ADVANCED
DATABASE
SYSTEMS

DuckDB / MotherDuck

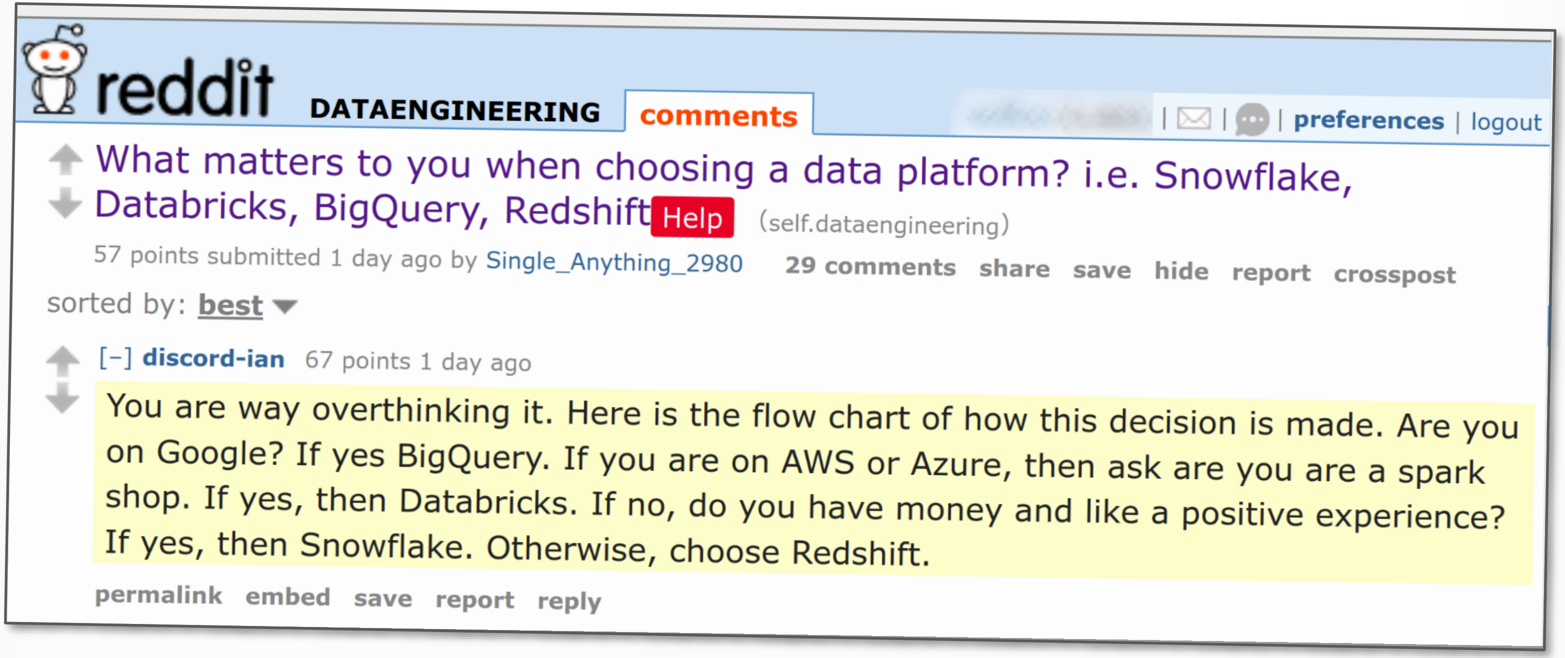
20

Andy Pavlo
CMU 15-721
Spring 2024

Carnegie Mellon University

LAST CLASS

Snowflake Data Warehouse



What matters to you when choosing a data platform? i.e. Snowflake, Databricks, BigQuery, Redshift Help (self.dataengineering)

57 points submitted 1 day ago by Single_Anything_2980 29 comments share save hide report crosspost

sorted by: best

[–] discord-ian 67 points 1 day ago

You are way overthinking it. Here is the flow chart of how this decision is made. Are you on Google? If yes BigQuery. If you are on AWS or Azure, then ask are you are a spark shop. If yes, then Databricks. If no, do you have money and like a positive experience? If yes, then Snowflake. Otherwise, choose Redshift.

permalink embed save report reply

HISTORICAL CONTEXT

CWI researchers recognized that data scientists do not use the full query capabilities of DBMSs due to the overhead of setting up and accessing data.

In 2017 they created an embedded version of MonetDB called MonetDBLite to run inside of R applications.

→ Running in-process reduces the cost of transferring data back and forth between the DBMS and the application.

But MonetDB had too much legacy baggage…

DUCKDB (2019)

Multi-threaded embedded (in-process, serverless) DBMS that executes SQL over disparate data files.

→ PostgreSQL-like dialect with quality-of-life enhancements.

→ ***"SQLite for Analytics"***

Provides zero-copy access to query results via Arrow to client code running in same process.

The core DBMS is nearly all custom C++ code with little to no third-party dependencies.

→ Relies on extensions ecosystem to expand capabilities.

DUCKDB: AN EMBEDDABLE ANALYTICAL DATABASE
SIGMOD 2019

DUCKDB

Shared-Everything

Push-based Vectorized Query Processing

Precompiled Primitives

Multi-Version Concurrency Control

Morsel Parallelism + Scheduling

PAX Columnar Storage

Sort-Merge + Hash Joins

Stratified Query Optimizer

→ Supports unnesting of arbitrary subqueries

DUCKDB: PUSH-BASED PROCESSING

System originally used pull-based vectorized query processing but found it unwieldly to expand to support more complex parallelism.

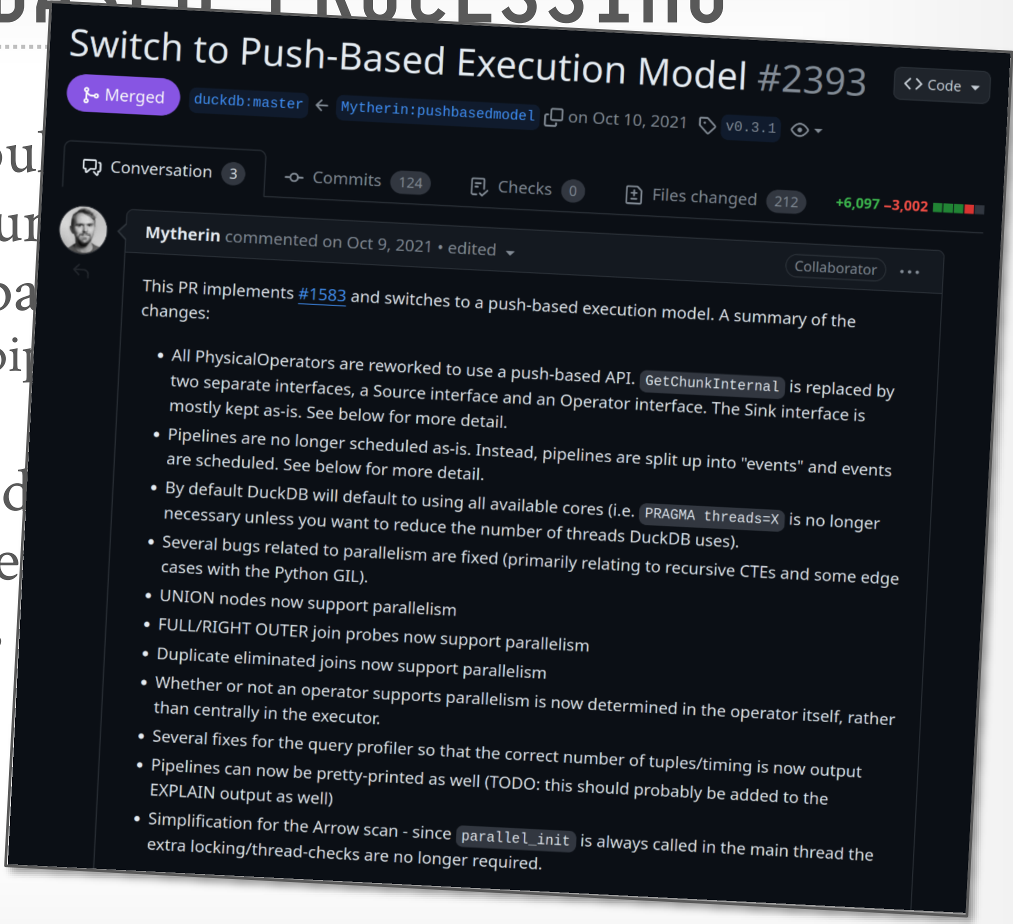
→ Cannot invoke multiple pipelines simultaneously.

Switched to a push-based query processing model in 2021. Each operator determines whether it will execute in parallel on its own instead of a centralized executor.

DUCKDB: PUSH-BASED PROCESSING

System originally used pu
processing but found it u
support more complex pa
→ Cannot invoke multiple pi

Switched to a push-based
2021. Each operator dete
execute in parallel on its
centralized executor.



Switch to Push-Based Execution Model #2393

duckdb:master ← Mytherin:pushbasedmodel on Oct 10, 2021 v0.3.1

Mytherin commented on Oct 9, 2021 • edited

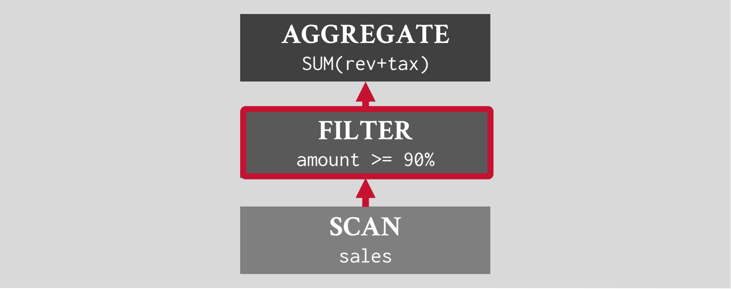
This PR implements #1583 and switches to a push-based execution model. A summary of the changes:

- All PhysicalOperators are reworked to use a push-based API. `GetChunkInternal` is replaced by two separate interfaces, a Source interface and an Operator interface. The Sink interface is mostly kept as-is. See below for more detail.
- Pipelines are no longer scheduled as-is. Instead, pipelines are split up into "events" and events are scheduled. See below for more detail.
- By default DuckDB will default to using all available cores (i.e. `PRAGMA threads=X` is no longer necessary unless you want to reduce the number of threads DuckDB uses).
- Several bugs related to parallelism are fixed (primarily relating to recursive CTEs and some edge cases with the Python GIL).
- UNION nodes now support parallelism
- FULL/RIGHT OUTER join probes now support parallelism
- Duplicate eliminated joins now support parallelism
- Whether or not an operator supports parallelism is now determined in the operator itself, rather than centrally in the executor.
- Several fixes for the query profiler so that the correct number of tuples/timing is now output
- Pipelines can now be pretty-printed as well (TODO: this should probably be added to the EXPLAIN output as well)
- Simplification for the Arrow scan - since `parallel_init` is always called in the main thread the extra locking/thread-checks are no longer required.

DUCKDB: FINE-GRAINED CONTROL

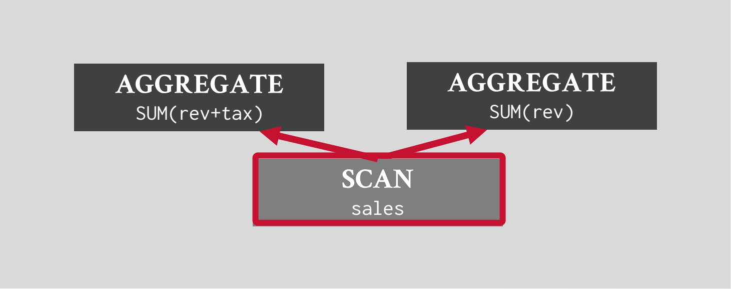
Vector Cache:

→ Buffer results between operators until it fills vector.



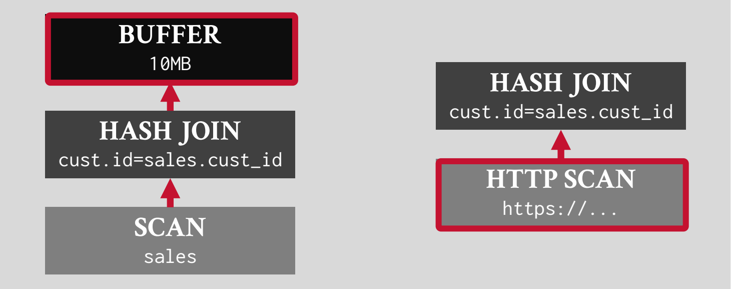
Scan Sharing:

→ Push results from one child operator to multiple parent operators (DAG plan).



Backpressure / Async IO

→ Pause operator execution when buffers are full or when waiting for remote I/O.

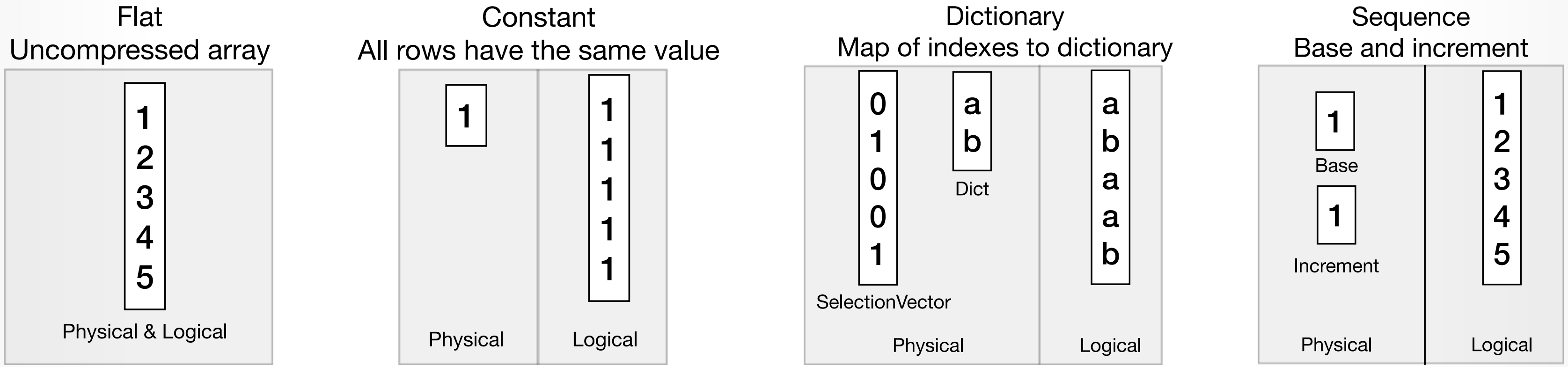


Source: Mark Raasveldt

DUCKDB: VECTORS

Custom internal vector layout for intermediate results that is compatible with Velox.

Supports multiple vector types:

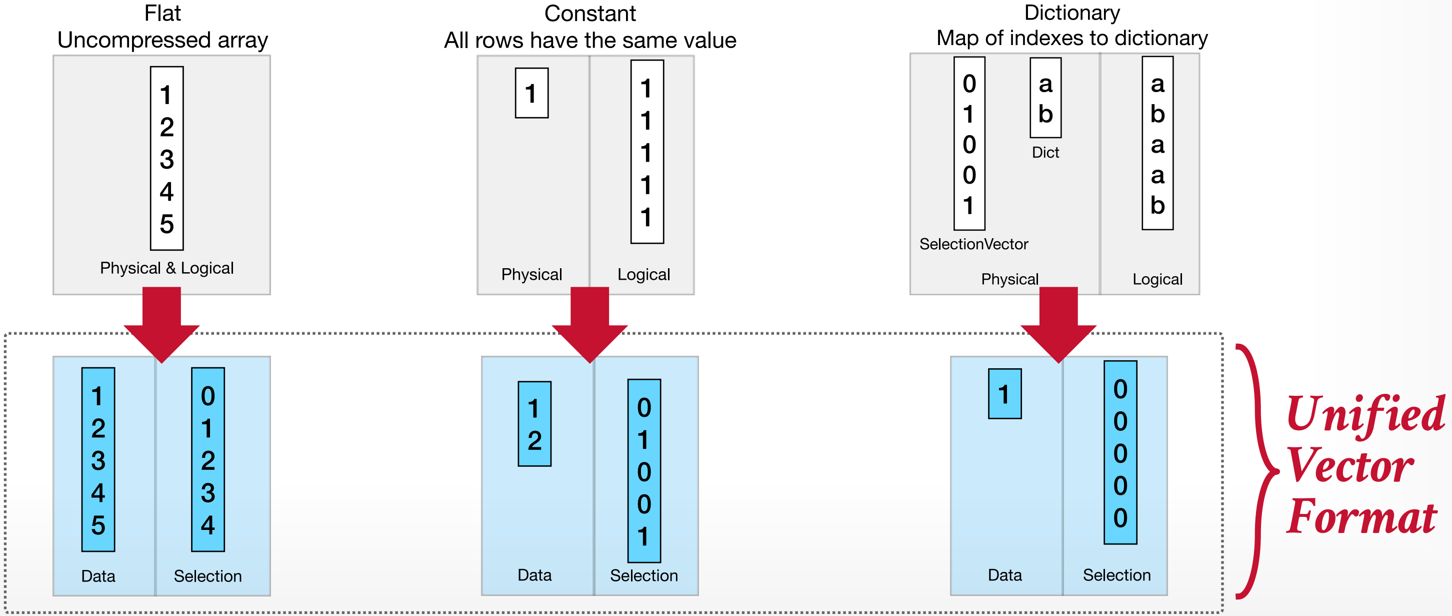


Source: Mark Raasveldt

DUCKDB: VECTORS

DuckDB uses a unified format to process all vector types without needing to decompress them first.

→ Reduce # of specialized primitives per vector type



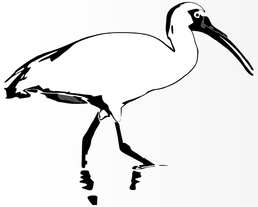
Source: Mark Raasveldt

DUCKDB: DATAFRAMES

DuckDB supports DataFrame libraries to query databases without using SQL.

→ **dpylr** (R-lang)

→ **Ibis** (Python)



Integration libraries generate DuckDB logical plans the DBMS converts into optimized physical plans.

→ Bypasses the SQL parser

Zero-copy result passing via Apache Arrow.

DUCKDB: STORAGE FORMAT

DBMS's built-in storage format maintains a single PAX-oriented file per database.

→ Splits tables into row groups with 120k tuples.

→ On-disk encoding is different than in-memory representation.

Two phase compression scheme:

→ **Analyze**: Sample a small portion of a column to determine the best encoding scheme

→ **Compress:** Encode the values and write it to disk.

DUCKDB: STORAGE FORMAT

DBMS's built-in s format maintains a single PAX-oriented file

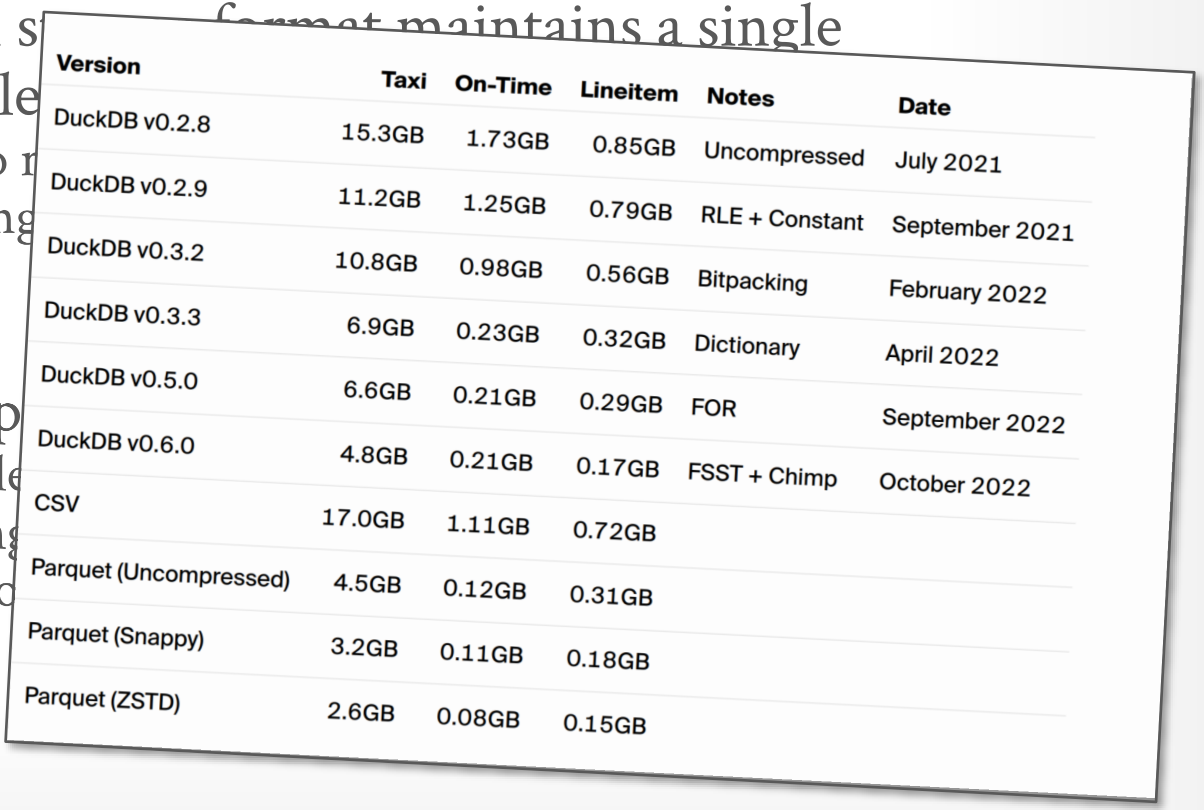
→ Splits tables into r

→ On-disk encoding representation.

Two phase comp

→ **Analyze**: Sample the best encoding

→ **Compress:** Enco



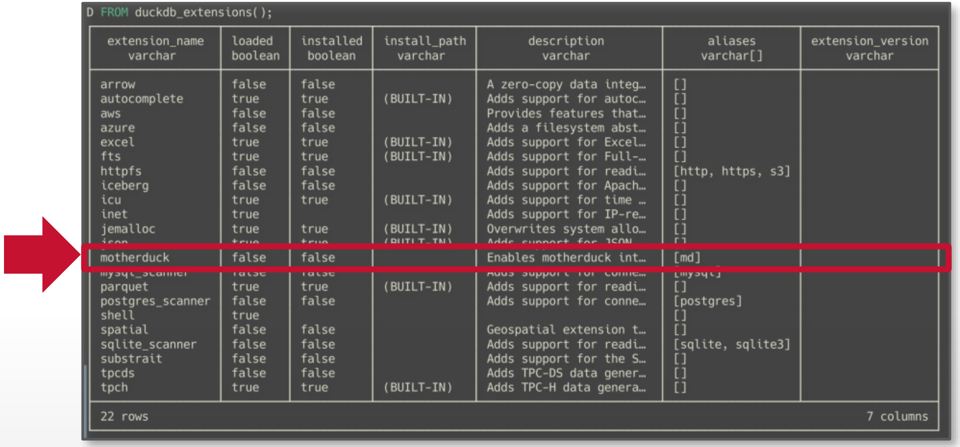
Version	Taxi	On-Time	Lineitem	Notes	Date
DuckDB v0.2.8	15.3GB	1.73GB	0.85GB	Uncompressed	July 2021
DuckDB v0.2.9	11.2GB	1.25GB	0.79GB	RLE + Constant	September 2021
DuckDB v0.3.2	10.8GB	0.98GB	0.56GB	Bitpacking	February 2022
DuckDB v0.3.3	6.9GB	0.23GB	0.32GB	Dictionary	April 2022
DuckDB v0.5.0	6.6GB	0.21GB	0.29GB	FOR	September 2022
DuckDB v0.6.0	4.8GB	0.21GB	0.17GB	FSST + Chimp	October 2022
CSV	17.0GB	1.11GB	0.72GB		
Parquet (Uncompressed)	4.5GB	0.12GB	0.31GB		
Parquet (Snappy)	3.2GB	0.11GB	0.18GB		
Parquet (ZSTD)	2.6GB	0.08GB	0.15GB		

DUCKDB: EXTERNAL TABLES

The DBMS can also access external data files via extensions.

→ Parquet, Arrow, SQLite, JSON,

Can also install extensions to retrieve files from remote filesystems (HTTP, S3)



```
D FROM duckdb_extensions();
┌────────────────┬─────────┬───────────┬──────────────┬──────────────────────────────┬──────────────────────┬───────────────────┐
│ extension_name │ loaded  │ installed │ install_path │         description          │       aliases        │ extension_version │
│    varchar     │ boolean │  boolean  │   varchar    │           varchar            │      varchar[]       │      varchar      │
├────────────────┼─────────┼───────────┼──────────────┼──────────────────────────────┼──────────────────────┼───────────────────┤
│ arrow          │ false   │ false     │              │ A zero-copy data integ…      │ []                   │                   │
│ autocomplete   │ true    │ true      │ (BUILT-IN)   │ Adds support for autoc…      │ []                   │                   │
│ aws            │ false   │ false     │              │ Provides features that…      │ []                   │                   │
│ azure          │ false   │ false     │              │ Adds a filesystem abst…      │ []                   │                   │
│ excel          │ true    │ true      │ (BUILT-IN)   │ Adds support for Excel…      │ []                   │                   │
│ fts            │ true    │ true      │ (BUILT-IN)   │ Adds support for Full-…      │ []                   │                   │
│ httpfs         │ false   │ false     │              │ Adds support for readi…      │ [http, https, s3]    │                   │
│ iceberg        │ false   │ false     │              │ Adds support for Apach…      │ []                   │                   │
│ icu            │ true    │ true      │ (BUILT-IN)   │ Adds support for time …      │ []                   │                   │
│ inet           │ true    │           │              │ Adds support for IP-re…      │ []                   │                   │
│ jemalloc       │ true    │ true      │ (BUILT-IN)   │ Overwrites system allo…      │ []                   │                   │
│ json           │ true    │ true      │ (BUILT-IN)   │ Adds support for JSON …      │ []                   │                   │
│ motherduck     │ false   │ false     │              │ Enables motherduck int…      │ [md]                 │                   │
│ mysql_scanner  │ false   │ false     │              │ Adds support for conne…      │ [mysql]              │                   │
│ parquet        │ true    │ true      │ (BUILT-IN)   │ Adds support for readi…      │ []                   │                   │
│ postgres_scanner │ false │ false     │              │ Adds support for conne…      │ [postgres]           │                   │
│ shell          │ true    │           │              │                              │ []                   │                   │
│ spatial        │ false   │ false     │              │ Geospatial extension t…      │ []                   │                   │
│ sqlite_scanner │ false   │ false     │              │ Adds support for readi…      │ [sqlite, sqlite3]    │                   │
│ substrait      │ false   │ false     │              │ Adds support for the S…      │ []                   │                   │
│ tpcds          │ false   │ false     │              │ Adds TPC-DS data gener…      │ []                   │                   │
│ tpch           │ true    │ true      │ (BUILT-IN)   │ Adds TPC-H data genera…      │ []                   │                   │
├────────────────┴─────────┴───────────┴──────────────┴──────────────────────────────┴──────────────────────┴───────────────────┤
│ 22 rows                                                                                                           7 columns │
└──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────┘
```

MOTHERDUCK

Cloud-based service that provides automatic execution of DuckDB queries on serverless compute nodes.

→ Remote nodes are DuckDB instances running inside of containers and connected to object stores.

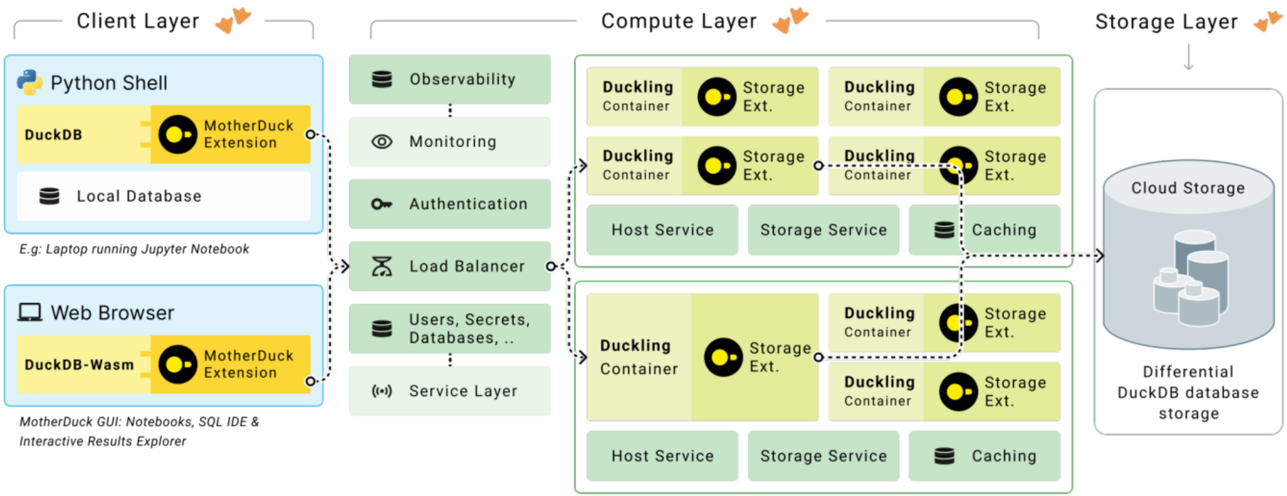
→ Exposes remote catalog to local instance.

The latest versions of DuckDB already include extension to connect to MotherDuck.

MOTHERDUCK: DUCKDB IN THE CLOUD AND IN THE CLIENT
CIDR 2024

MOTHERDUCK

Cloud-based service that provides automatic



Source: MotherDuck

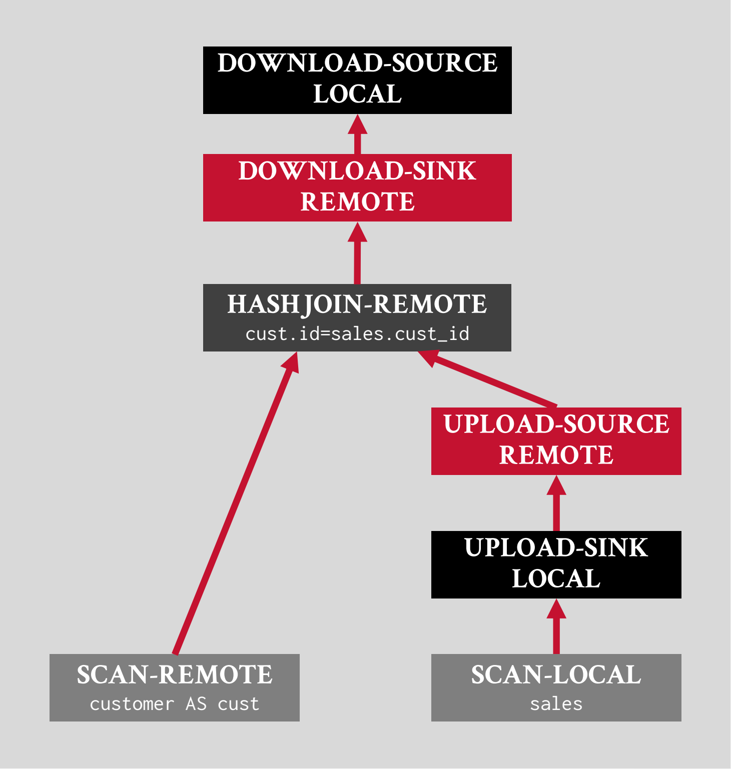
MOTHERDUCK: DUCKDB IN THE CLOUD AND IN THE CLIENT
CIDR 2024

MOTHERDUCK: HYBRID QUERY PROCESSING

Introduces a new "bridge" operators that passes tuple streams between local and remote DuckDB instances.

→ Leverages operator pausing feature that DuckDB added from switching to push-based execution.

Query optimization occurs on the local instance as normal and then uses cost-based rules to decide what to run locally vs. remote.



PARTING THOUGHTS

DuckDB is brilliant and its adoption is enviable.
→ Right place. Right time. Right problem.

Andy bet his earlier research agenda wrongly on in-memory DBMSs.

This is what HyPer/Umbra could have become if they were open-source…

PARTING THOUGHTS

DuckDB is brilliant and its adoption is enviable.

→ Right place. Right time. Right problem.

Andy bet his earlier research agenda wrongly on in-memory DBMSs.

This is what HyPer/Umbra could have become if they were open-source…

NEXT CLASS

Yellowbrick