MotherDuck

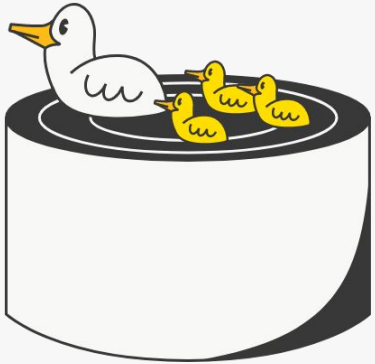
WHAT CAN POSTGRES LEARN FROM DUCKDB?

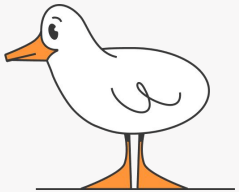
Jelte Fennema-Nio (@JelteF)

Postgres Contributor

Maintainer of PgBouncer & pg_duckdb

2025-05-15



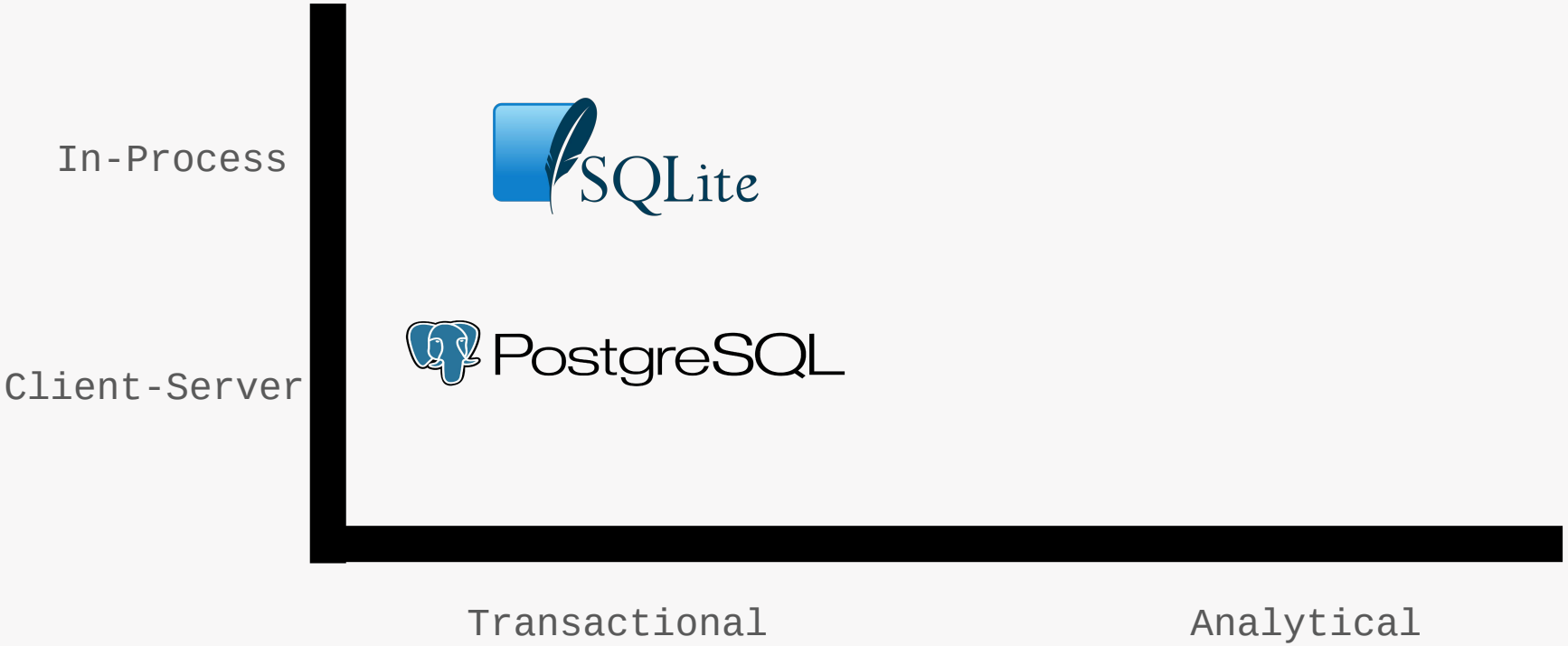


What is **DuckDB**?

Lightweight in-process SQL Analytics Engine

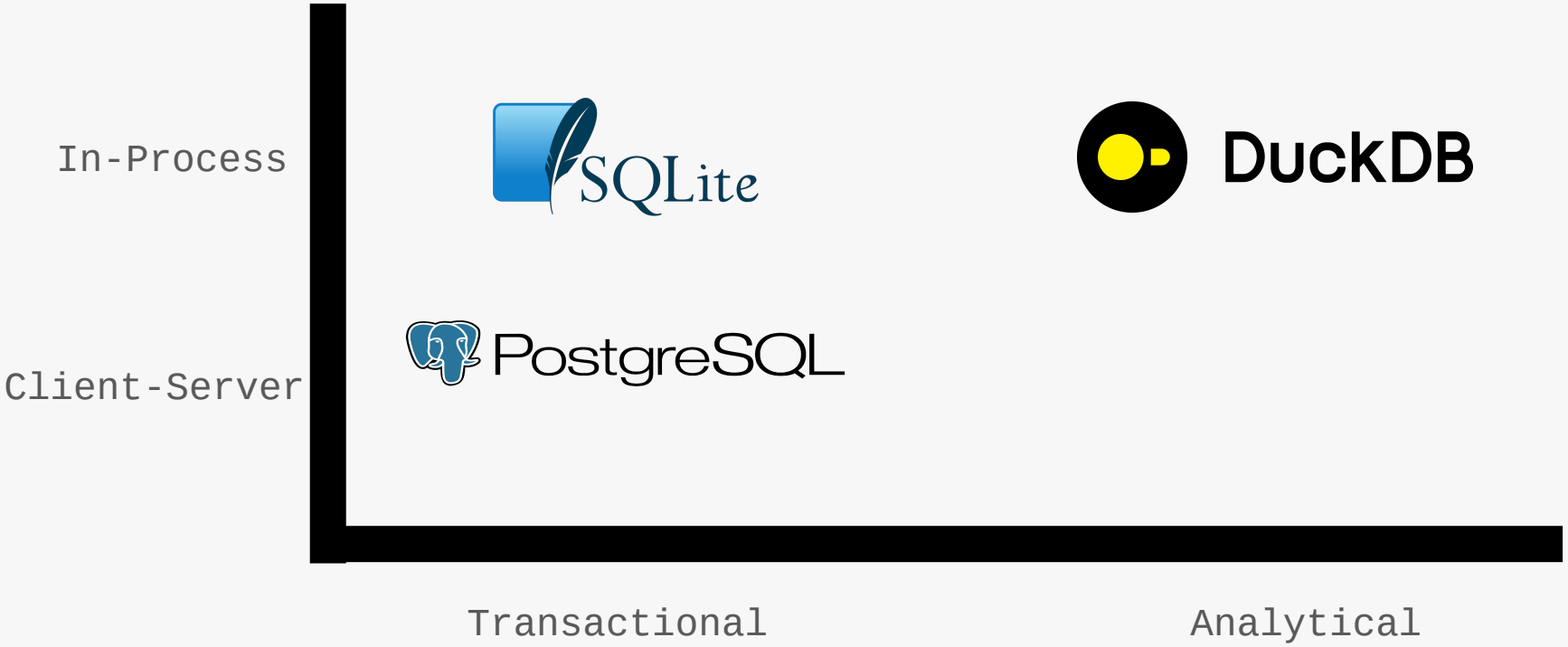
MotherDuck

DuckDB is a new category of database



MotherDuck

DuckDB is a new category of database



MotherDuck

DuckDB

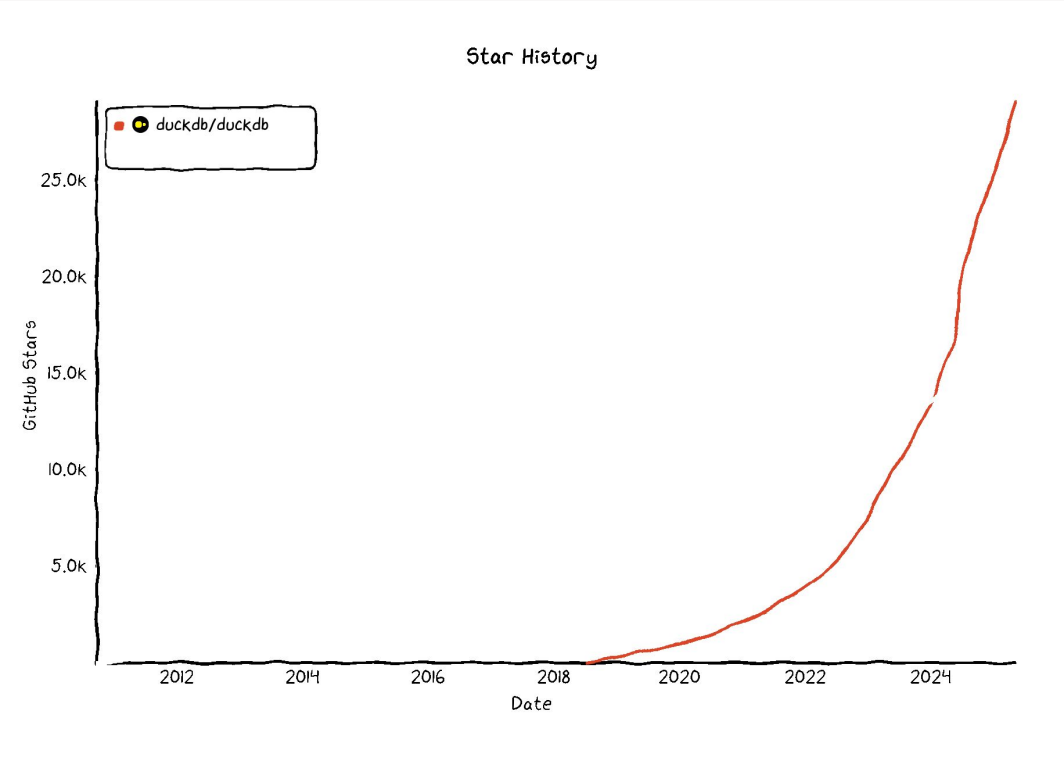
created at: CWI Research

created by:

maintained by: DuckDB Labs

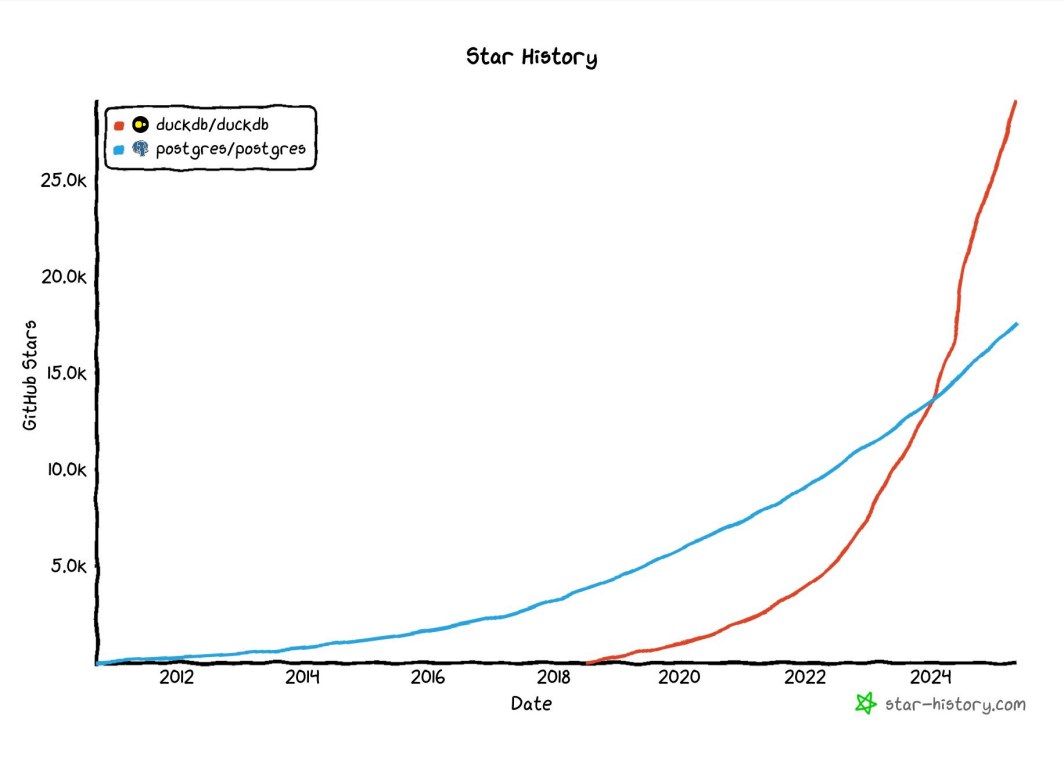
DuckDB Community & Foundation

And it’s very popular



MotherDuck

And it’s very popular



MotherDuck

Swiss army-knife for data

World’s best CSV parser

- An absurd amount of the world runs on CSV files
- An absurd amount of the world has broken / wonky CSV files
- An absurd amount of data engineering time is spent dealing with CSV file peculiarities
- Wouldn’t it be nice if they just … worked?

Multi-Hypothesis CSV Parsing

Till Döhmen
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
tilldoehmen@gmail.com

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

Peter Boncz
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
boncz@cwi.nl

ABSTRACT

Comma Separated Value (CSV) files are commonly used to represent data. CSV is a very simple format, yet we show that it gives rise to a surprisingly large amount of ambiguities in its parsing and interpretation. We summarize the state-of-the-art in CSV parsers, which typically make a linear series of parsing and interpretation decisions, such that any wrong decision at an earlier stage can negatively affect all downstream decisions. Since computation time is much less scarce than human time, we propose to turn CSV parsing into a ranking problem. Our quality-oriented *multi-hypothesis* CSV parsing approach generates several concurrent hypotheses about dialect, table structure, etc. and ranks these hypotheses based on quality features of the resulting table. This approach makes it possible to create an advanced CSV parser that makes many different decisions, yet keeps the overall parser code a simple plug-in infrastructure. The complex interactions between these decisions are taken care of by searching the hypothesis space rather than by having to program these many interactions in code. We show that our approach leads to better parsing results than the state of the art and facilitates the parsing of large corpora of heterogeneous CSV files.

CCS CONCEPTS

• **Information systems → Inconsistent data**;

ACM Reference format:
Till Döhmen, Hannes Mühleisen, and Peter Boncz. 2017. Multi-Hypothesis CSV Parsing. In *Proceedings of SSDBM '17, Chicago, IL, USA, June 27-29, 2017,* 12 pages.
https://doi.org/http://dx.doi.org/10.1145/3085504.3085520

1 INTRODUCTION

Data scientists typically lose much time in importing and cleaning data, and large data repositories such as open government collections with tens of thousands of datasets remain under-exploited due to the high human cost of discovering, accessing and cleaning this data. CSV is the most commonly used data format in such repositories. The lack of explicit information on the CSV dialect, the table structure, and data types makes proper parsing tedious and error-prone.

SSDBM '17, June 27-29, 2017, Chicago, IL, USA

ACM ISBN 978-1-4503-5282-6/17/06…$15.00
https://doi.org/http://dx.doi.org/10.1145/3085504.3085520

Figure 1: Ambiguous CSV file which is at risk to be parsed incorrectly, because the number of commas and the number of semi-colons per row are the same.

Tools currently popular among data scientists, such as R and Python offer robust CSV parsing libraries, which try to address parsing of messy CSV files with a number of practical heuristics. These libraries makes a linear sequence of parsing and interpretation decisions, such that any wrong decision at an earlier stage (e.g. determining the separator character) will negatively affect all downstream decisions. Interlinking different parsing steps (backtracking on prior decisions) is not done, because if all parsing decisions affect each other, the parsing code becomes very complex (code size would need to grow quadratically in the amount of decisions or even worse).

Since CPU-cycles are currently plentiful but human time is not, this research pursues an approach where CSV parsing becomes an computerized search problem. Our quality-oriented CSV parsing approach generates several concurrent hypotheses about dialect, table structure, etc. and in the end ranks these hypotheses based on quality features of the resulting table, such that the top-1 would be the automatic parsing result, or a top-K of parsed tables could be presented to the user. A high absolute score from the quality function can also be used to automatically parse large amounts of files. Only ambigious cases would be presented to a user. This can strongly reduce human data interpretation effort.

This very practical problem touches on various areas of related work. In the extended version of this paper [6], we survey the state-of-the art on this topic, which covers areas such as computer-assisted data-cleaning (*data-wrangling*), table-interpretation (e.g. on the web), automatic list extraction and even automated semantic (web) enrichment; covered more briefly in the related work Section 5.

Outline. In Section 2 we explaine CSV parsing problem by example, and introduce our multi-hypothesis parsing framework in Section 3. We demonstrate the improved parsing quality of our approach with computed quality metrics on the full *data.gov.uk* dataset collection, as well as on a sample of this collection using human ground truth in Section 4. We summarize related work in Section 5 and describe next steps in Section 6 before concluding in Section 7.

DuckDB runs anywhere

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

npm install duckdb

pip install duckdb

org.duckdb:duckdb_jdbc

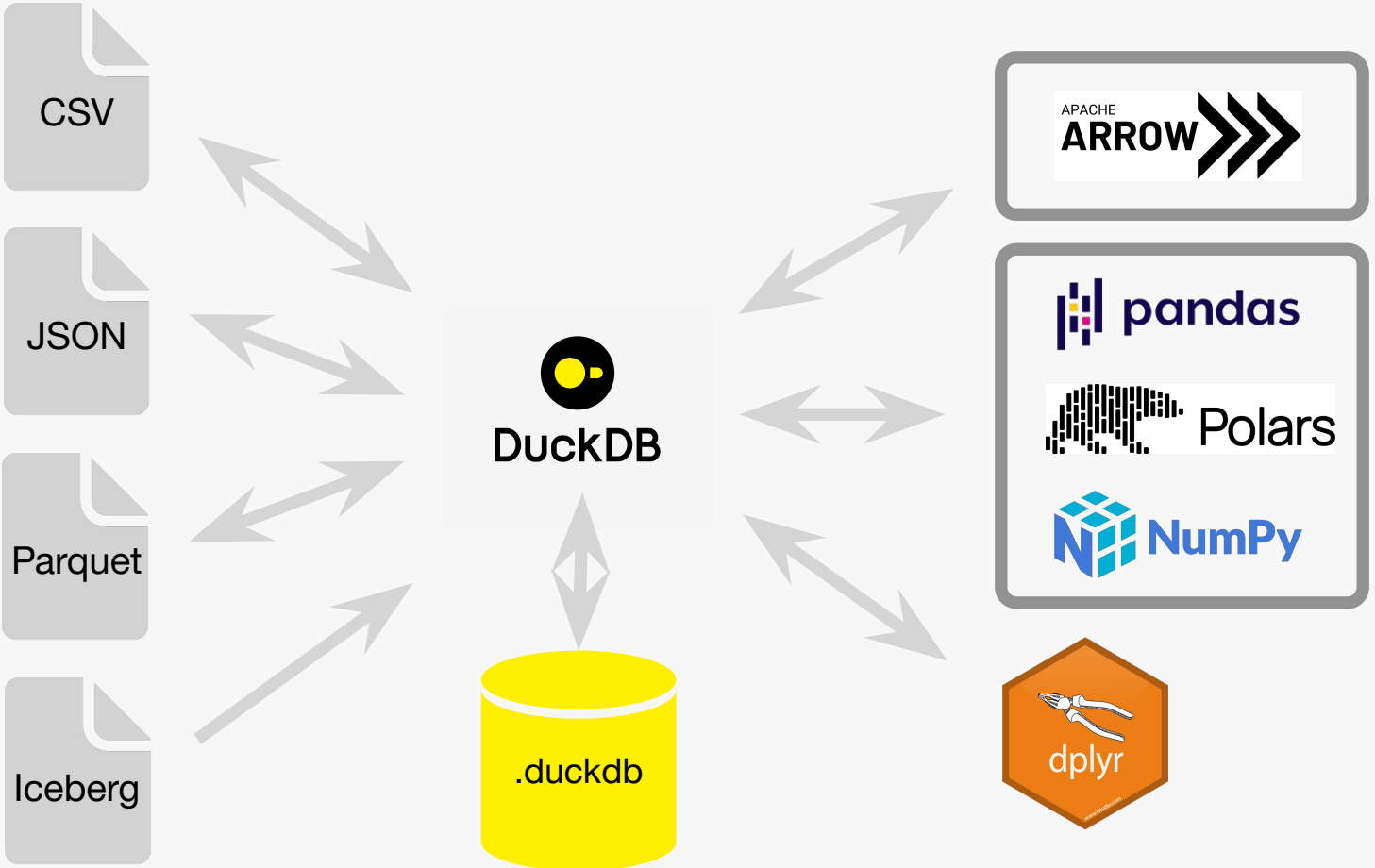
install.packages('duckdb')

Web browsers (WebAssembly)

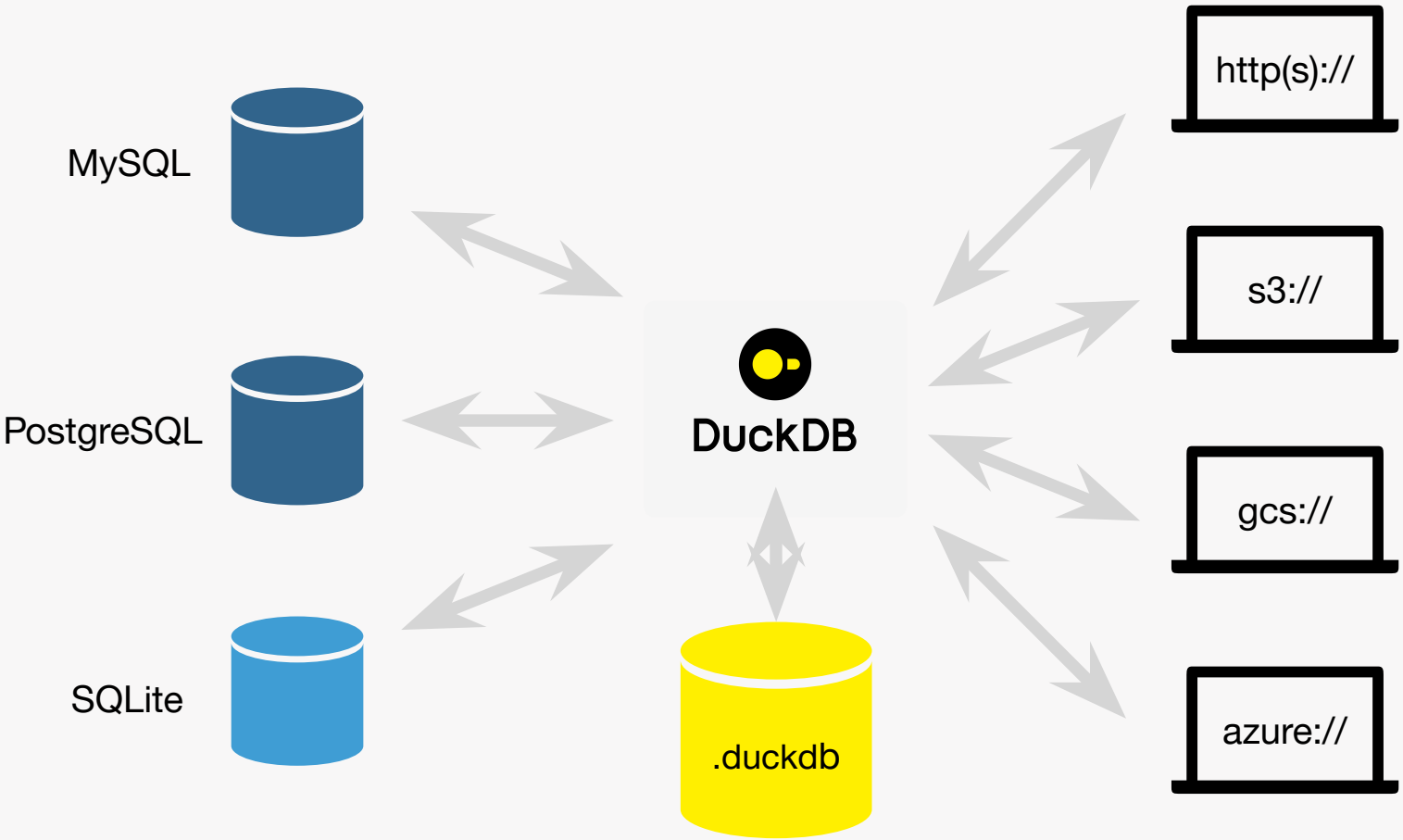
cargo add duckdb

ODBC, ADBC, & ~10 more languages!

Input and output formats

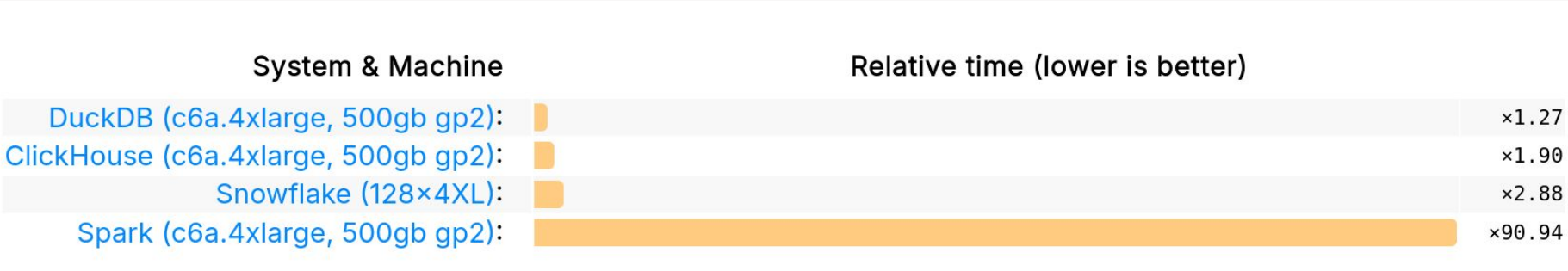


Data sources and destinations



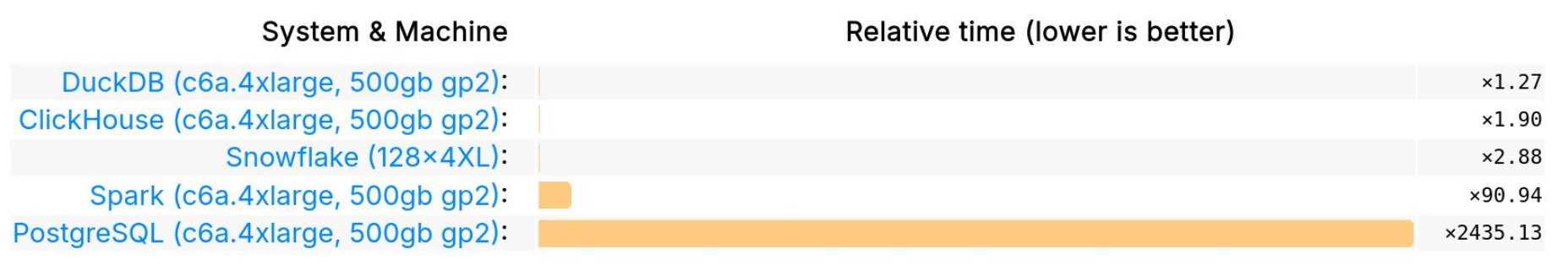
But is it fast???

YES!



MotherDuck

YES!



MotherDuck

What can Postgres learn?

What can Postgres learn?

1. **Ease of use**
2. **Performance**
3. **Extensibility**

MotherDuck

Ease of use

Demo time

```
$ duckdb -ui
```

MotherDuck

Generic types

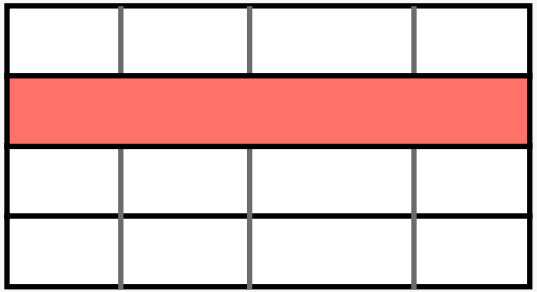
1. `MAP(TEXT, INT)`
2. `UNION(i INT, f FLOAT)`
3. `STRUCT(t TEXT, i INTEGER)`
4. `Sensible arrays: TEXT[2][6] and TEXT[][]`

MotherDuck

Performance

Postgres storage format

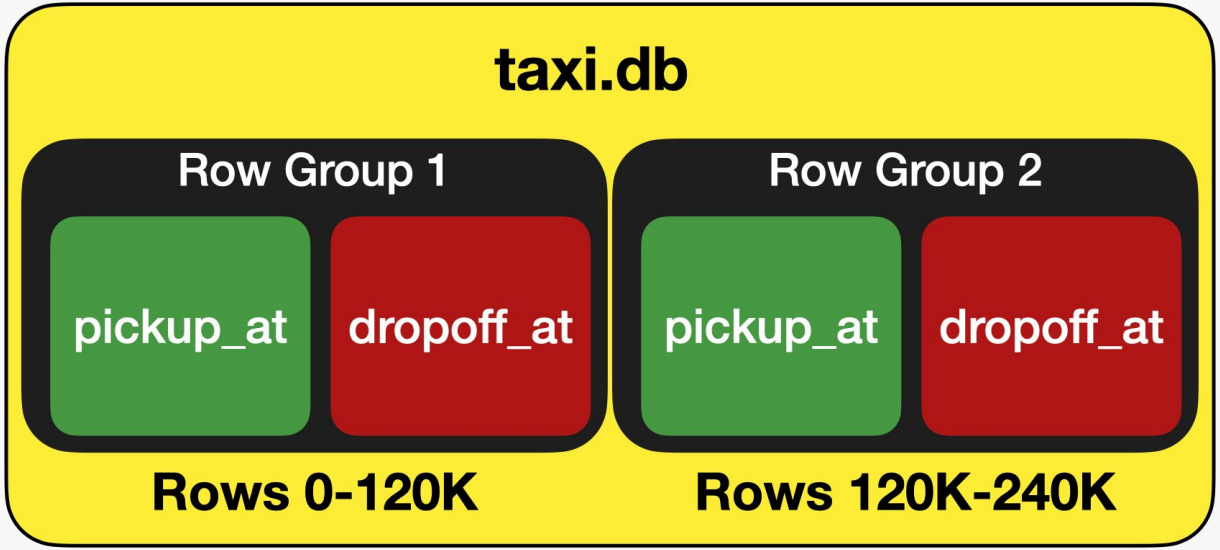
Row-based (tuples)



Optimized for:

* low memory footprint
* transactional workloads

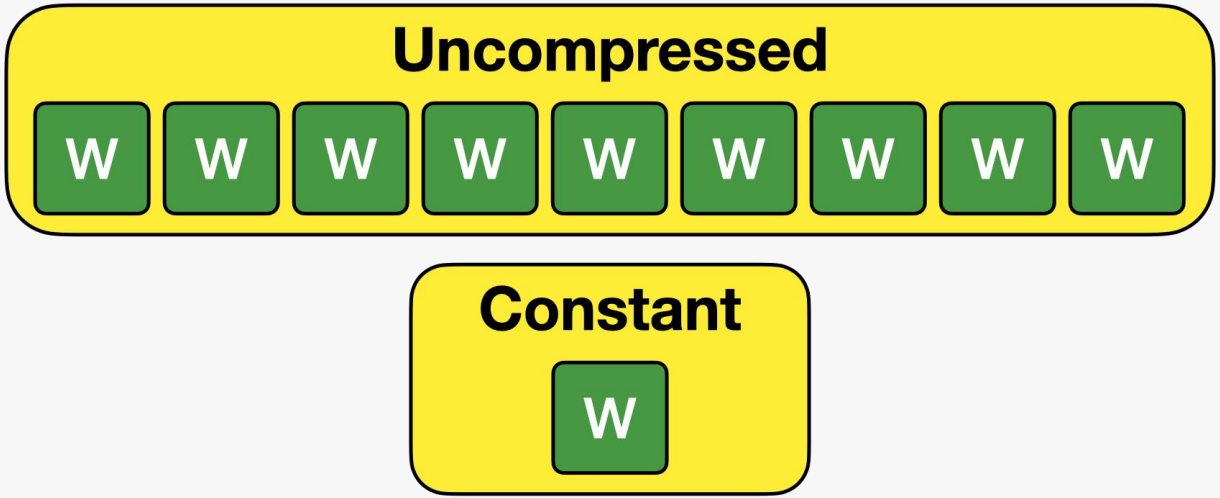
DuckDB storage format



Source: https://duckdb.org/2022/10/28/lightweight-compression.html

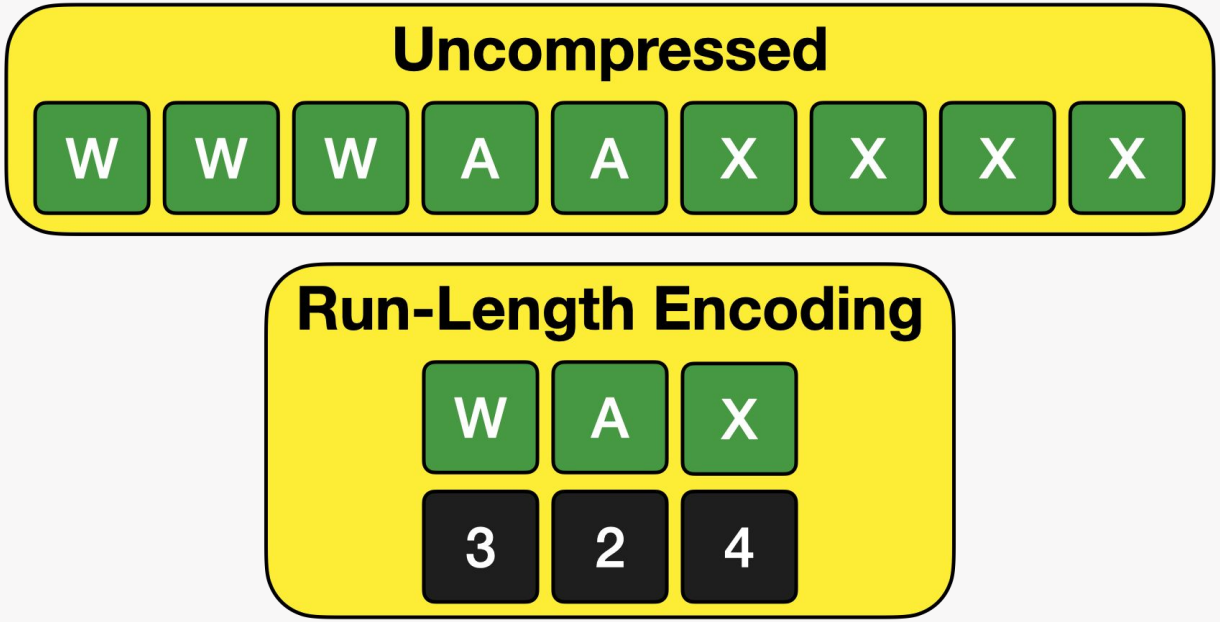
… with lightweight compression

Constant vectors



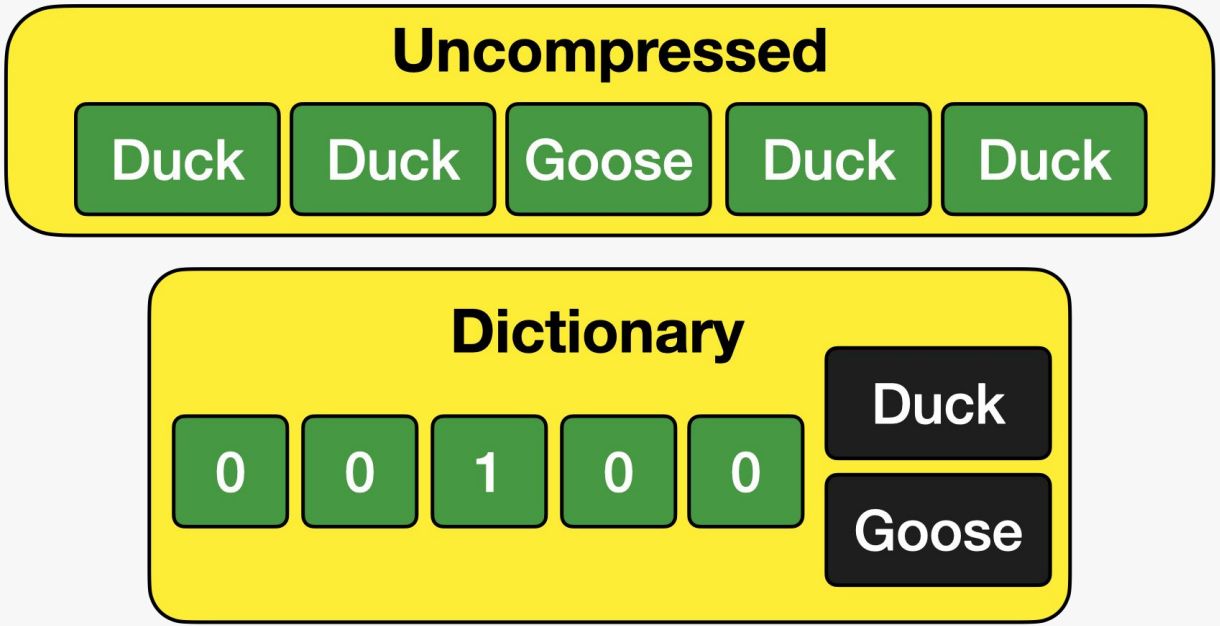
Source: https://duckdb.org/2022/10/28/lightweight-compression.html

Run-Length Encoding (RLE)



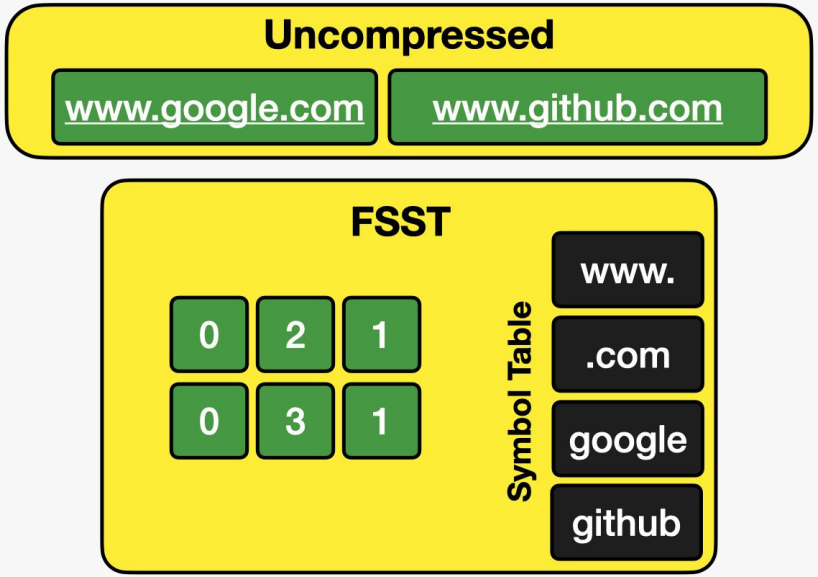
Source: https://duckdb.org/2022/10/28/lightweight-compression.html

Dictionary Encoding



Source: https://duckdb.org/2022/10/28/lightweight-compression.html

Fast Static Symbol Table



Source: https://duckdb.org/2022/10/28/lightweight-compression.html

How about execution?

Morsel-Driven Parallelism

Morsel-Driven Parallelism: A NUMA-Aware Query Evaluation Framework for the Many-Core Age

Viktor Leis* Peter Boncz† Alfons Kemper* Thomas Neumann*

* Technische Universität München † CWI

* {leis,kemper,neumann}@in.tum.de † p.boncz@cwi.nl

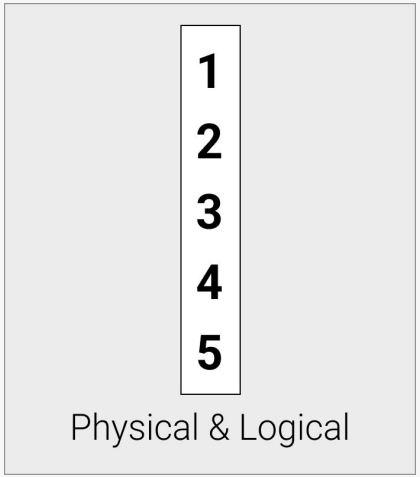
Source: https://db.in.tum.de/~leis/papers/morsels.pdf

Morsel-Driven Parallelism

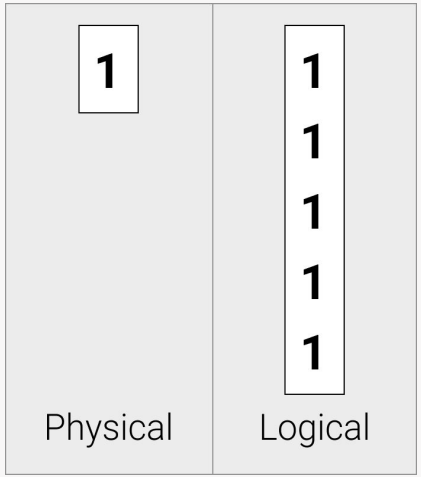
- **Morsel-driven query execution** is a new parallel query evaluation framework that fundamentally differs from the traditional Volcano model in that it distributes work between threads dynamically using work-stealing. This prevents unused CPU resources due to load imbalances, and allows for *elasticity*, i.e., CPU resources can be reassigned between different queries at any time.

Execution on compressed data

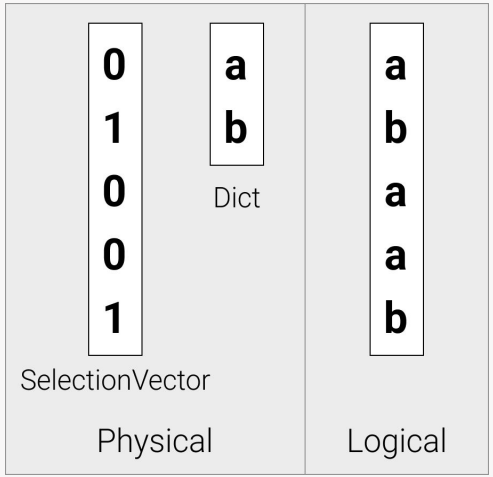
Flat



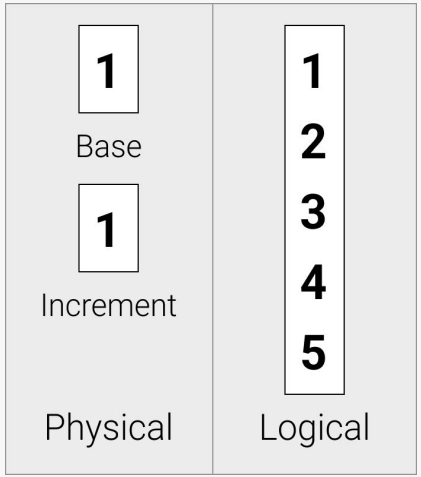
Constant



Dictionary



Sequence

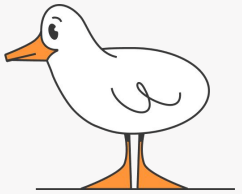


Source: https://duckdb.org/docs/stable/internals/vector

Factorized joins

Extensibility

Built-in package manager



```
INSTALL spatial
```

MotherDuck

FDWs on steroids

```
ATTACH 'host=my-postgres.com' AS postgres (TYPE postgres);

FROM postgres.myschema.users WHERE age > 21;
```

MotherDuck

Custom filesystems

```
FROM 's3://my-bucket/myfile.parquet';

FROM iceberg_scan('az://metadata/v1.metadata.json');
```

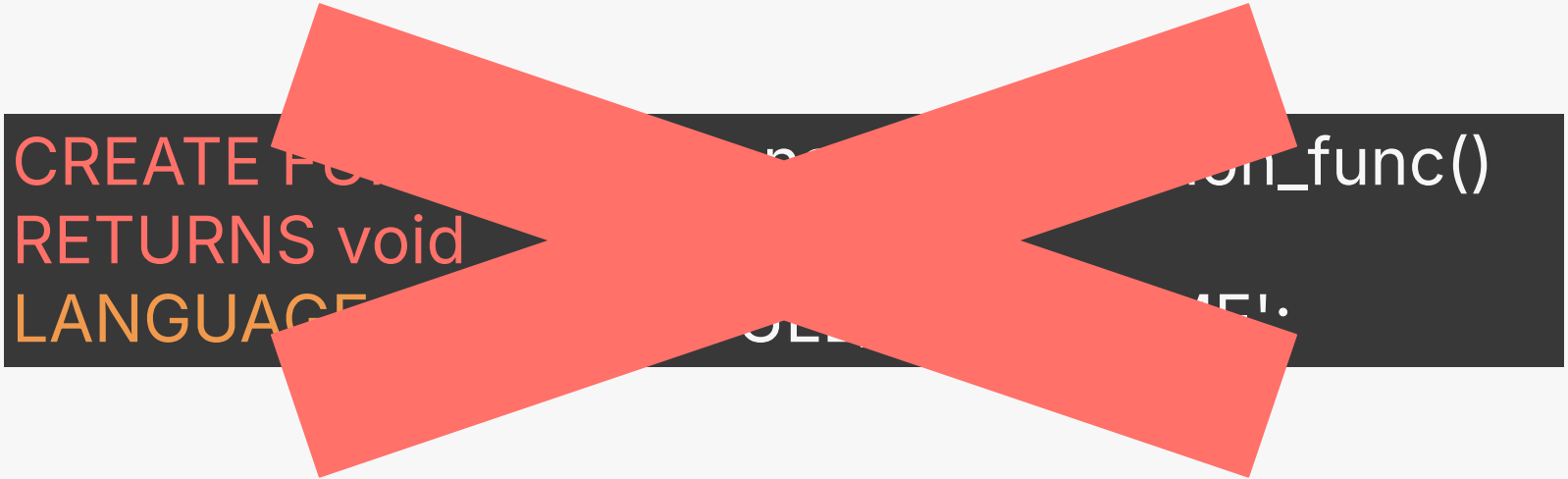
MotherDuck

Runtime added native functions

```
CREATE FUNCTION my_native_extension_func()
RETURNS void
LANGUAGE C AS 'MODULE_PATHNAME';
```

MotherDuck

Runtime added native functions



MotherDuck

There’s more!

- duckdb.org/news
- github.com/duckdb/duckdb
- github.com/duckdb/pg_duckdb
- MIT licensed

Questions?

MotherDuck