goto;

DuckDB: Crunching data anywhere, from laptops to servers

Dr. Gábor Szárnyas

2024-06-12

DuckDB Labs

About me

PhD
2014–2019

graph databases

postdoc
2020–2023

database benchmarks

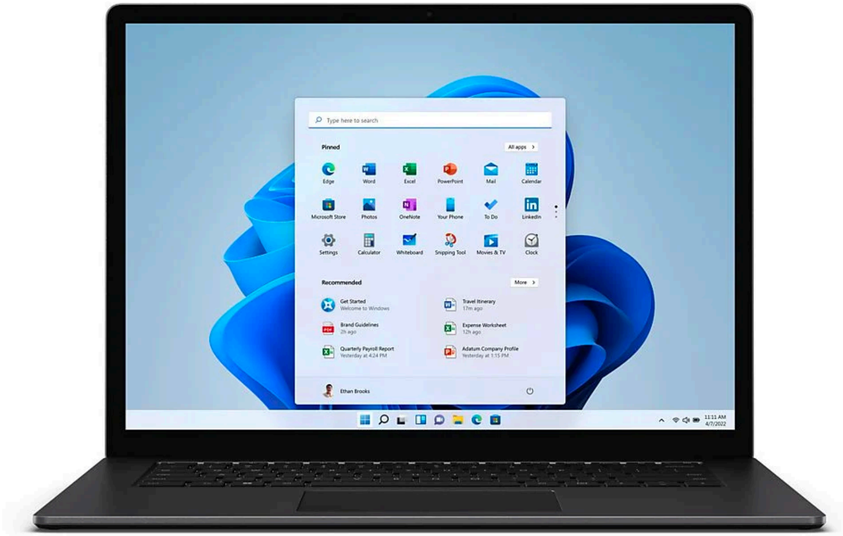
developer relations
2023–

DuckDB database

What is DuckDB?

DuckDB = an analytical database management system for processing 100 GB+ data sets on end-user devices

Apple MacBook Pro
up to 16 cores

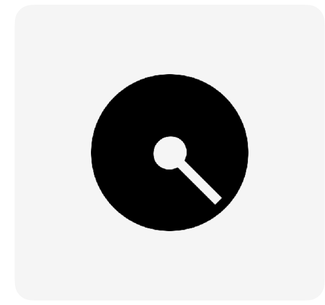


Microsoft Copilot+
up to 12 cores

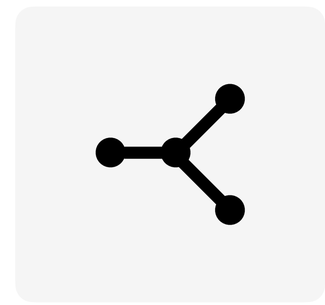
DuckDB's key values

Simple

Free



Fast



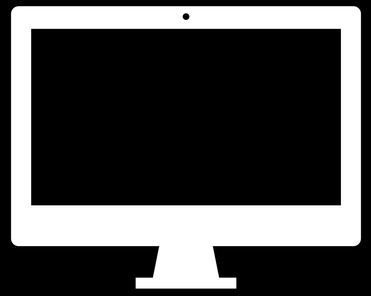
Feature-rich

Demo data set

Rijden de Treinen

Open railway datasets:

- train services
- stations
- distances



Demo

What we did *not* do

Simple

We did not

Register an account

Start a server /
Configure a client

What we did *not* do

Simple

Free

We did not

Register an account

Start a server /
Configure a client

We also did not

Enter credit card details

Ask for a quota increase

What we did

Fast

Analyze 15GB data
in seconds

Performance

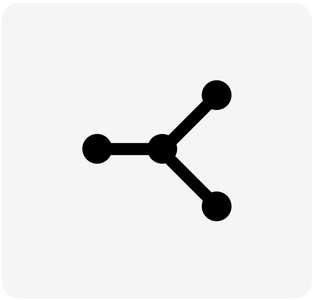
Fast

Boost from 5 years → 200 years of services

- 600 GB CSV files = 5.4 billion rows
- Load time: 16 minutes
- Aggregation: 22 seconds

Features

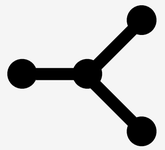
Fast



Feature-rich

Features

Fast



Feature-rich

Full SQL support: PostgreSQL dialect
Plus "friendly" SQL extensions

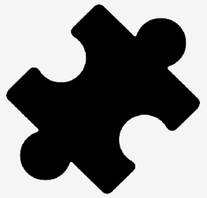
...back to the demo

More characteristics of DuckDB

In-process

Portable

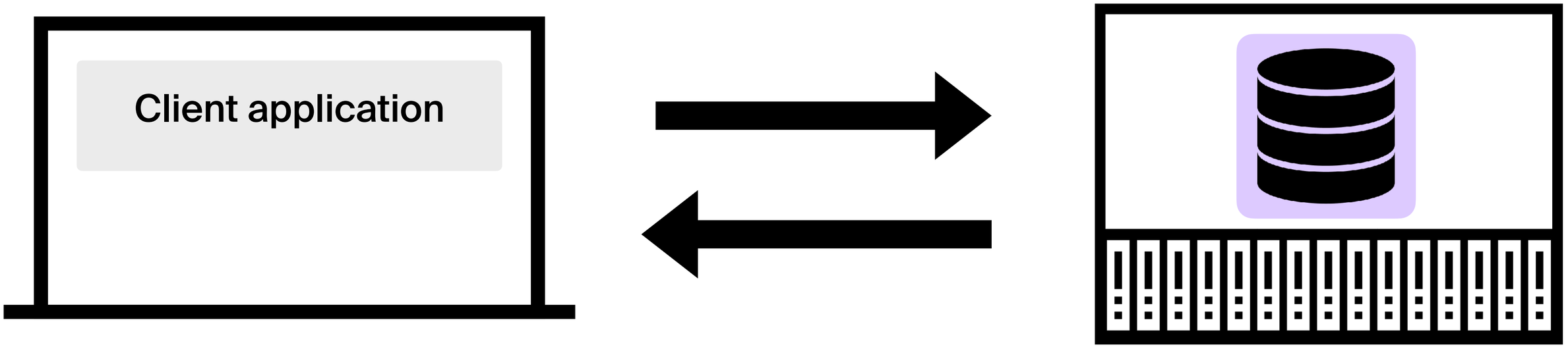
Ergonomic



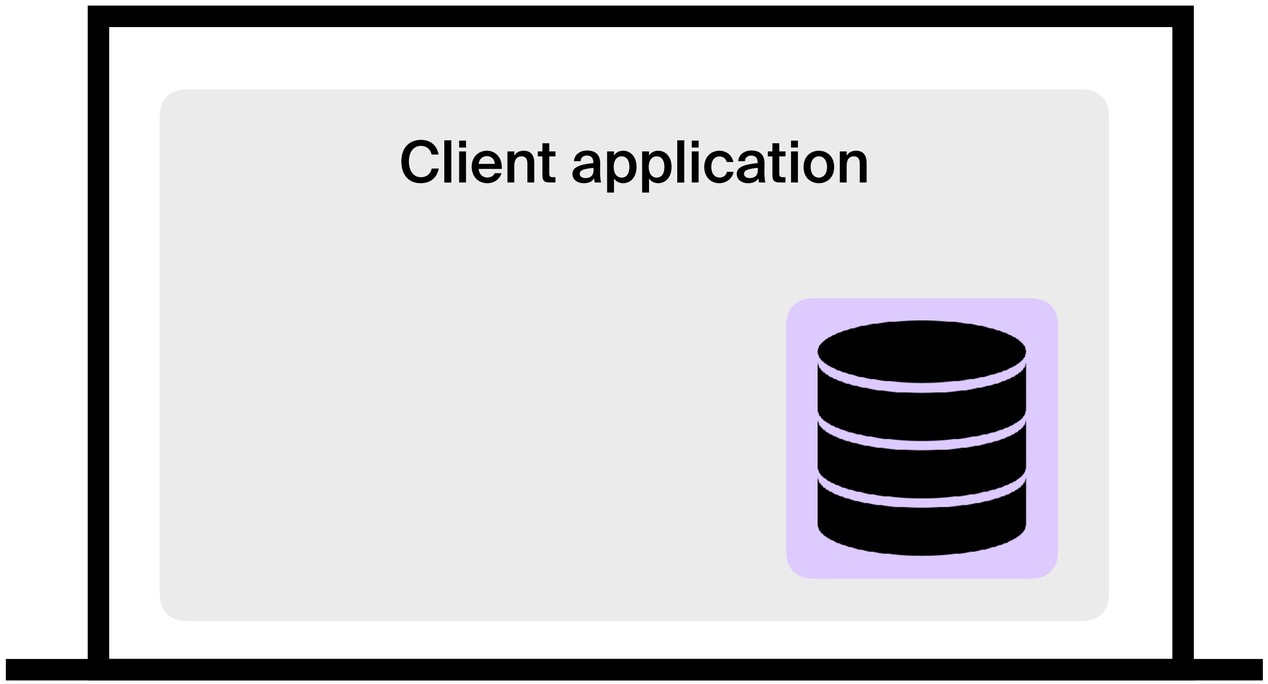
Extensible

In-process architecture

Client-server architecture

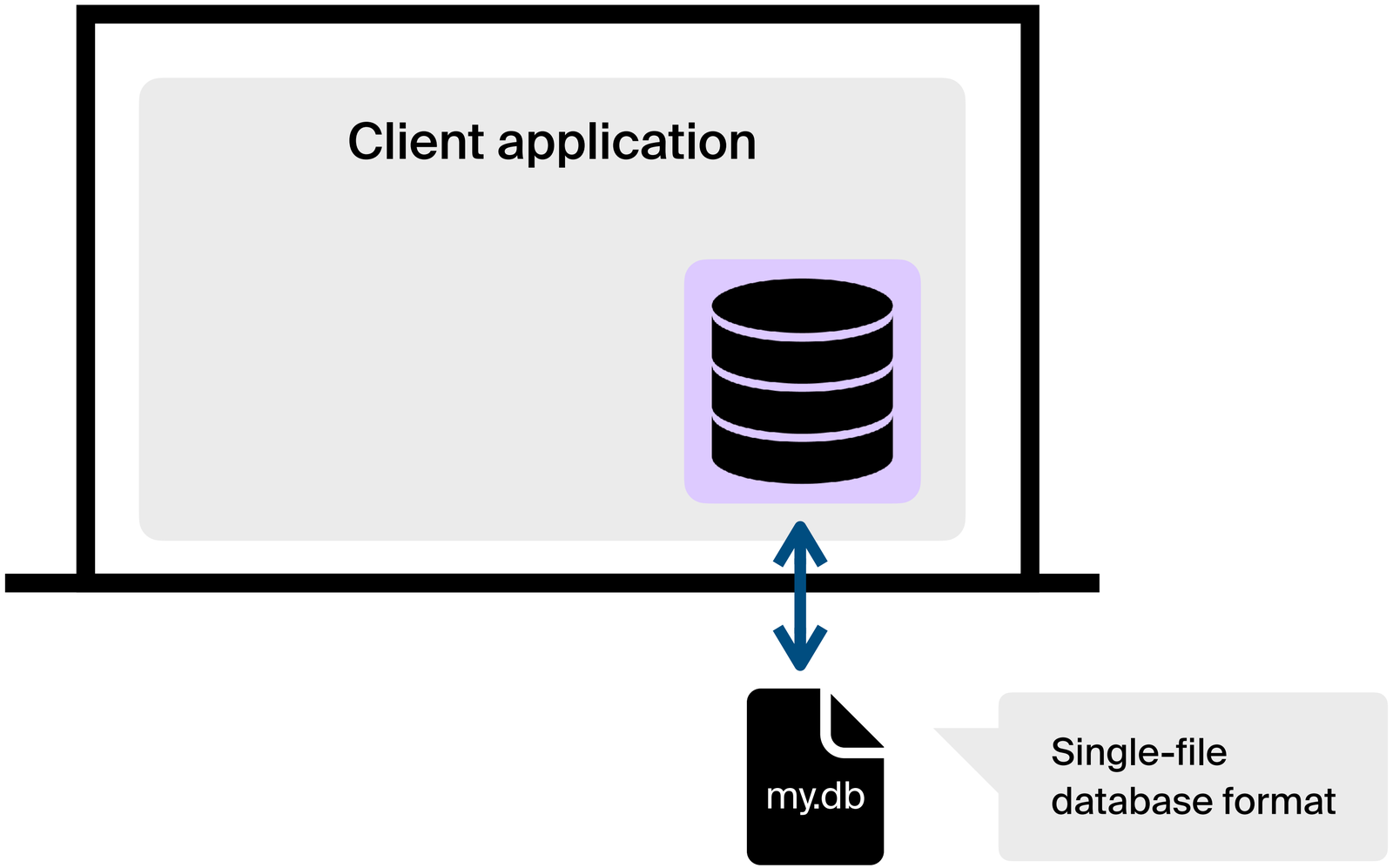


In-process architecture



Truly serverless!

In-process architecture



Database systems

	Transactional	Analytical
In-process	SQLite	DuckDB
Client–server	MySQL, PostgreSQL	snowflake, VERTICA

Internals:
Storage and execution

Storage

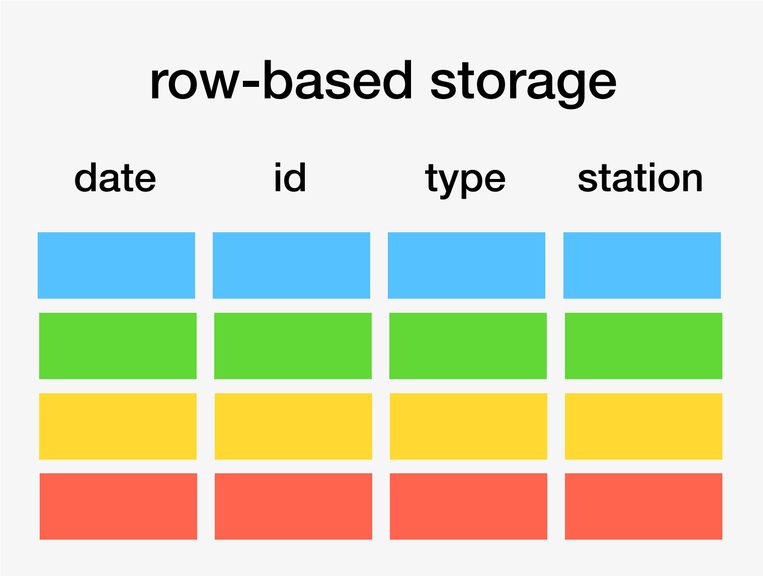
row-based storage

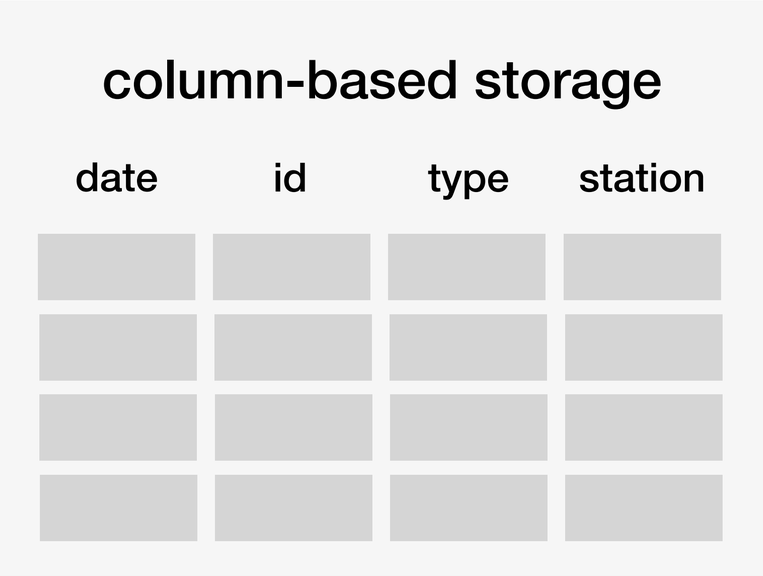
date	id	type	station

column-based storage

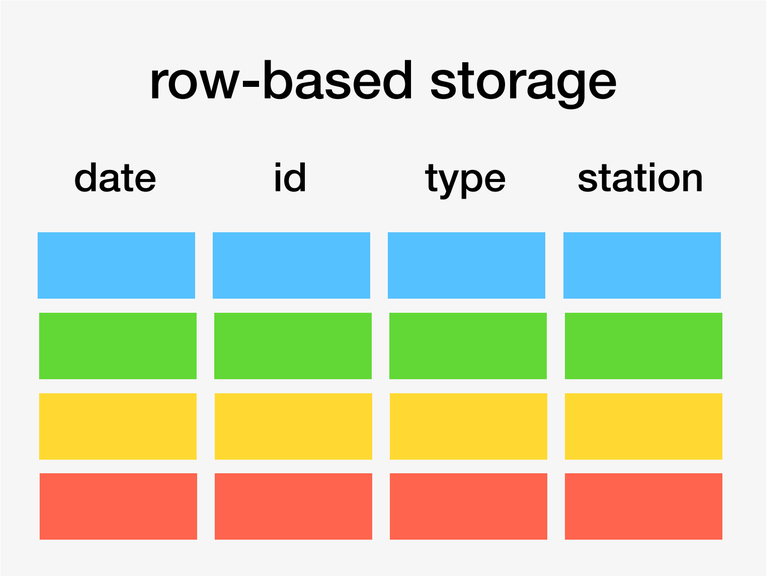
date	id	type	station

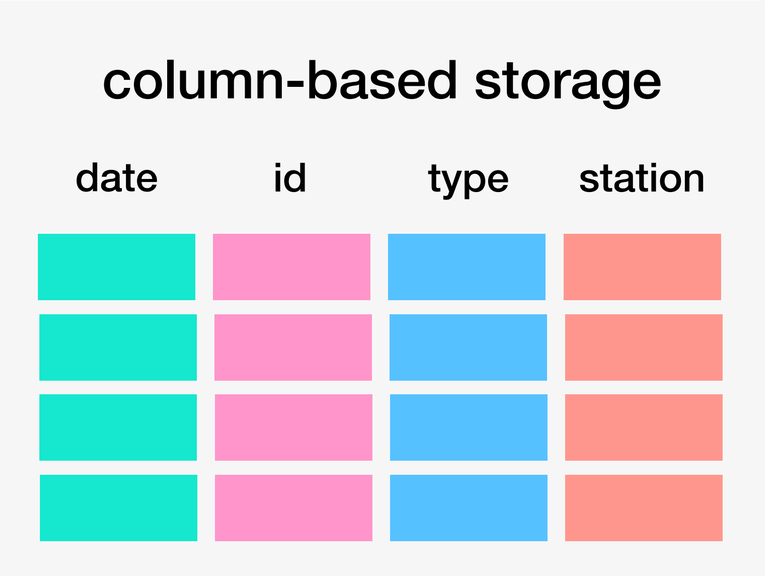
Storage


row-based storage
date
id
type
station


column-based storage
date
id
type
station

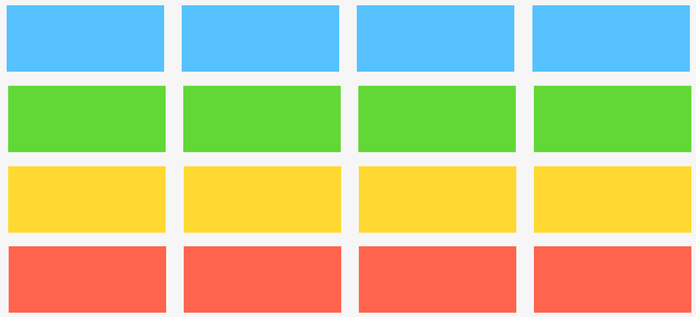
Storage


row-based storage
date
id
type
station

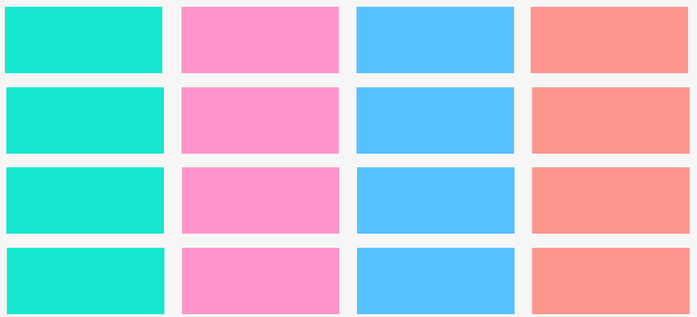

column-based storage
date
id
type
station

Execution

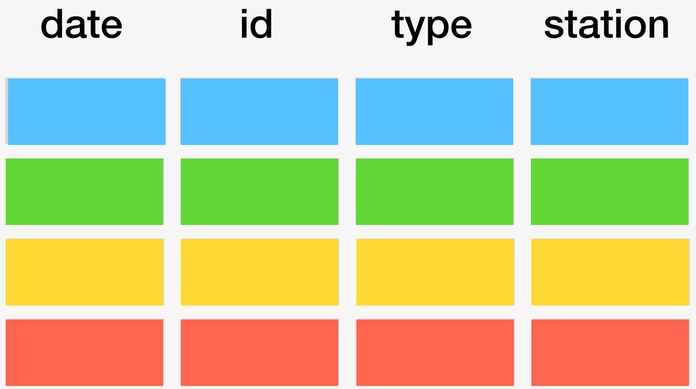
row-based storage



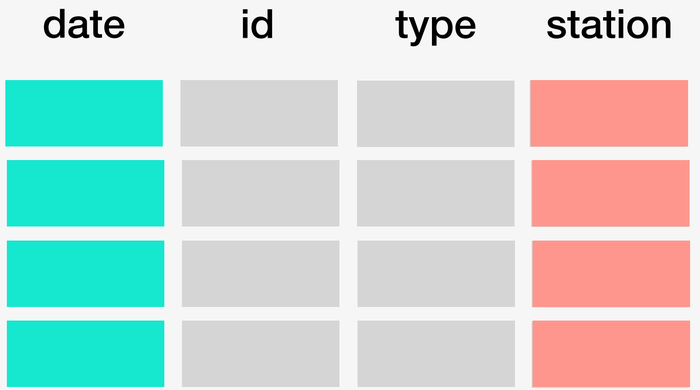
column-based storage



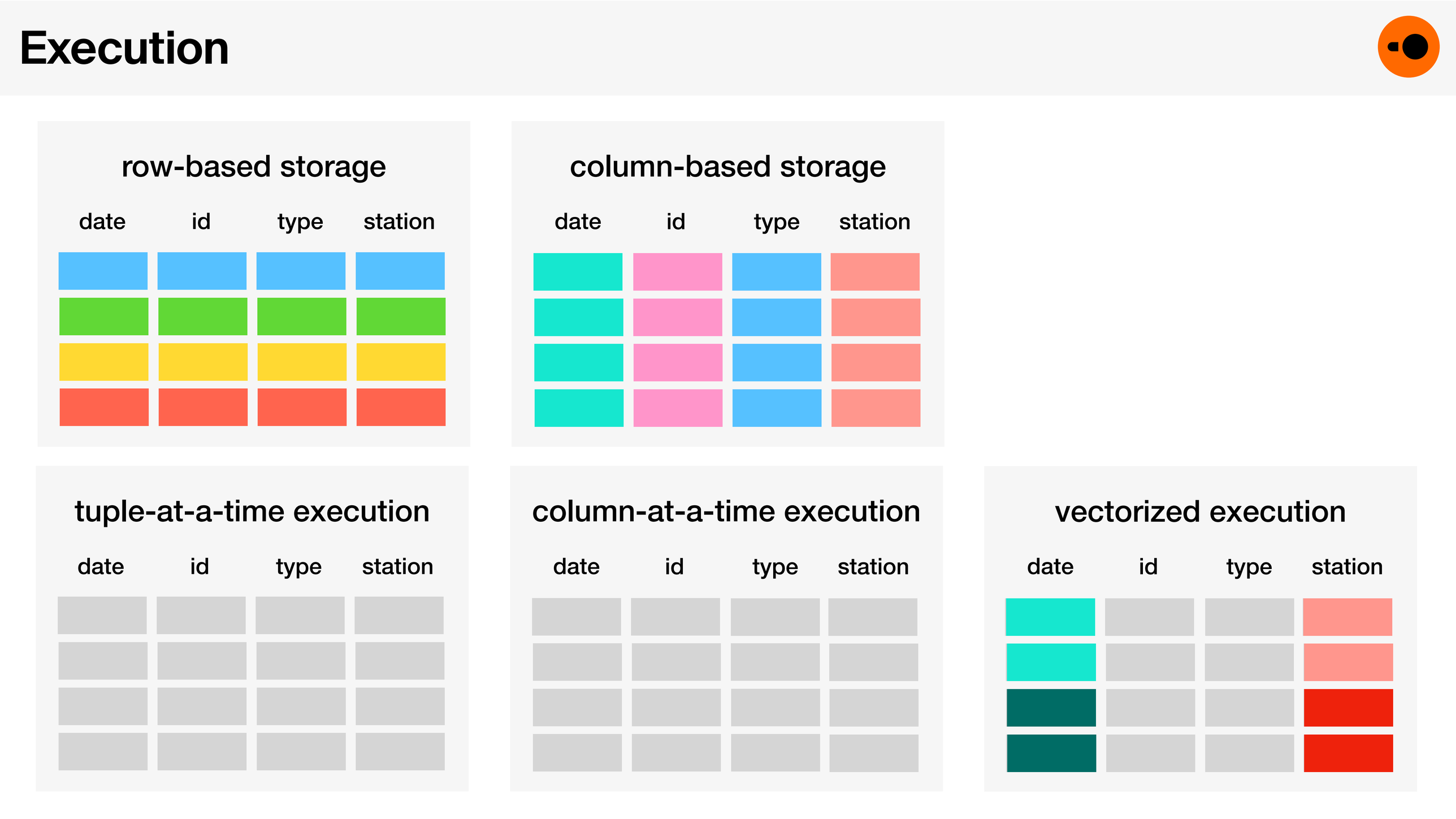
tuple-at-a-time execution



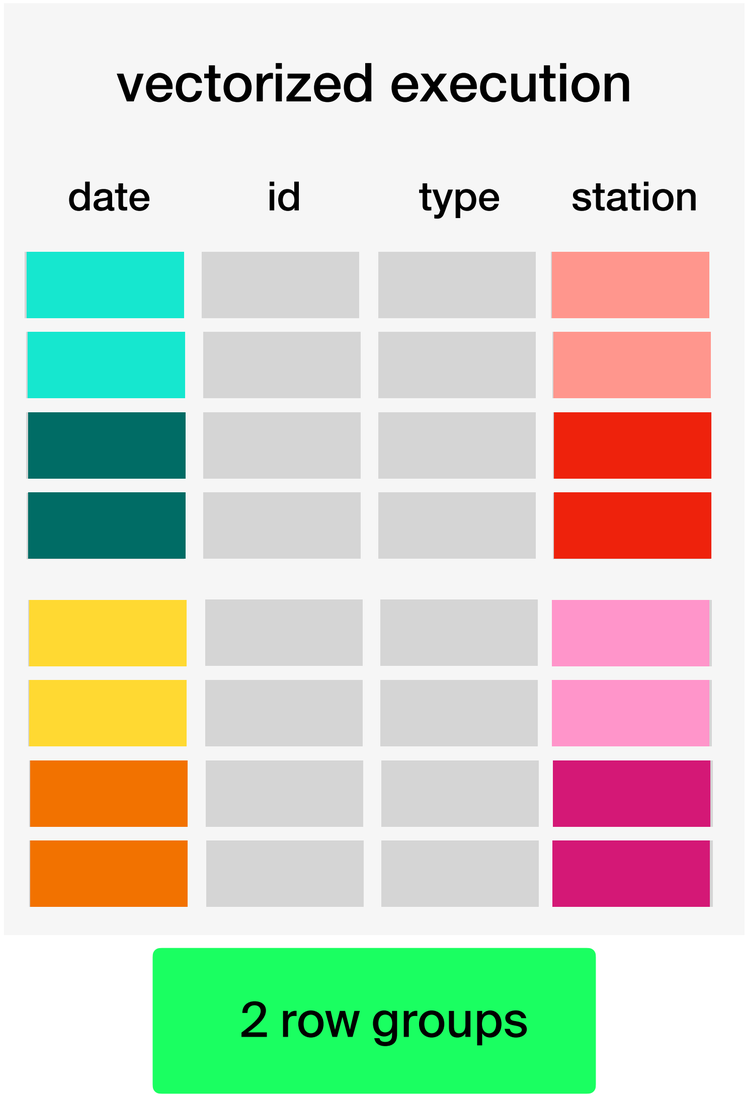
column-at-a-time execution



Execution


row-based storage
date
id
type
station
column-based storage
date
id
type
station
tuple-at-a-time execution
date
id
type
station
column-at-a-time execution
date
id
type
station
vectorized execution
date
id
type
station

Vectorized execution



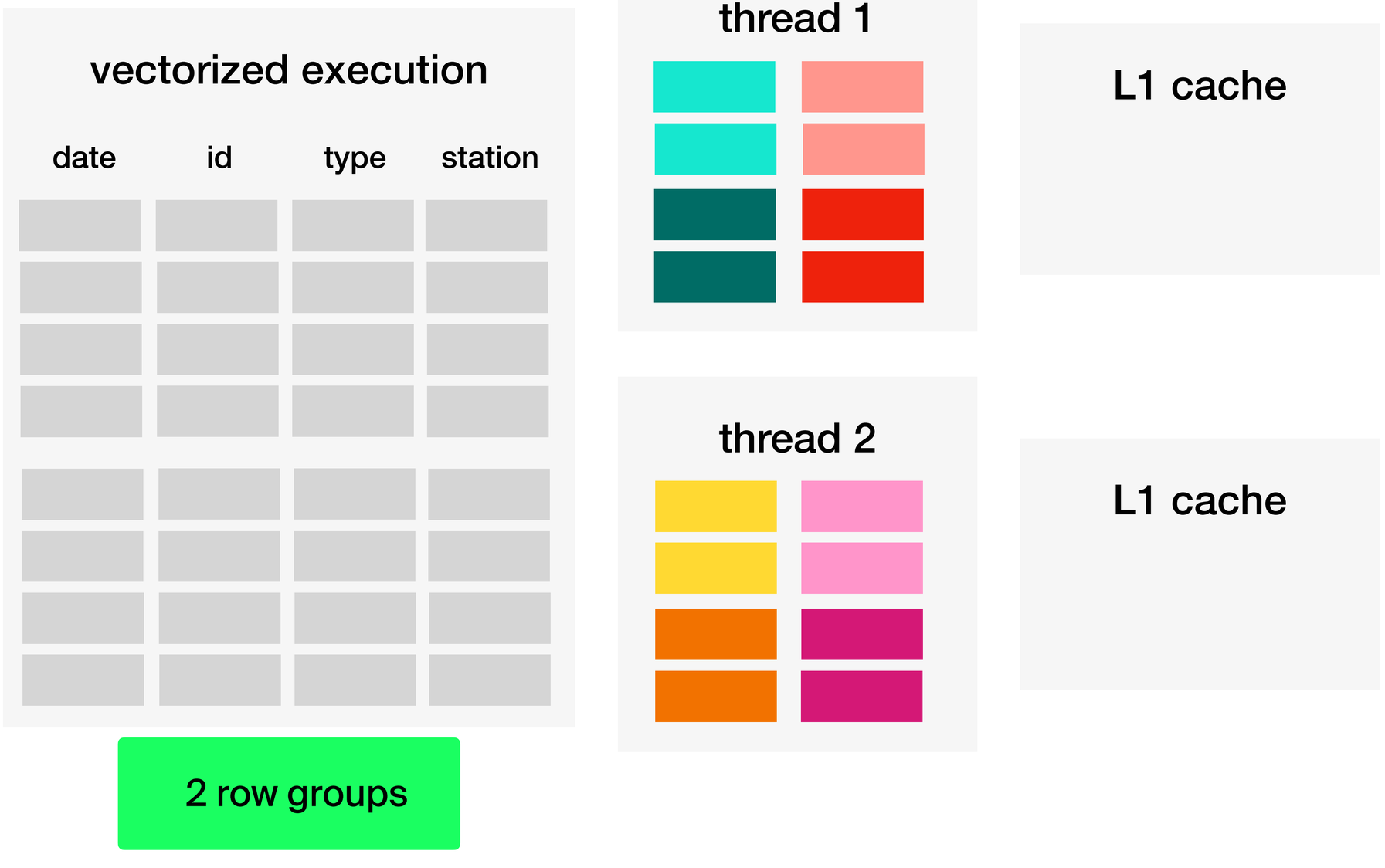
thread 1

L1 cache

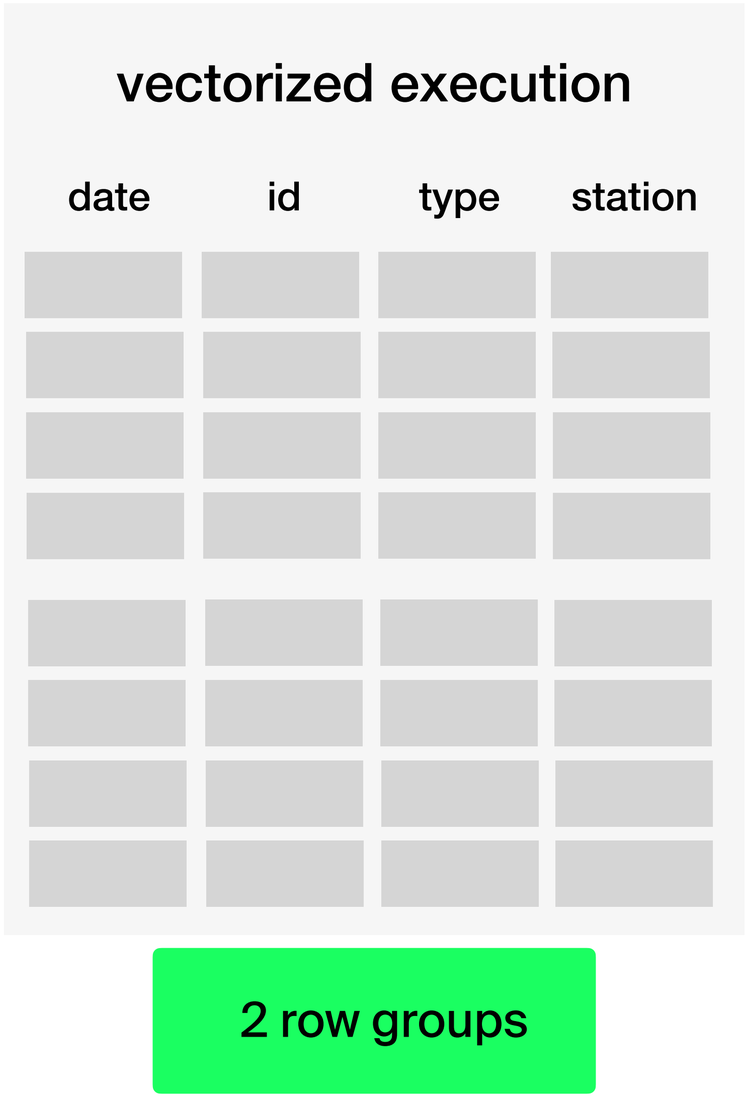
thread 2

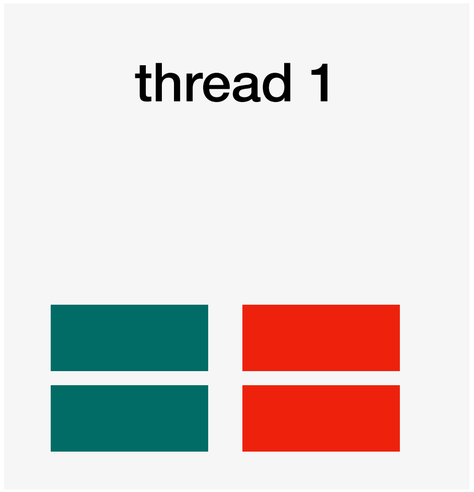
L1 cache

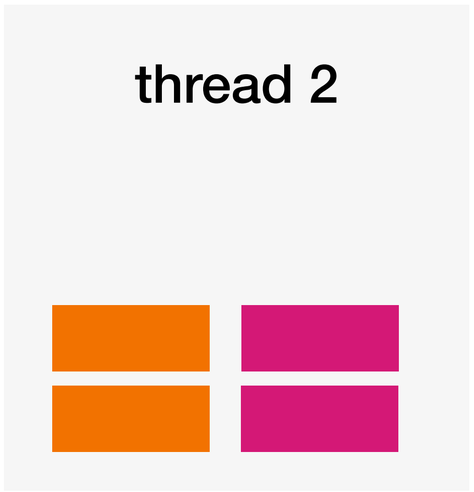
Vectorized execution

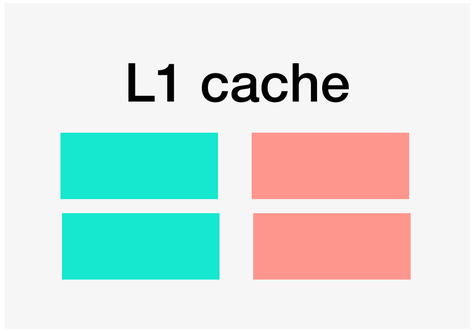


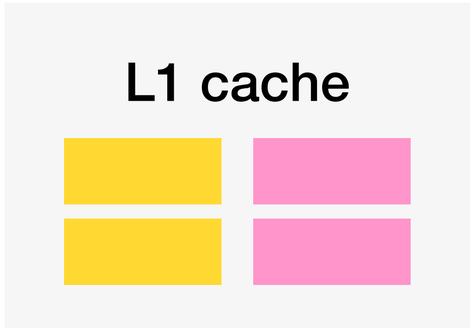
Vectorized execution



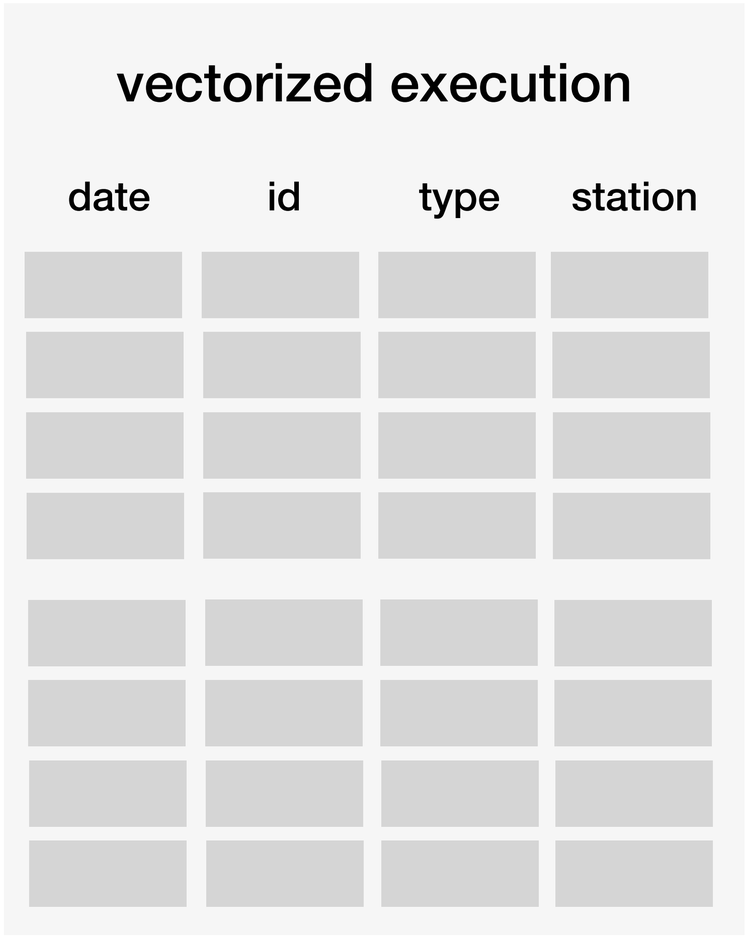








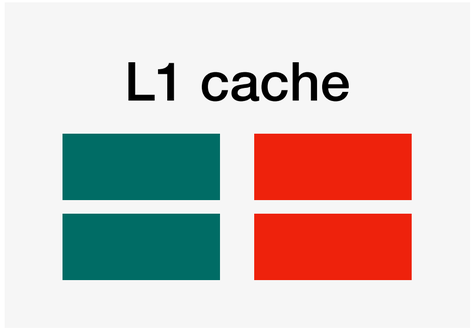
Vectorized execution

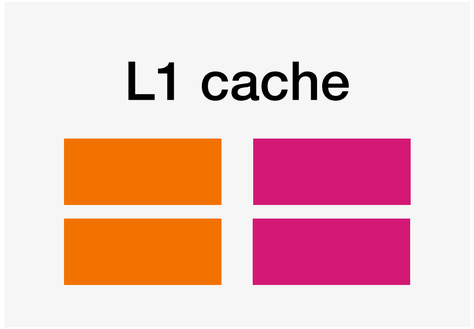


2 row groups

thread 1

thread 2





Vectors fit into
the L1 cache (32–128kB)

Modern compilers
auto-vectorize code
using SIMD instructions

Some relational
operators (e.g., join)
need hash tables

Indexing: Zonemaps

Zonemaps (min-max indexes) are created for **each column** in **each row group**

row group 1

date	id	type	station
May 30	1	Intercity	Ams C
May 30	2	Sprinter	Utrecht
May 31	3	ICE Intl	Ams C
May 31	4	Intercity	Schiphol

min	May 30	1	ICE Intl	Ams C
max	May 31	4	Sprinter	Utrecht

row group 2

date	id	type	station
June 1	5	Sprinter	Schiphol
June 1	6	Sprinter	Utrecht
June 1	7	Intercity	Utrecht
June 2	8	Intercity	Ams C

min	June 1	5	Intercity	Ams C
max	June 2	8	Sprinter	Utrecht

Portability

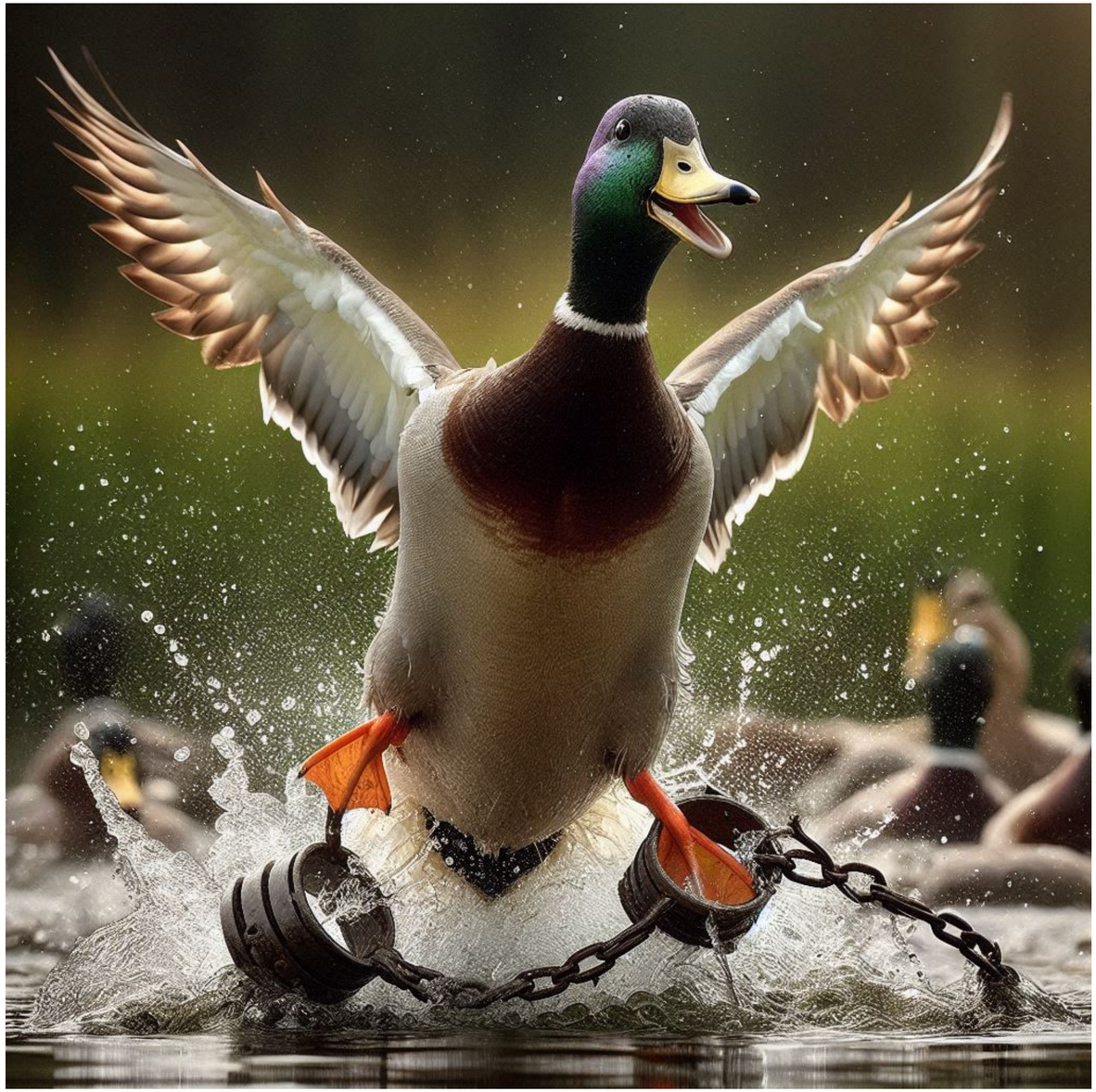
Portability

DuckDB is written in C++11

No external dependencies

Instead: a few inlined dependencies

→ DuckDB is **very portable**



DuckDB clients

```
pip install duckdb
```

```
npm install duckdb
```

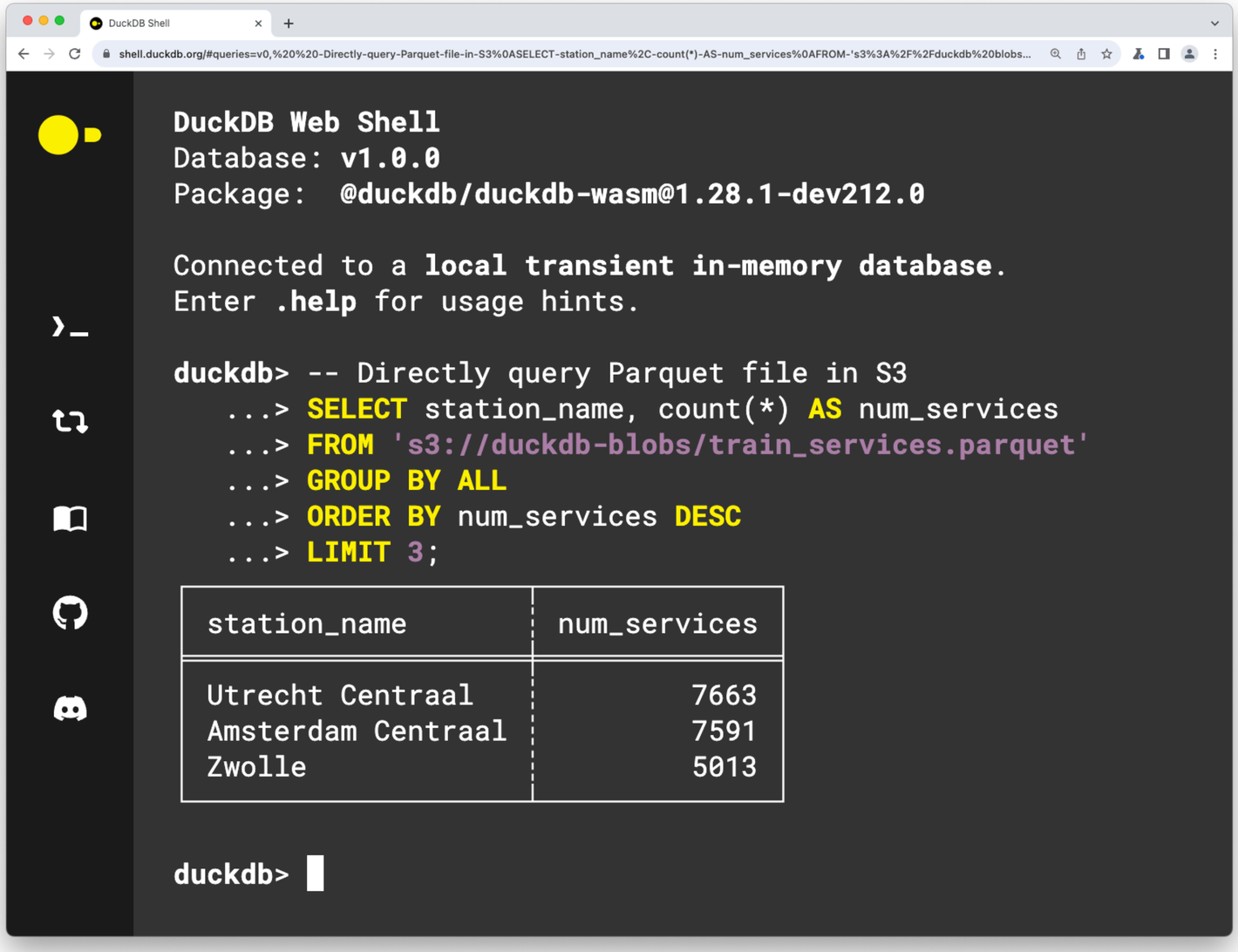
```
install.packages("duckdb")
```

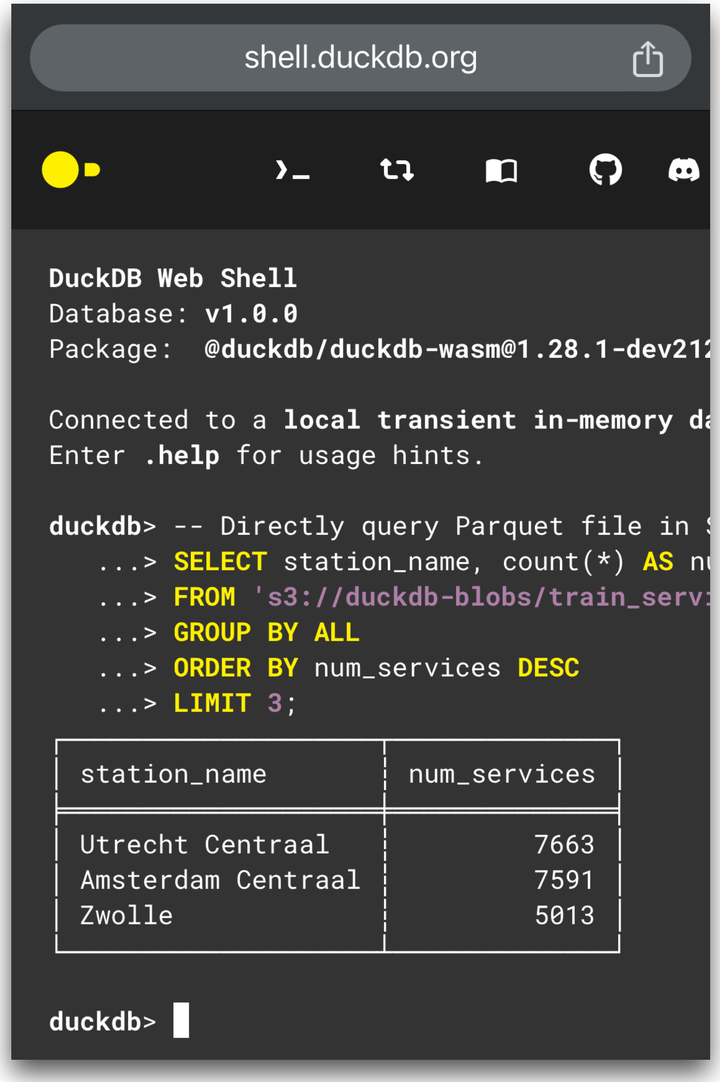
```
org.duckdb:duckdb_jdbc
```

```
brew install duckdb
```

```
cargo add duckdb
```

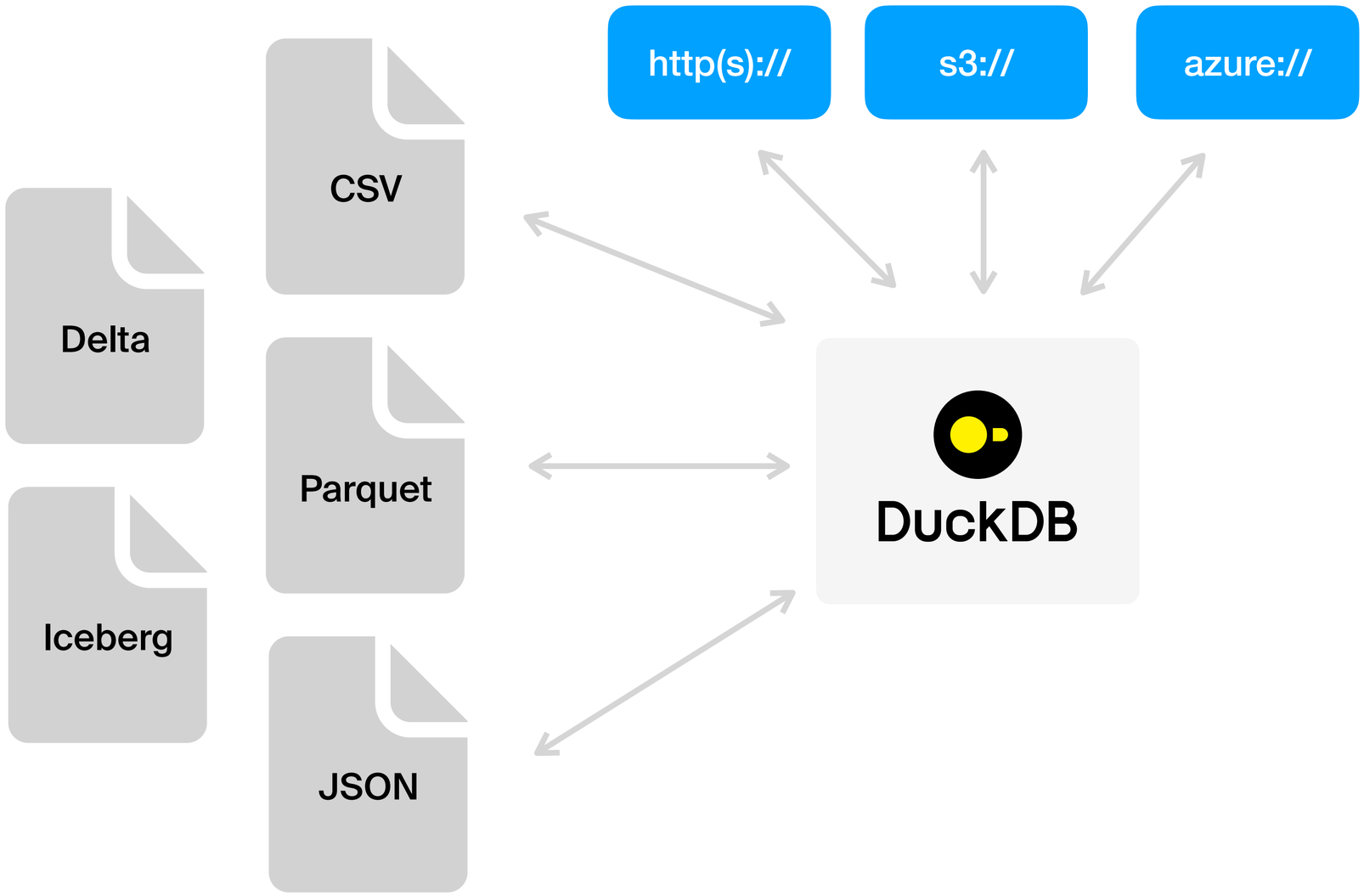
WebAssembly: DuckDB in the browser





Ergonomic

Supported formats and protocols



Partial reading on Parquet

Parquet files also have zonemaps

These can be used for HTTP range requests (https://, s3://, etc.)

row group 1

date	id	type	station
May 30	1	Intercity	Ams C
May 30	2	Sprinter	Utrecht
May 31	3	ICE Intl	Ams C
May 31	4	Intercity	Schiphol

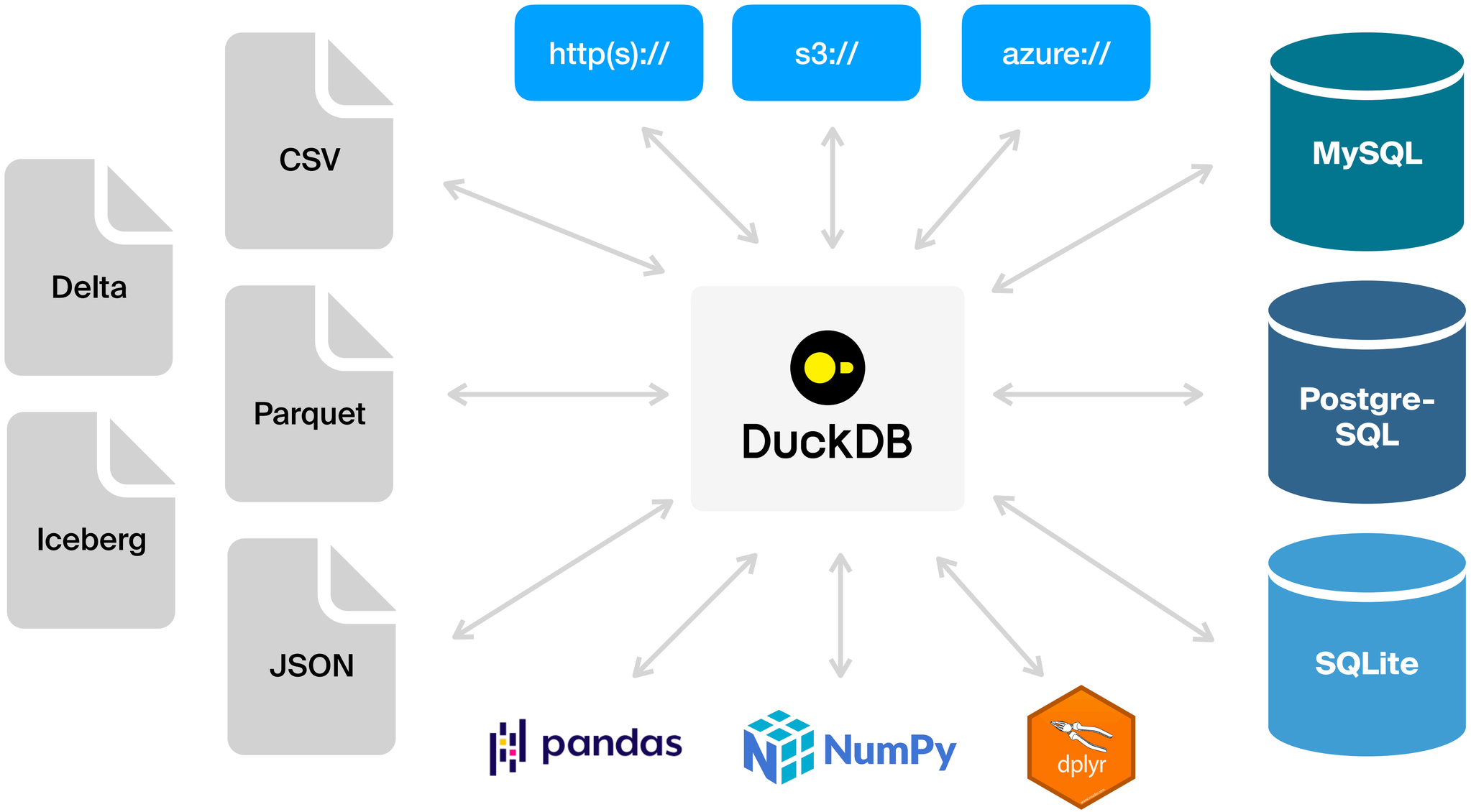
min	May 30	1	ICE Intl	Ams C
max	May 31	4	Sprinter	Utrecht

row group 2

date	id	type	station
June 1	5	Sprinter	Schiphol
June 1	6	Sprinter	Utrecht
June 1	7	Intercity	Utrecht
June 2	8	Intercity	Ams C

min	June 1	5	Intercity	Ams C
max	June 2	8	Sprinter	Utrecht

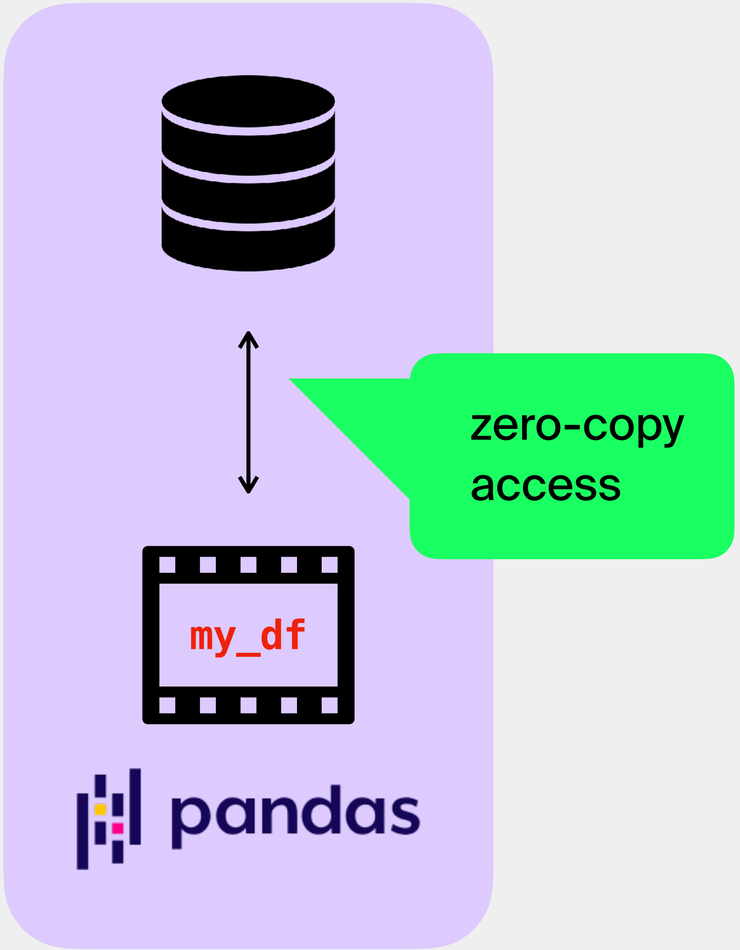
Supported formats and protocols



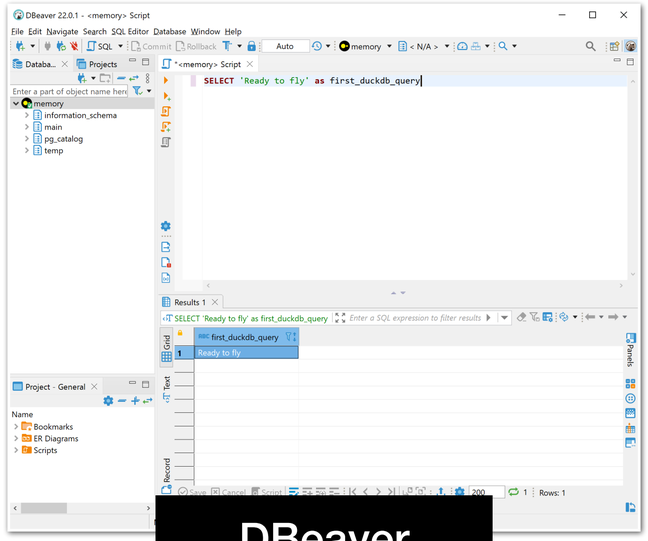
Pandas integration

```
import duckdb
import pandas as pd

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

r1 = duckdb.sql("SELECT avg(a) AS s FROM my_df")

#
#   ┌─────────┐
#   │    s    │
#   │ double  │
#   ├─────────┤
#   │    42.5 │
#   └─────────┘
#

r2 = r1.df()

#       s
# 0  42.5
```

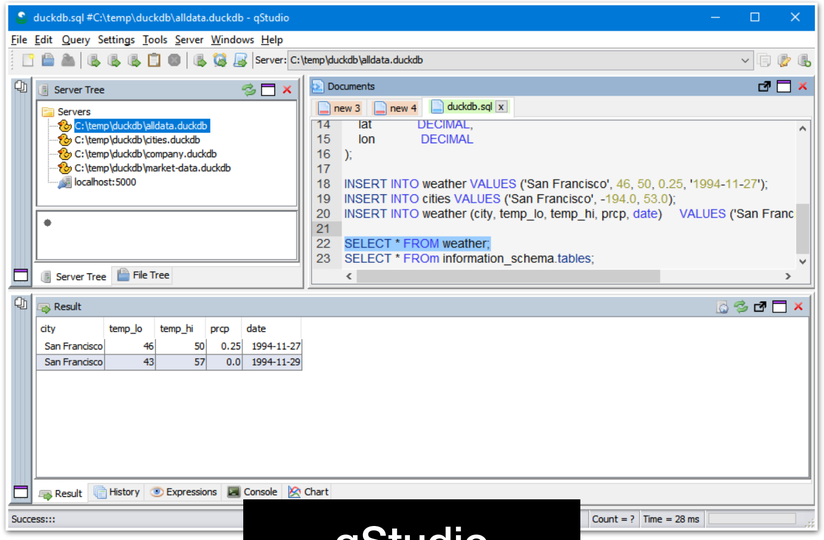
replacement scan



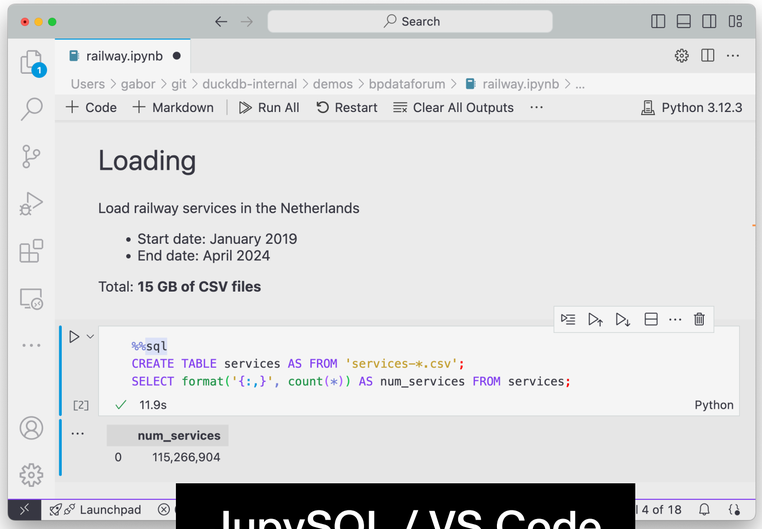
IDEs for DuckDB



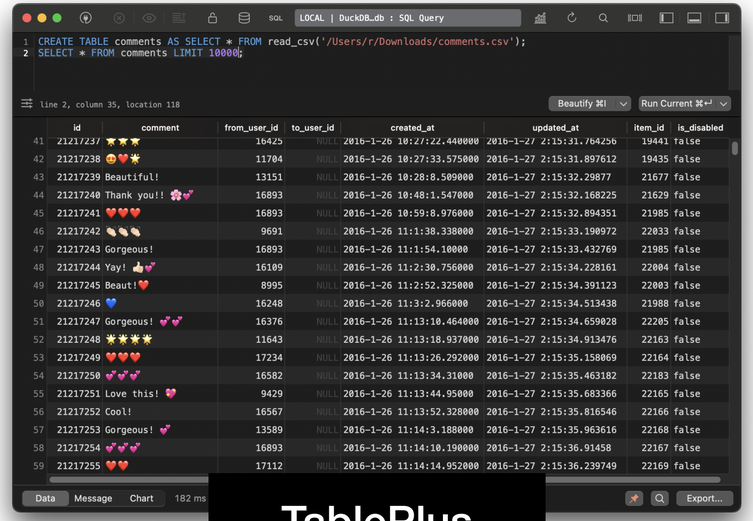
DBeaver



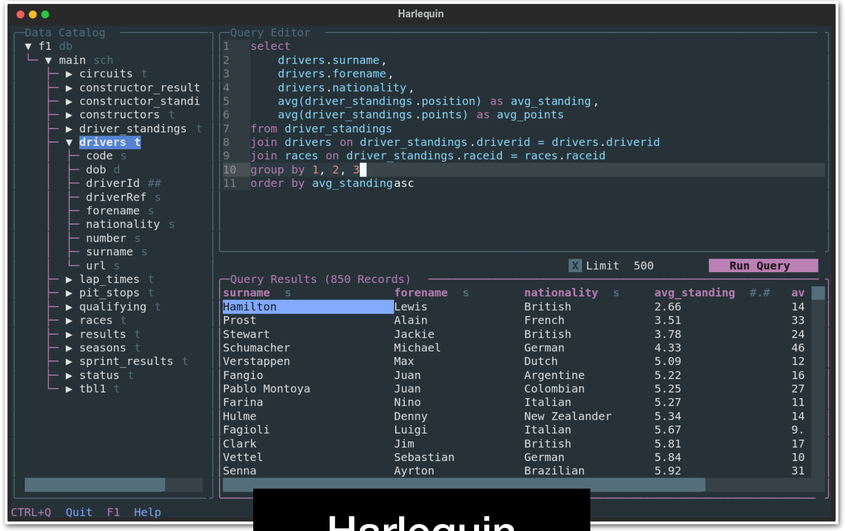
qStudio



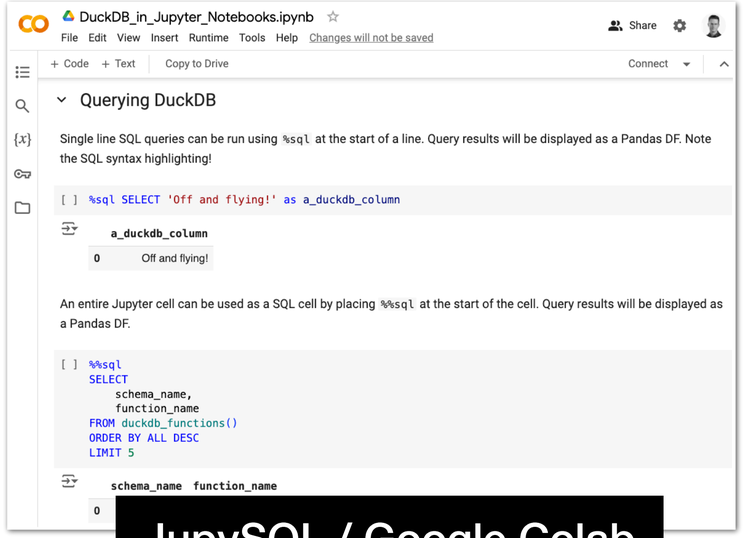
JupySQL / VS Code



TablePlus



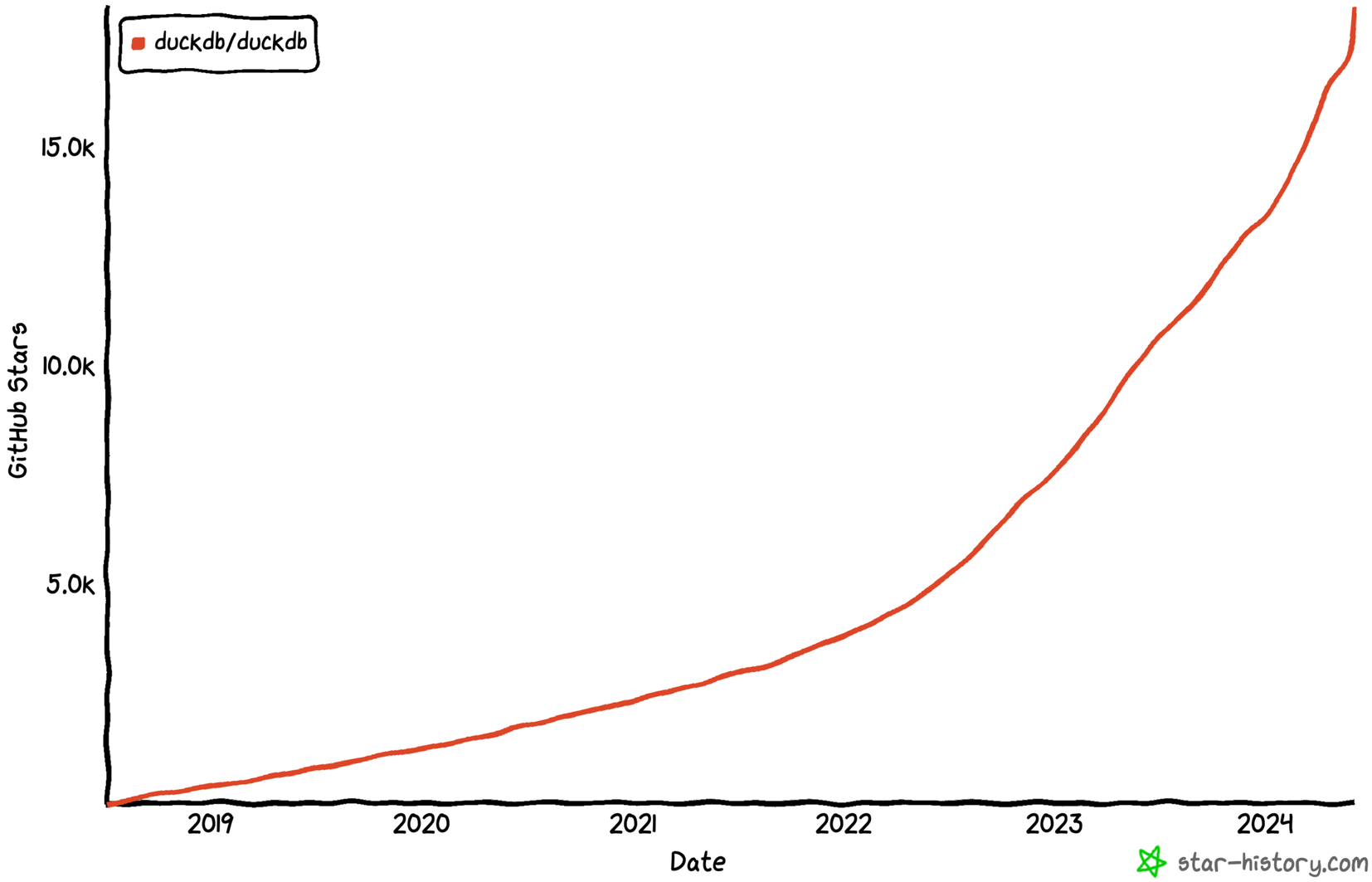
Harlequin



JupySQL / Google Colab

State of DuckDB

Adoption

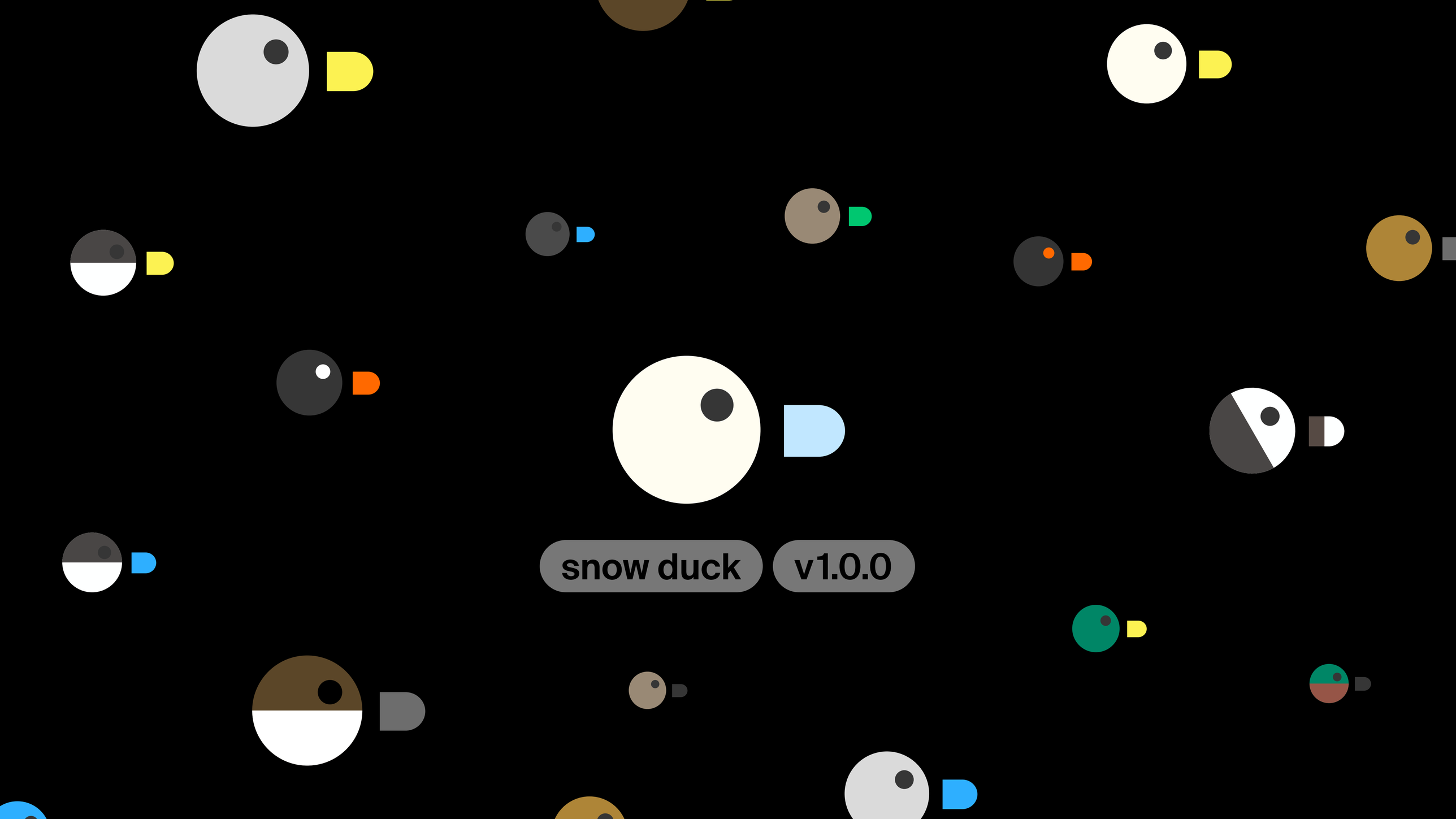


19k GitHub stars

~30k followers on LinkedIn/Twitter

1M visits per month

4M+ PyPI installs/month


snow duck
v1.0.0

DuckDB v1.0.0: Snow Duck

2018-07-13: Initial commit

2024-06-03: v1.0.0 release

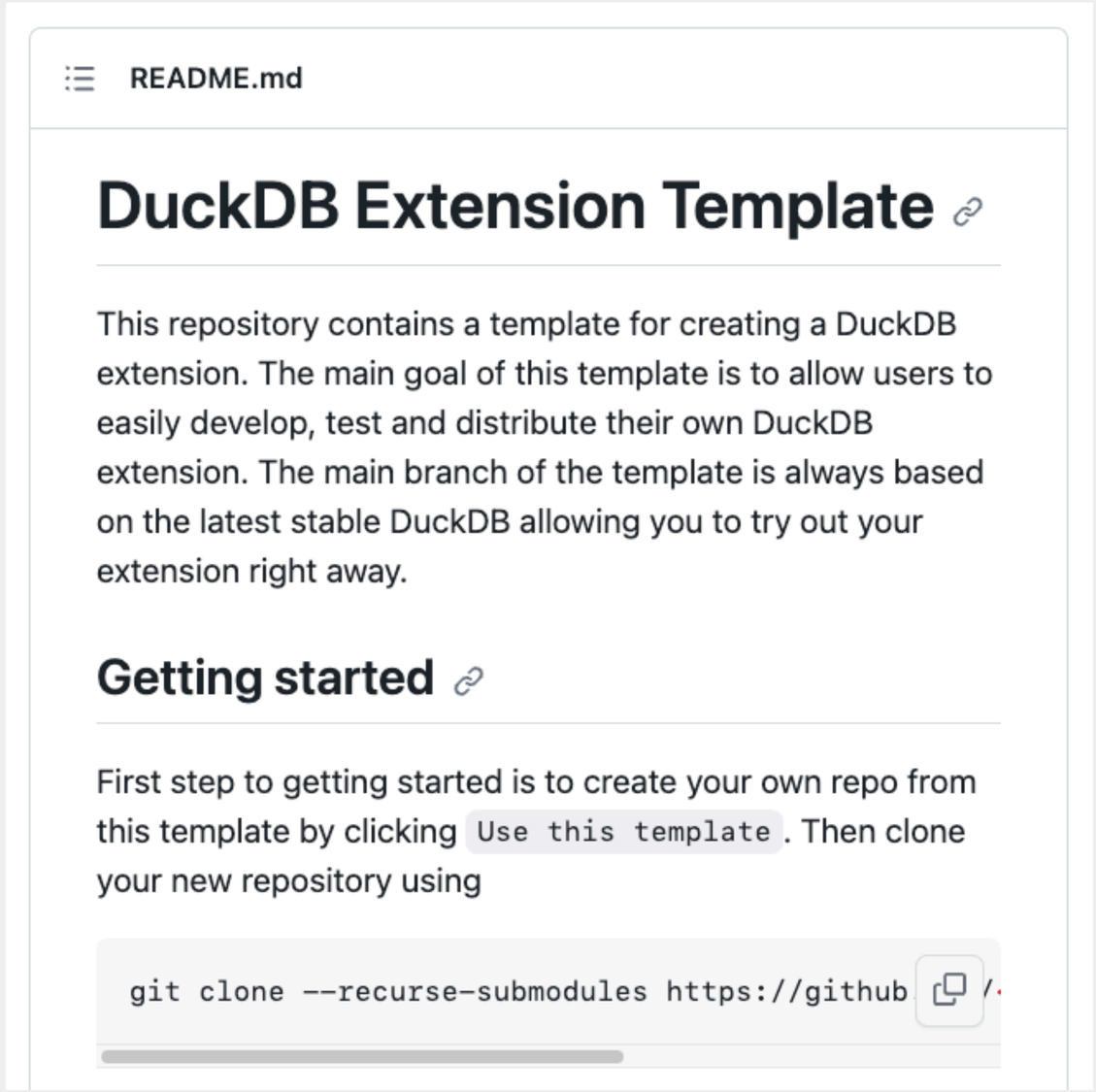
Stable storage: DuckDB's file format is backwards-compatible

Extensions

Extensions

- Powerful extension mechanism:
  - new types and functions
  - data formats
  - operators
  - SQL syntax
  - memory allocator
  - compression

- We are dogfooding – many key features are DuckDB extensions:
  - httpfs
  - JSON
  - Parquet


README.md

DuckDB Extension Template

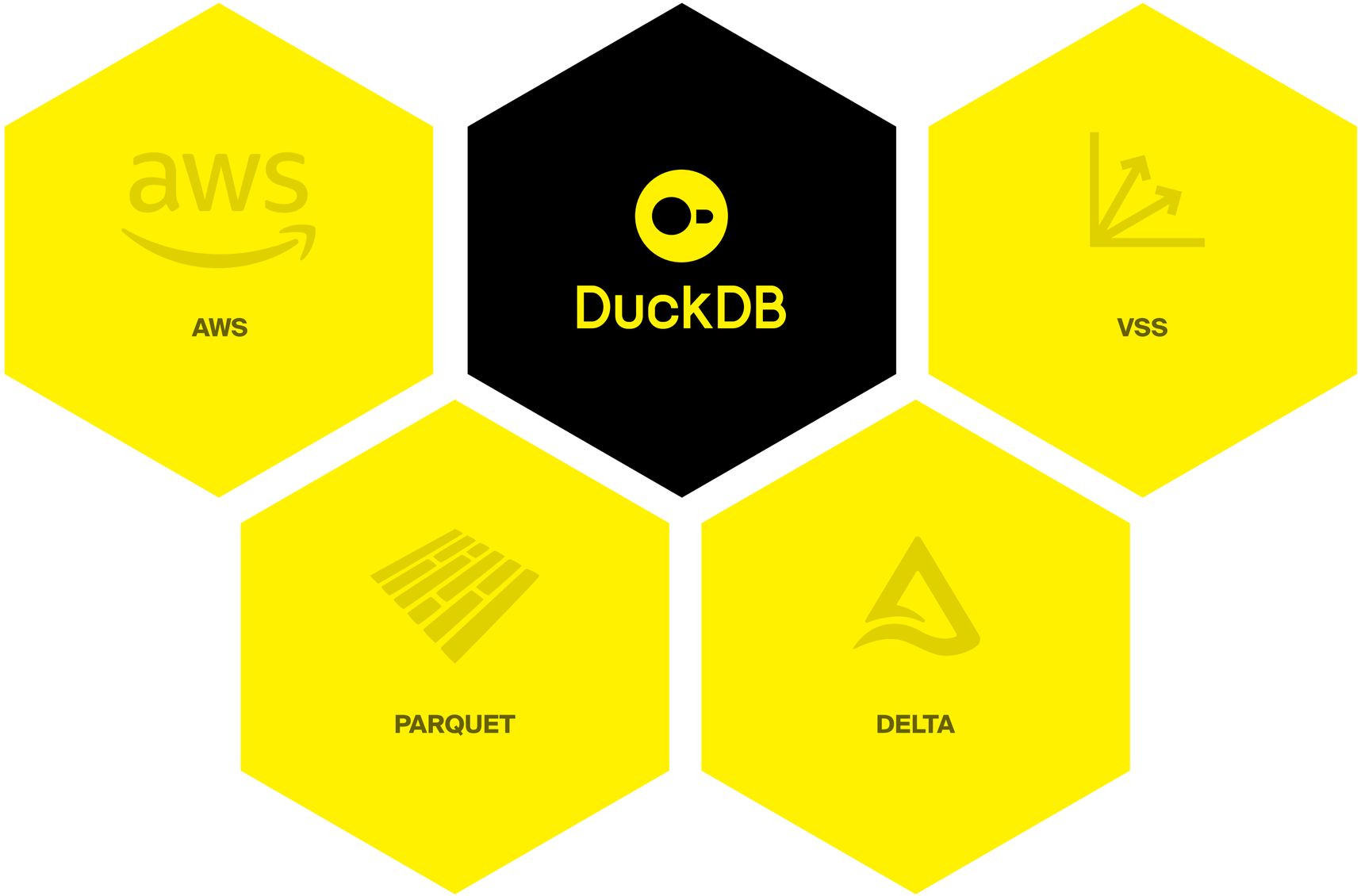
This repository contains a template for creating a DuckDB extension. The main goal of this template is to allow users to easily develop, test and distribute their own DuckDB extension. The main branch of the template is always based on the latest stable DuckDB allowing you to try out your extension right away.

Getting started

First step to getting started is to create your own repo from this template by clicking `Use this template`. Then clone your new repository using

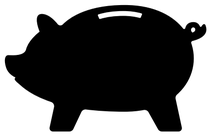
```
git clone --recurse-submodules https://github
```

Extension ecosystem



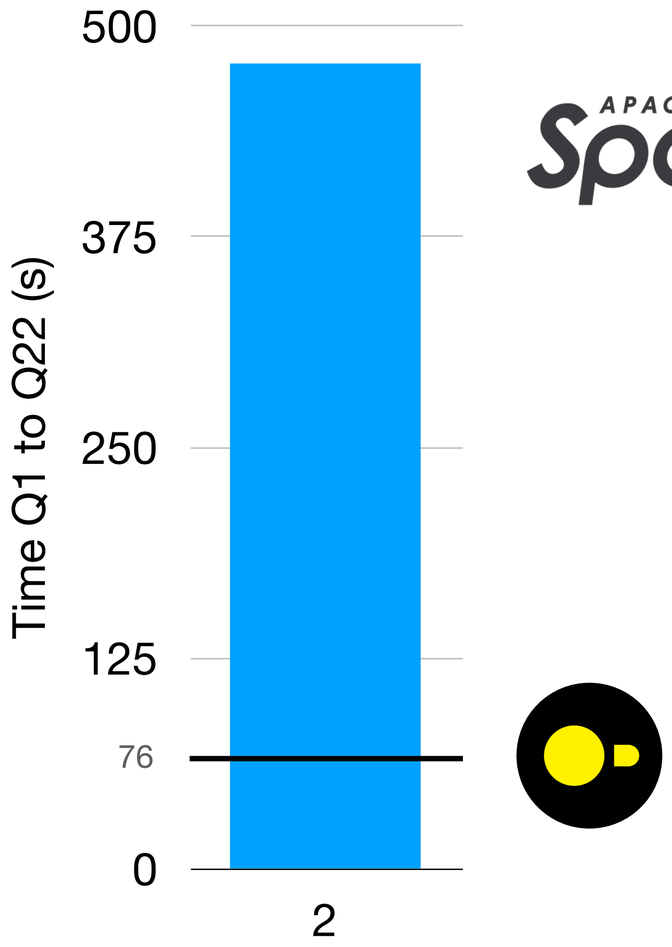
Use cases

Reducing costs

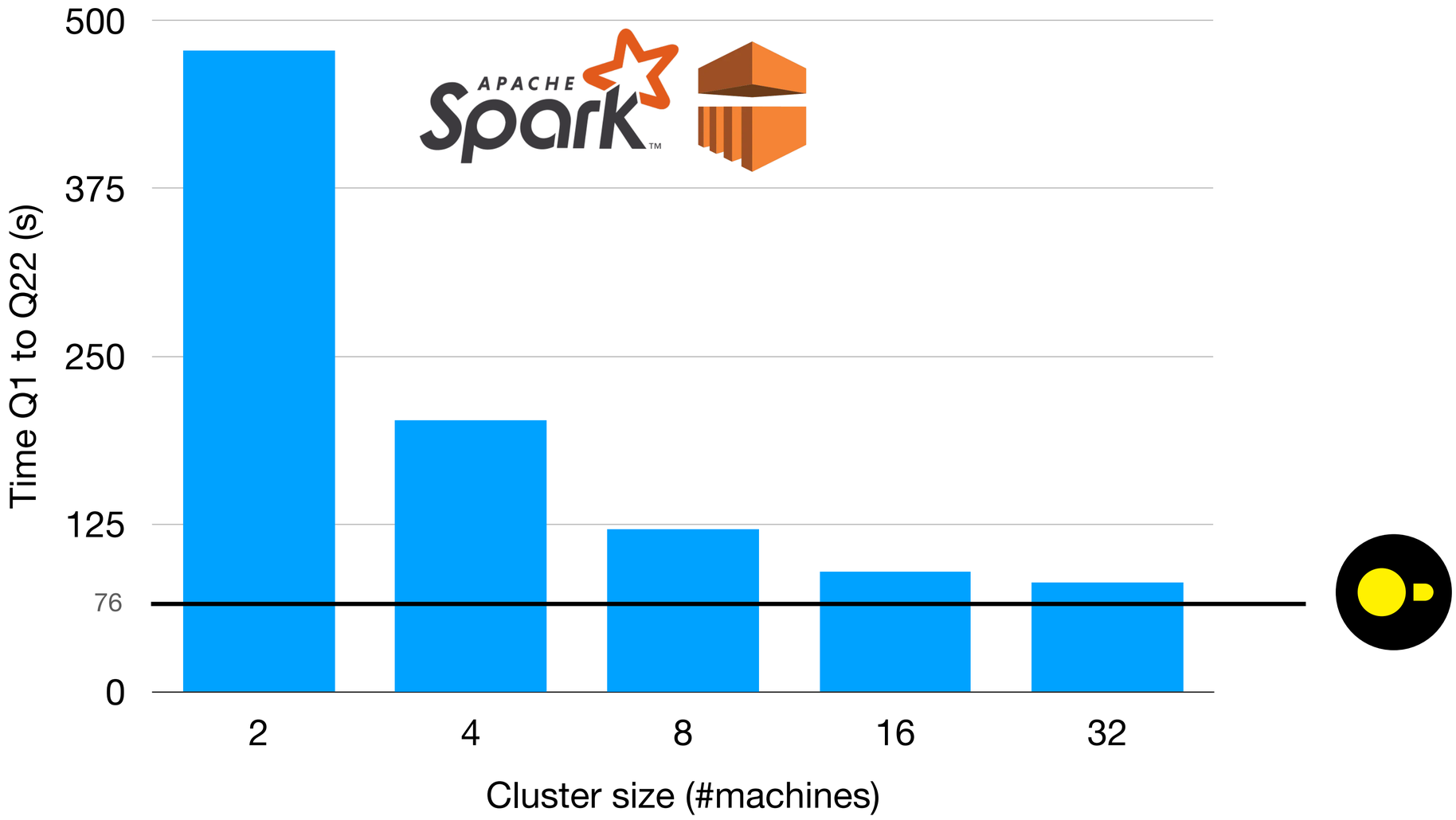


- Local data processing
- Avoid egress fees
- Replace proprietary systems: DuckDB is free
- Replace distributed systems: DuckDB is more efficient

TPC-H experiments on Parquet files

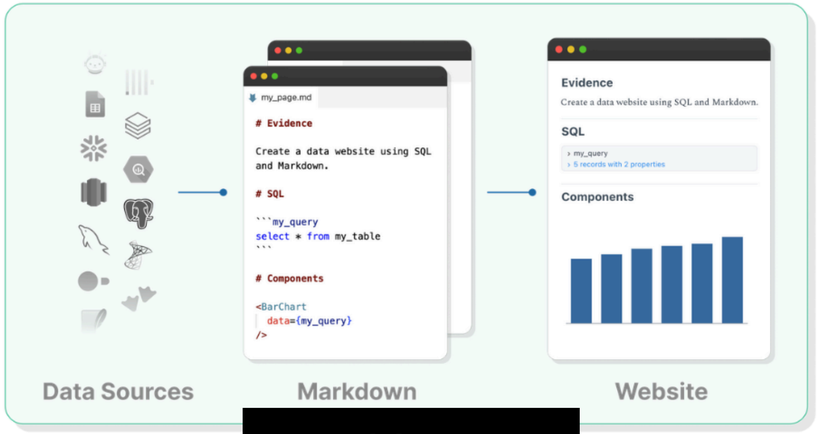


TPC-H experiments on Parquet files

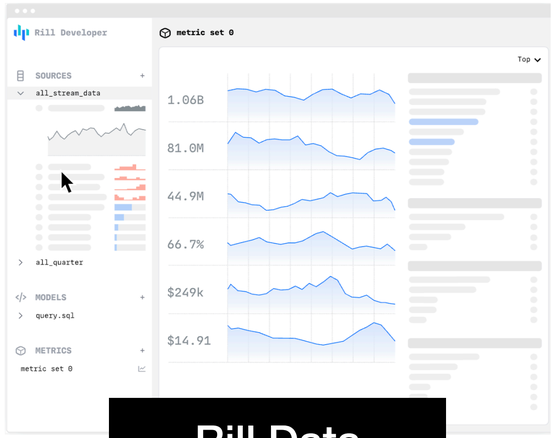


Last mile analytics

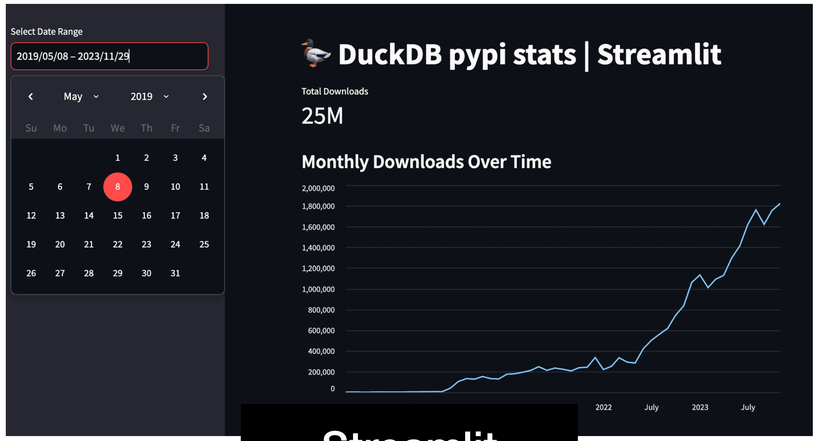
- E.g., 100TB log file processed by Spark to 50GB summary
- The lasts steps can be done interactively using DuckDB
- Build 120 FPS dashboards!



evidence

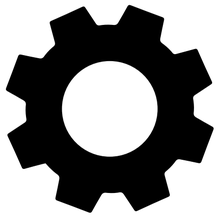


Rill Data



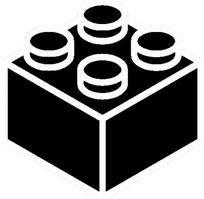
Streamlit

Local prototyping



- Experimenting with SQL queries
- In a local environment
- ...with quick response time
- ...for free
- Using a popular SQL dialect helps!

Building block



- A single data processing step in a larger system
- Parquet → CSV conversion, loading data, etc.
- Not a database use case!

Feature/fix operation stream loading #165

Merged **szarnyasg** merged 19 commits into `main` from `feature/fix-operation-stream-loading`

Conversation 0 · Commits 19 · Checks 0 · Files changed 102

GLaDAP commented on Jun 23, 2022 • edited · Member

This PR contains the following:

- QueryEventStreams are merged into 1 class
- Operation streams are loaded using DuckDB
- Queries moved to their own namespace

GLaDAP added 19 commits last year

`Move queries to separate namespace` 5bf4581

`Add DuckDb for CSV parsing` 3c6f682

Reviewers: szarnyasg

Assignees: No one assigned

Labels: None yet

Projects: None yet

+1,634 −5,270

Output validation using matching in SQL #217

Merged **szarnyasg** merged 10 commits into `main` from `output-validation-using-matching`

Conversation 0 · Commits 10 · Checks 1 · Files changed 25

szarnyasg commented on Aug 24, 2022 • edited · Member

Will fix #205.

We can use the DuckDB appender to populate the tables.

Current validation scripts are in:

- https://github.com/ldbc/ldbc_graphalytics/tree/master/graphalytics-core/src/main/java/science/atlarge/graphalytics/validation
- https://github.com/ldbc/ldbc_graphalytics/tree/master/graphalytics-core/src/main/java/science/atlarge/graphalytics/validation/rule

A lot of time is spent parsing the results back from CSVs to Java data structures, this could also be improved by using DuckDB's

Reviewers: No reviews

Assignees: No one assigned

Labels: None yet

Projects: None yet

Milestone

+338 −457

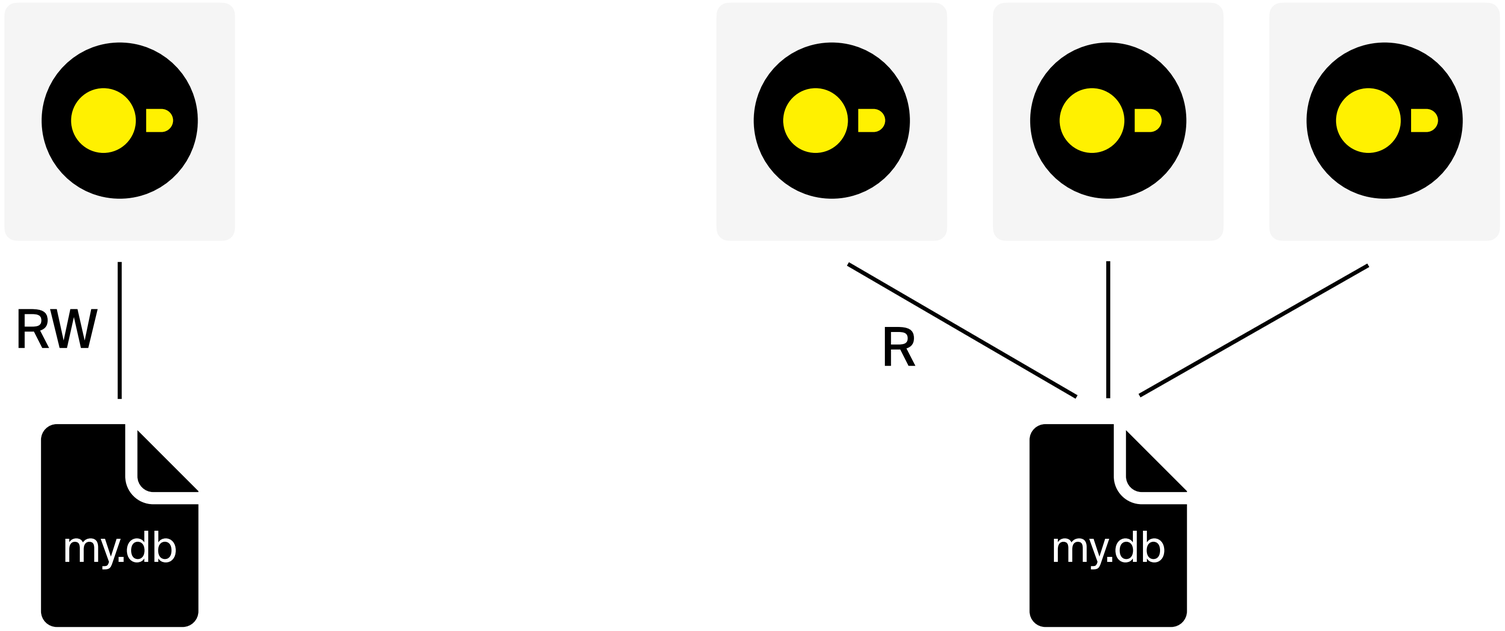
Education

- Easy to install
- Open-source
- Uses *de facto* standards (PostgreSQL dialect, Parquet, ...)
- Does not need configuration
- Does not need a DBA to run it

Limitations

Concurrency control

- ACID compliance via multi-version concurrency control (MVCC)
- Recovery using a write-ahead log (WAL)
- But: Not a good fit for write-heavy workloads



No distributed execution

DuckDB only supports **single-node** execution

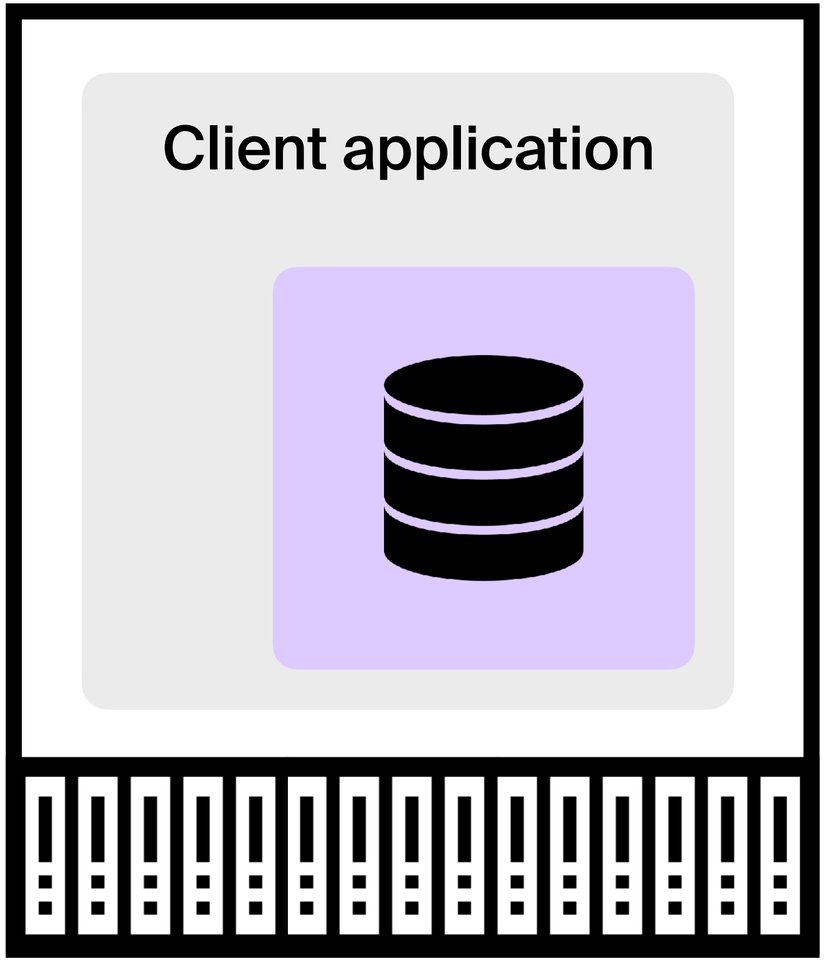
DuckDB can **scale up:**

- r6id.32xlarge: 1TB RAM, $10/h
- x1e.32xlarge: 4TB RAM, $28/h
- u7in-32tb.224xlarge, 36TB, $408/h :)

Allows scaling for TBs of data

Typical setup:

- Store the data in S3 or S3 Express One
- Run short bursts of workloads



Business model

DuckDB Labs

A company of ~15, based in Amsterdam

Operations funded by revenue (no VC)

Consulting and support for DuckDB

DuckDB Labs

MotherDuck databricks HYDRA Fivetran Rill Spice AI Hugging Face dltHub

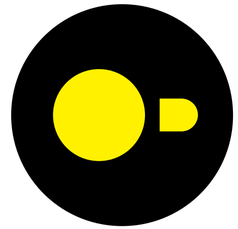
VOLTRON DATA TigerEye tecton evidence STOIC HEX

DuckDB Foundation

A non-profit organization

Owns the intellectual property of the project

- 12 silver supporters
- 3 gold supporters



DuckDB

MotherDuck

posit™

VOLTRON DATA

MotherDuck

Venture capital-funded, HQ in Seattle

MotherDuck

Builds a cloud data warehouse using DuckDB

Reached *General Availability* yesterday!

CLOUD DATABASE STORAGE

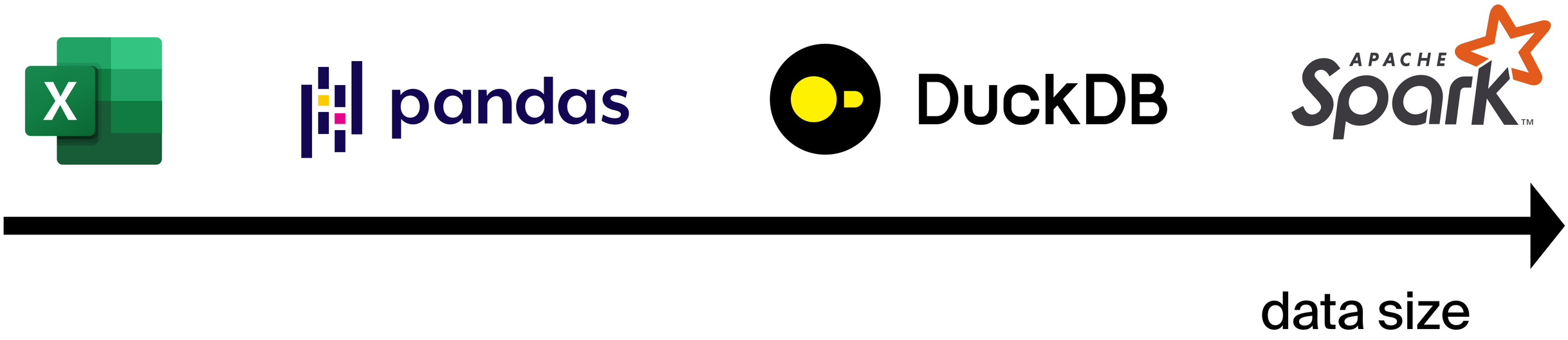
SIMPLIFIED DATABASE SHARING

HYBRID QUERY EXECUTION

STRONG DUCKDB ECOSYSTEM

Summary

Spectrum of data processing systems



DuckDB is an old-school analytical database for modern hardware

Open-source

MIT license

Portable

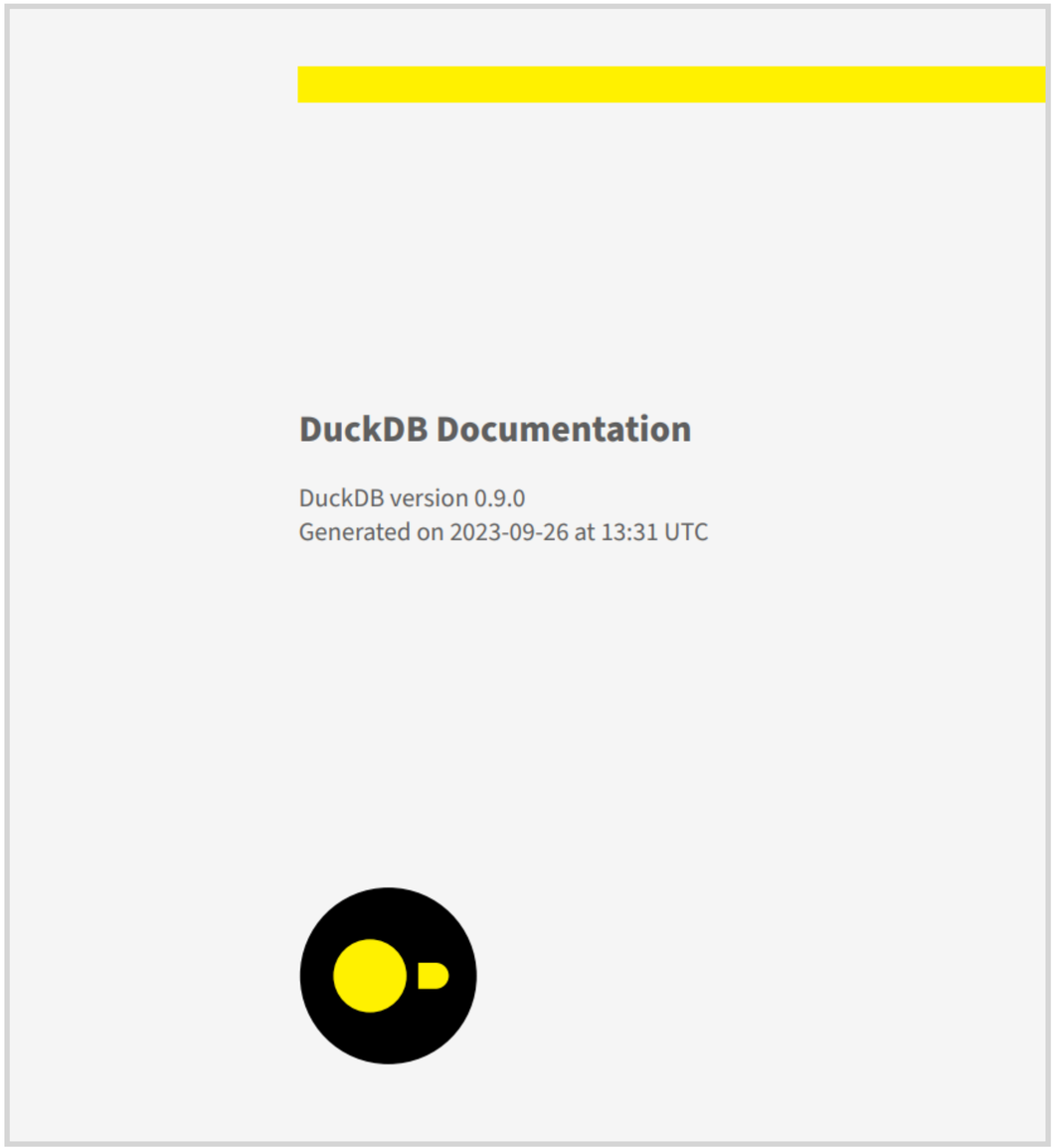
Built on open standards

Easy to avoid vendor lock-in

Works offline (PDF, ZIP)

No need to be permanently connected

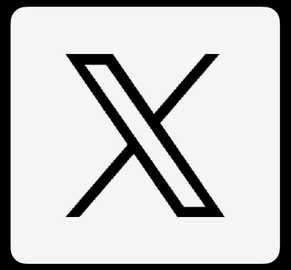
No tracking, no cookies



duckdb.org

Stay in touch

discord.duckdb.org



@duckdb

duckdb.org

Why DuckDB?