DuckDB:

Extraindo toda a potência do seu Notebook.

Pedro Holanda

Introdução

CWI

CWI

Algoritmo de Dijkstra

CWI

Algoritmo de Dijkstra

Internet

CWI

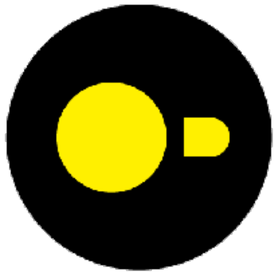
Algoritmo de Dijkstra

Internet

Python

Arquitetura de Banco de Dados

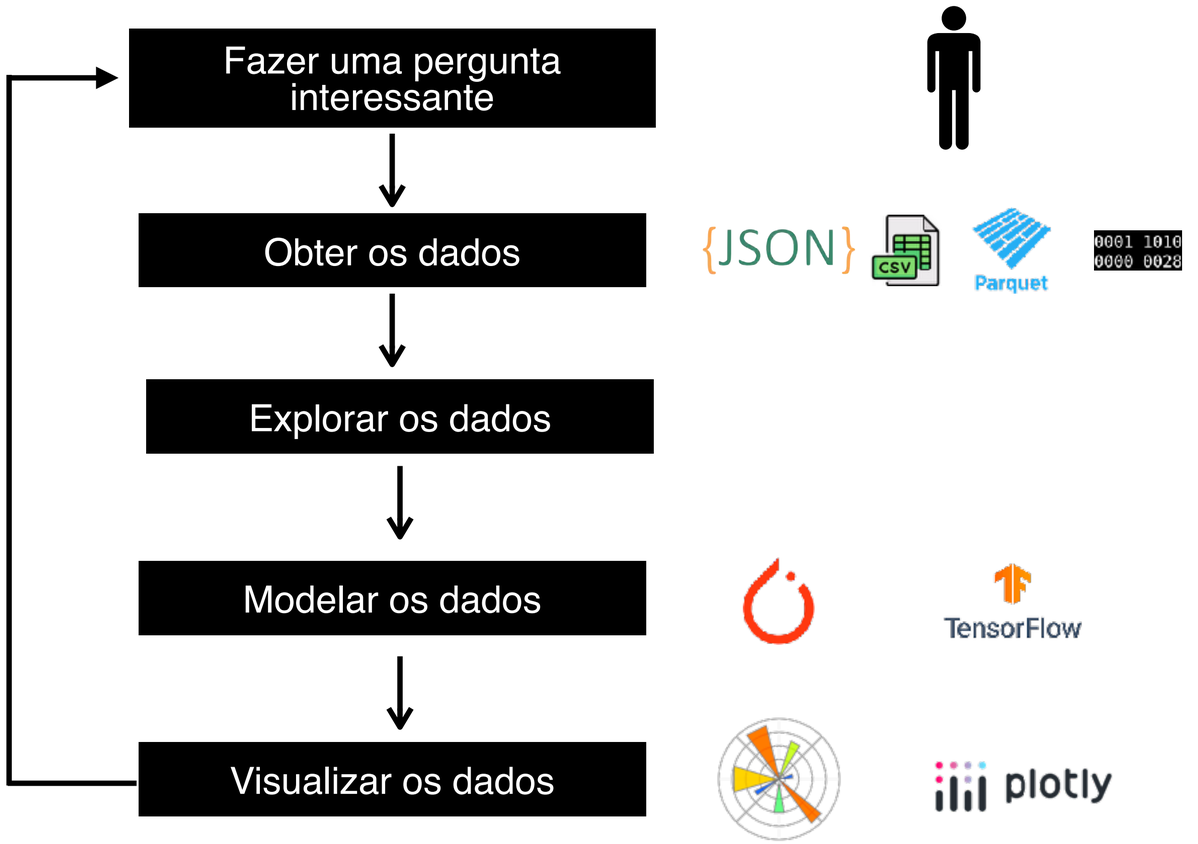
CWI
Database Architectures



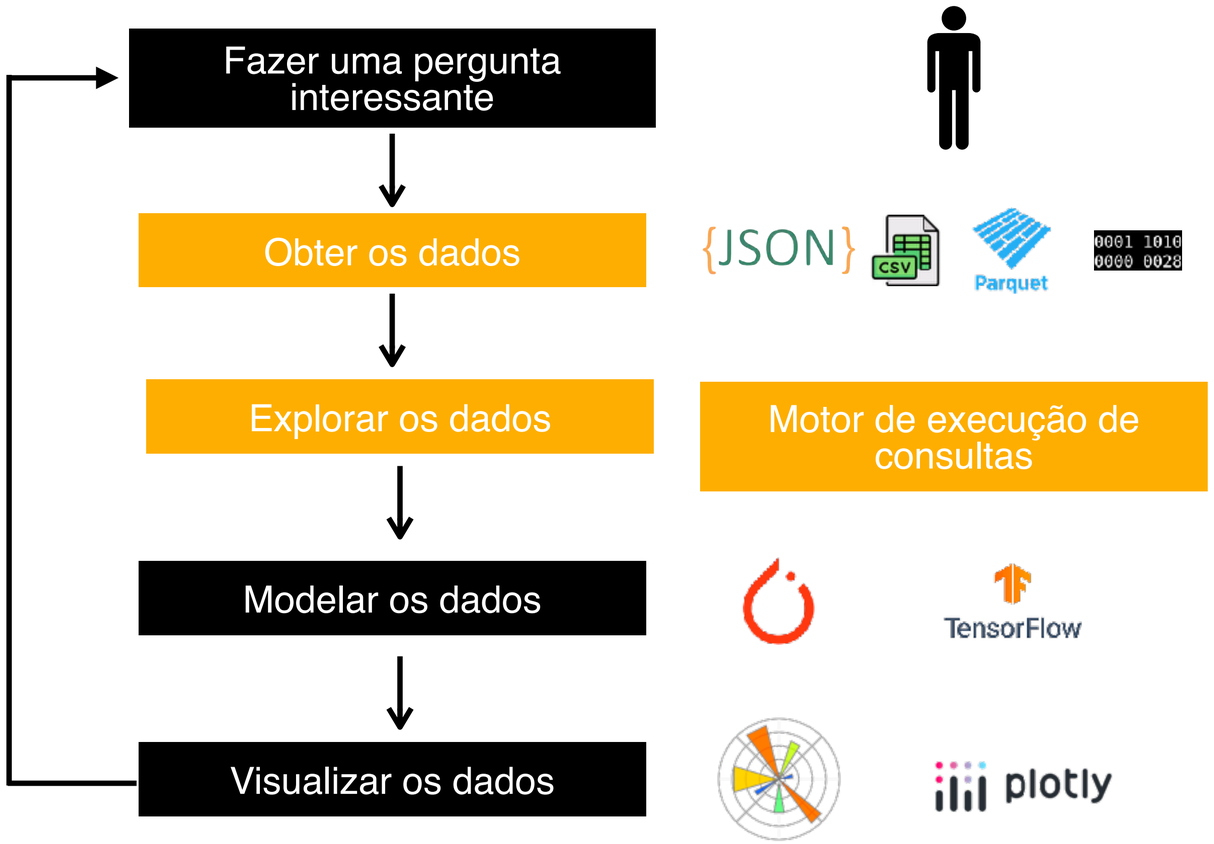
DuckDB

Motivação

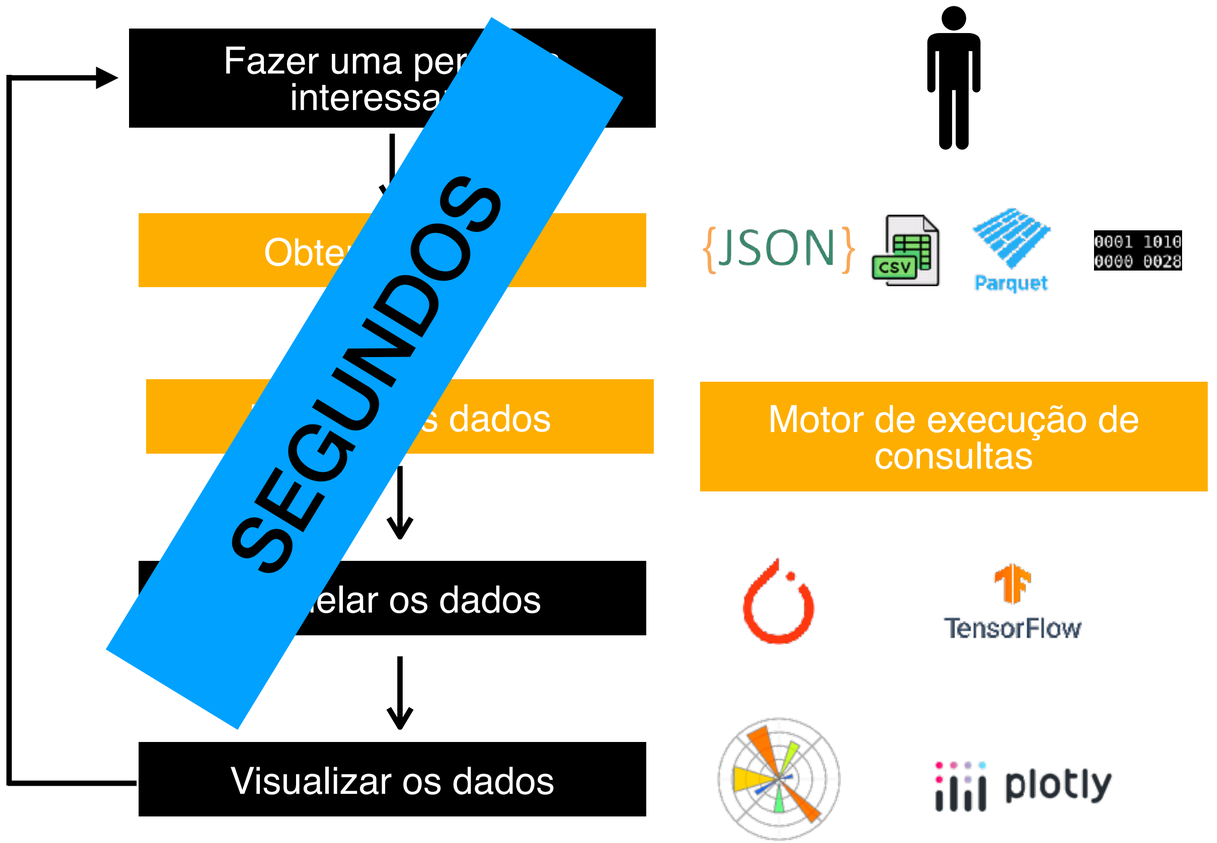
Análise de Dados



Análise de Dados



Análise de Dados



Motor de Execução de consultas

- **Ler** diferentes arquivos.
  - CSV, JSON, Parquet, Binários
- **Integrar** com outras ferramentas.
  - Gerar imagens
  - Aprendizado de máquina
- **Fácil de usar.**
- **Alta performance (Usuário no loop)**
  - Execução além da memória
  - Paralelismo
  - Otimização de consulta

Obviamente é um SGBD.

ORACLE

monetdb


pandas
APACHE
Spark™

Aplicação

```
import psycopg2
con = psycopg2.connect(
  host="1.2.3.4",
  port=8000,
  user="my_user",
  password="my_password",
  db_name="my_database")
cur = con.cursor()
cur.execute("SELECT ...")
```

Protocolo de comunicação

Servidor

Aplicação

```
import psycopg2
con = psycopg2.connect(
  host="1.2.3.4",
  port=8000,
  user="my_user",
  password="my_password",
  db_name="my_database")
cur = con.cursor()
cur.execute("SELECT ...")
```

Configuração do servidor

Protocolo de comunicação

Servidor

Aplicação

```
import psycopg2
con = psycopg2.connect(
  host="1.2.3.4",
  port=8000,
  user="my_user",
  password="my_password",
  db_name="my_database")
cur = con.cursor()
cur.execute("SELECT ...")
```

Configuração do servidor

Protocolo de comunicação

Transferência de dados.

Servidor

Aplicação

```
import psycopg2
con = psycopg2.connect(
  host="1.2.3.4",
  port=8000,
  user="my_user",
  password="my_password",
  db_name="my_database")
cur = con.cursor()
cur.execute("SELECT ...")
```

Configuração do servidor

Protocolo de comunicação

Transferência de dados.

Servidor

{JSON} 

0001 1010
0000 0028

Falta de suporte

pandas

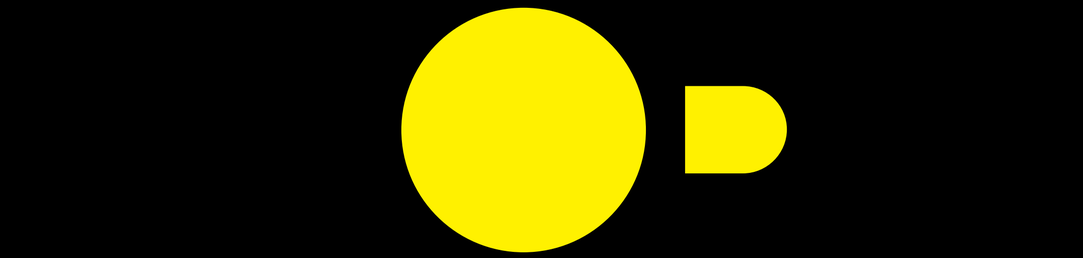
APACHE Spark™

Experiência
do usuário

Motor de execução
de consultas


MacBook Pro

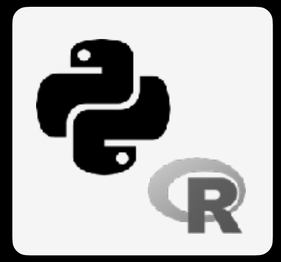
DuckDB



Embarcado

Rapido

SQL

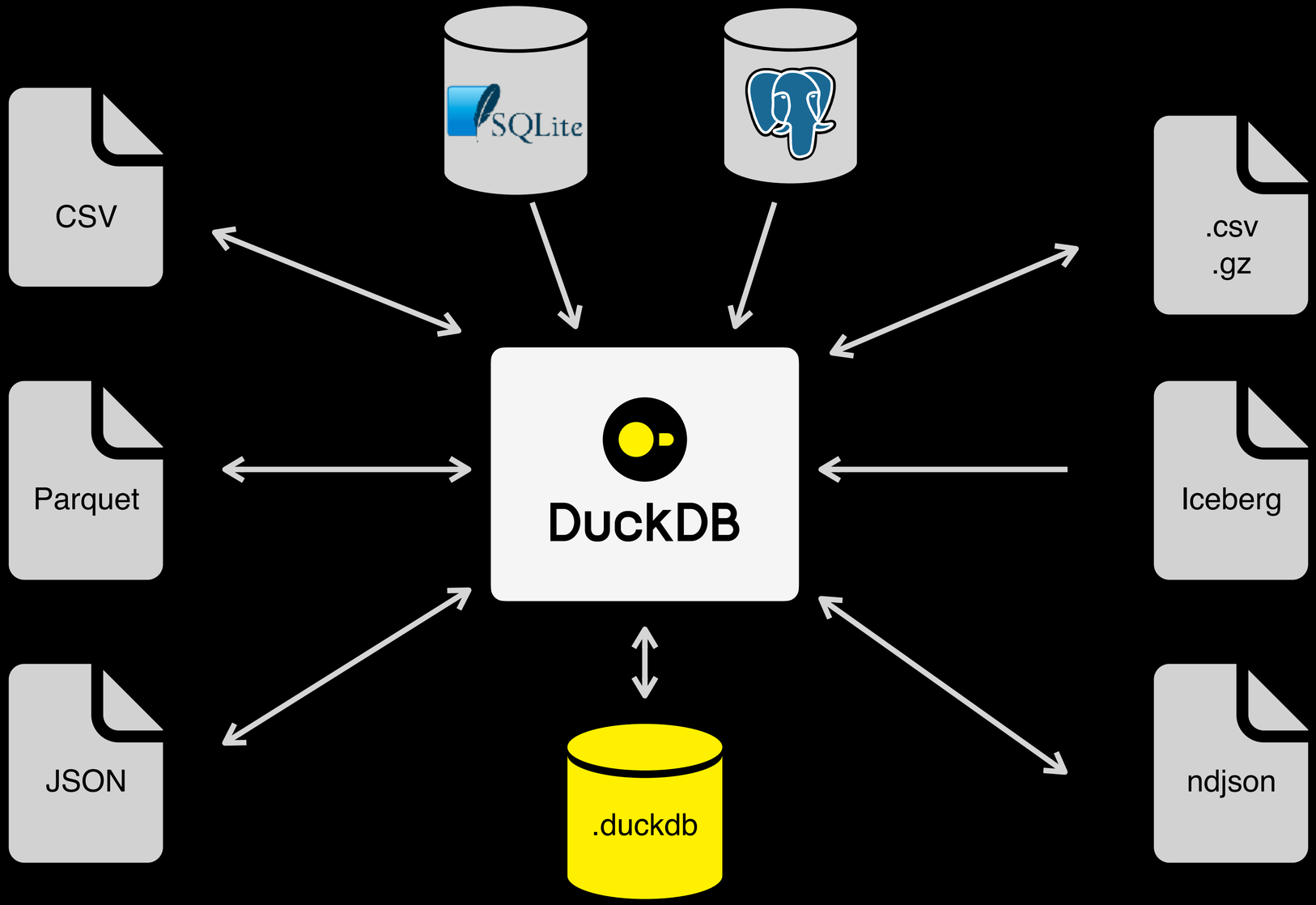


Python/R

Gratuito

Código Aberto

```python
import duckdb

con = duckdb.connect("my.db")
con.sql("SELECT ...")
```

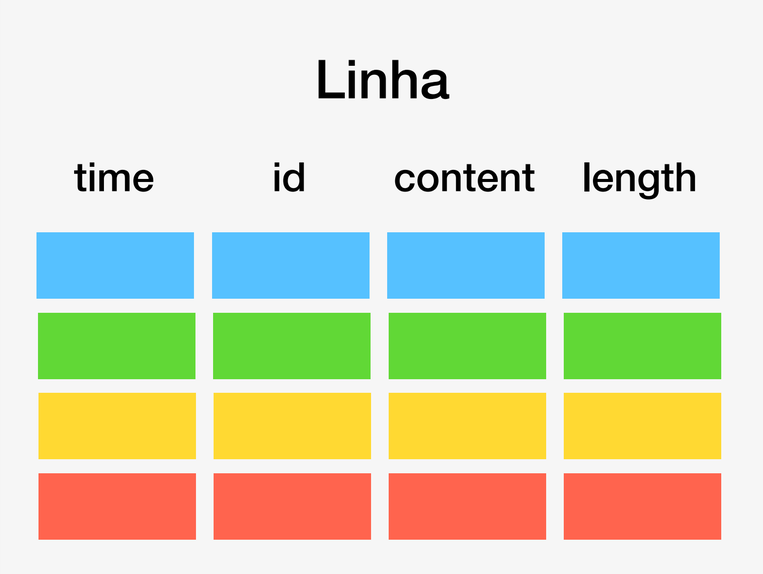

SQLite
CSV
.csv
.gz
Parquet
DuckDB
Iceberg
JSON
.duckdb
ndjson

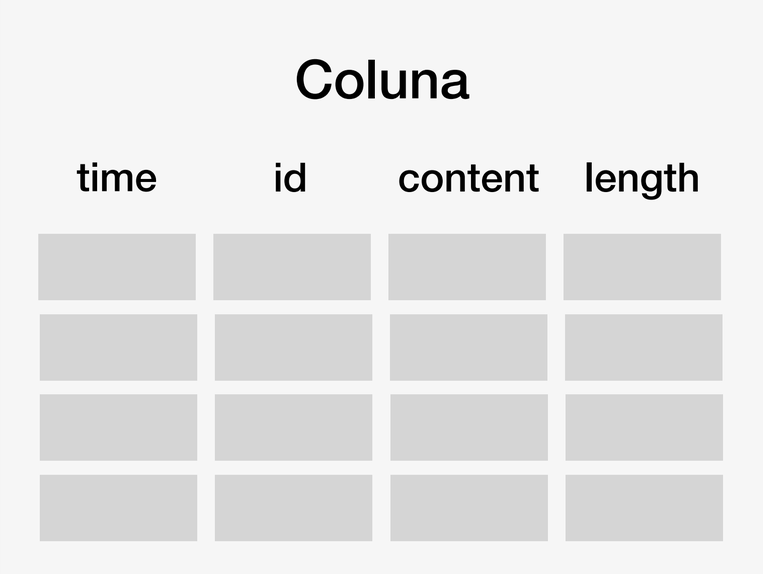
```
for table
 in ["Person", “Comment”,…]:
  con.sql(f"""
   CREATE TABLE {table} AS
     SELECT *
     FROM ‘{dir}/*/{tbl}/*.csv';
        """)
```

Tamanho CSV	Tempo (s)	Tamanho .db
2.4 GB	8 s	0.6 GB
26 GB	20 s	6 GB
253 GB	221 s	62 GB
2.6 TB	2701 s	624 GB

Design

Tipos de Armazenamento



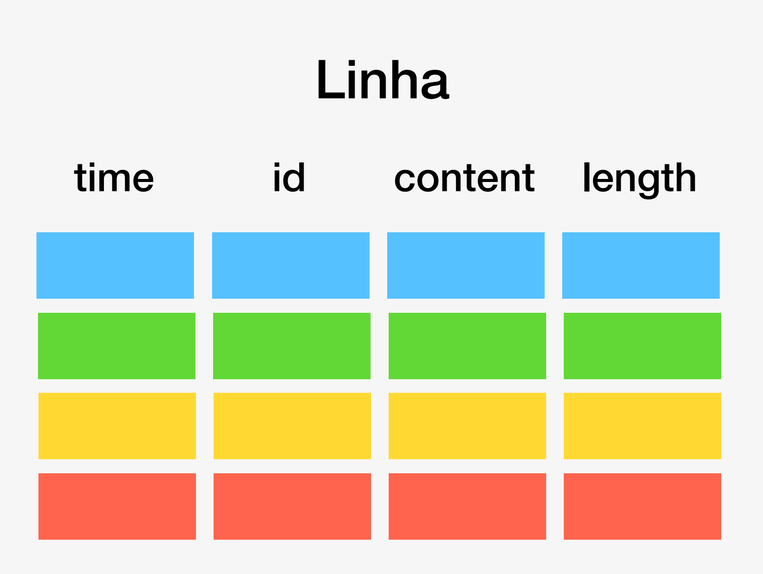


Leituras pequenas

Inserções e updates

Baixa utilização de memória

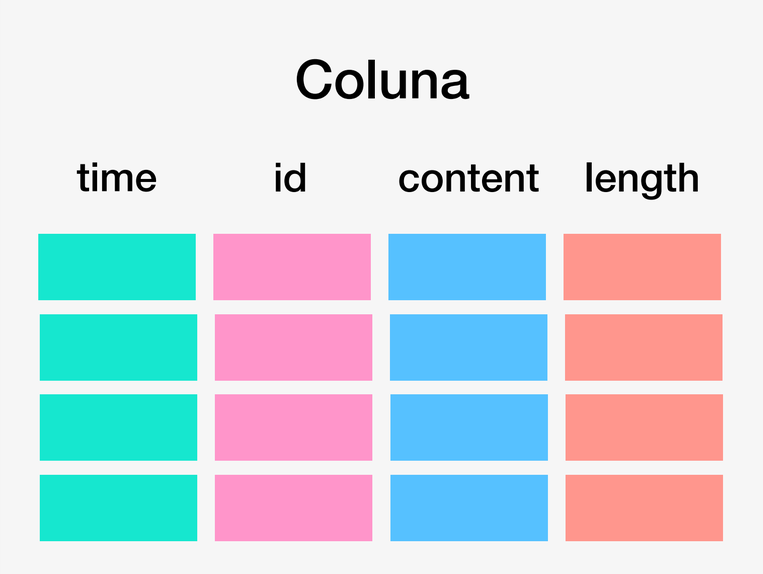
Tipos de Armazenamento (Colunar = Análise de Dados)



Leituras pequenas

Inserções e updates

Baixa utilização de memória



Compressão

Leituras maiores

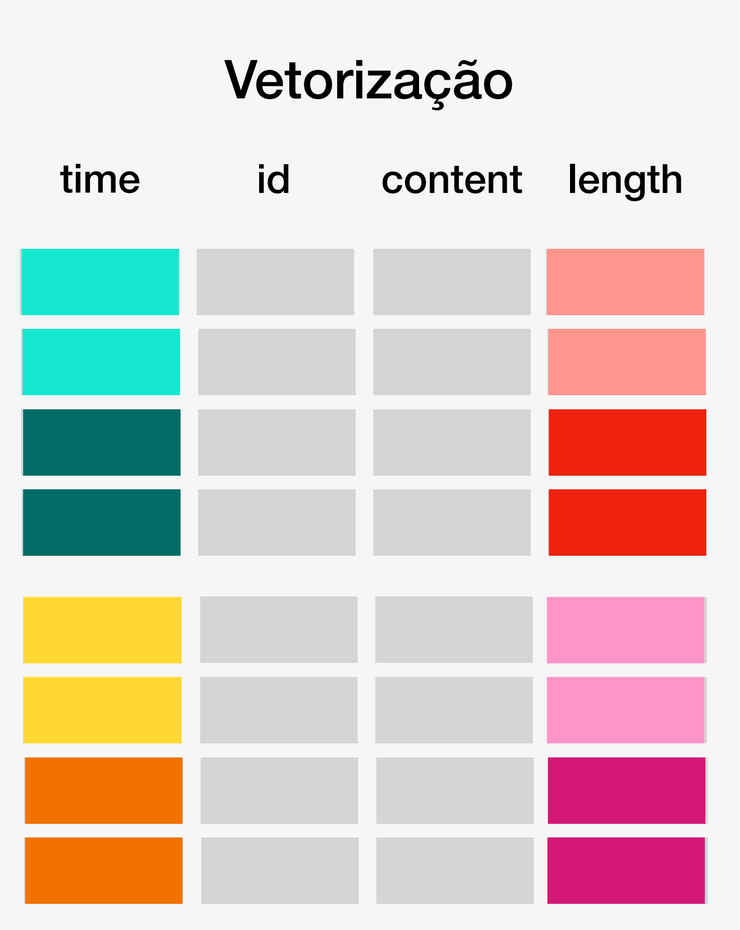
Foco em inserções grandes

Leitura de algumas colunas

Compressão

Versão DuckDB	Taxi	Ratio	Lineitem	Ratio	Compressões Adicionadas	Data
0.2.8	15.3 GB	1	0.85 GB	1	None	07/21
0.2.9	11.2 GB	1.36x	0.79 GB	1.07x	RLE + Constant	09/21
0.3.2	10.8 GB	1.41x	0.56 GB	1.51x	Bitpacking	02/22
0.3.3	6.9 GB	2.21x	0.32 GB	2.64x	Dictionary	24/22
0.5.0	6.6 GB	2.31x	0.29 GB	2.93x	For	09/22
0.6	4.8GB	3.18x	0.17 GB	5x	FSST + CHIMP	11/22

Execução Vetorizada e Paralelização



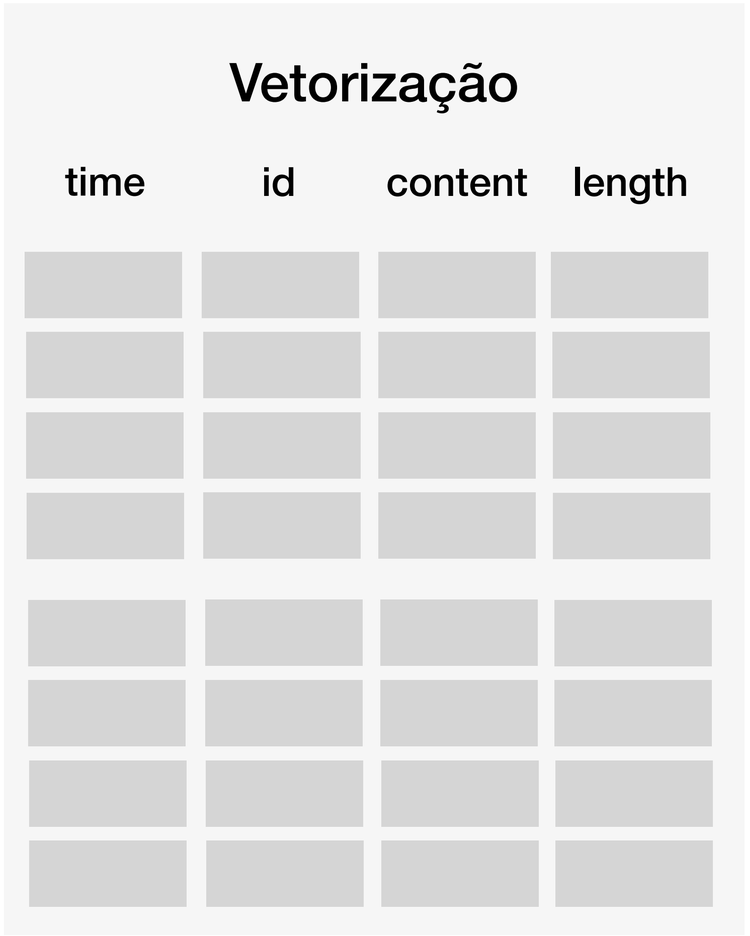
Thread 1

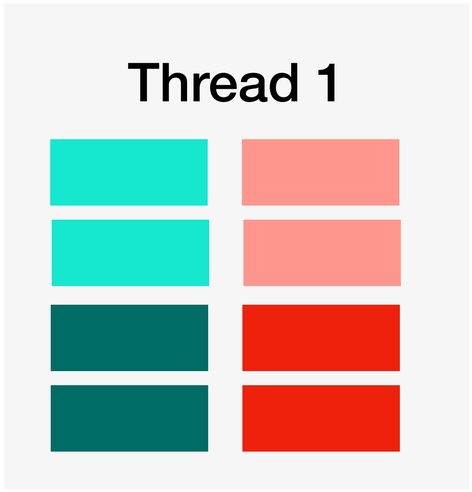
L1 cache

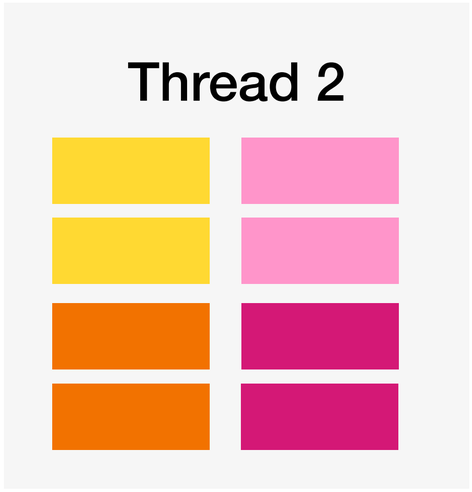
Thread 2

L1 cache

Execução Vetorizada e Paralelização



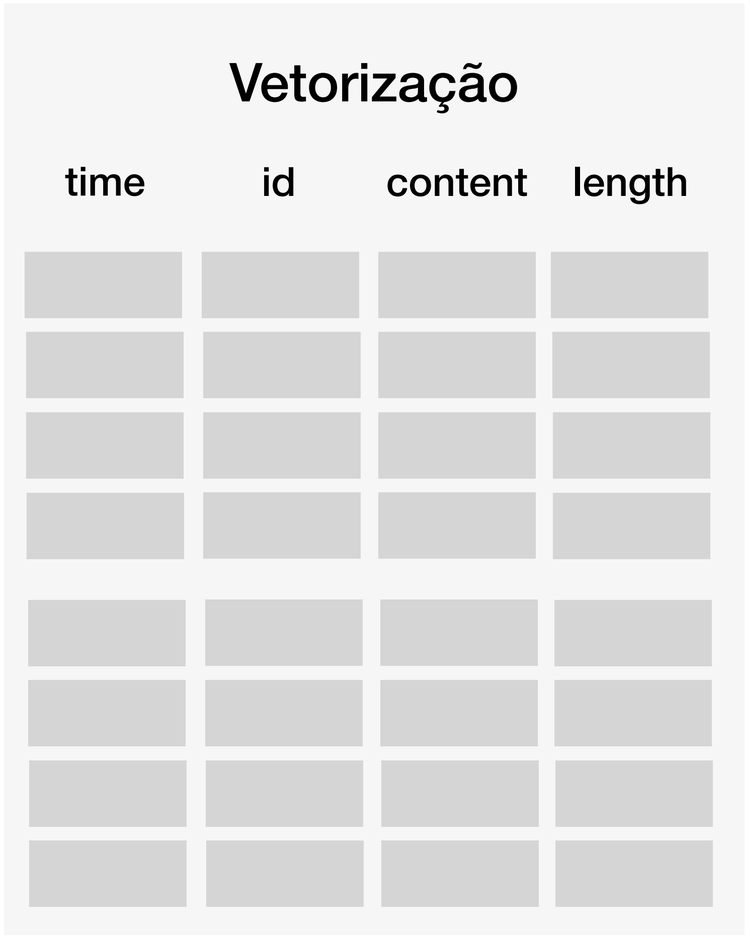


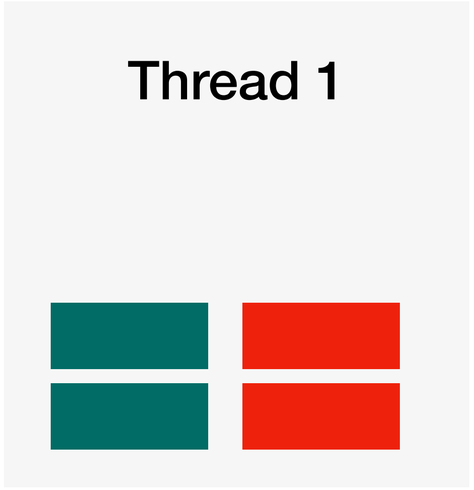


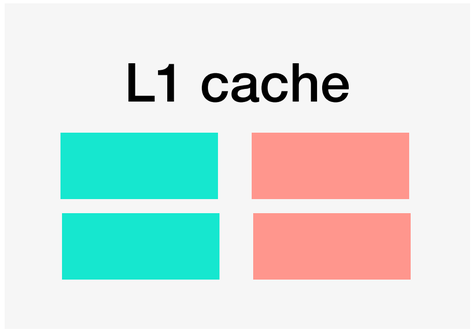
L1 cache

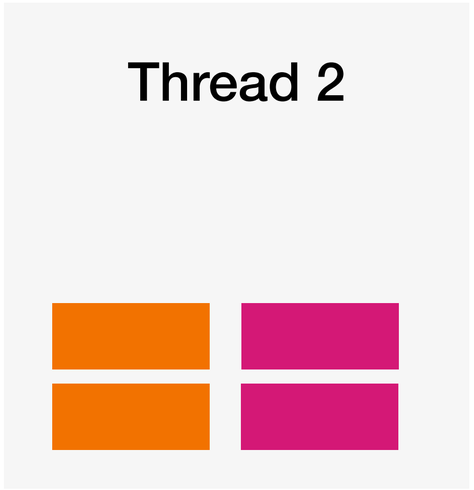
L1 cache

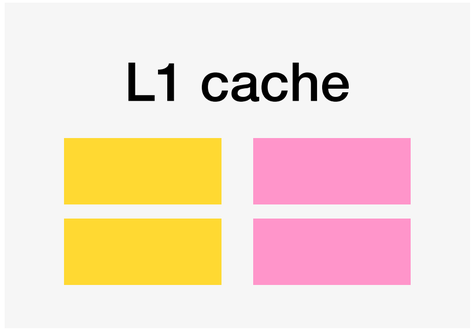
Execução Vetorizada e Paralelização











Execução Vetorizada e Paralelização

Vetorização

time	id	content	length

Thread 1

Thread 2

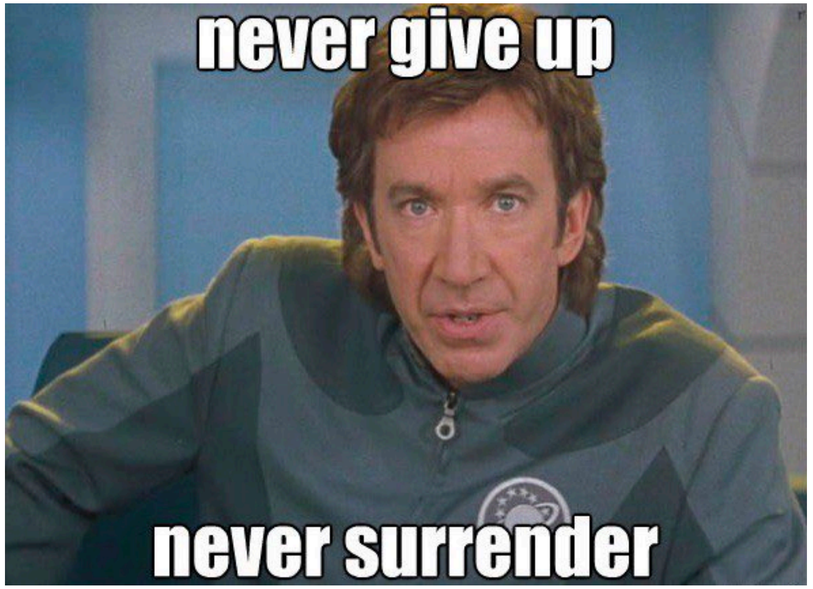
L1 cache

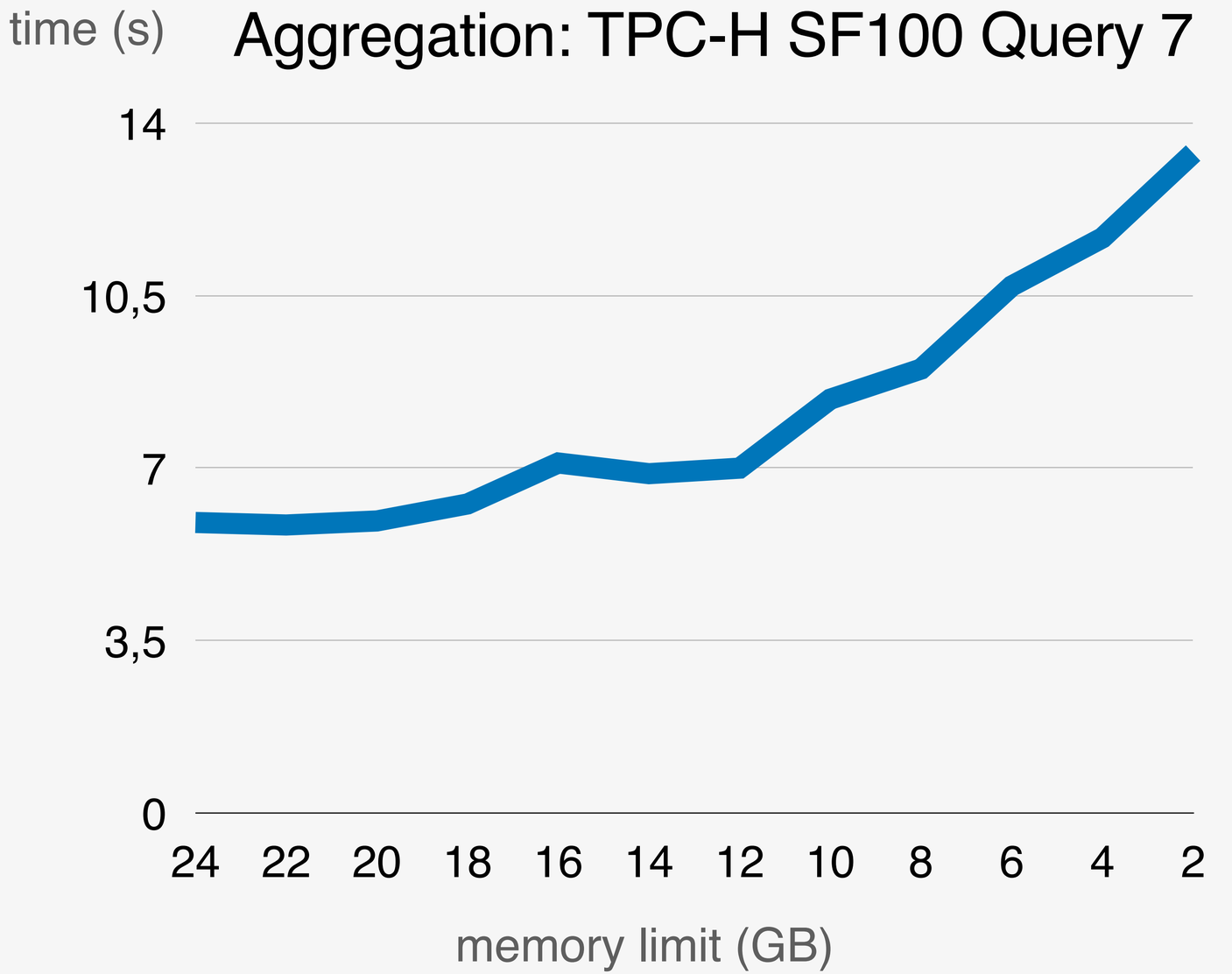
L1 cache

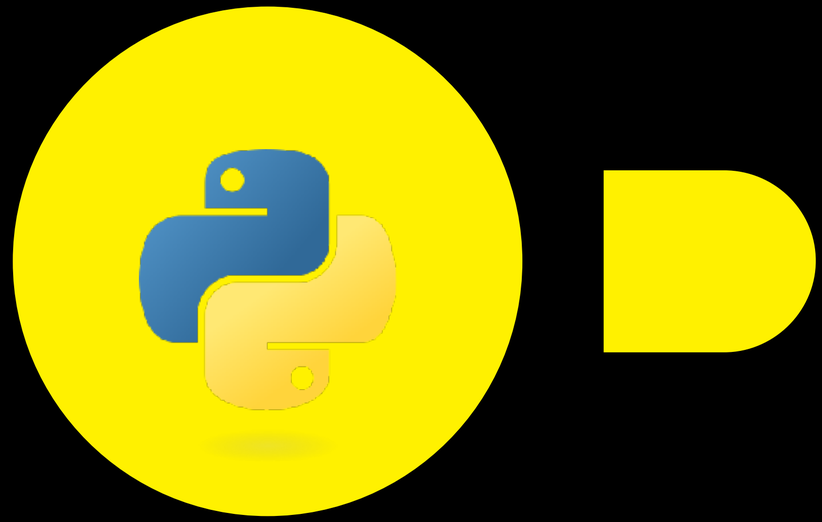
- Vetores cabem em L1 (32–128kB)
- SIMD (AVX-512: 8×64-bit ints)
- Compiladores fazem SIMD automático
- Paralelização em grupos de linhas

Execução em dados maiores que a da memória

- Todos os operadores são suportados: Junções, Agregações, Ordenações, Funções de Janela.
- Degradação graciosa.
- Objetivo é nunca falhar!







Python

Hello-World DuckDB!

```
[1] # !pip install duckdb
    import duckdb

    duckdb.sql("SELECT 42").fetchall()
```

```
[(42,)]
```

Dataframe ⇔ DuckDB em um a linha

```
[1] # !pip install duckdb
    import duckdb

    duckdb.sql("SELECT 42").fetchall()
```

```
[(42,)]
```

```
[3] # !pip install pandas
    import pandas as pd
    my_df = pd.DataFrame([{"c1":"duck","c2":"duck","c3":"goose"}])

    duckdb.sql("SELECT * FROM my_df").df()
```

	c1	c2	c3
0	duck	duck	goose

Python User Defined Functions (UDFs)

```
[1]  import duckdb
     from duckdb.typing import *

     # !pip install pyarrow
     import pyarrow as pa
     import pyarrow.compute as pc
```

```
[2]  def swap_case(x):
       return pc.utf8_swapcase(x)
```

```
[3]  con = duckdb.connect()
     con.create_function('swap_case', swap_case, [VARCHAR], VARCHAR, type='arrow')
     con.sql("SELECT swap_case('duck duck GOOSE')").fetchall()
```

```
     [('DUCK DUCK goose',)]
```

Python API Relacional

```
[18] from io import StringIO
     in_memory_csv = StringIO("joke,answer,laughs\nWhat do you call a bird that can f
     rel = con.read_csv(in_memory_csv)
```

```
[19] (rel.project('joke','laughs')
         .filter('laughs > 0')
         .aggregate('joke, sum(laughs)')
         .order('2')
     )
```

joke varchar	sum(laughs) int128
What do you call an under cover duck?	1
What do you call a bird that can fix anything?	2
What do you call a duck burglar?	5

Python API Relacional

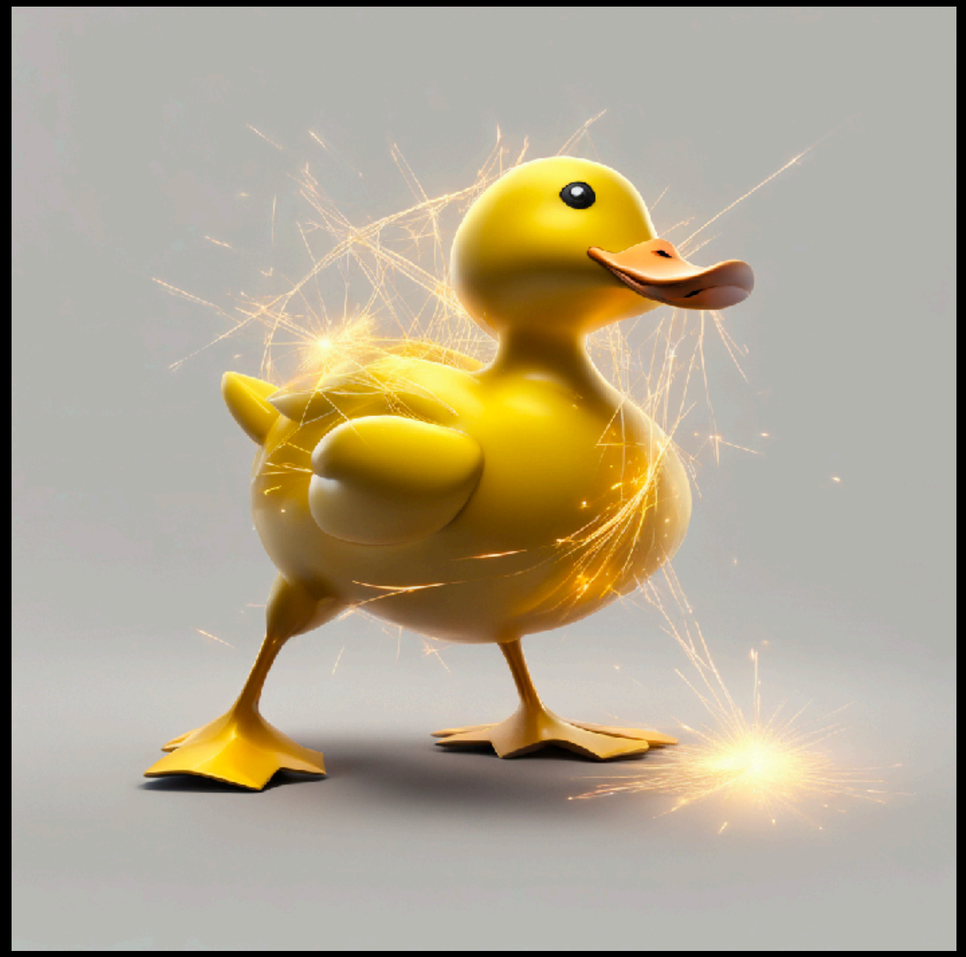
```
[18] from io import StringIO
     in_memory_csv = StringIO("joke,answ
     rel = con.read_csv(in_memory_csv)
```

```
[19] (rel.project('joke','laughs')
         .filter('laughs > 0')
         .aggregate('joke, sum(laughs)')
         .order('2')
     )
```

Benefícios da API Relacional

- Combinável
- Possível inspecionar resultados intermediários
- Mesmo otimizador de consulta do SQL.

joke varchar	sum(laughs) int128
What do you call an under cover duck?	1
What do you call a bird that can fix anything?	2
What do you call a duck burglar?	5



PySpark API

DuckDB - Spark API

```
[22] from duckdb.experimental.spark.sql import SparkSession as session
     from duckdb.experimental.spark.sql.functions import lit, col

     spark = session.builder.getOrCreate()
```

```
[23] spark_df = spark.createDataFrame(pandas_df)

     (spark_df
       .groupBy(col('joke'))
       .sum('laughs')
       .sort('sum(laughs)')
     )
```

joke varchar	sum(laughs) int128
What do you call an under cover duck?	1
What do you call a bird that can fix anything?	2
What do you call a duck burglar?	5

DuckDB - Spark API

```
[22] from duckdb.experimental.spark.sql import SparkSession as session
     from duckdb.experimental.spark.sql.functions import lit, col

     spark = session.builder.getOrCreate()
```

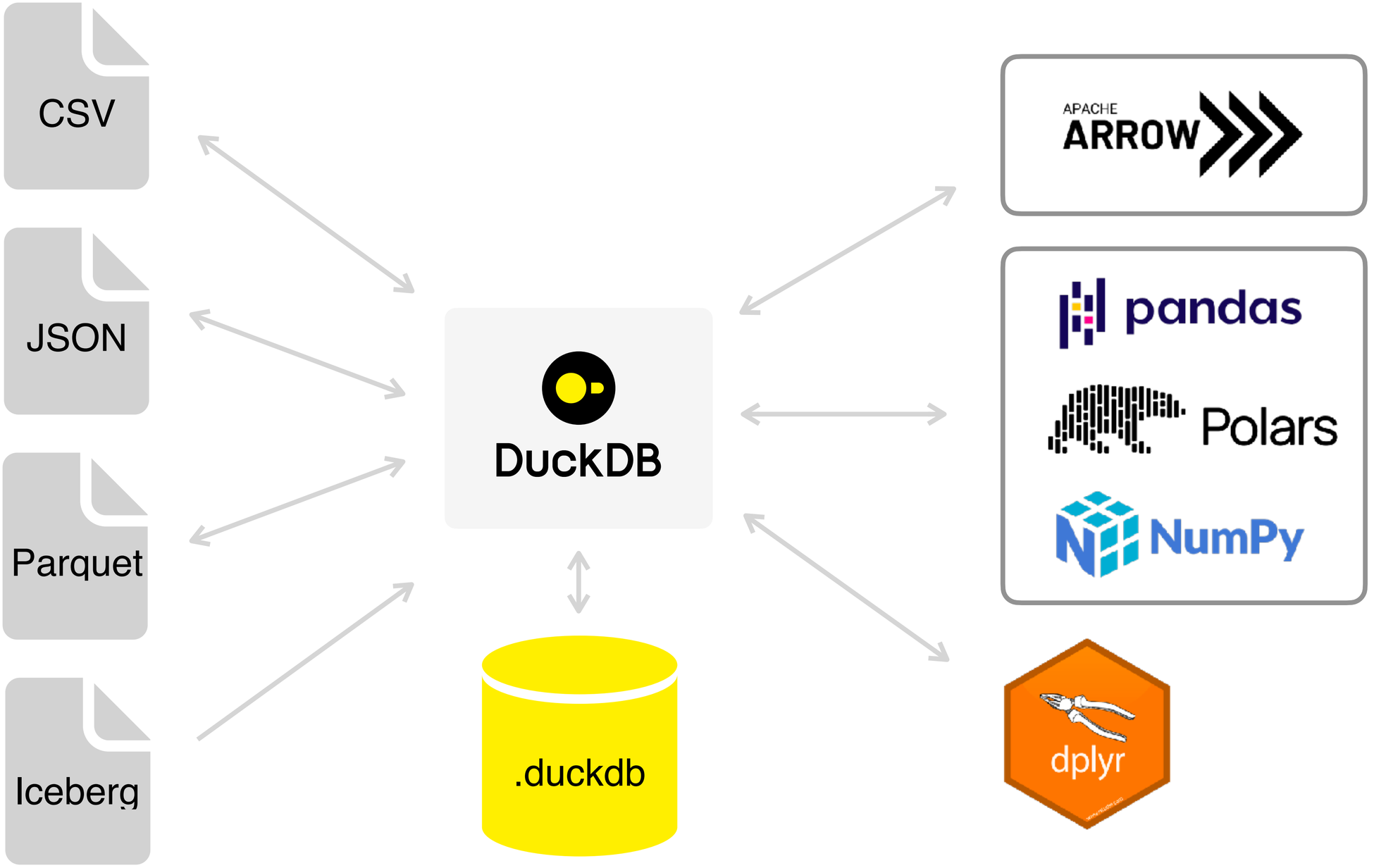
```
[23] spark_df = spark.createDataFrame(pandas_df)

     (spark_df
       .groupBy(col('joke'))
       .sum('laughs')
       .sort('sum(laughs)')
     )
```

joke varchar
What do you call an under cover duck?
What do you call a bird that can fix anything?
What do you call a duck burglar?

Benefícios

- 1:1 com a API do Spark
- Sem gerenciamento de cluster
- Inicialização imediata
- Escrito em Python

Input and output formats



Python Ecosystem Integrations

Formatos de Dados

NumPy
pandas
APACHE ARROW
polars
TensorFlow
PyTorch

Engenharia de dados

dbt
Apache Airflow
dagster
Airbyte

APIs de dataframes

The Ibis Project
PySpark API
Siuba

Visualização

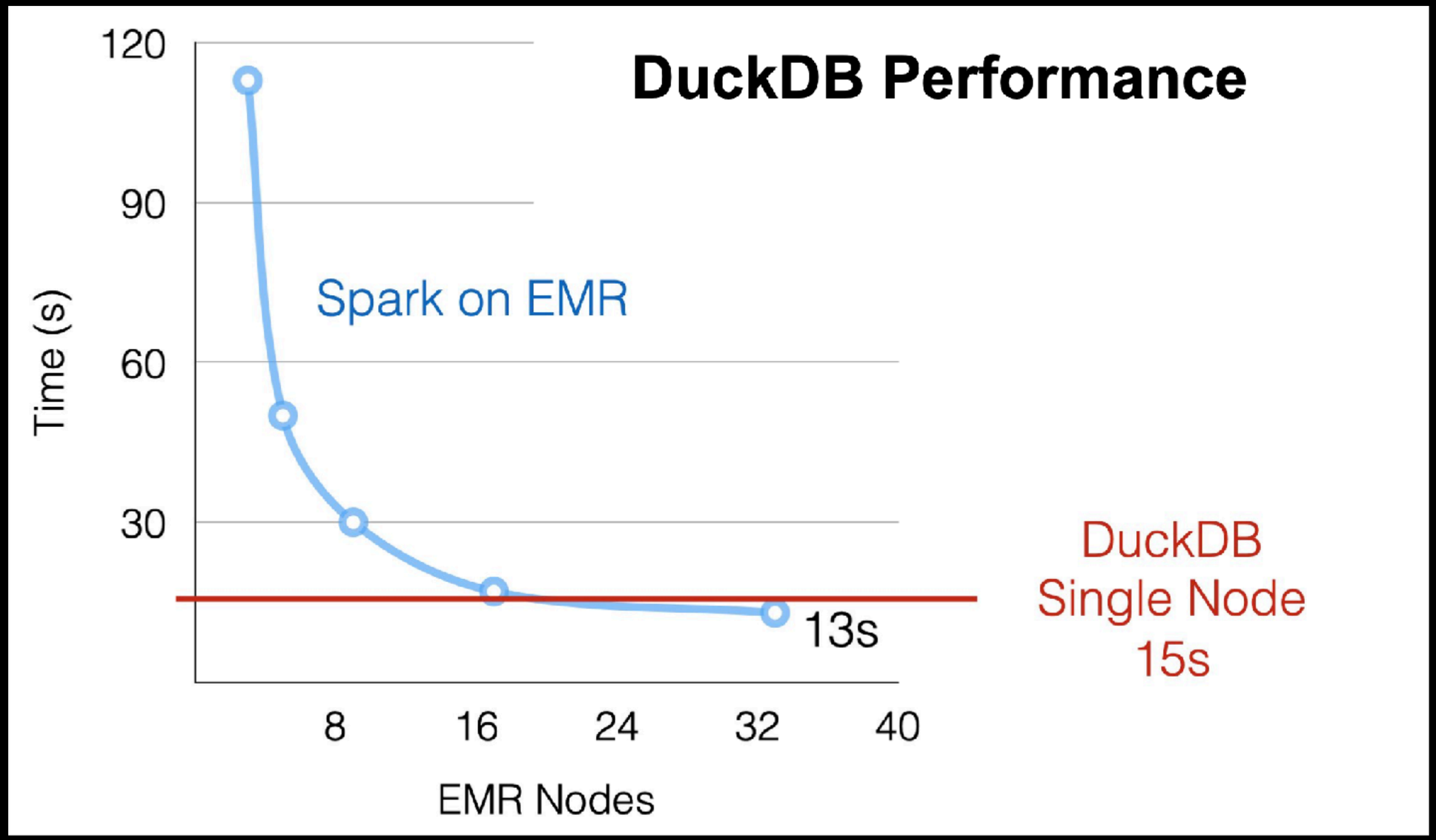
HEX
Mode
Apache Superset
Vega Fusion

Outros

SQLAlchemy
fsspec
JupySQL

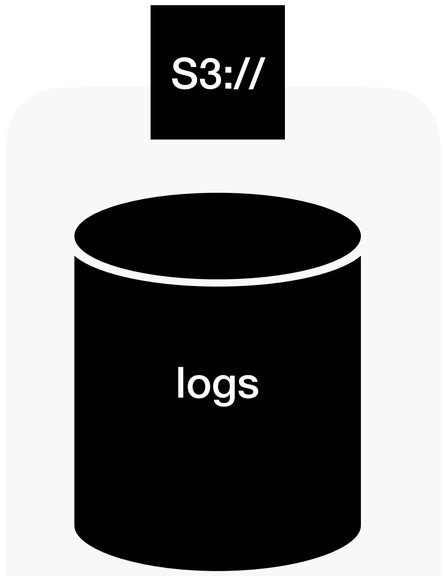
Demo

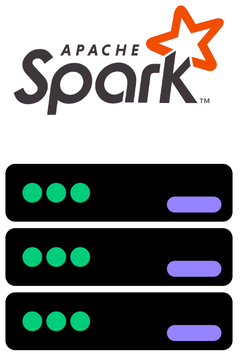
PySpark Vs DuckDB - Um nó

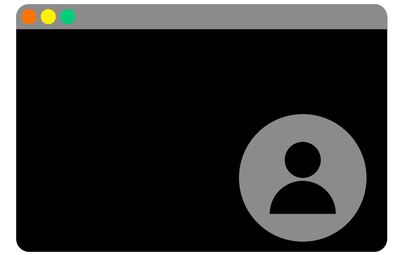

DuckDB Performance
120
90
60
30
Time (s)
Spark on EMR
13s
DuckDB
Single Node
15s
8
16
24
32
40
EMR Nodes

BIG Data

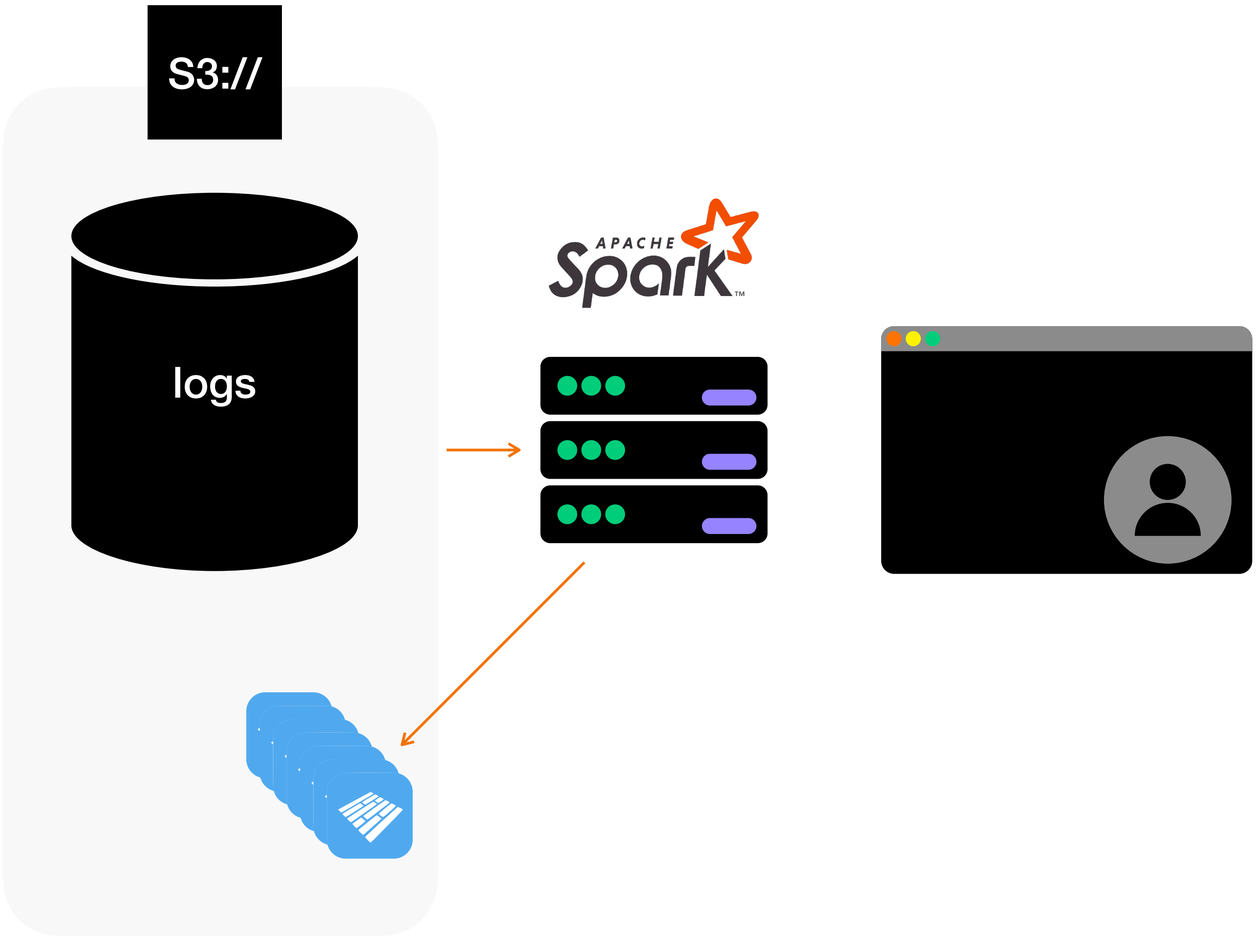
Data Lake



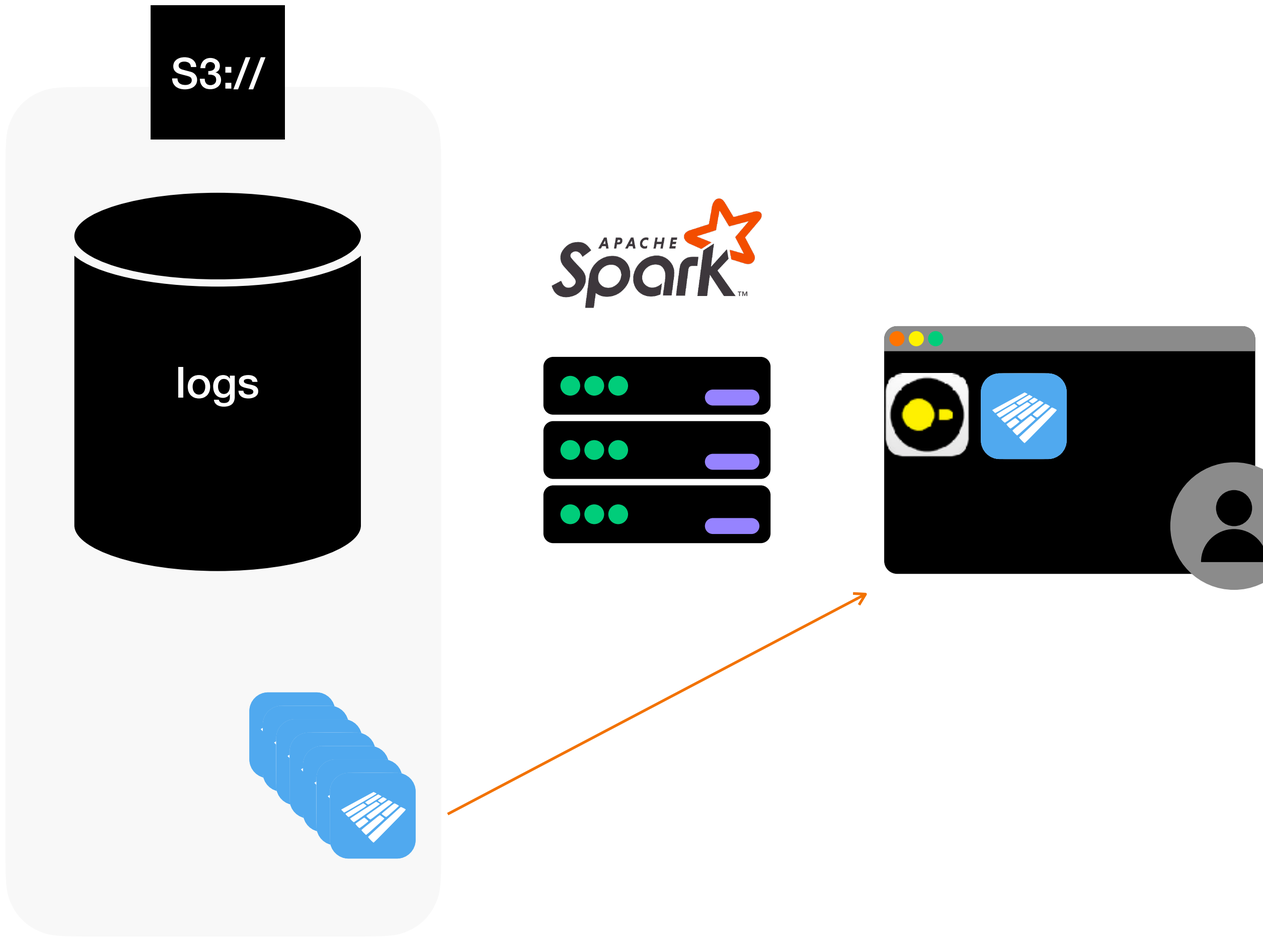




ETL



Exploração e Análise



Demo

Spark (Databricks) + DuckDB