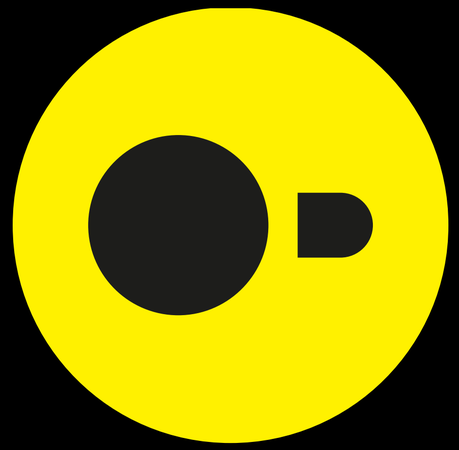
Super-Secret Next Big Thing for DuckDB

Hannes Mühleisen

Previously...



DuckDB

DuckDB

40 M

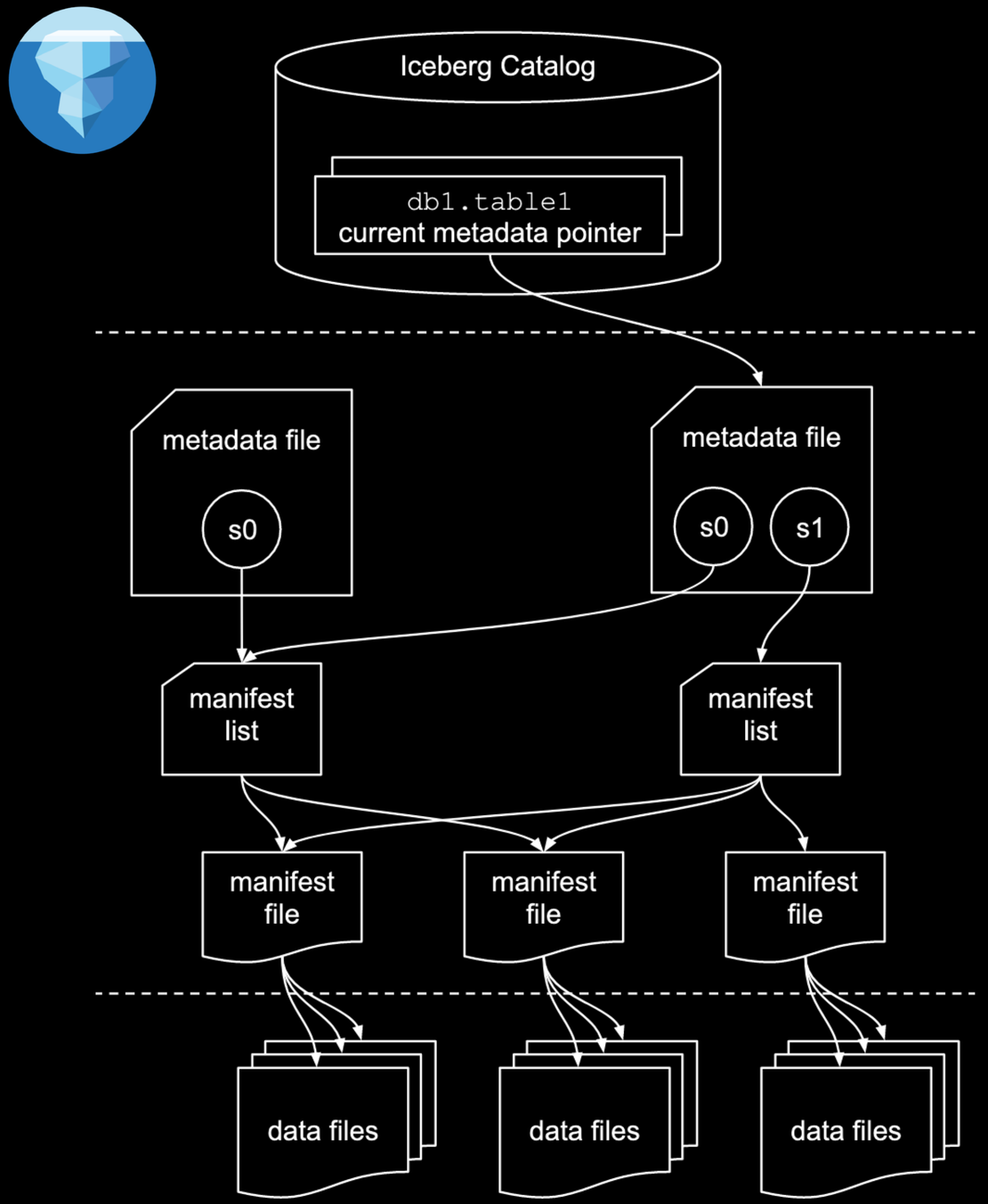
Extensions

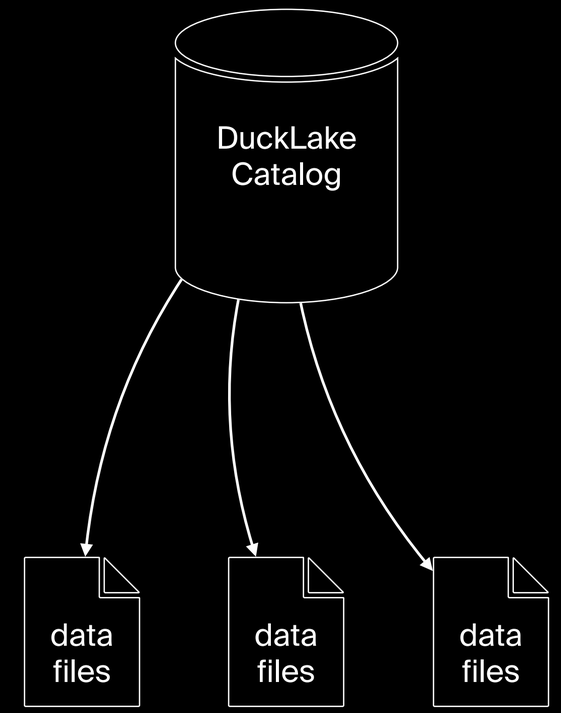
160 M

Downloads / Month


@duckdb.org
@hannes.muehleisen.org

DuckLake


Iceberg Catalog
db1.table1
current metadata pointer
metadata file
s0
metadata file
s0
s1
manifest list
manifest list
manifest file
manifest file
manifest file
data files
data files
data files


DuckLake Catalog
data files
data files
data files

ESSAY

DuckDB and DuckLake: Why We Bet the Company on the Duck Stack

MIKE RITCHIE · FEBRUARY 6, 2026 · 12 MIN READ

DuckLake v1.0: The Lakehouse Format Built on SQL Reaches Production-Readiness

The DuckDB team
2026-04-13 · 23 min

TL;DR: We are happy to release DuckLake v1.0, a production-ready lakehouse format specification built on SQL. Its reference implementation, the `ducklake` DuckDB extension, is available as of today in DuckDB v1.5.2.

In May 2025, we published the DuckLake manifesto, where we explained what motivated us to work on DuckLake. Here's a quick recap: we basically outlined how, in our view, it makes much more sense to store all metadata of a lakehouse in a **database** rather than in scattered files in object storage. This is why we created DuckLake.

> The manifesto is much more compelling, we recommend you read it!

Today, we are happy to announce **DuckLake v1.0**, almost a year after we released our first sketch of the specification. This is a production-ready release with guaranteed backward-compatibility. DuckLake v1.0 ships a stable specification, a feature-rich and fast reference implementation (the DuckDB `ducklake` extension), as well as a roadmap for future

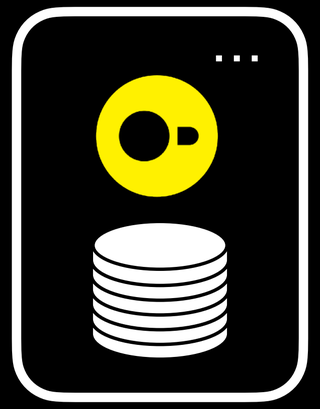
2.9 M

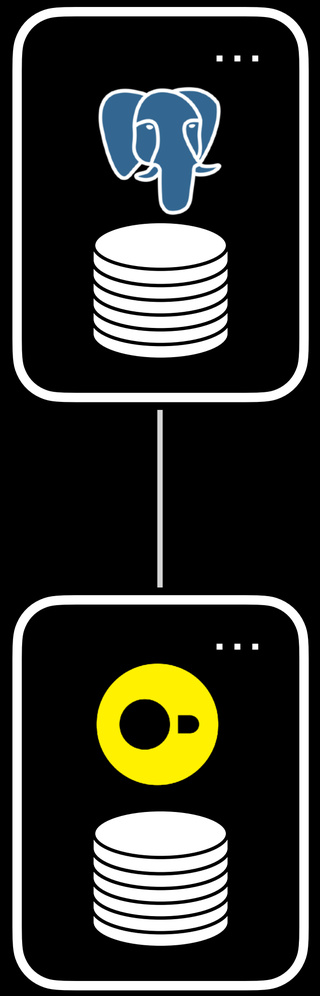
2.5 M

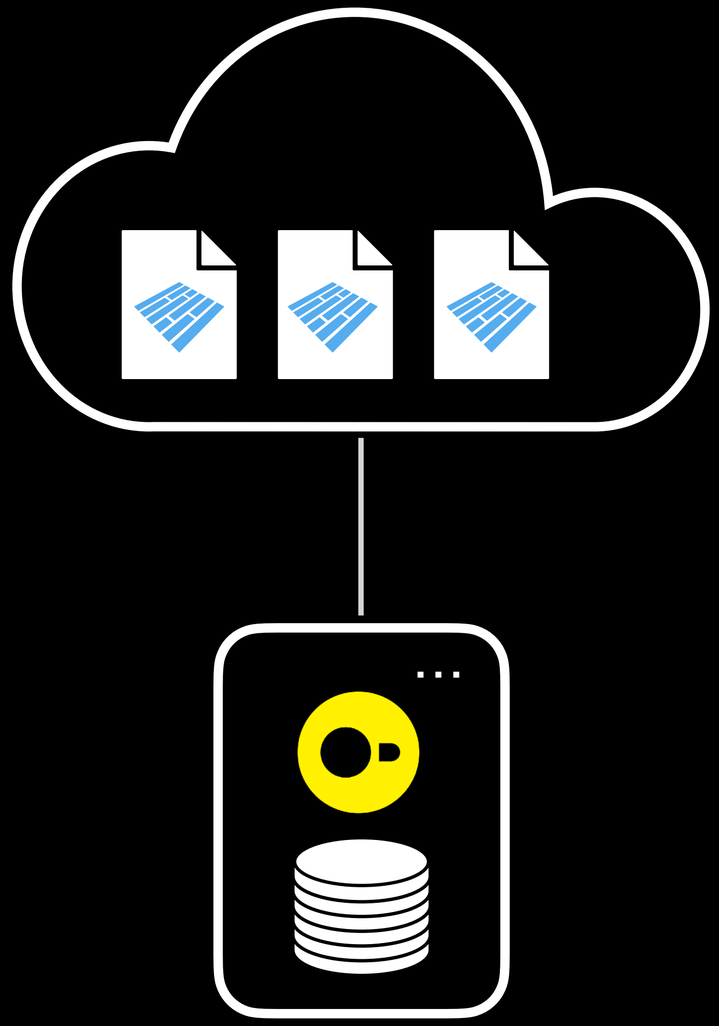
2.3 M

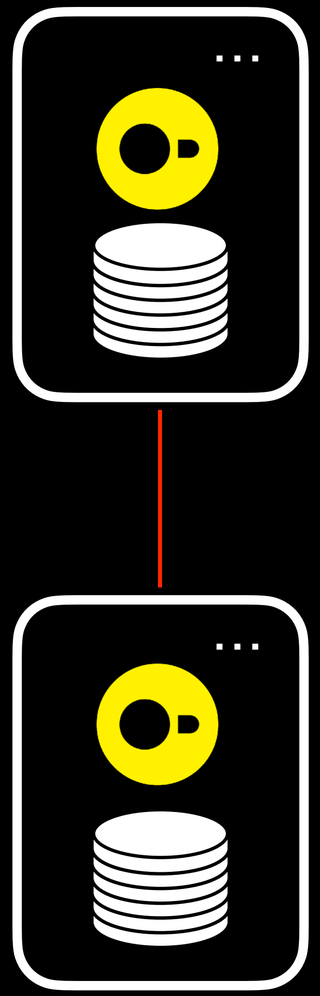
Downloads / Month

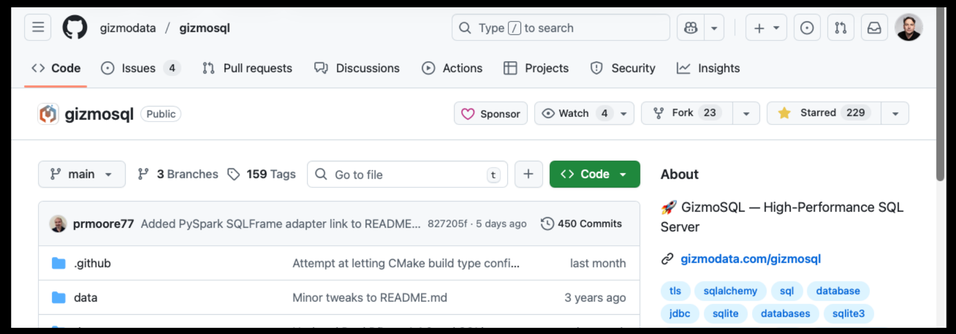
... and now?

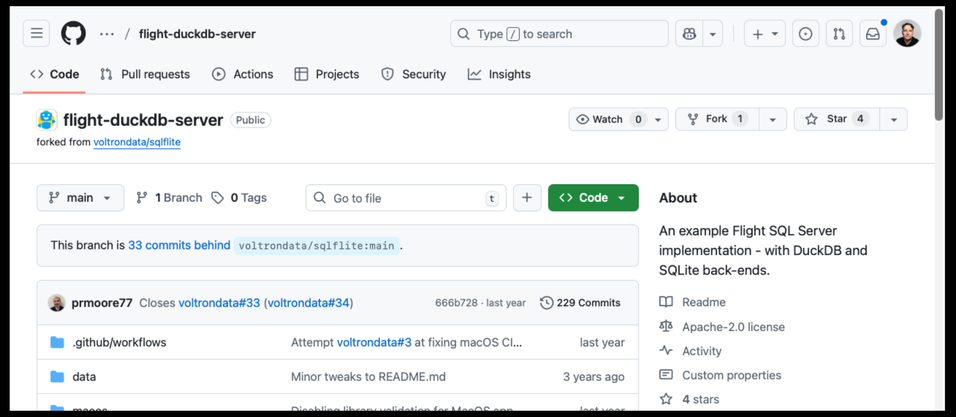


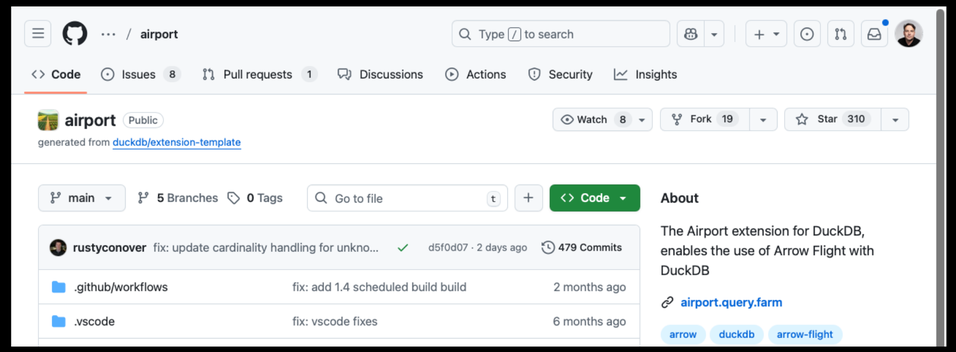


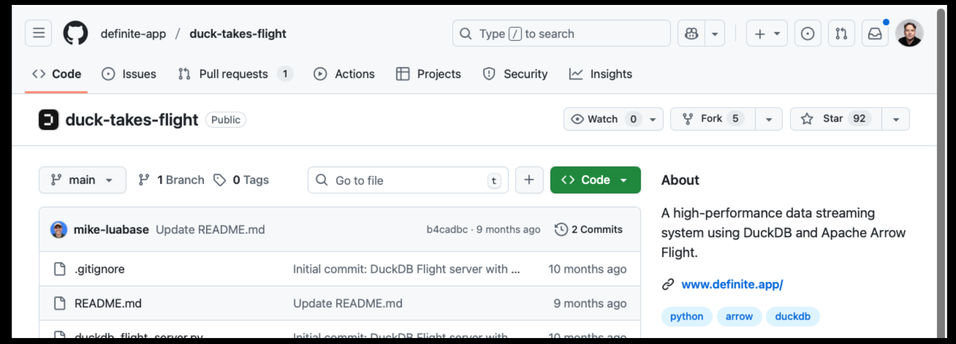


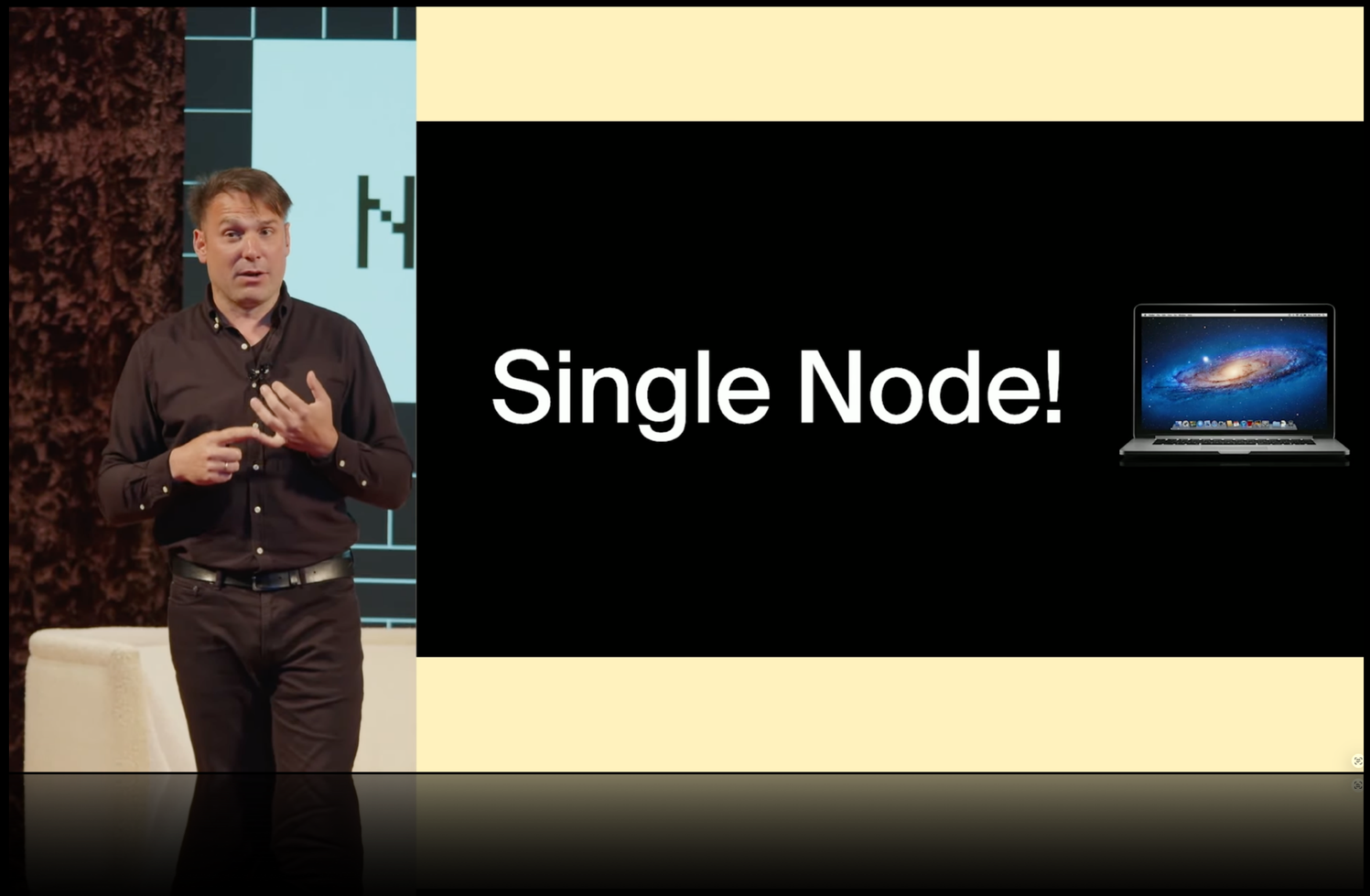


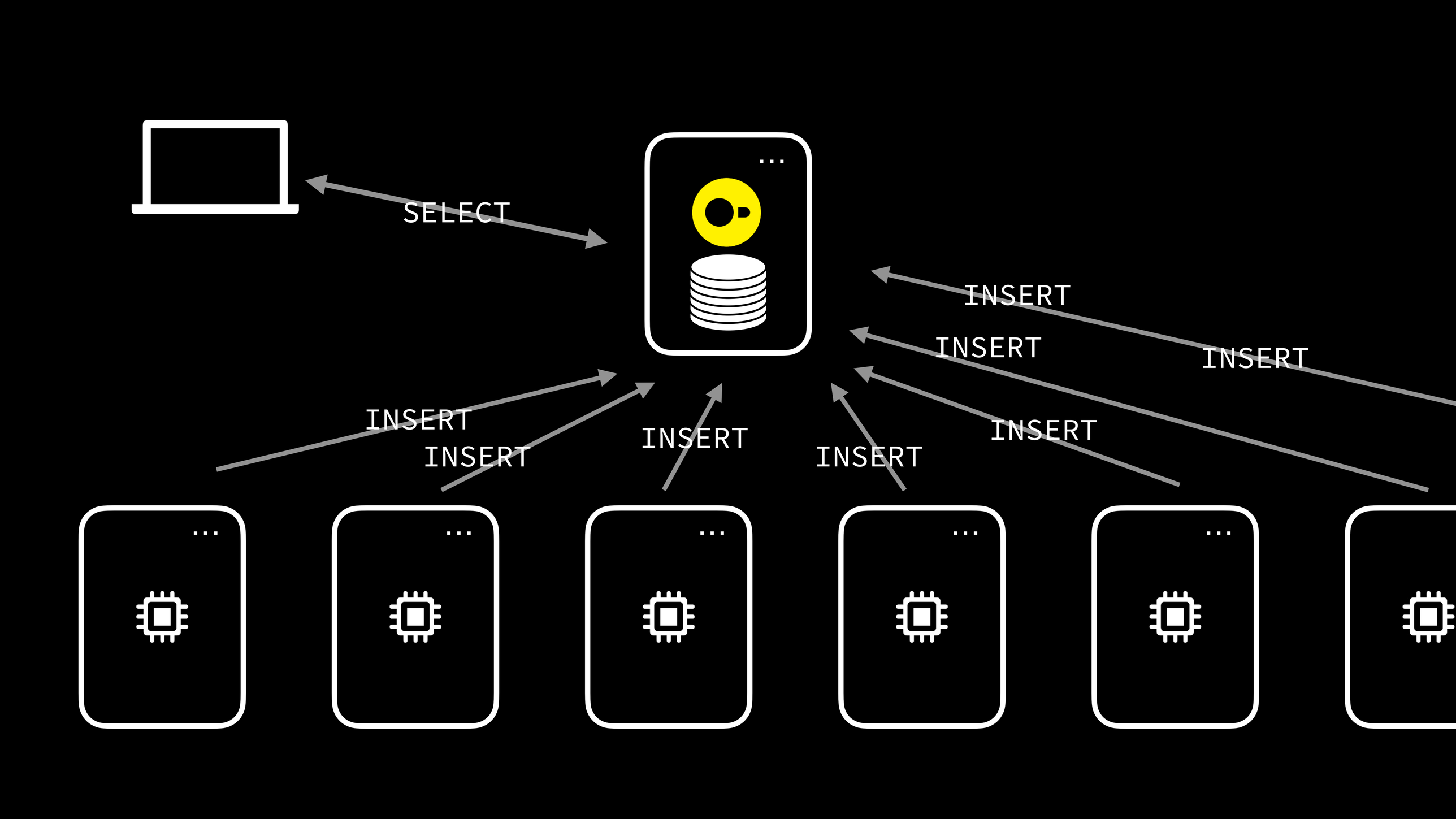

gizmodata / gizmosql
Type / to search
Code
Issues 4
Pull requests
Discussions
Actions
Projects
Security
Insights
gizmosql
Public
Sponsor
Watch 4
Fork 23
Starred 229
main
3 Branches
159 Tags
Go to file
Code
prmoore77 Added PySpark SQLFrame adapter link to README... 827205f · 5 days ago
450 Commits
.github
Attempt at letting CMake build type confi...
last month
data
Minor tweaks to README.md
3 years ago
About
🚀 GizmoSQL — High-Performance SQL Server
gizmodata.com/gizmosql
tls
sqlalchemy
sql
database
jdbc
sqlite
databases
sqlite3


flight-duckdb-server
Type / to search
Code
Pull requests
Actions
Projects
Security
Insights
flight-duckdb-server
Public
forked from voltrondata/sqlflite
Watch 0
Fork 1
Star 4
main
1 Branch
0 Tags
Go to file
Code
This branch is 33 commits behind voltrondata/sqlflite:main.
prmoore77 Closes voltrondata#33 (voltrondata#34)
666b728 · last year
229 Commits
.github/workflows
Attempt voltrondata#3 at fixing macOS CI...
last year
data
Minor tweaks to README.md
3 years ago
About
An example Flight SQL Server implementation - with DuckDB and SQLite back-ends.
Readme
Apache-2.0 license
Activity
Custom properties
4 stars


airport
Type / to search
Code
Issues 8
Pull requests 1
Discussions
Actions
Security
Insights
airport
Public
generated from duckdb/extension-template
Watch 8
Fork 19
Star 310
main
5 Branches
0 Tags
Go to file
Code
rustyconover fix: update cardinality handling for unkno...
d5f0d07 · 2 days ago
479 Commits
.github/workflows
fix: add 1.4 scheduled build build
2 months ago
.vscode
fix: vscode fixes
6 months ago
About
The Airport extension for DuckDB, enables the use of Arrow Flight with DuckDB
airport.query.farm
arrow
duckdb
arrow-flight


definite-app / duck-takes-flight
Type / to search
Code
Issues
Pull requests 1
Actions
Projects
Security
Insights
duck-takes-flight
Public
Watch 0
Fork 5
Star 92
main
1 Branch
0 Tags
Go to file
Code
mike-luabase Update README.md
b4cadbc · 9 months ago
2 Commits
.gitignore
Initial commit: DuckDB Flight server with ...
10 months ago
README.md
Update README.md
9 months ago
About
A high-performance data streaming system using DuckDB and Apache Arrow Flight.
www.definite.app/
python
arrow
duckdb


Single Node!


SELECT
INSERT
INSERT
INSERT
INSERT
INSERT
INSERT
INSERT
INSERT

Quack

```
CALL quack_serve('quack:localhost');
CREATE TABLE fuu AS FROM VALUES(42);

FROM fuu;
FROM fuu;
```

```
ATTACH 'quack:localhost' AS remote;

FROM remote.fuu;
FROM remote.query('FROM fuu');
```

```
FORCE INSTALL quack
  FROM core_nightly;
LOAD quack;

CALL quack_serve('quack:localhost',
  token='asdf');


SELECT 42;
```

```
FORCE INSTALL quack
  FROM core_nightly;
LOAD quack;

CREATE SECRET (TYPE quack,
  TOKEN 'asdf');
ATTACH 'quack:localhost' AS remote;

FROM remote.query('SELECT 42');
```

Internals

TCP/IP

HTTP(S)

TCP/IP

duckdb::Serializer

HTTP(S)

TCP/IP

duckdb::Message

duckdb::Serializer

HTTP(S)

TCP/IP

AuthN A12n Authentication?

AuthZ A11n Authorization?

Experiments!

Don’t Hold My Data Hostage – A Case For Client Protocol Redesign

Mark Raasveldt
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
m.raasveldt@cwi.nl

Hannes Mühleisen
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
hannes@cwi.nl

ABSTRACT

Transferring a large amount of data from a database to a client program is a surprisingly expensive operation. The time this requires can easily dominate the query execution time for large result sets. This represents a significant hurdle for external data analysis, for example when using statistical software. In this paper, we explore and analyse the result set serialization design space. We present experimental results from a large chunk of the database market and show the inefficiencies of current approaches. We then propose a columnar serialization method that improves transmission performance by an order of magnitude.

Keywords

Databases, Client Protocols, Data Export

1. INTRODUCTION

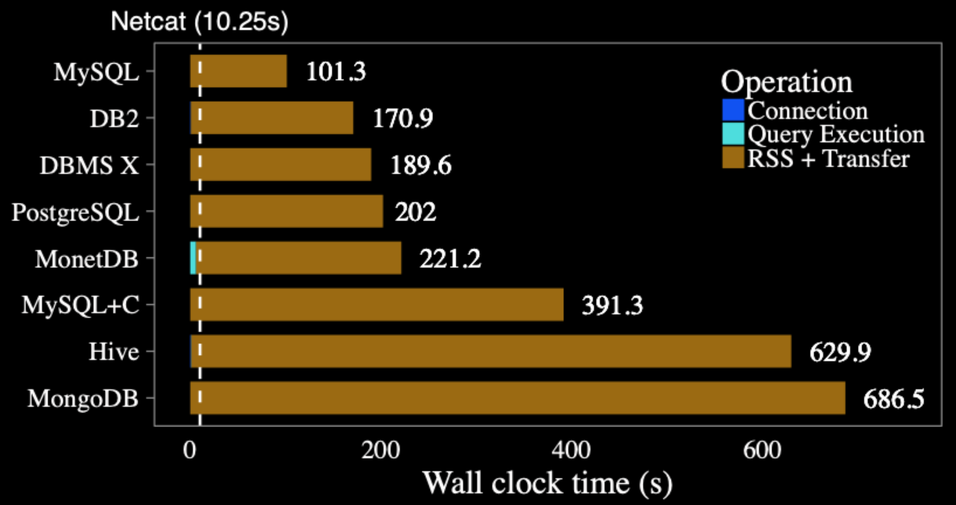
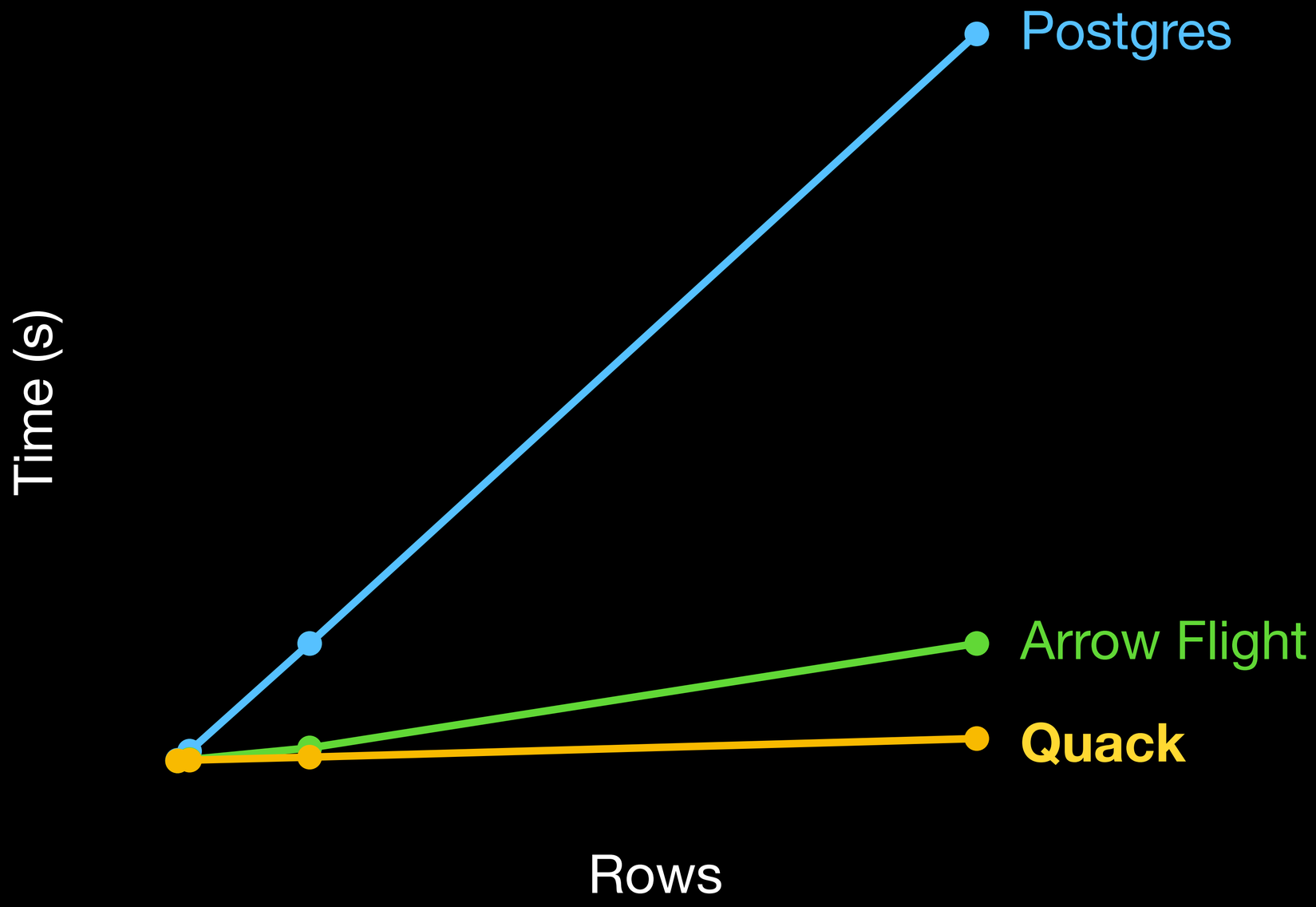


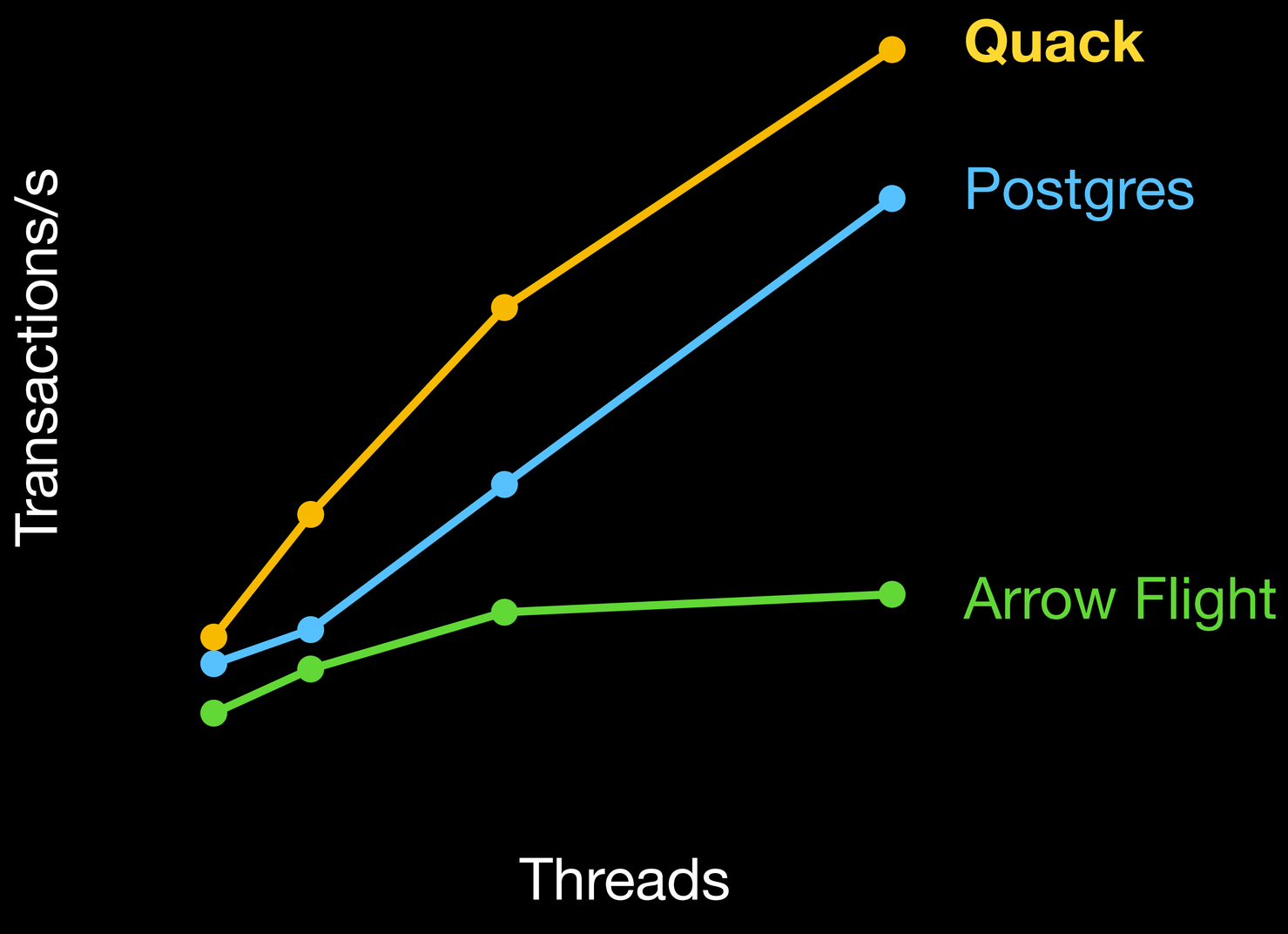
Figure 1: Wall clock time for retrieving the `lineitem` table (SF10) over a loopback connection. The dashed line is the wall clock time for netcat to transfer a CSV of the data.


Client
Server

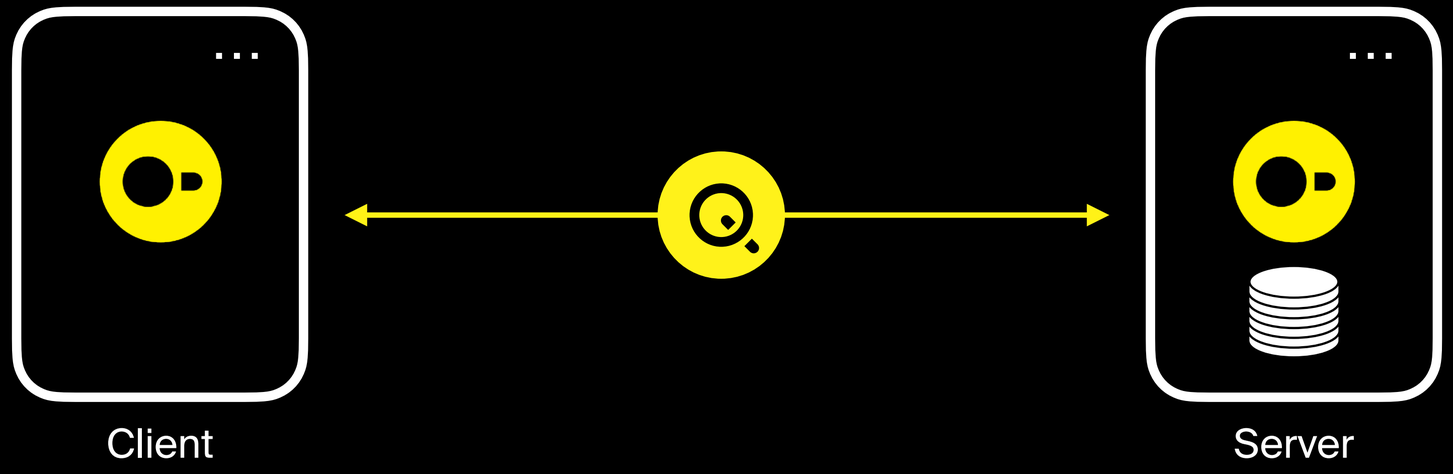


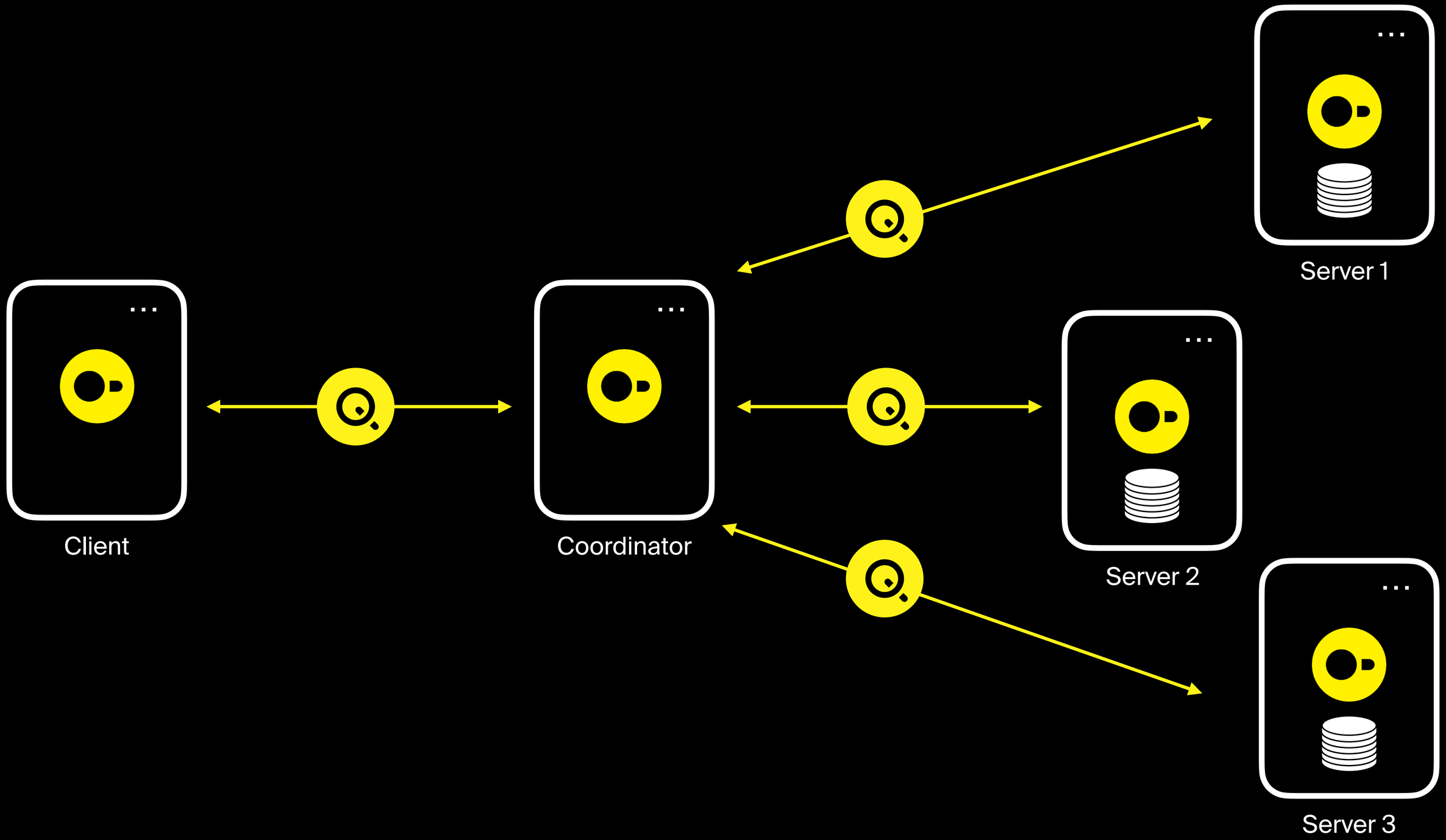
Lower is better

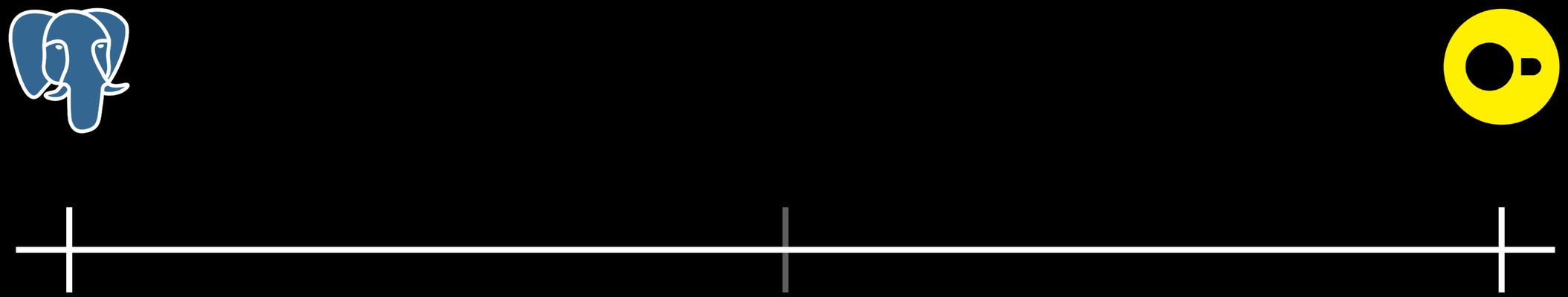
Transactions?!



Higher is better


Client
Server


Server 1
Client
Coordinator
Server 2
Server 3



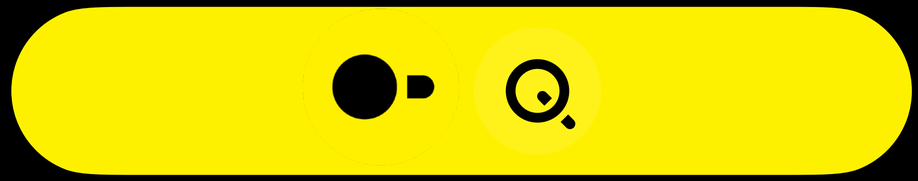
Yesterday

OLTP

GPTP

OLAP

Yesterday, actually

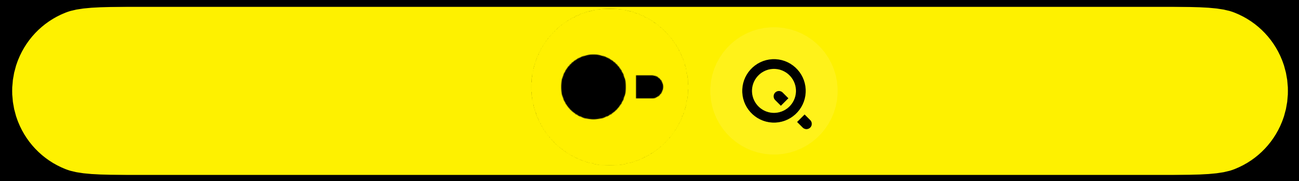


OLTP

GPTP

OLAP

Today

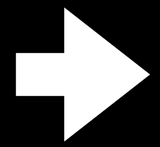


OLTP

GPTP

OLAP

Tomorrow?

Data Fear  Data Confidence

Time for Demo?

click

Quack

quack.duckdb.org