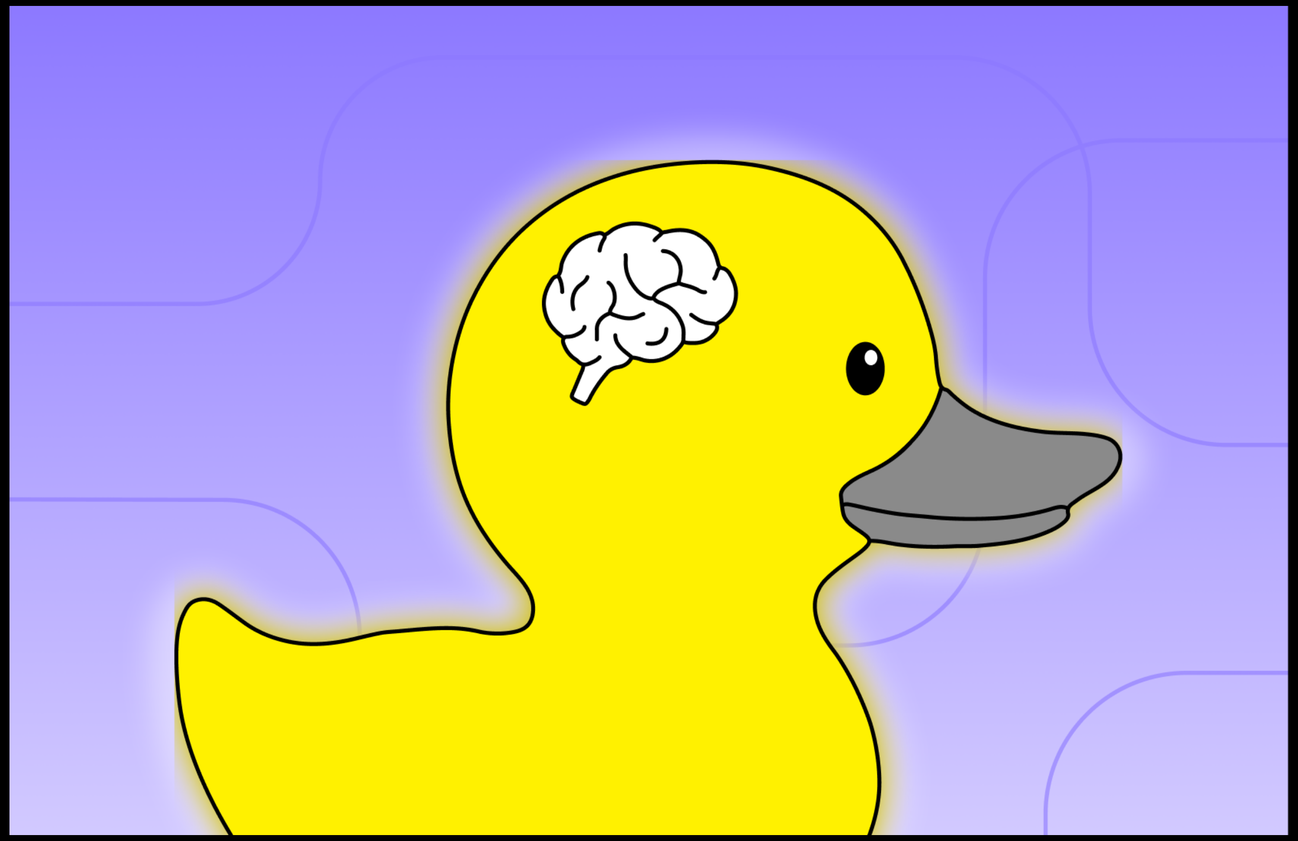
An open-source in-process SQL database for analytics


Intro
Data lakes
DuckDB in action
DuckDB's insights
Summary

Data processing
in increasing levels of enlightenment

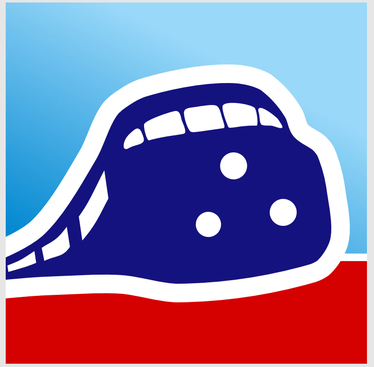


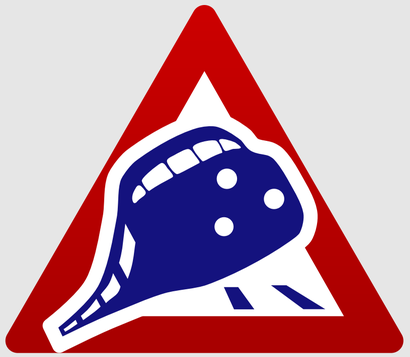
Text files



Dutch railway dataset by “Rijden de Treinen”

Which train stations had the worst delays over the weekends of February 2025?





```
$ wget https://blobs.duckdb.org/trains-2025-feb.csv.gz
$ gunzip trains-2025-feb.csv.gz
$ head -n 2 trains-2025-feb.csv


id,date,service_type,company,train_number,...
15277134,2025-02-01,Intercity,NS,1410,...


$ db_client -u admin -p admin

# CREATE TABLE services (
      id BIGINT, date DATE, service_type VARCHAR, ...
  );


# COPY services
  FROM 'trains-2025-feb.csv'
  (HEADER true, DELIMITER ',');
```

```
# SELECT
      station,
      avg(delay)
  FROM services
  WHERE strftime('%a', date) IN ('Sat', 'Sun')
  GROUP BY station
  ORDER BY #2 DESC
  LIMIT 3;
```

station	avg(delay)
Siegburg/Bonn	5.58
Emmerich-Elten	3.97
Brussel-Zuid	3.86

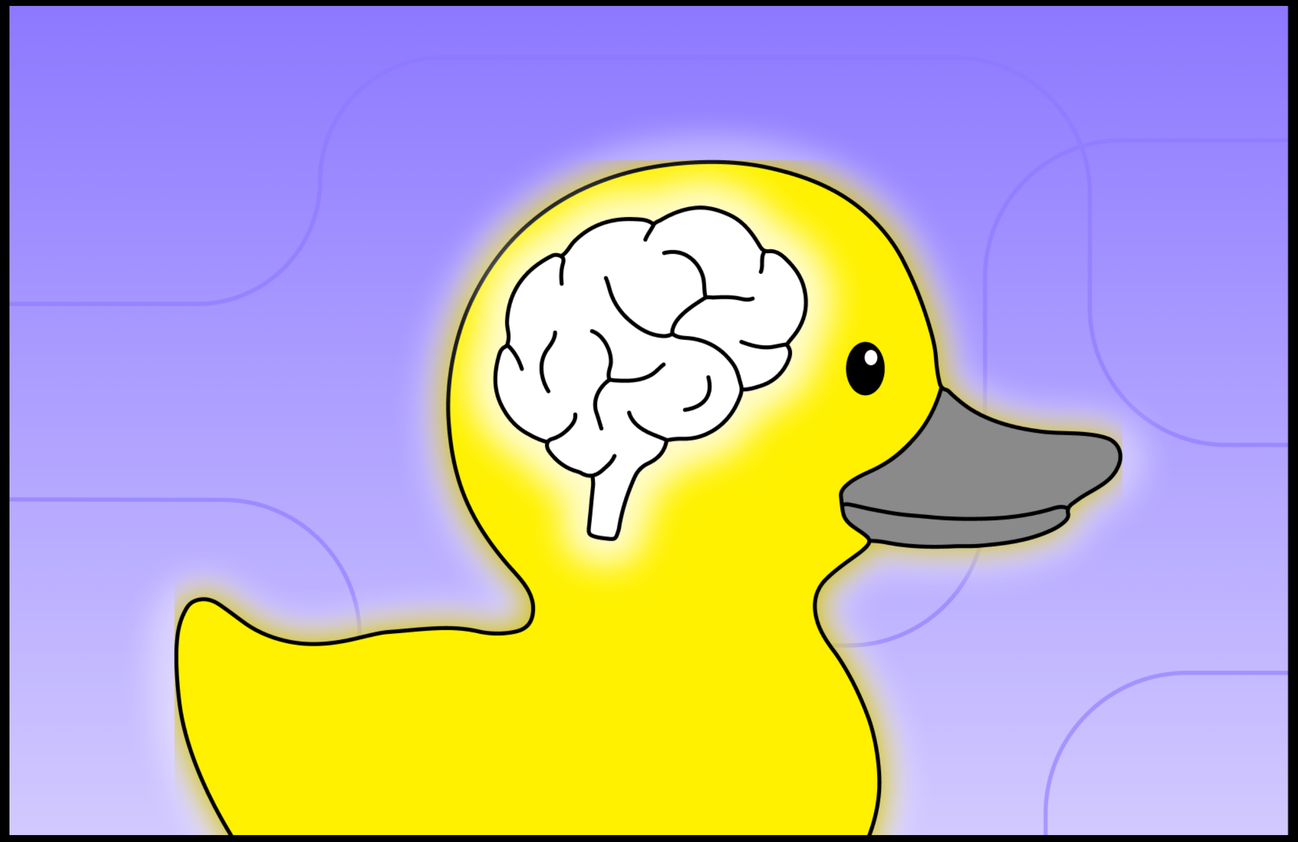
```
$ wget https://blobs.duckdb.org/trains-2025-feb.csv.gz
$ gunzip trains-2025-feb.csv.gz
$ head -n 2 trains-2025-feb.csv
$ db_client -u admin -p admin

# CREATE TABLE services (id BIGINT, date DATE,
service_type VARCHAR, company VARCHAR, train_number
BIGINT, cancelled BOOLEAN, partly_cancelled BOOLEAN,
maximum_delay BIGINT, stop_id BIGINT, station_code
VARCHAR, station VARCHAR, arrival_time TIMESTAMP WITH
TIME ZONE, delay BIGINT, arrival_cancelled BOOLEAN,
departure_time TIMESTAMP WITH TIME ZONE, departure_delay
BIGINT, departure_cancelled BOOLEAN);

# COPY services FROM 'trains-2025-feb.csv' (HEADER true,
DELIMITER ',');

# SELECT station, avg(delay) FROM services WHERE
strftime('%a', date) IN ('Sat', 'Sun') GROUP BY station
ORDER BY #2 DESC LIMIT 3;
```

750+ keystrokes
for a simple analysis

Legacy solution

Works fine in DuckDB!

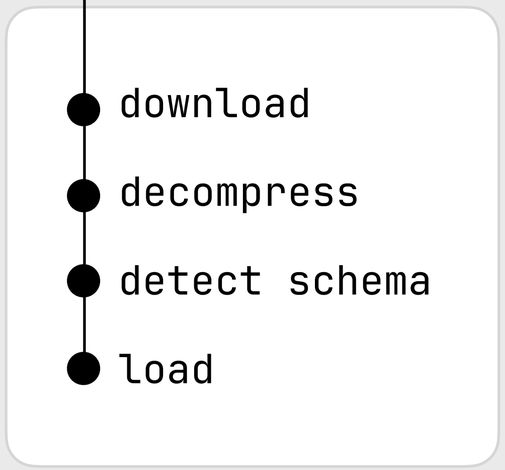
Text files (deluxe)



```
$ duckdb

D SELECT
      station,
      avg(delay)
  FROM 'https://blobs.duckdb.org/trains-2025-feb.csv.gz'
  WHERE strftime('%a', date) IN ('Sat', 'Sun')
  GROUP BY station
  ORDER BY #2 DESC
  LIMIT 3;
```

station	avg(delay)
Siegburg/Bonn	5.58
Emmerich-Elten	3.97
Brussel-Zuid	3.86

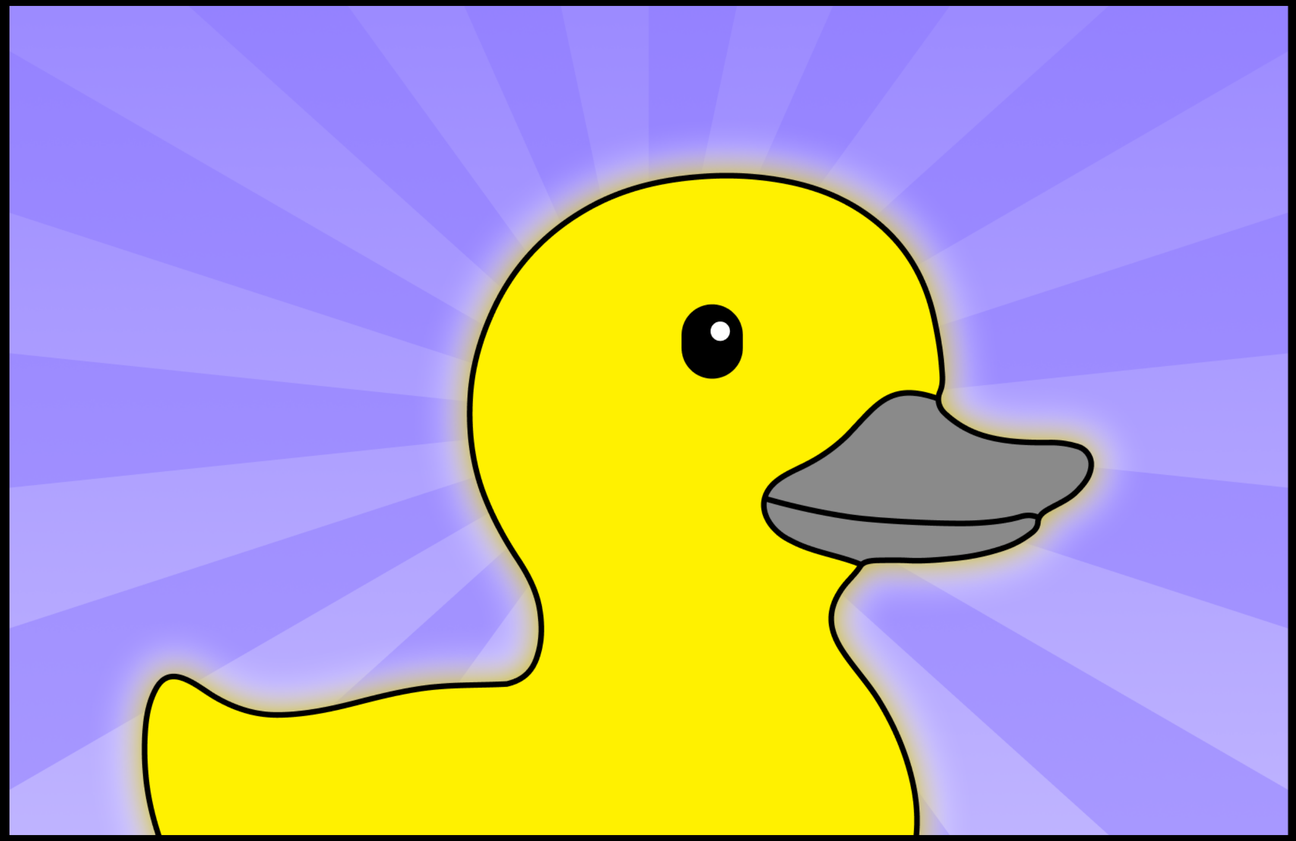
(4 seconds)

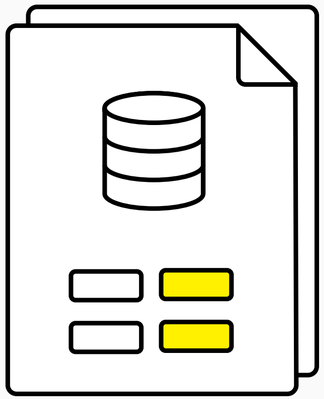


200 characters =
70% reduction

Modern SQL

Binary files





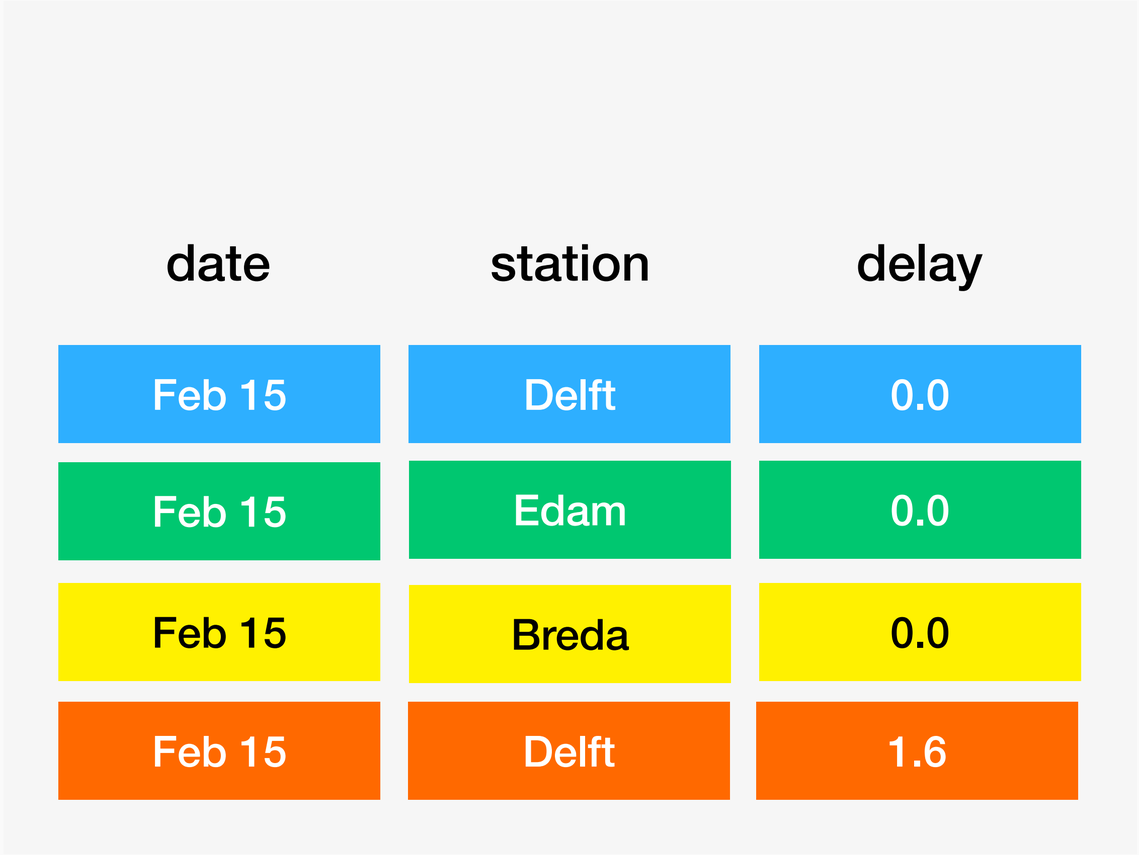
Data layout:

Row-based or column-based?

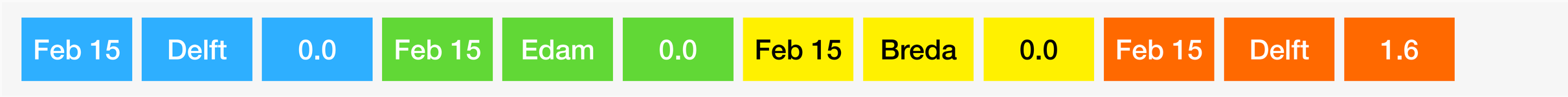
Row-based storage

date	station	delay
Feb 15	Delft	0.0
Feb 15	Edam	0.0
Feb 15	Breda	0.0
Feb 15	Delft	1.6

Row-based storage

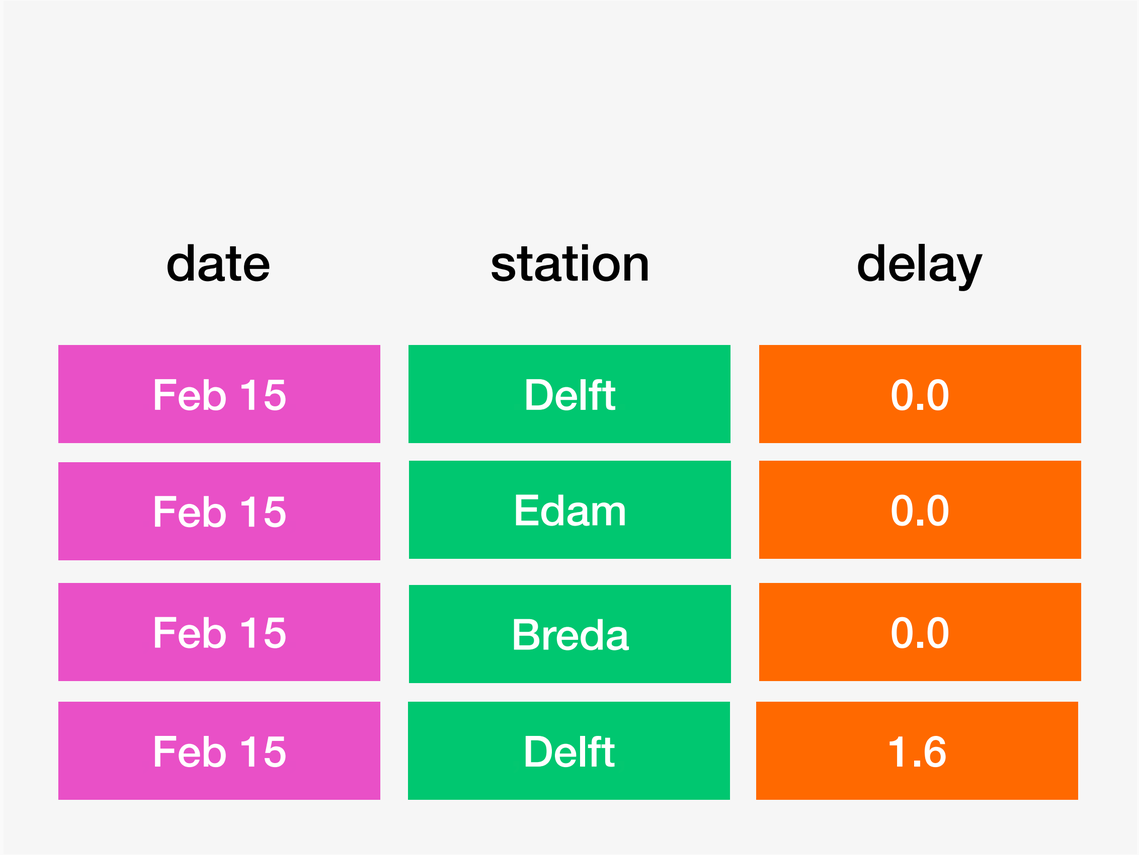


Row-based storage

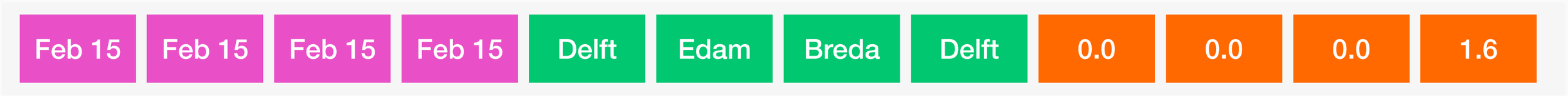


e.g.: Avro, Thrift

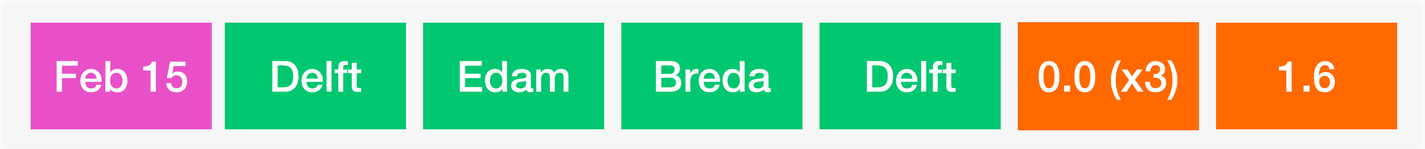
Column-based storage



Column-based storage



Column-based storage



e.g.: Parquet, Vortex

```
$ duckdb

D SELECT
      station,
      avg(delay)
  FROM 'https://blobs.duckdb.org/trains-2025-feb.parquet'
  WHERE strftime('%a', date) IN ('Sat', 'Sun')
  GROUP BY station
  ORDER BY #2 DESC
  LIMIT 3;
```

station	avg(delay)
Siegburg/Bonn	5.58
Emmerich-Elten	3.97
Brussel-Zuid	3.86

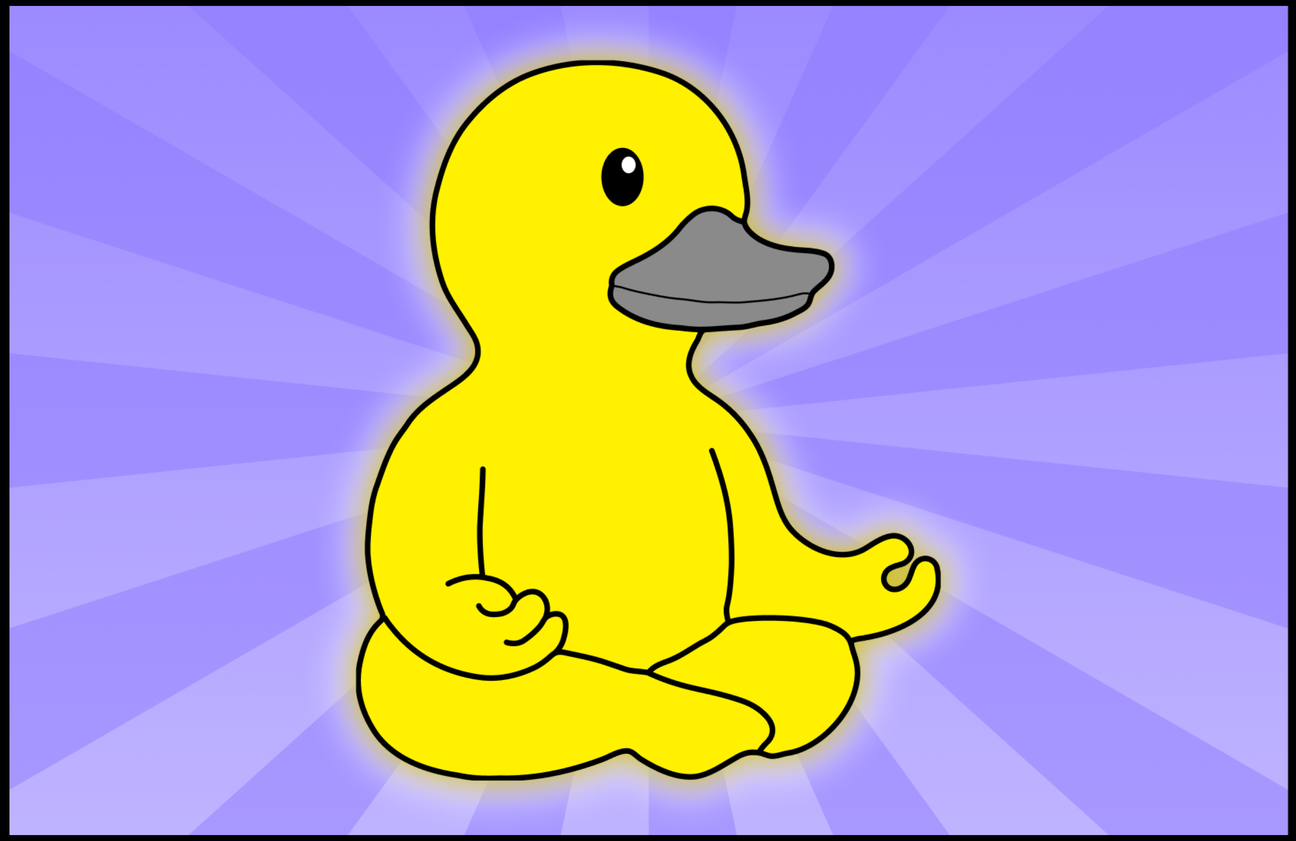
(1 second)

Concise

Safe

Fast

Database



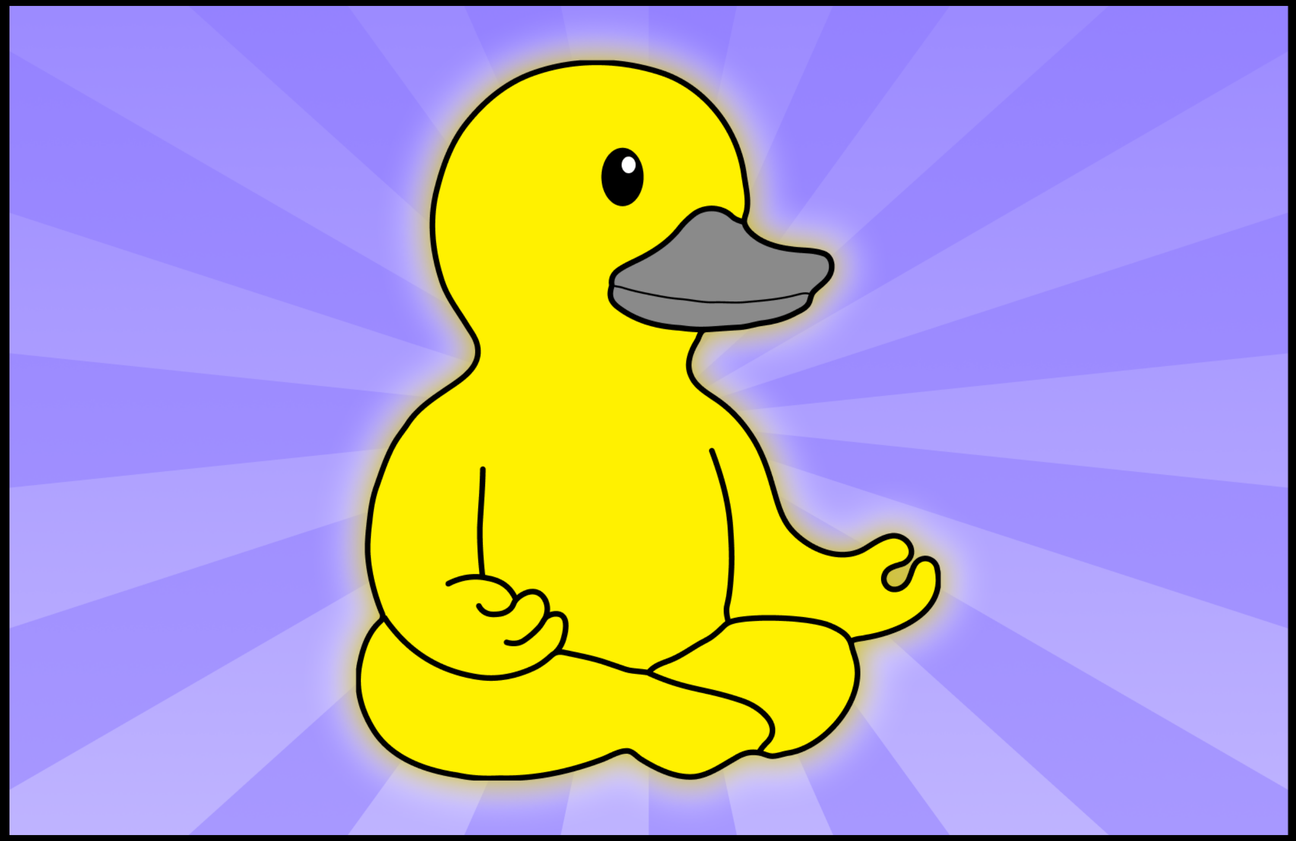
```
$ duckdb train-services.db

D SELECT
      station,
      avg(delay)
  FROM services
  WHERE strftime('%a', date) IN ('Sat', 'Sun')
  GROUP BY station
  ORDER BY #2 DESC
  LIMIT 3;
```

station	avg(delay)
Siegburg/Bonn	5.58
Emmerich-Elten	3.97
Brussel-Zuid	3.86

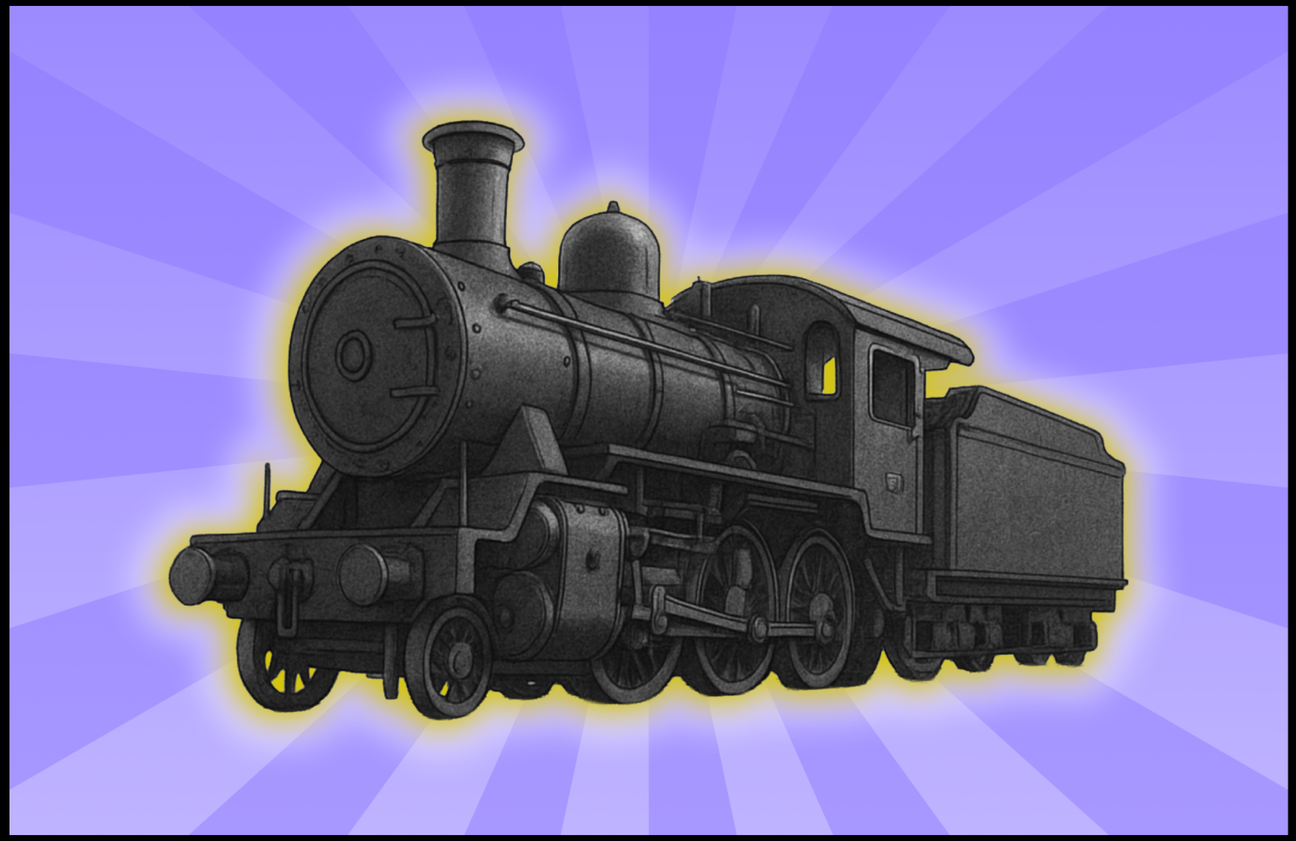
(0.01 seconds)

Super fast!

Database



Railway operators



The worst delay in Amsterdam Centraal is almost 3 hours!

```sql
SELECT date, train_number, delay
FROM services
WHERE date = '2025-02-27'
  AND station = 'Amsterdam Centraal'
  AND train_number = 420;
```

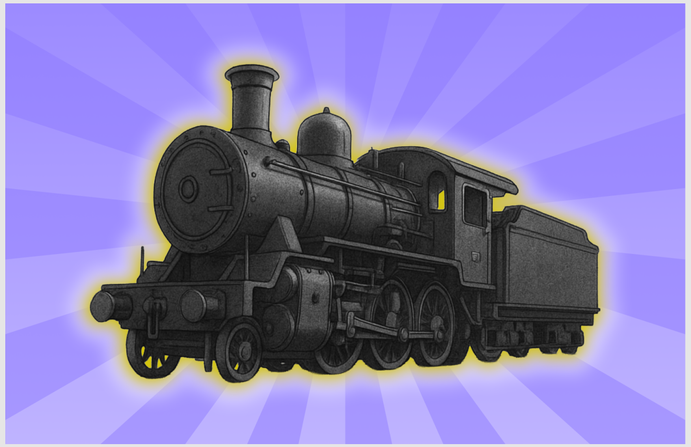
date	train_number	delay
2025-02-27	420	174

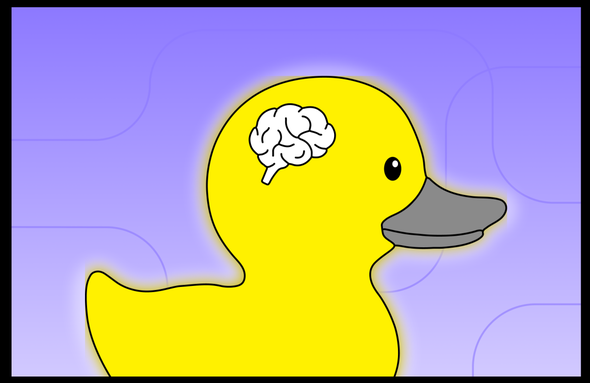
A fictional story
from a railway operator

Change the train's number!

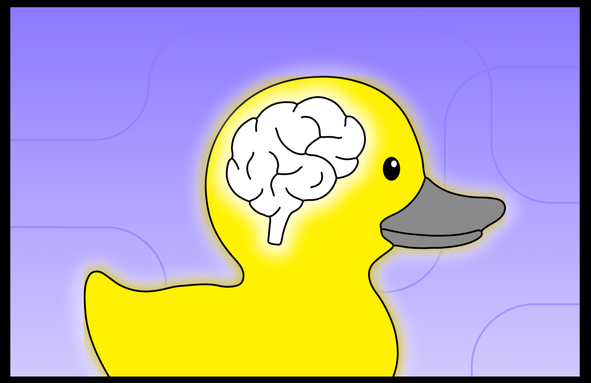
```
UPDATE services
SET train_number =
    train_number + (random() * 100 + 1)::INT
WHERE delay > 150;

(23 rows updated)
```

date	train_number	delay
0 rows		

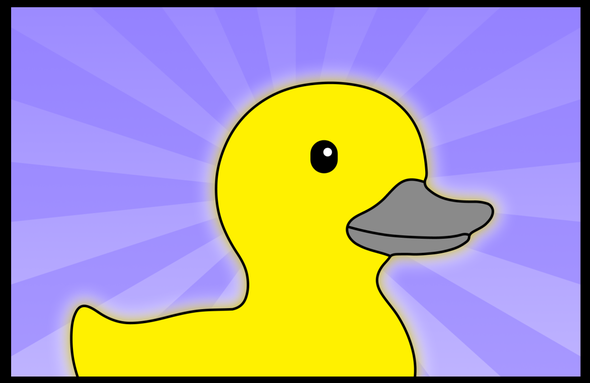




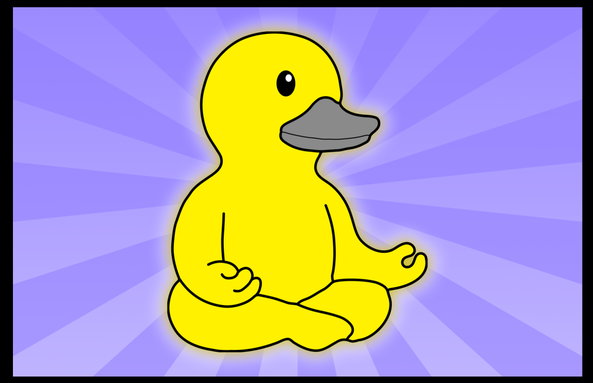
Text files



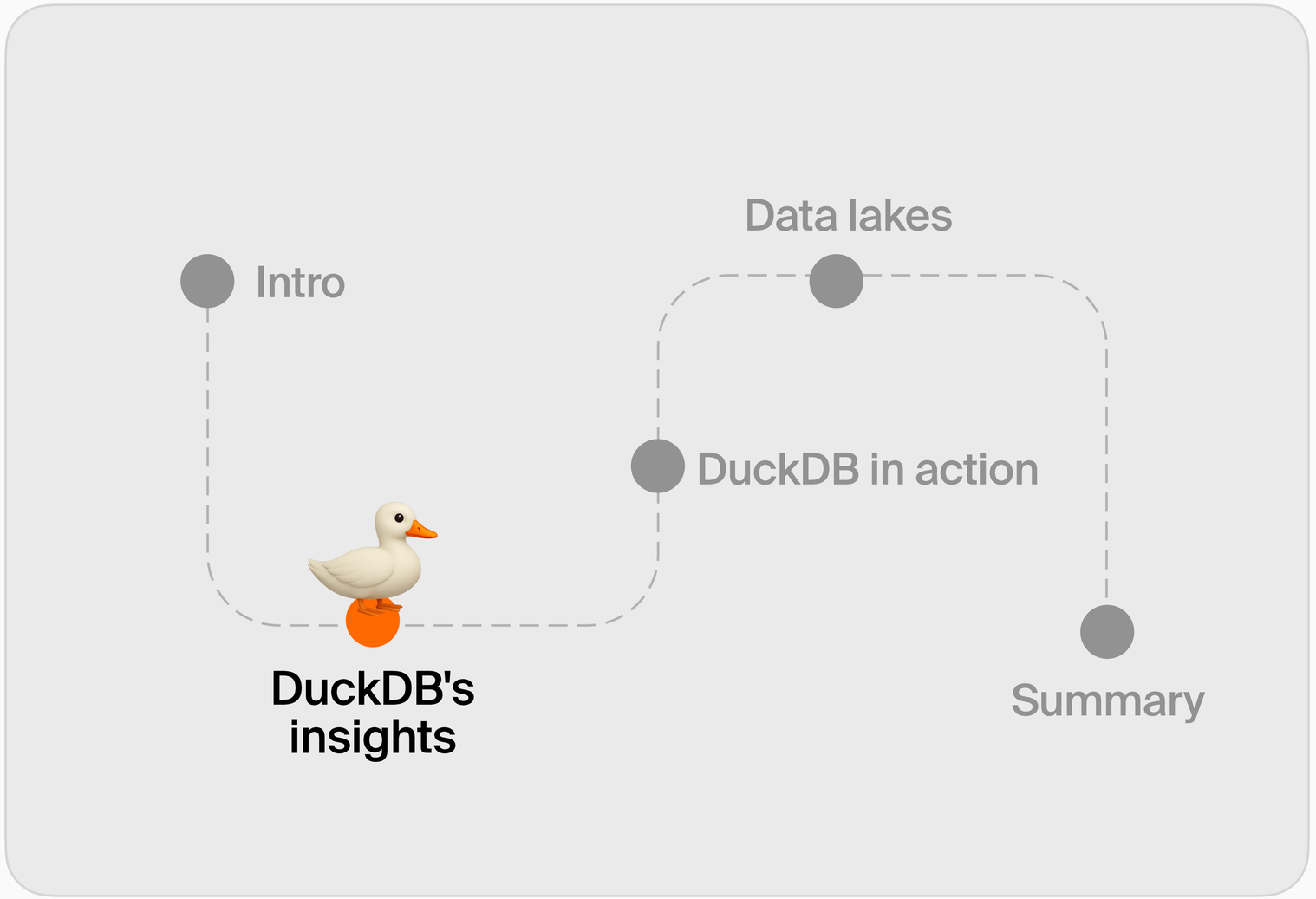
Text files (deluxe)

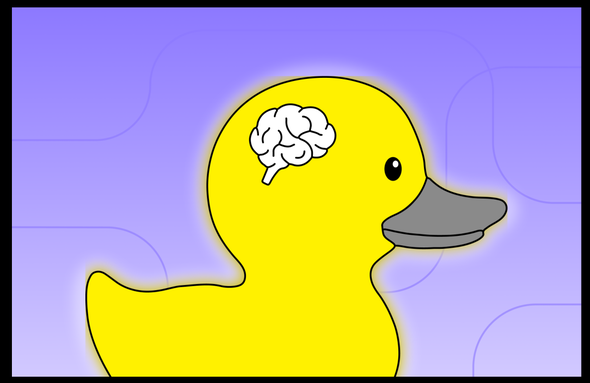


Binary files

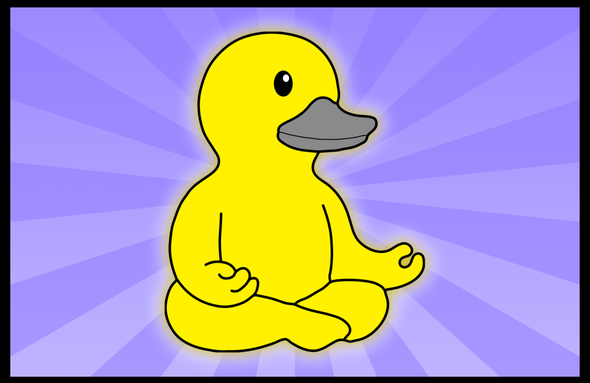


Database


Intro
Data lakes
DuckDB in action
DuckDB's insights
Summary



Text files



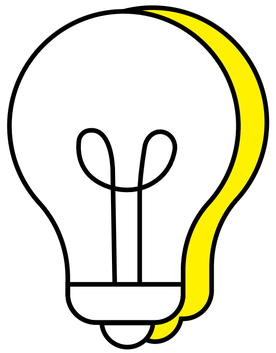
Database

Concise queries

Safe

Fast

Updatable



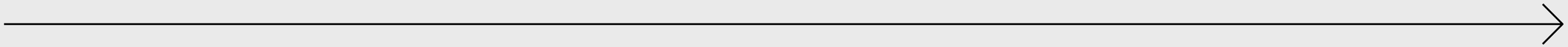
INSIGHT #1

Databases can target data science workloads

Data science landscape, 2015

?

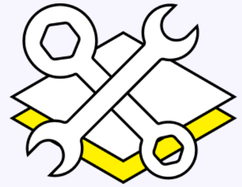
`data.tables`



data size

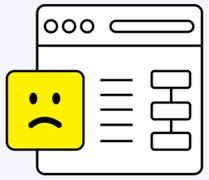
Resistance: Why would I use a database?

```
$ sudo apt install your_favorite_database
$ ... configure for 60 minutes ...
$ sudo service start your_favorite_database
$ wget https://blobs.duckdb.org/trains-2025-feb.csv.gz
$ gunzip trains-2025-feb.csv.gz
$ db_client -u admin -p admin

# CREATE TABLE ...; COPY ...; SELECT ...;
```

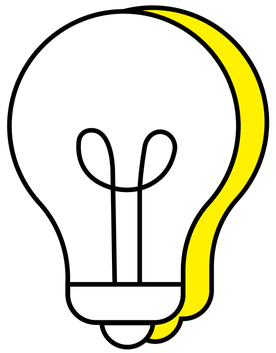


Difficult operation

Expensive



Bad UX



INSIGHT #2

Focus on usability

End-to-end performance

End-to-end performance

setup

define schema

write queries

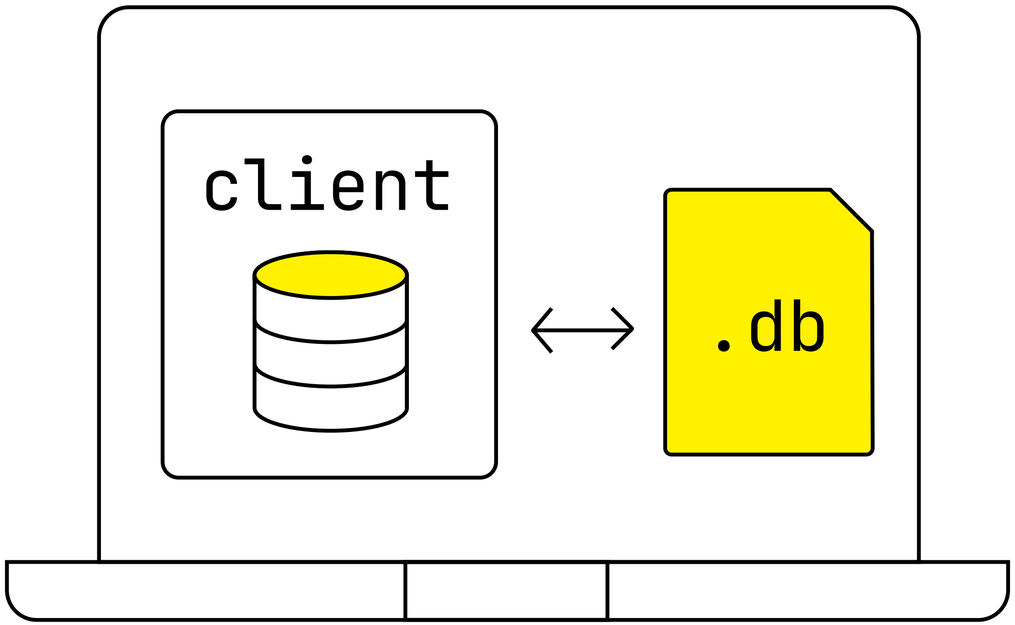
load

run queries

End-to-end performance

In-process architecture

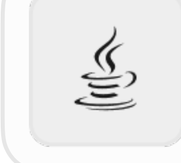
setup



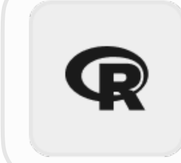
In-process architecture

```
pip install duckdb
```

```
curl https://install.duckdb.org | sh
```



```
go get github.com/duckdb/duckdb-go/v2
```



End-to-end performance

Friendly SQL

define schema

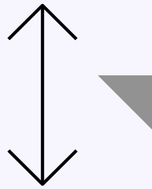
write queries

```sql
SELECT
    date, station, avg(delay),
--  min(delay)
FROM 'https://blobs.duckdb.org/trains-2025-feb.csv.gz'
GROUP BY ALL;
```

Pandas integration

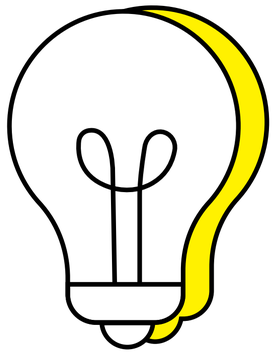
```
import duckdb
import pandas as pd

my_df = pd.DataFrame \
          .from_dict({'a': [42, 43]})

res = duckdb \
        .sql("SELECT avg(a) FROM my_df")

res.df()

#       s
# 0  42.5
```

Replacement scan



Zero-copy access

pandas

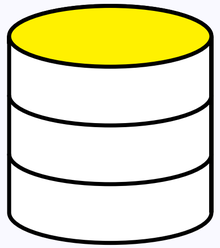


INSIGHT #3

“Big Data” is overrated

2010 Data processing landscape

Lots of data



Weak hardware

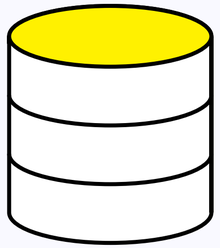
Inefficient software

```
  0[||||100.0%]   1[       0.0%]
  2[       0.0%]  3[       0.0%]
  4[       0.0%]  5[       0.0%]
  6[       0.0%]  7[       0.0%]
Mem[||||                362M/27.4G]
```

2025 Data processing landscape

Lots of data

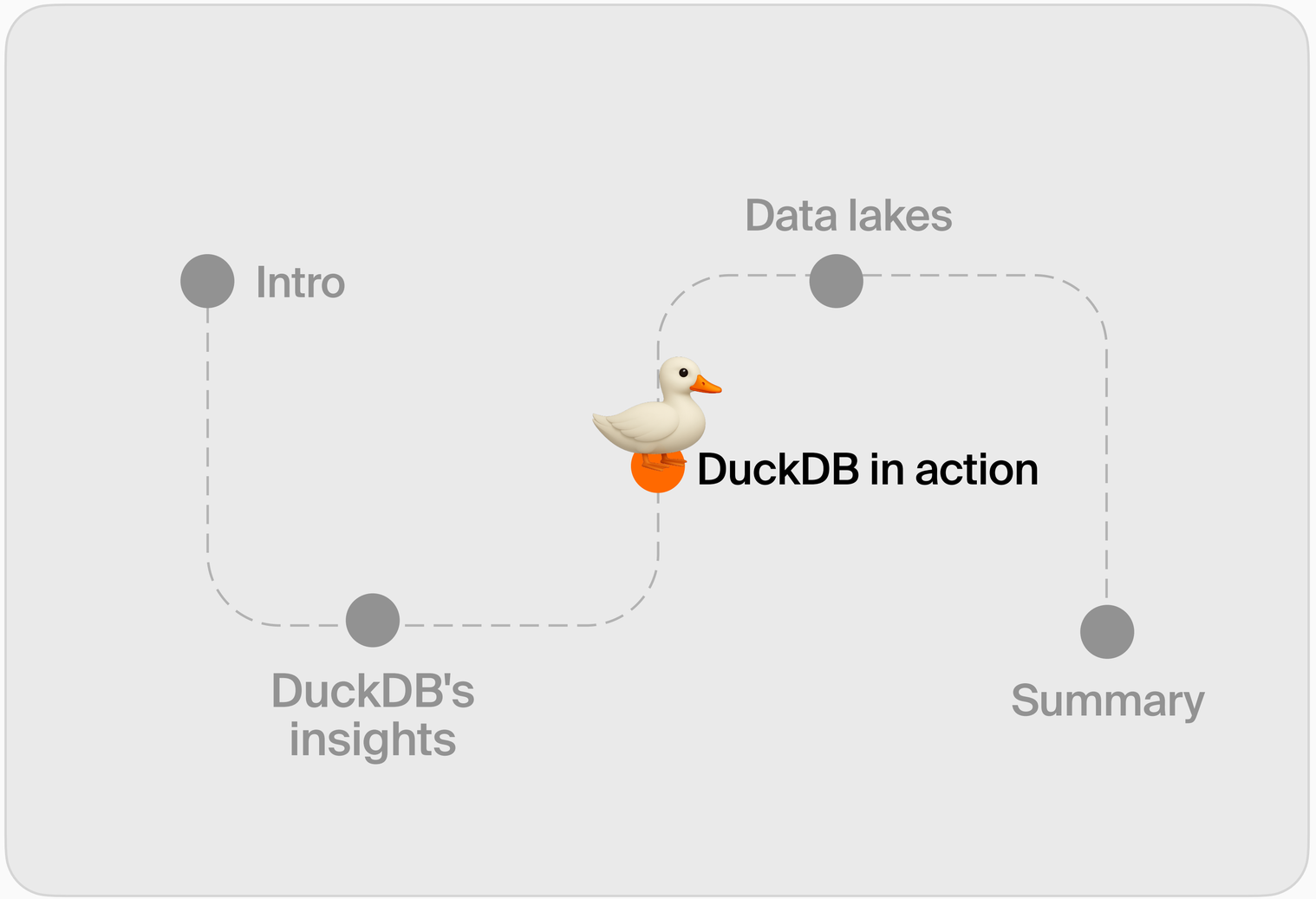
but most queries
only access a small amount

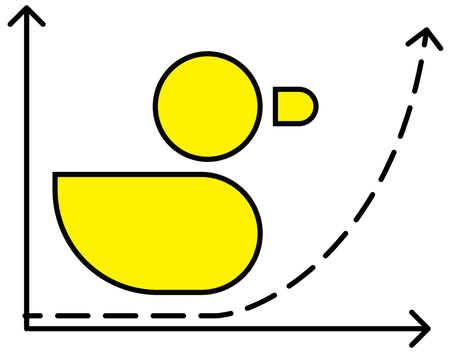


Strong hardware

New techniques

columnar storage,
lightweight compression,
vectorized query execution


Intro
Data lakes
DuckDB in action
DuckDB's insights
Summary



Performance

TPC-H benchmark

TPC®

Alibaba.com · AMD · 柏睿数据 · cisco · DELL Technologies · Hewlett Packard Enterprise · HITACHI
HUAWEI · IEIT SYSTEMS · 浪潮云 · intel · Lenovo · Microsoft · NUTANIX
NVIDIA · ORACLE · Red Hat · timecho · TRANSWARP · TTA · vmware

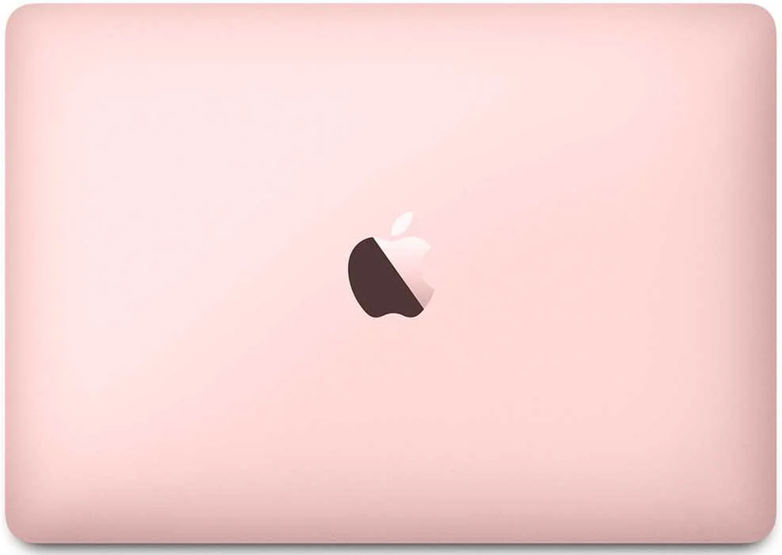
Home | About the TPC | Benchmarks/Results | Downloads | TPCTC | Miscellaneous | Search | Newsletter | Member Login

1,000 GB Results

Rank	Company	System	QphH	Price/kQphH	Watts/KQphH	System Availability	Database	Operating System	Date Submitted
1	Hewlett Packard Enterprise	HPE ProLiant DL345 Gen11 (with AMD EPYC 9375F)	1,531,016	192.37 USD	NR	10/01/25	Microsoft SQL Server 2022 Enterprise Edition 64 bit	Microsoft Windows Server 2022 Standard Edition	10/01/25
2	Hewlett Packard Enterprise	HPE Proliant DL345 Gen11 (with AMD EPYC 9374F Proc	1,257,628	233.57 USD	NR	07/29/25	Microsoft SQL Server 2022 Enterprise Edition 64 bit	Microsoft Windows Server 2022 Standard Edition	07/29/25
3	Hewlett Packard Enterprise	HPE ProLiant DL380 Gen12	1,184,211	263.13 USD	NR	06/02/25	Microsoft SQL Server 2022 Enterprise Edition 64 bit	Microsoft Windows Server Datacenter 2025 Edition	04/23/25

TPC-H unplugged

100 GB



1 000 GB

10 000 GB

Lightweight indexing

Which stations are affected by delays > 2 min?

row group 1

	date	station	delay
min	Feb 15	Breda	0.0
max	Feb 15	Edam	1.6

✗

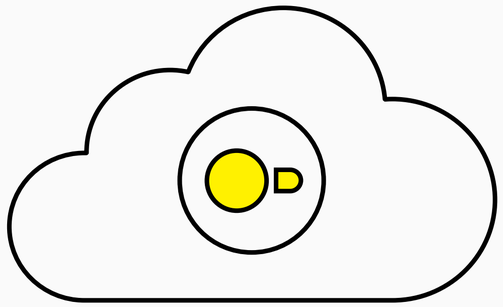
date	station	delay
Fe	Delft	.0
Feb 15		0.0
Feb 15		0.0
Fe	Delft	6

row group 2

	date	station	delay
min	Feb 16	Delft	0.0
max	Feb 23	Utrecht	3.7

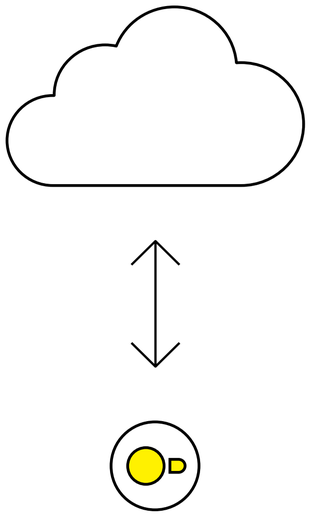
✓

date	station	delay
Feb 16	Utrecht	2.5
Feb 16	Gouda	0.0
Feb 16	Edam	0.8
Feb 23	Delft	3.7



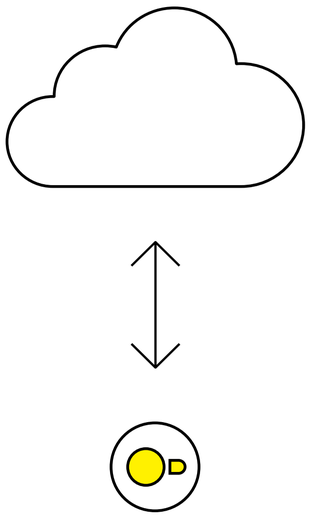
DuckDB and the cloud

Which stations are affected by delays > 2 min?



```sql
SELECT station, delay
FROM 's3://my-bucket/trains-2025-feb.parquet'
WHERE delay > 2;
```

Which stations have delays > 2 min?



row group 1

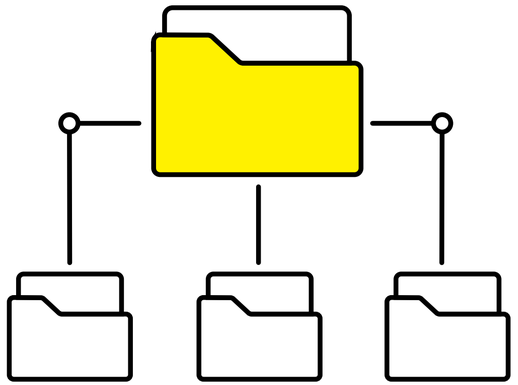
	date	station	delay
min	Feb 15	Breda	0.0
max	Feb 15	Edam	1.6

row group 2

	date	station	delay
min	Feb 16	Delft	0.0
max	Feb 23	Utrecht	3.7

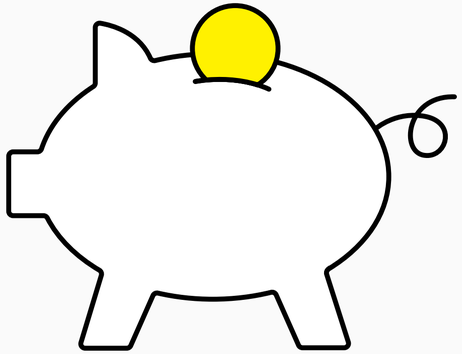
date	station	delay
Feb	Delft	0.0
Feb 15		0.0
Feb 15		0.0
Fe	Delft	

date	station	delay
Feb 16	Utrecht	2.5
Feb 16	Gouda	0.0
Feb 16	Edam	0.8
Feb 23	Delft	3.7



Use cases

Cost saving



Fast local development:
time & egress fees

Replace proprietary systems:
license cost

Replace distributed systems:
hardware cost

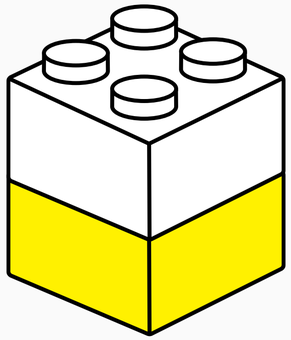
Last mile analytics

1 PB log →
pre-process with Spark

Process the 200 GB
aggregate DuckDB

Build fast dashboards

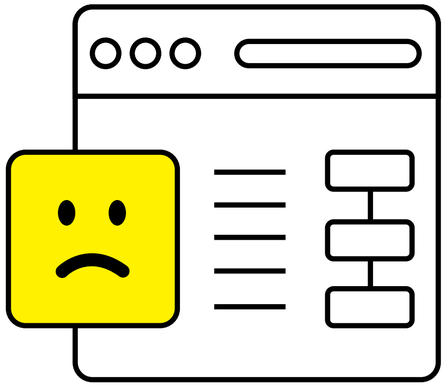
Modernizing codebases



Postgres-compatible SQL dialect

Builds on open formats (CSV, Parquet)

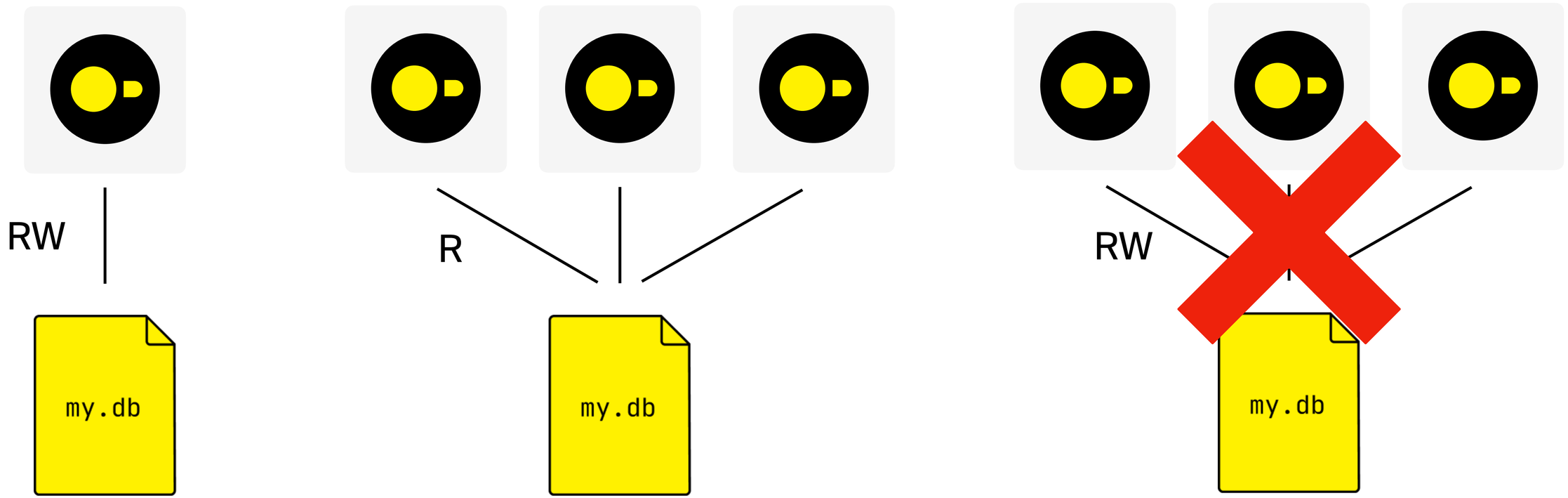
LLMs can refactor code to use DuckDB



Limitations

What's not so great about DuckDB?

Single-player experience: just one writer



What's not so great about DuckDB?

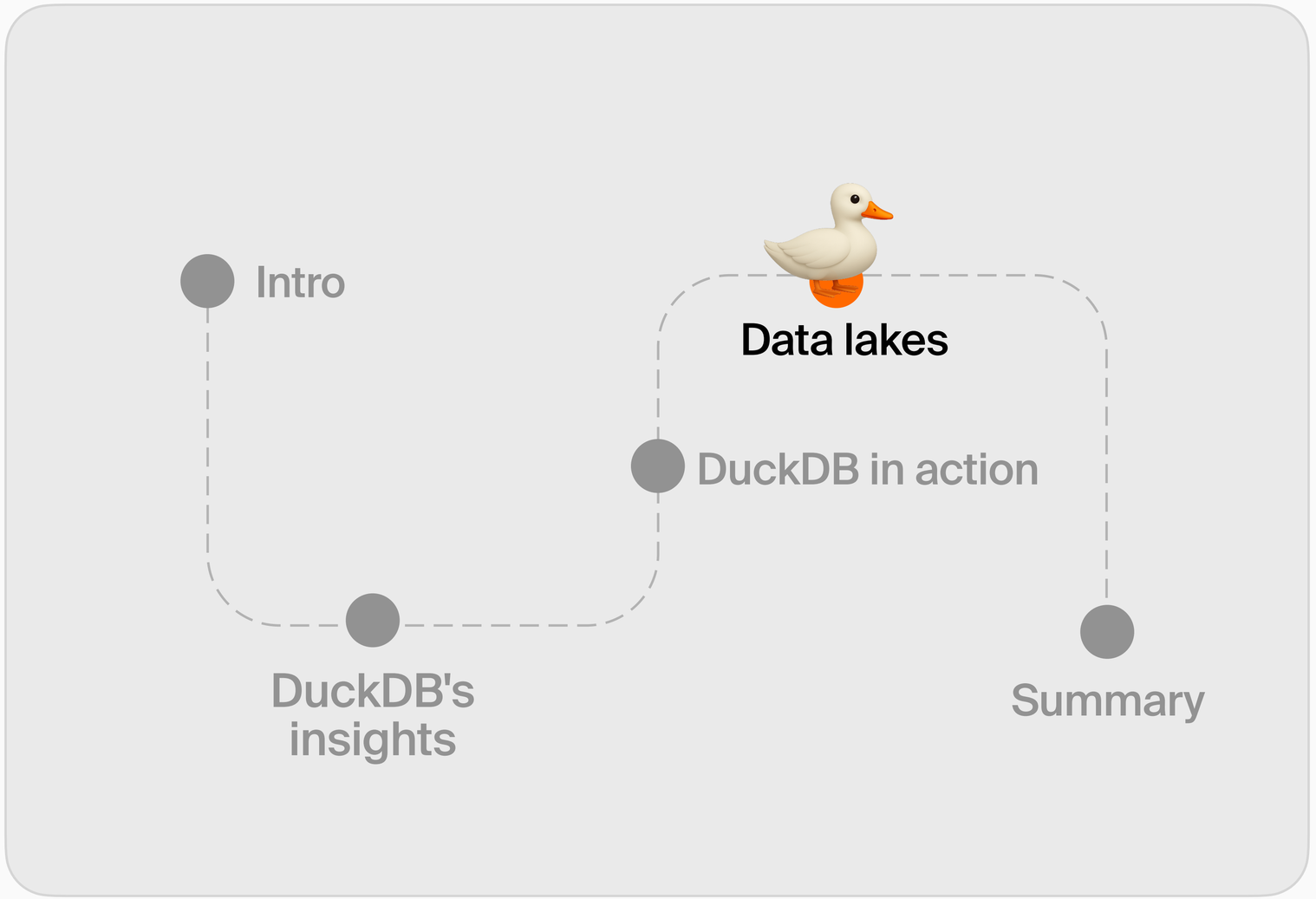
10 TB CSV = 2.7 TB database file

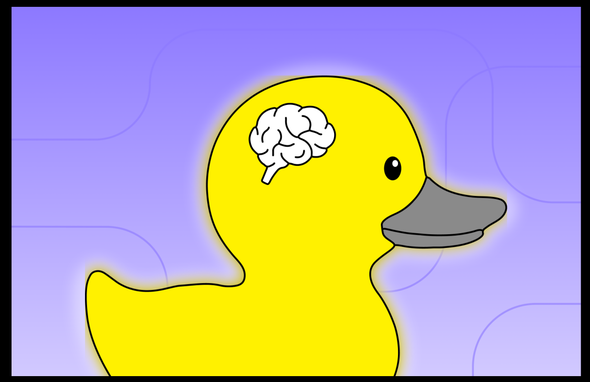
Optimized for bulk insert

small writes are slow

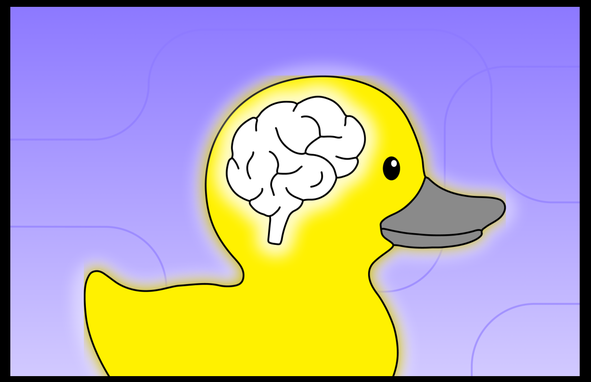
Huge database files

can be unwieldy

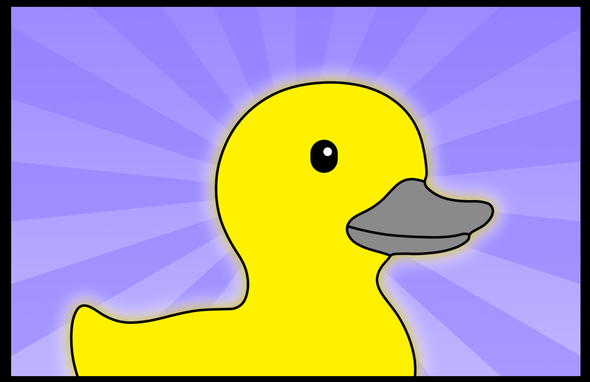

Intro
Data lakes
DuckDB in action
DuckDB's insights
Summary



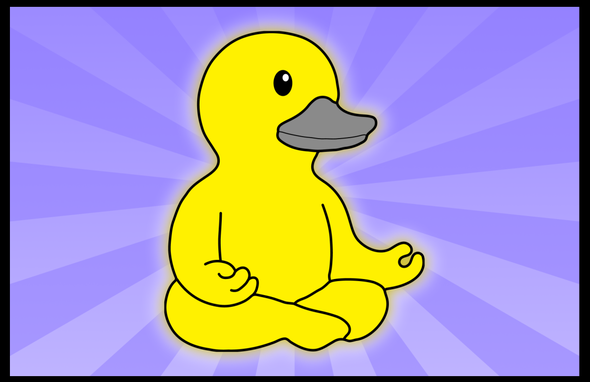
Text files



Text files (deluxe)



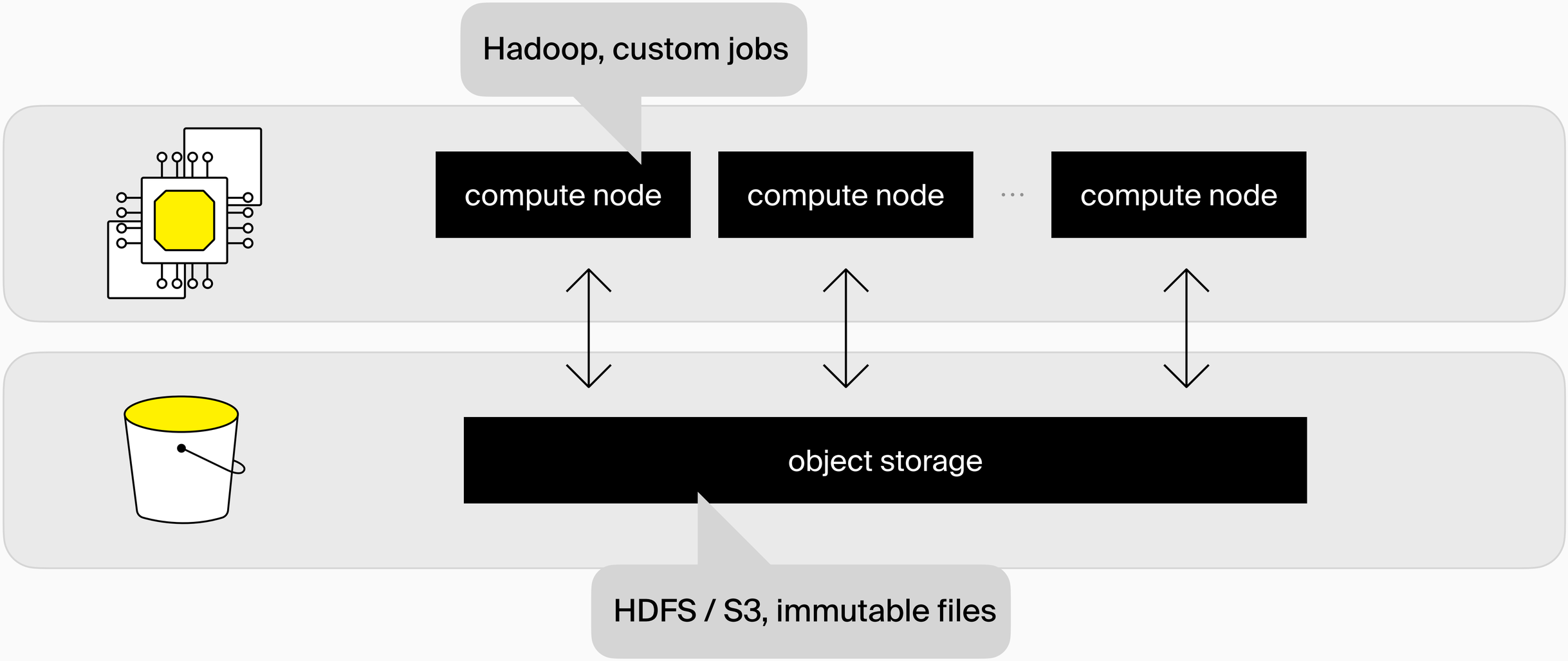
Binary files



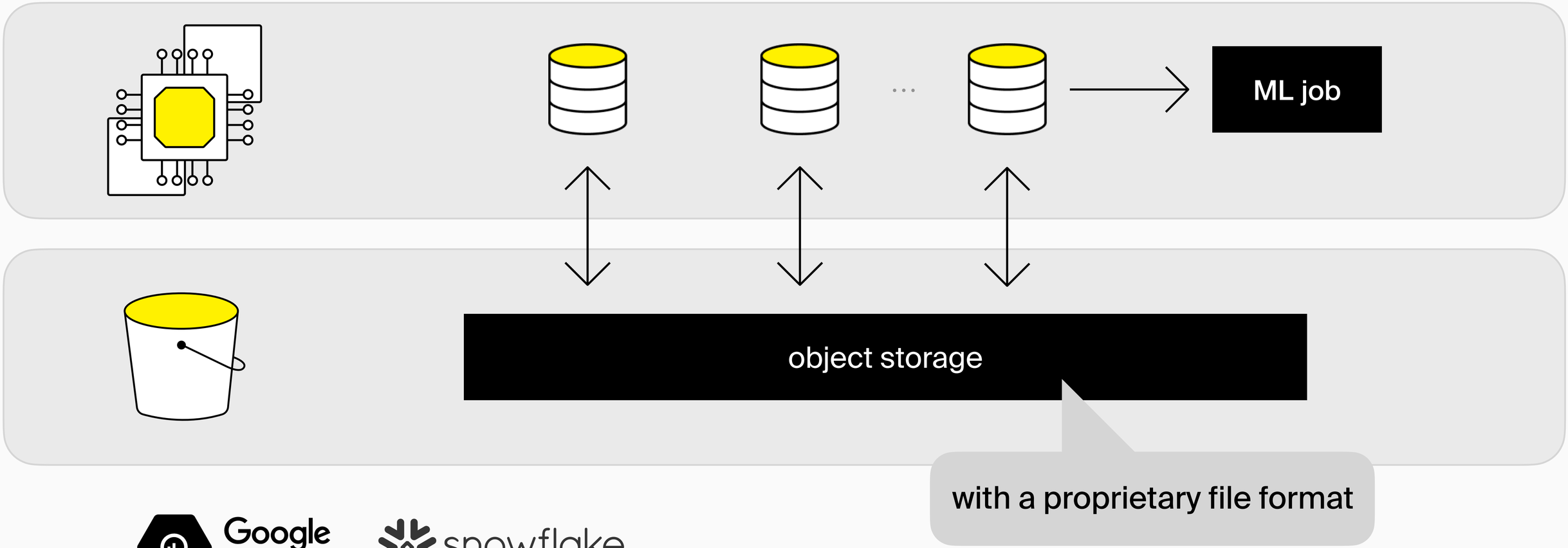
Database

Data lake

2005 Disaggregated storage



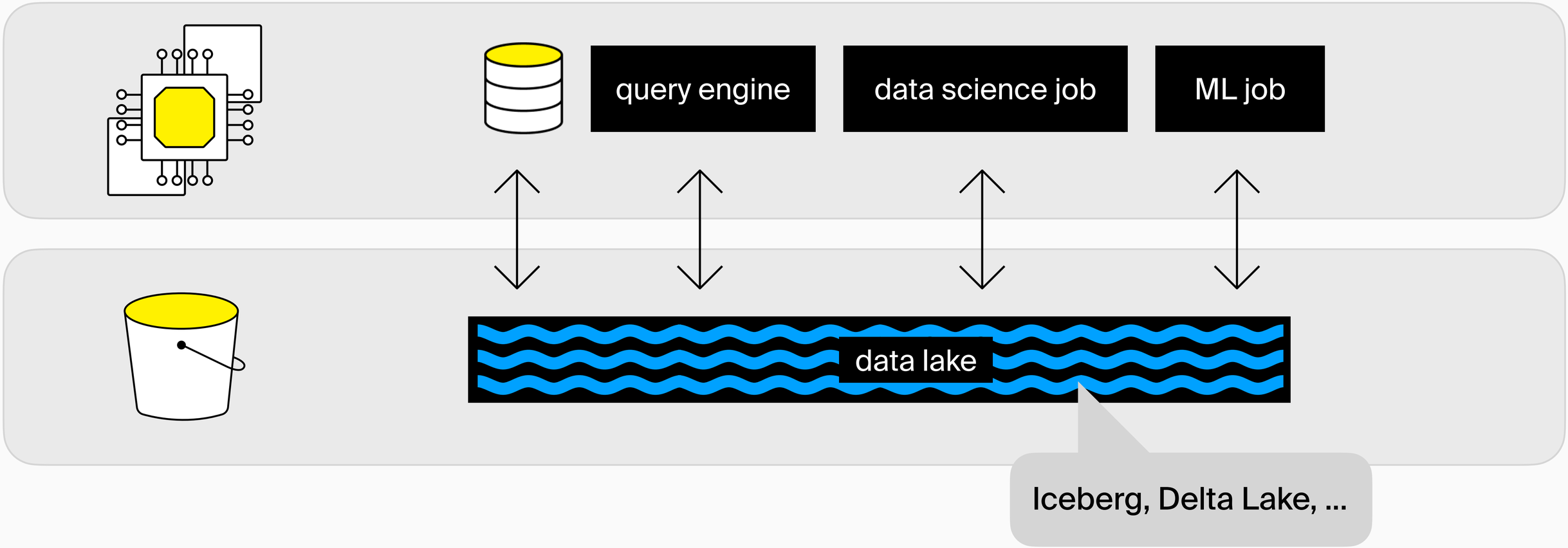
2010 Cloud data warehouses



Google Big Query

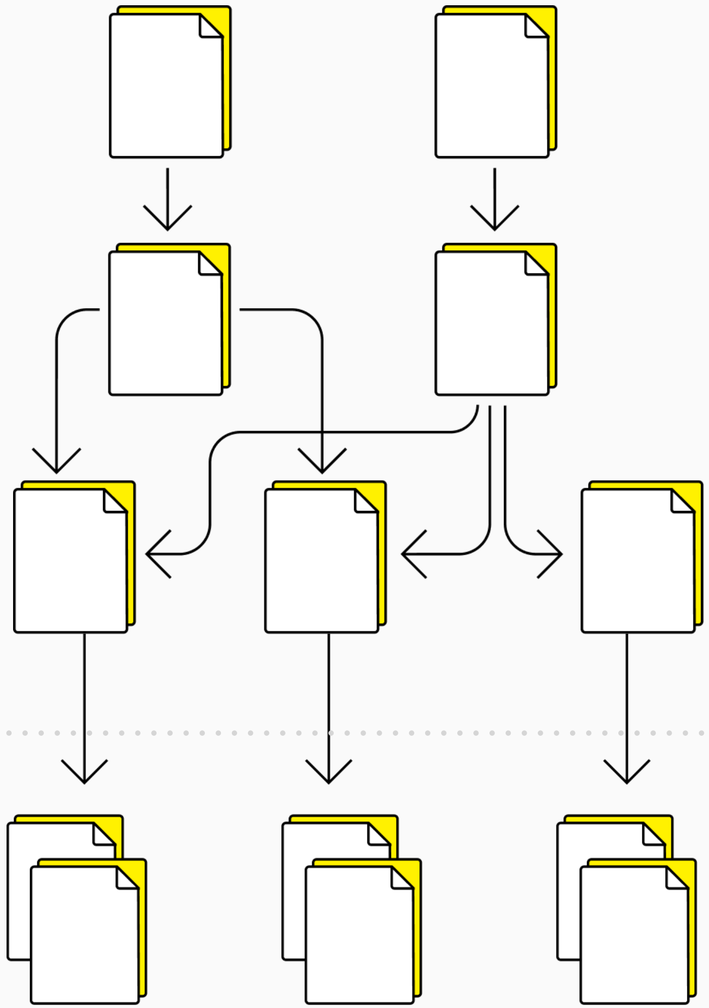
snowflake

2015 Data lake architecture



ICEBERG

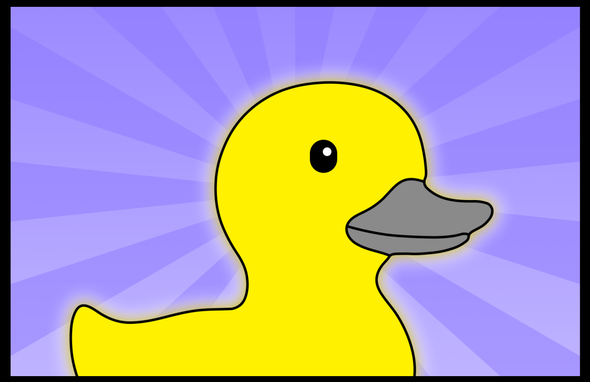
metadata layer
(json and avro)



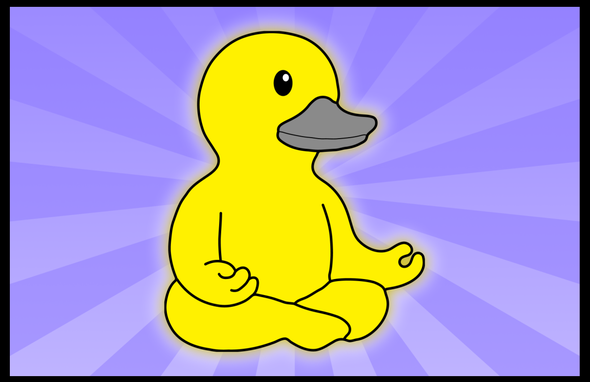
data layer
(parquet)

- File-only architecture
- Transactional updates
- Time travel queries
- Schema evolution

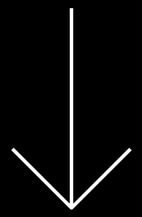
- Performance issues
- “Small files” problem
- Limited to a single table



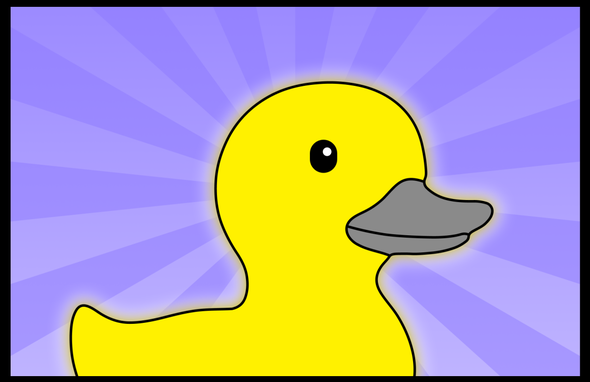
Binary files



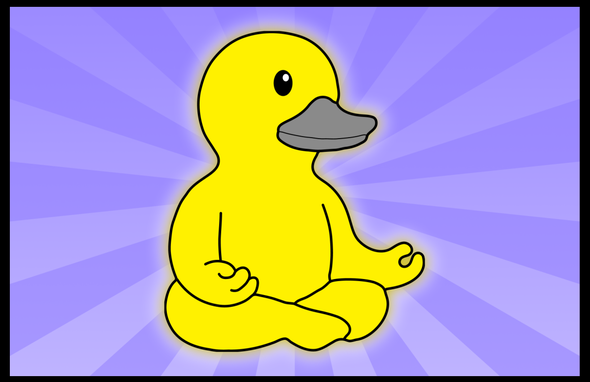
Database



Data lake



Binary files

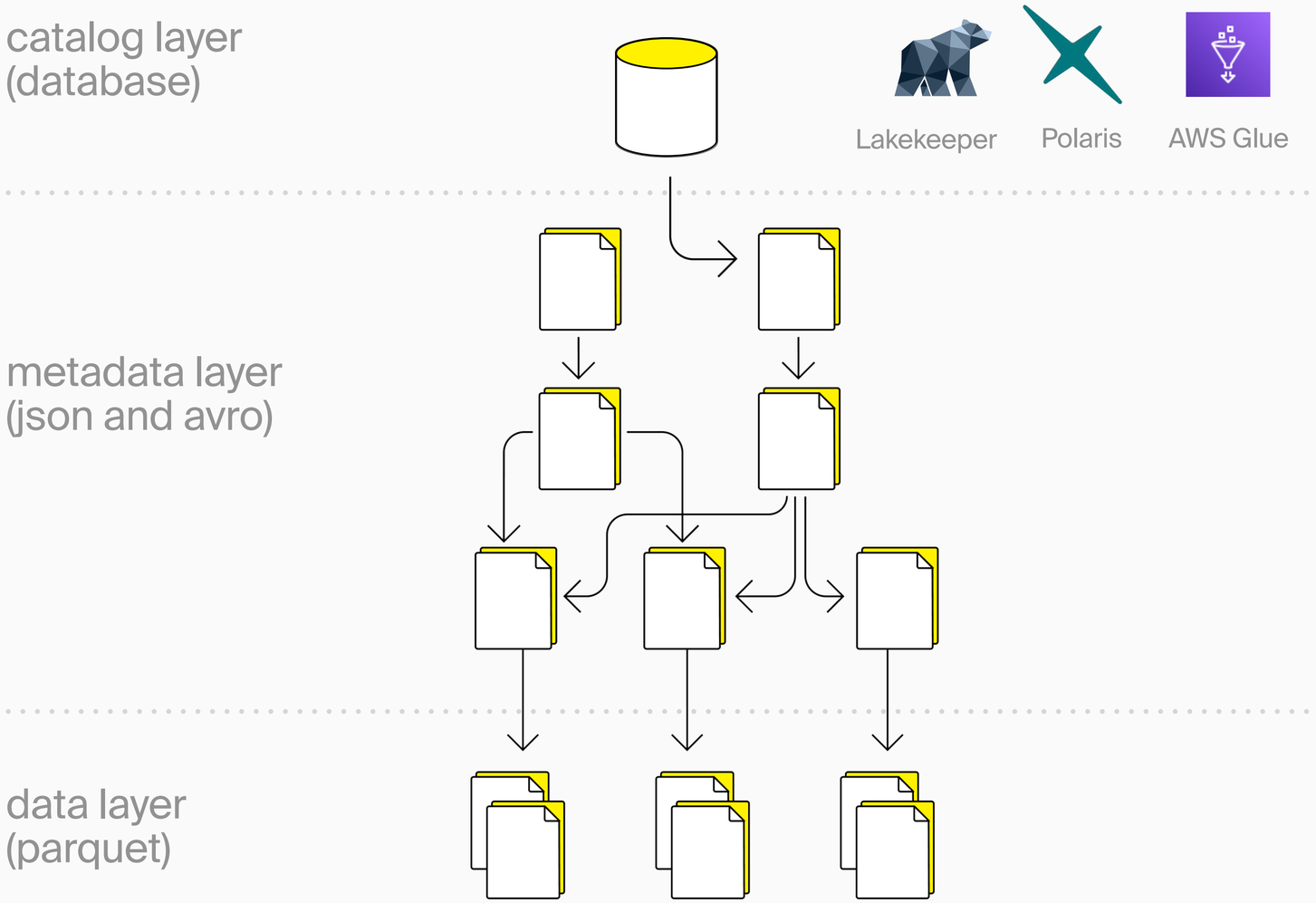


Database

Data lake

Lakehouse

2020 Lakehouse



- Cheap object storage
- Small catalog database
- Full database features
- Scalable

- Performance issues
- Operational complexity

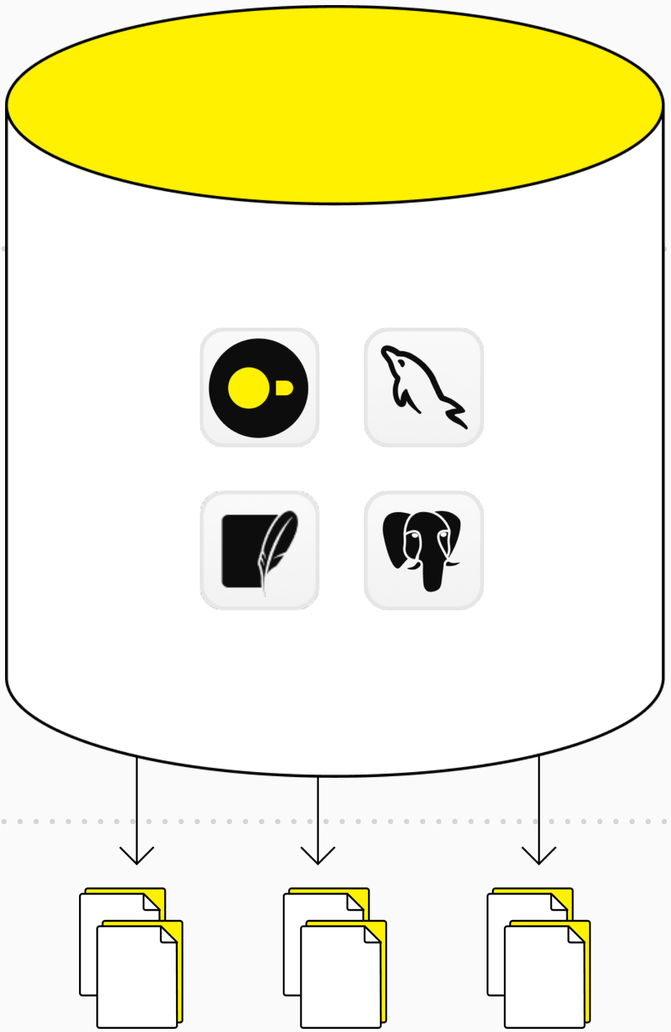
2025

DuckLake

catalog layer
(database)

metadata layer
(also a database)

data layer
(parquet)



- Cheap object storage
- Small catalog database
- Full database features
- Scalable

- Lightweight snapshots
- Multi-table ACID
- High performance

DuckLake

Quick setup

```
ATTACH 'ducklake:trains.ducklake' AS trains;
USE trains;

CREATE TABLE services AS
    FROM 'trains-2025-feb.csv';
```

DuckLake

Change the train's number!

```
UPDATE services
SET train_number =
    train_number + (random() * 100 + 1)::INT
WHERE delay > 150;

SELECT snapshot_id, changes
FROM ducklake_snapshots('trains');
```

snapshot_id int64	changes map(varchar, varchar[])
0	{schemas_created=[main]}
1	{tables_created=[main.services], tables_inserted_into=[1]}
2	{tables_inserted_into=[1], tables_deleted_from=[1]}

DuckLake

Use time travel

```
SELECT date, train_number, delay
FROM services AT (VERSION => 1)
WHERE date = '2025-02-27'
  AND station = 'Amsterdam Centraal'
  AND train_number = 420;
```

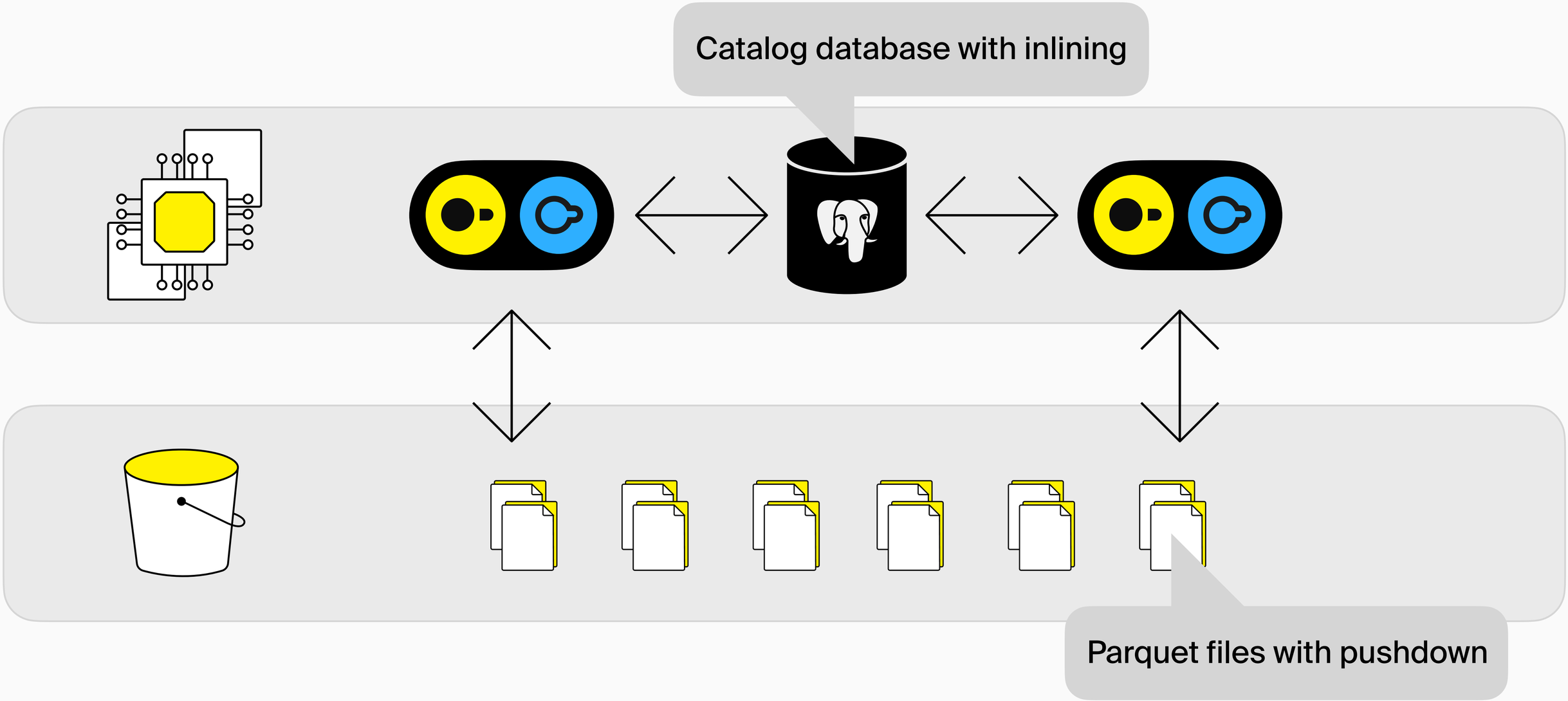
date	train_number	delay
2025-02-27	420	174

You can look back in time: train company busted

Also great for re-running fixed scripts on old data

DuckLake

Multiplayer DuckDB architecture



Laboratory experiment,
do not replicate!

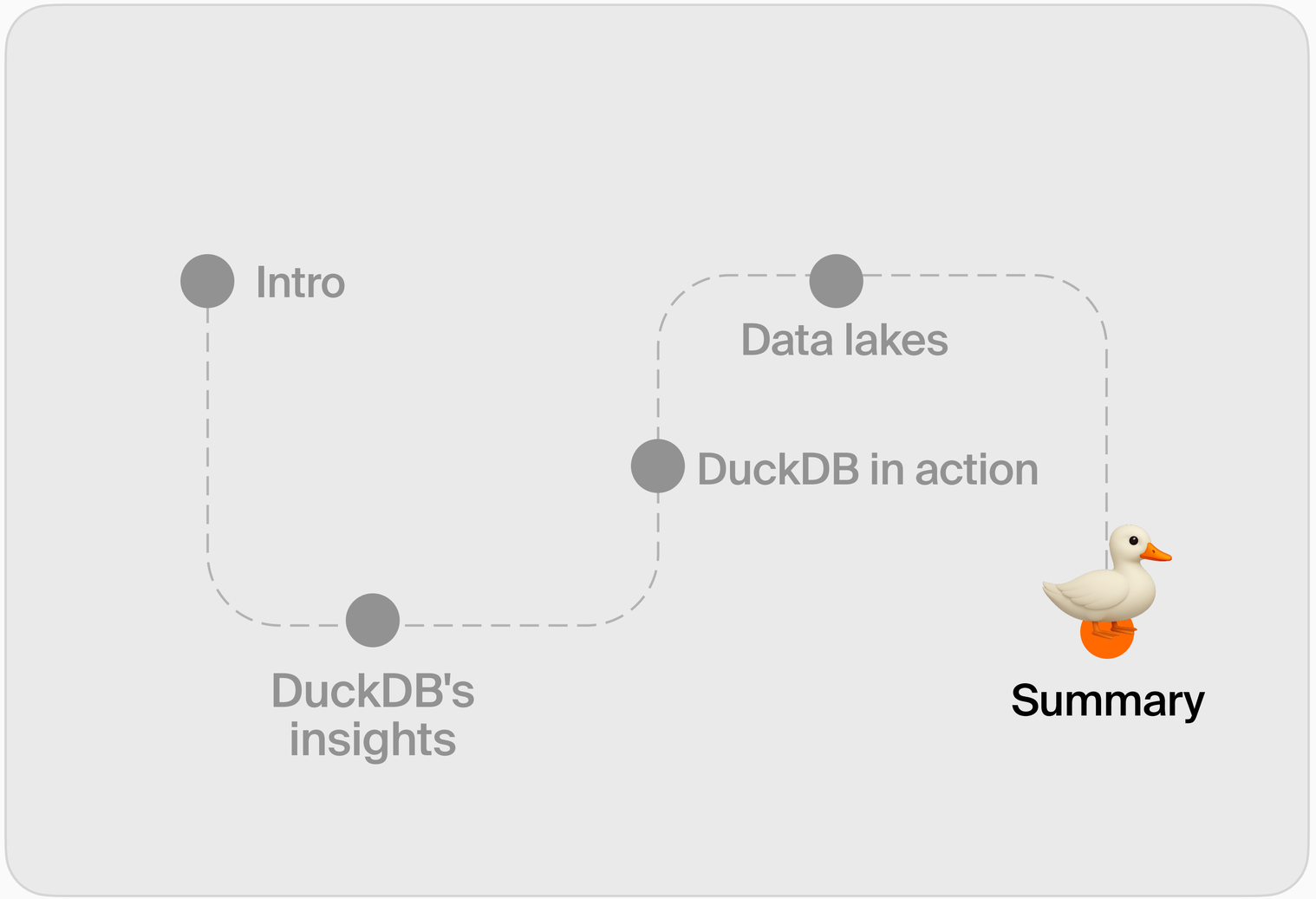
Demineralized water

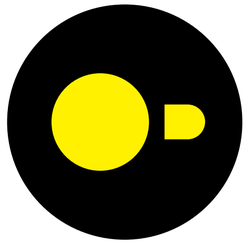
Brand new phone

Limited time underwater

Still a bad idea:
Wi-Fi and 5G signals
drop while underwater!

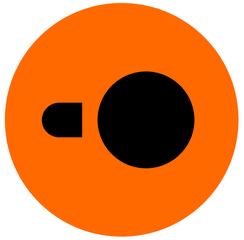
DuckLake on a phone *underwater*


Intro
Data lakes
DuckDB in action
DuckDB's insights
Summary



DuckDB Foundation

Intellectual property and TM

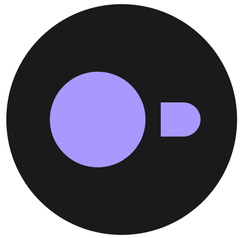


DuckDB Labs

Core DuckDB / Amsterdam

MotherDuck

Cloud data warehouse / Seattle



Community Extensions

Third-party contributions

DuckLake

Lakehouse standard

pg_duckdb

Run DuckDB in Postgres