DuckDB as backend to build optimization models in JuMP.jl

JuMP-dev 2026

Abel Soares Siqueira - Netherlands eScience Center
Diego A. Tejada Arango - TNO

netherlands eScience center

Summary

- Linearise indices by storing them as tables
- Separate indices creation from model creation
- Use DuckDB to facilitate data manipulation

netherlands eScience center

TulipaEnergyModel.jl circa Dec 2023

- Energy System Optimization
- Time blocks instead of time steps: `1:2, 3:6, 7:7, 8:9`, etc.
- Each flow is defined on a time block:

  ```
  @variable(_, flow[e in edges, timeblocks[e]])
  ```

- Each balance equation happens for another time block:

  ```
  @constraint(_, cons[a in assets, timeblocks[a]], _)
  ```

- `flow[e, block_var] * intersect(block_var, block_cons)`

Issues

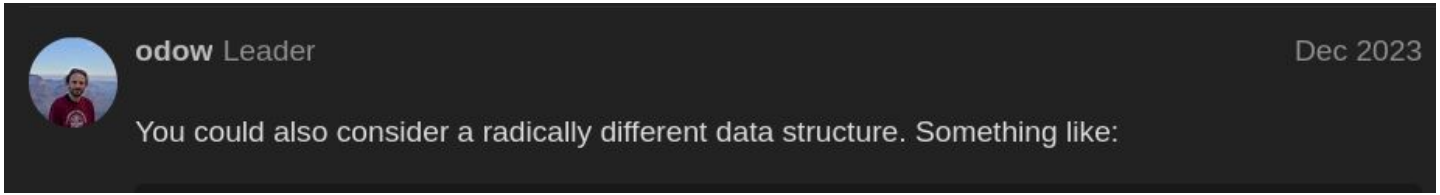
- Sparse containers
- “Sum-if problem”

netherlands eScience center

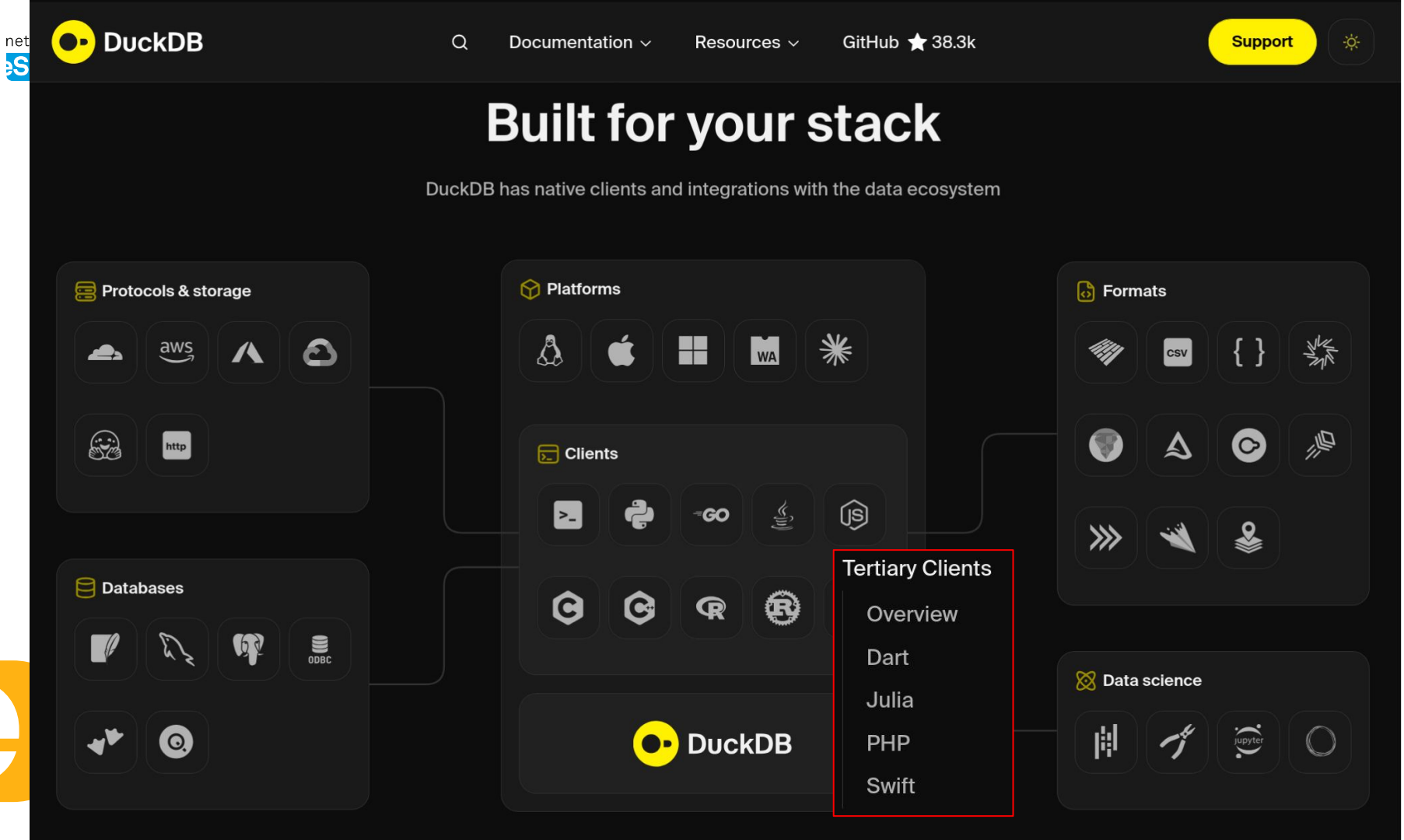
Issues

- Sparse containers
- “Sum-if problem”

Solution

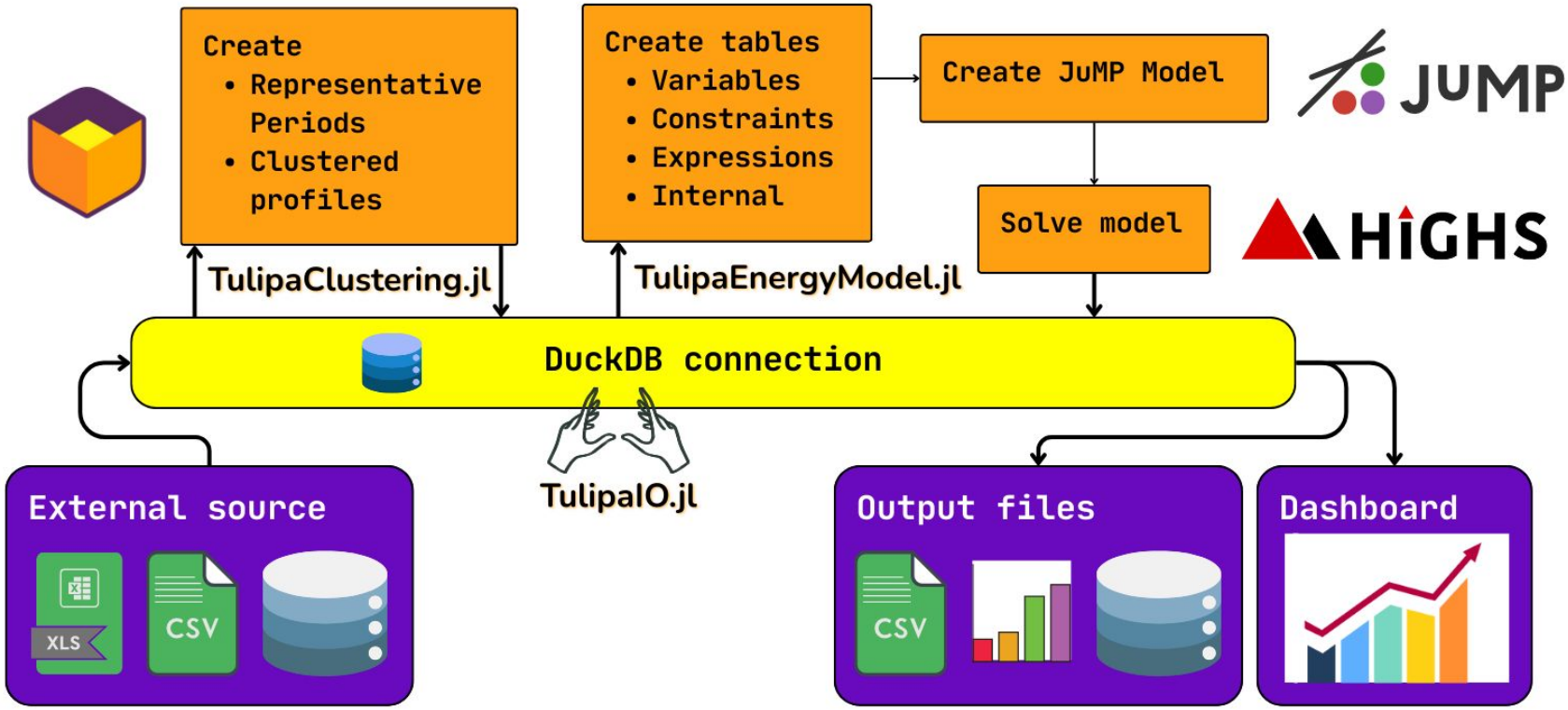


- Linearized indices
- Separate indices creation from model creation


DuckDB
Documentation
Resources
GitHub 38.3k
Support
Built for your stack
DuckDB has native clients and integrations with the data ecosystem
Protocols & storage
aws
http
Platforms
WA
Formats
CSV
Clients
GO
Databases
ODBC
DuckDB
Tertiary Clients
Overview
Dart
Julia
PHP
Swift
Data science
jupyter

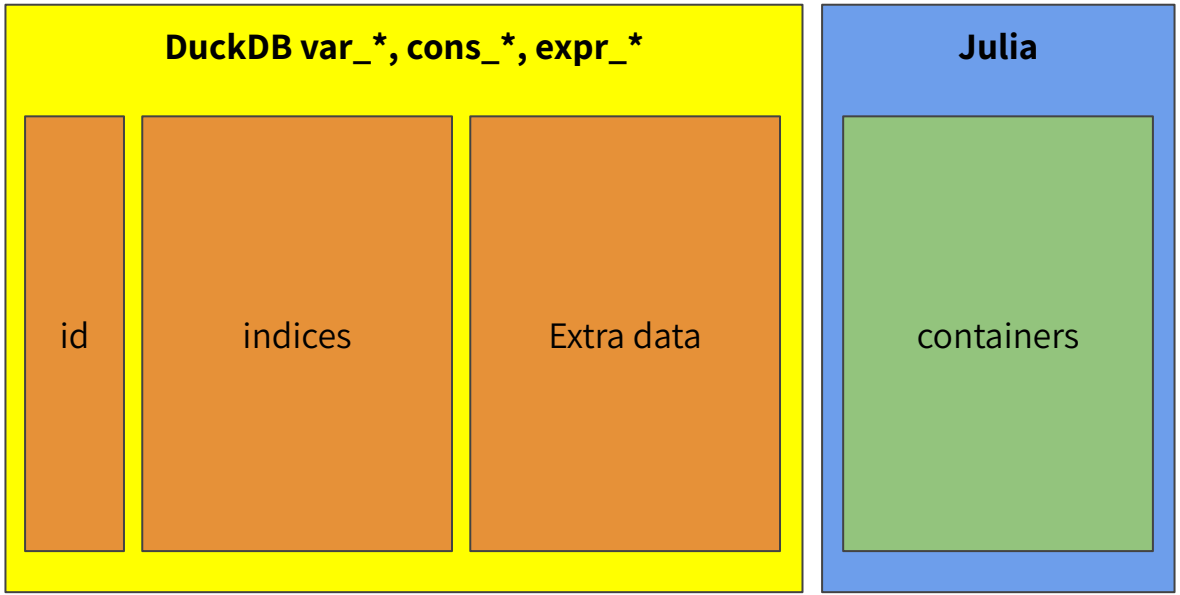
netherlands eScience center

Tulipa ecosystem overview



netherlands eScience center

Indices and containers



netherlands eScience center

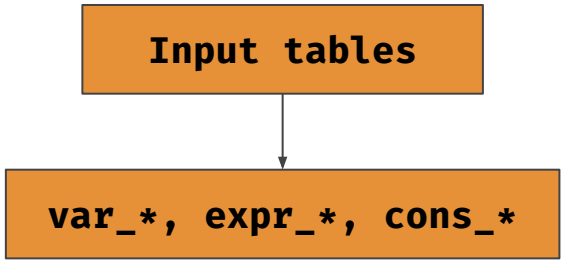
Pipeline

Input tables

```
-- SQL

CREATE OR REPLACE edges AS FROM
read_csv('edges.csv');

CREATE OR REPLACE edge_resolution AS FROM
read_csv('edge_resolution.csv');

…
```

netherlands eScience center

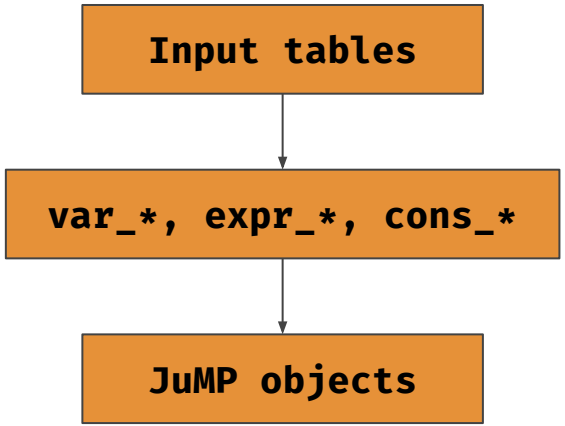
Pipeline




```
-- SQL

CREATE SEQUENCE var_id START 1;

CREATE OR REPLACE var_flow AS (
  SELECT
    nextval('var_id') AS id,
    edges.src_node_id AS src_node_id,
    edges.dst_node_id AS dst_node_id,
    … AS time_block_start
  FROM edges
  LEFT JOIN edge_resolution
    …
);
…

CREATE OR REPLACE cons_balance AS (
  SELECT
    …
    ARRAY_AGG(...) AS incoming_ids,
    …
  FROM …
);
```

netherlands eScience center

Pipeline

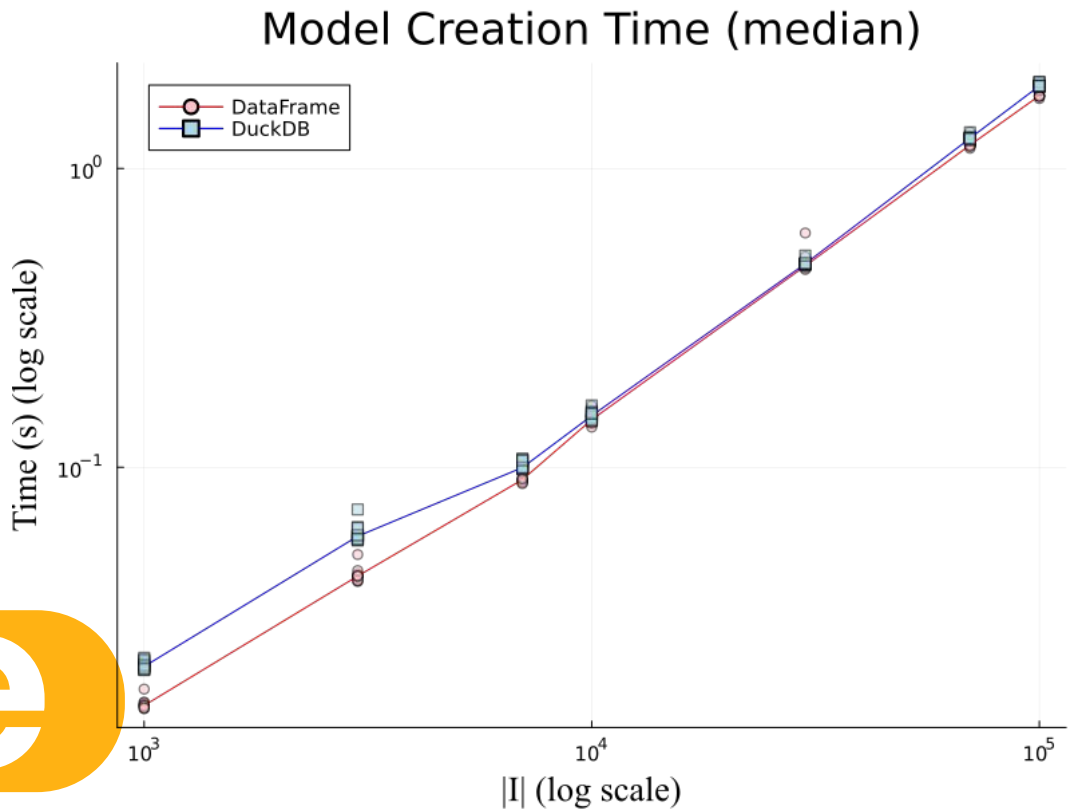


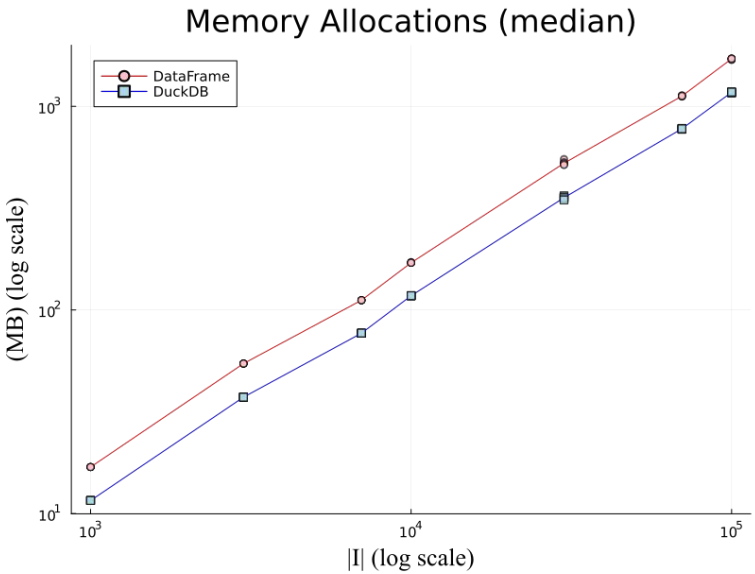

```julia
# Julia

sql(s) = DuckDB.query(connection, s)

var_flow = model[:var_flow] = [
  @variable(
    model,
    lower_bound = 0.0,
    base_name = "flow[$(row.id)]",
  )
  for row in sql("FROM var_flow")
]

model[:cons_balance] = [
  @constraint(
    model,
    sum(
      var_flow[id] for id in row.incoming_ids
    ) == sum(
      var_flow[id] for id in row.outgoing_ids
    )
  )
  for row in sql(“FROM cons_balance”)
]
```

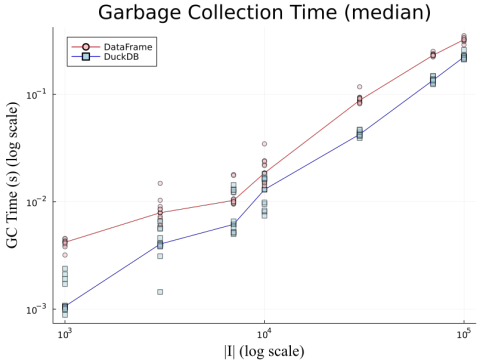
netherlands eScience center

https://github.com/TulipaEnergy/2025-juliacon-poster

IJKLM problem







netherlands eScience center

Pros and cons

+ Model creation speed is similar to **DataFrames.jl** (same **Tables.jl** backend)
+ Data ops in SQL => less memory and faster
+ Easier to integrate in a more complex pipeline
+ Same indices can be used for other frameworks
- Requires SQL knowledge for modeling
- Loses JuMP indexing (but lookup can be created)
- DuckDB.jl has limited support and dynamically dispatches

netherlands eScience center

Thanks


GH: abelsiqueira


abel.siqueira@esciencecenter.nl