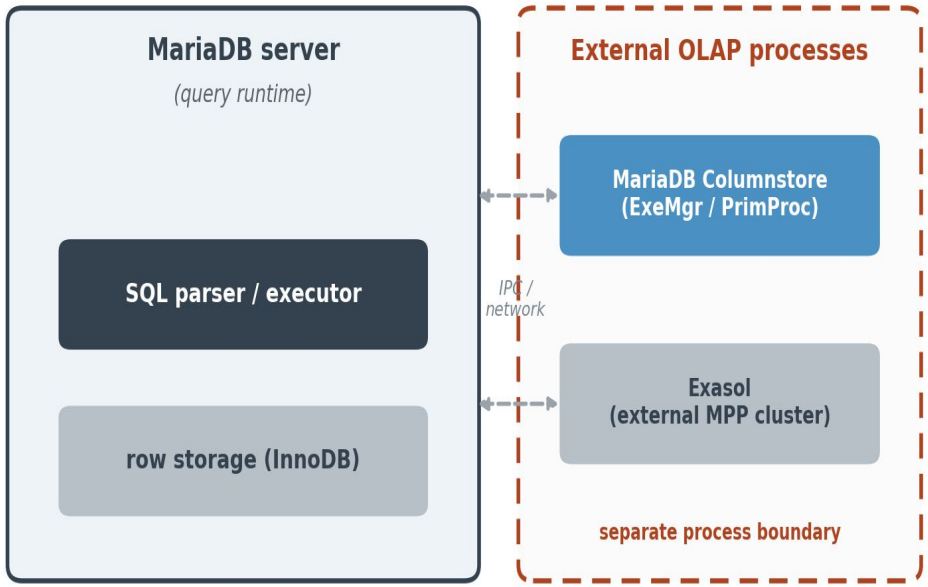
When Sea Lion Learns to Quack: DuckDB as a MariaDB storage engine

Roman Nozdrin
MariaDB Corporation

DuckCon 2026 @Amsterdam

The Sea Lion


MariaDB®
The Pool

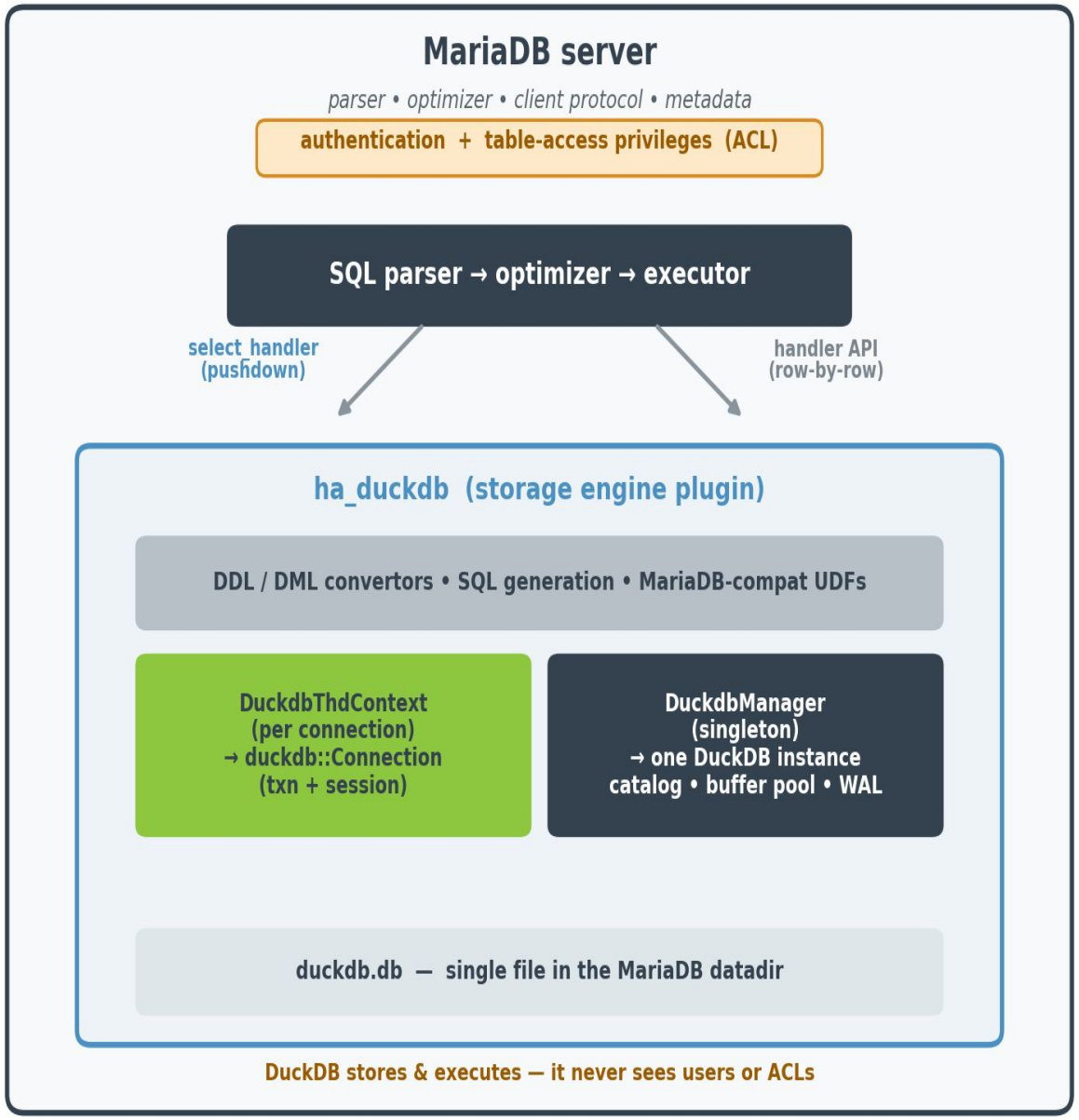


The Pool
OLAP in MariaDB Today

- MariaDB Columnstore: a columnar engine for analytics — but it runs as its own processes (ExeMgr, PrimProc) beside the server.
- Exasol: a high-performance MPP database — data is offloaded to a separate external cluster.
- Both run outside the MariaDB runtime: a separate process boundary, network hops, data to keep in sync.

MariaDB

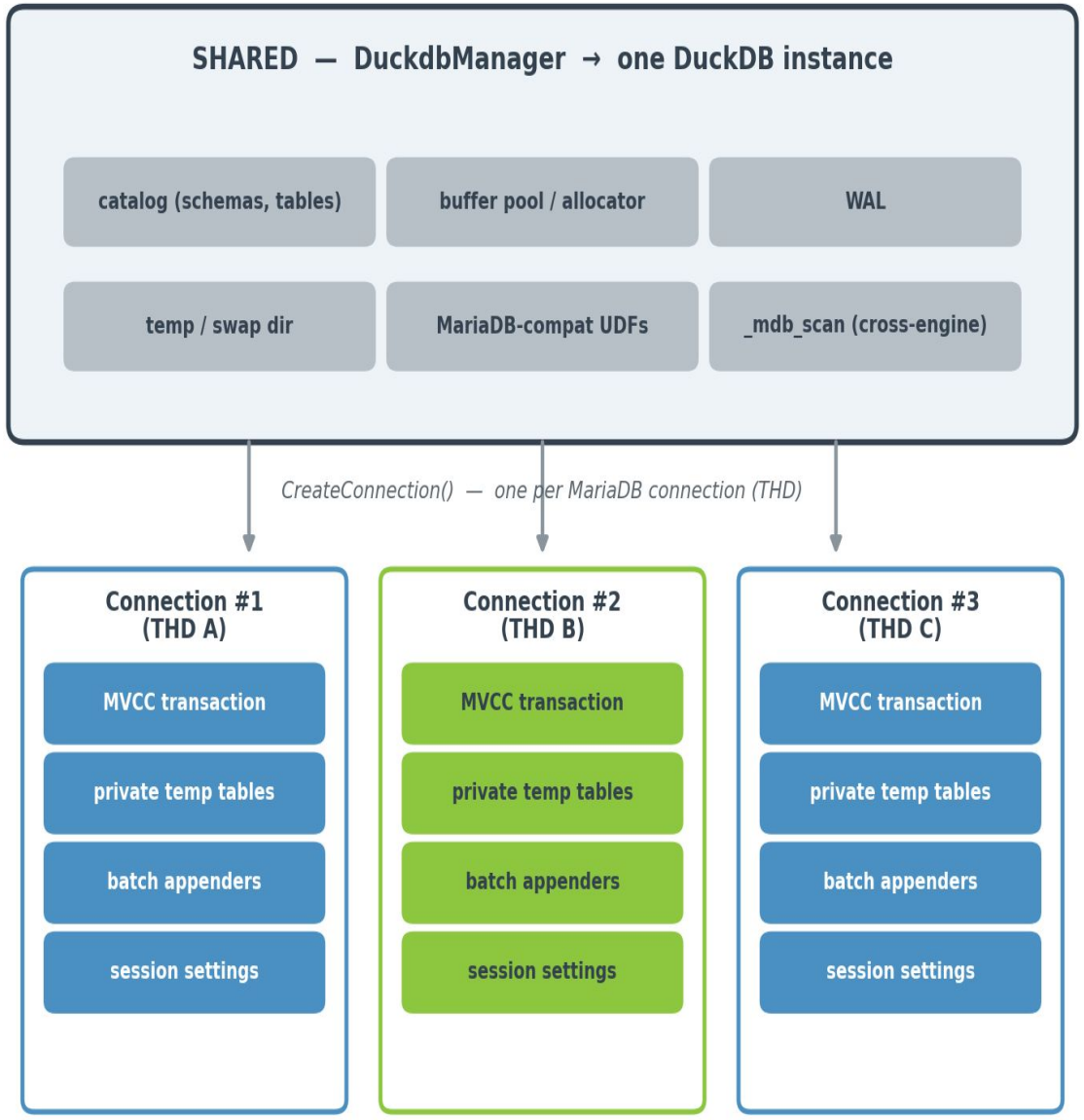
The Duck



DuckDB
Smart Storage Engine

- ha_duckdb is a storage-engine plugin: MariaDB keeps parsing, client protocol and metadata.
- DuckDB owns columnar storage, execution and MVCC - one shared instance (catalog, buffer pool, WAL).
- User authentication and table-access privileges stay in MariaDB - DuckDB never sees ACLs; per connection a DuckdbThdContext wraps a duckdb::Connection.

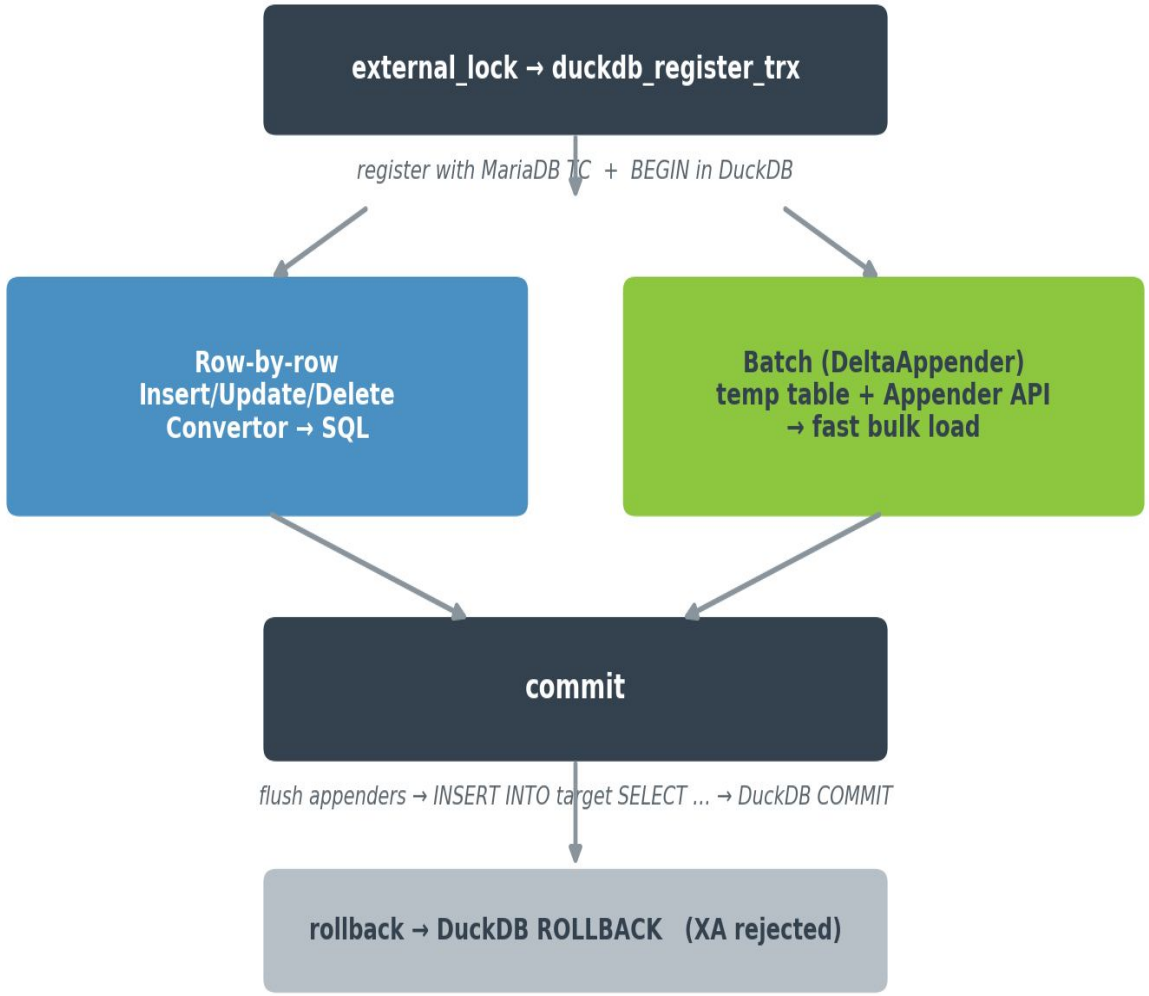
MariaDB



Shared Resources Across Connections

- Shared, one per server: the DuckDB instance - catalog, buffer pool, WAL, temp/swap, plus the compat UDFs and macros.
- Per connection: its own duckdb::Connection - an MVCC transaction, private temporary tables, batch appenders and session settings.
- Trade-off: reads are snapshot-isolated and writes serialized by DuckDB's single-writer lock - but the shared buffer pool means one big scan can evict another's pages.

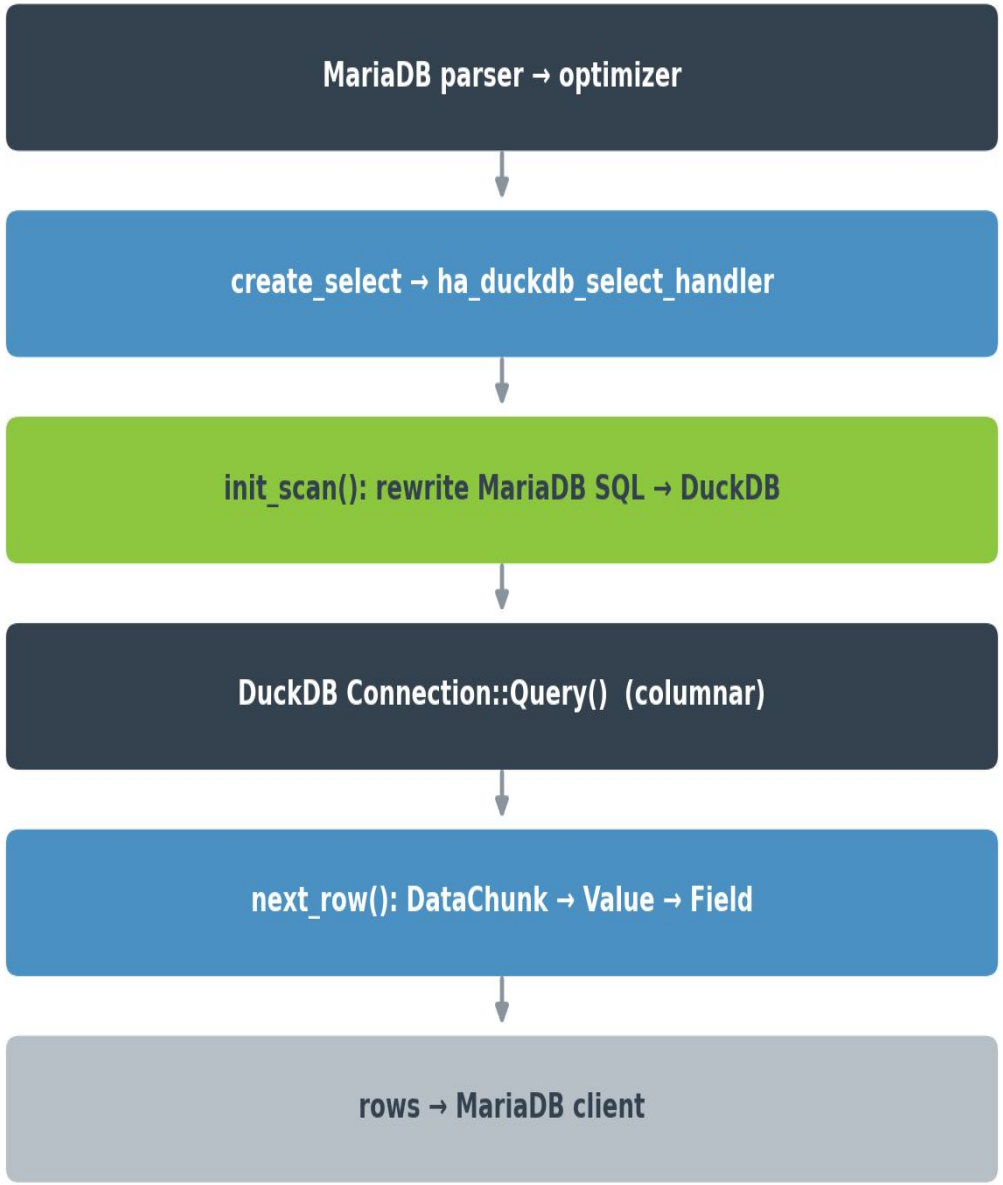
MariaDB



Write Path & Batch DML

- DDL maps through convertors (CREATE / ALTER / DROP)
  - ALTER supports both INSTANT and COPY ALTER TABLE protocols of MariaDB.
- Each statement registers with MariaDB's transaction coordinator; a matching BEGIN / COMMIT runs in DuckDB (XA rejected).
- Small writes go row-by-row via DML convertors; bulk loads use DeltaAppender - temp table + Appender API, flushed as one INSERT ... SELECT at commit.
- Multi-engine txns are supported

MariaDB



Happy Read Path: SELECT Handler Pushdown

For analytics, MariaDB pushes the whole SELECT to DuckDB via select_handler.

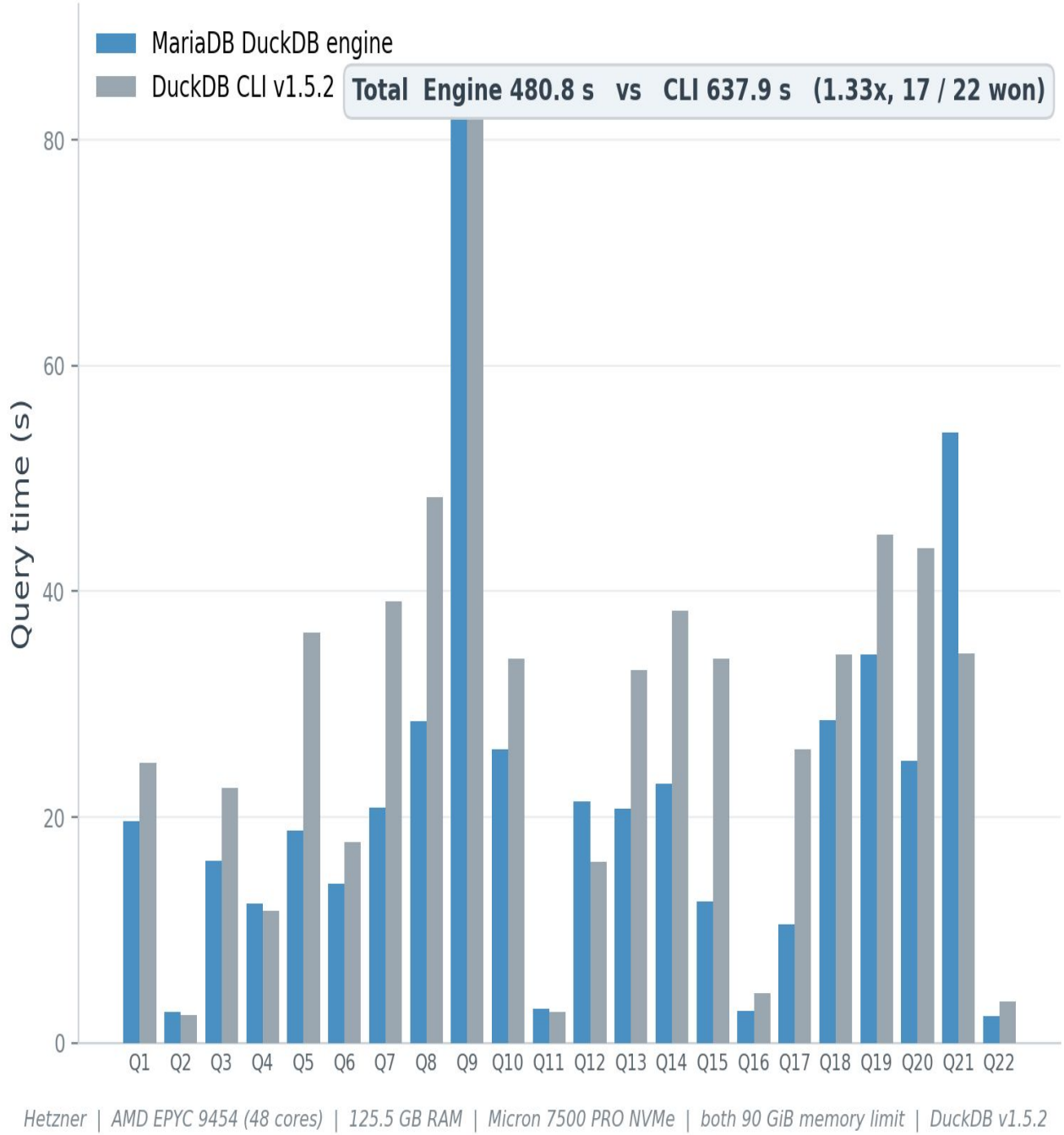
init_scan() rewrites MariaDB syntax to DuckDB: WITH ROLLUP → ROLLUP(), CONVERT → CAST, REGEXP → ~, LIMIT a,b → LIMIT b OFFSET a; hints stripped.

Results stream back DataChunk-by-DataChunk (Value → Field).

Cross-engine joins reach InnoDB/ARIA via a replacement scan + stackful fiber.

MariaDB

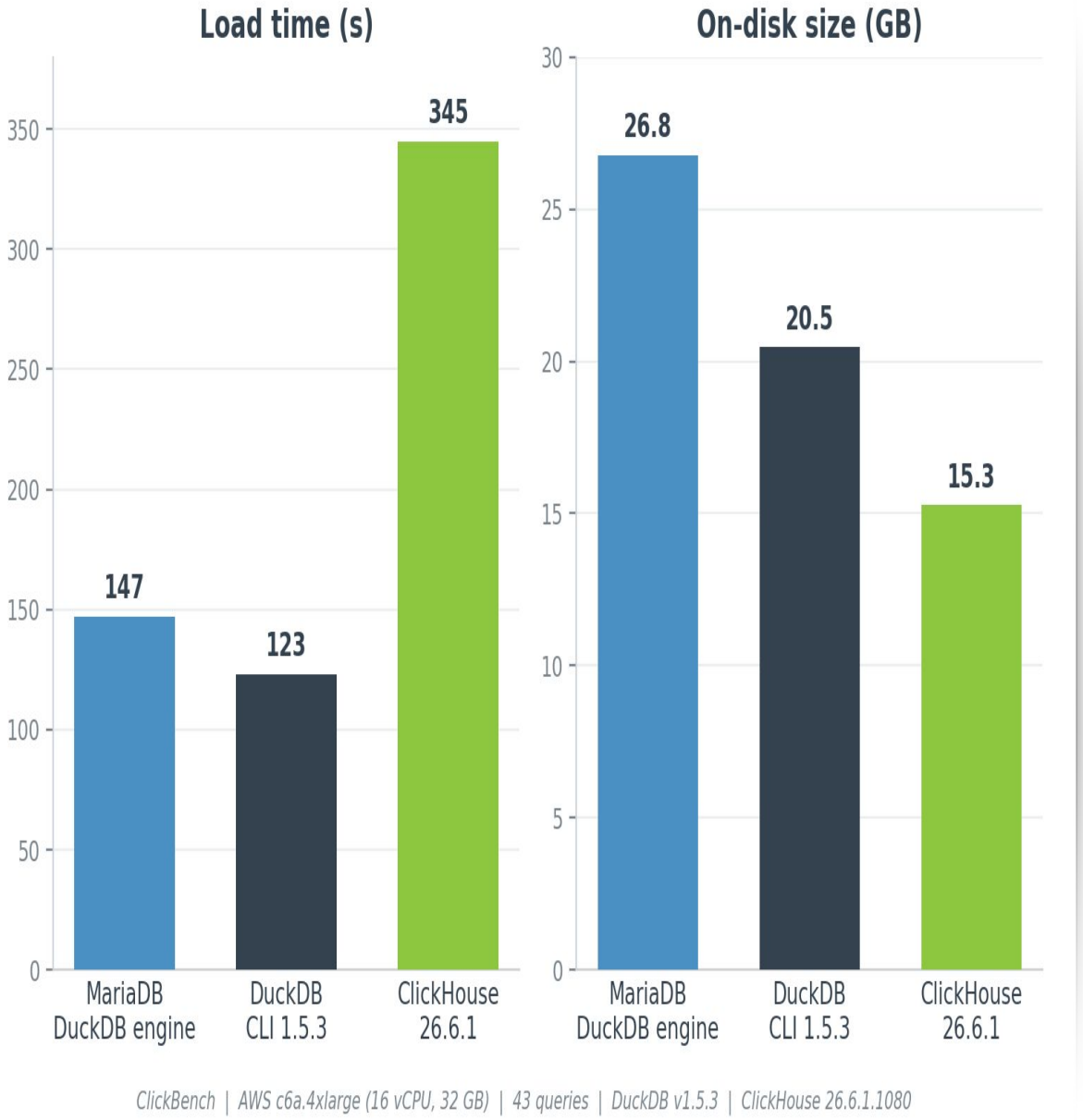
The First Quack in Numbers



TPC-H SF1000: Engine vs DuckDB 1.5.2 CLI

- On TPC-H SF1000 (about 8.7 B rows, 400 GB Parquet), the engine totals 480.8 s vs 637.9 s for the DuckDB CLI v1.5.2: 1.33x faster, winning 17 of 22. Best case Q15 at 2.72x.
- Why: the engine is a persistent in-process DuckDB instance inside mariadbd with a warm buffer pool. The CLI spawns a fresh process per query, cold-starting the buffer manager every time. That cold start dominates, not raw execution.
- Honest read: single run, no repetition; a single-session CLI run would narrow the gap. Peak RSS is similar (112 vs 115 GB). One regression is Q21 (EXISTS patterns), where the CLI wins.

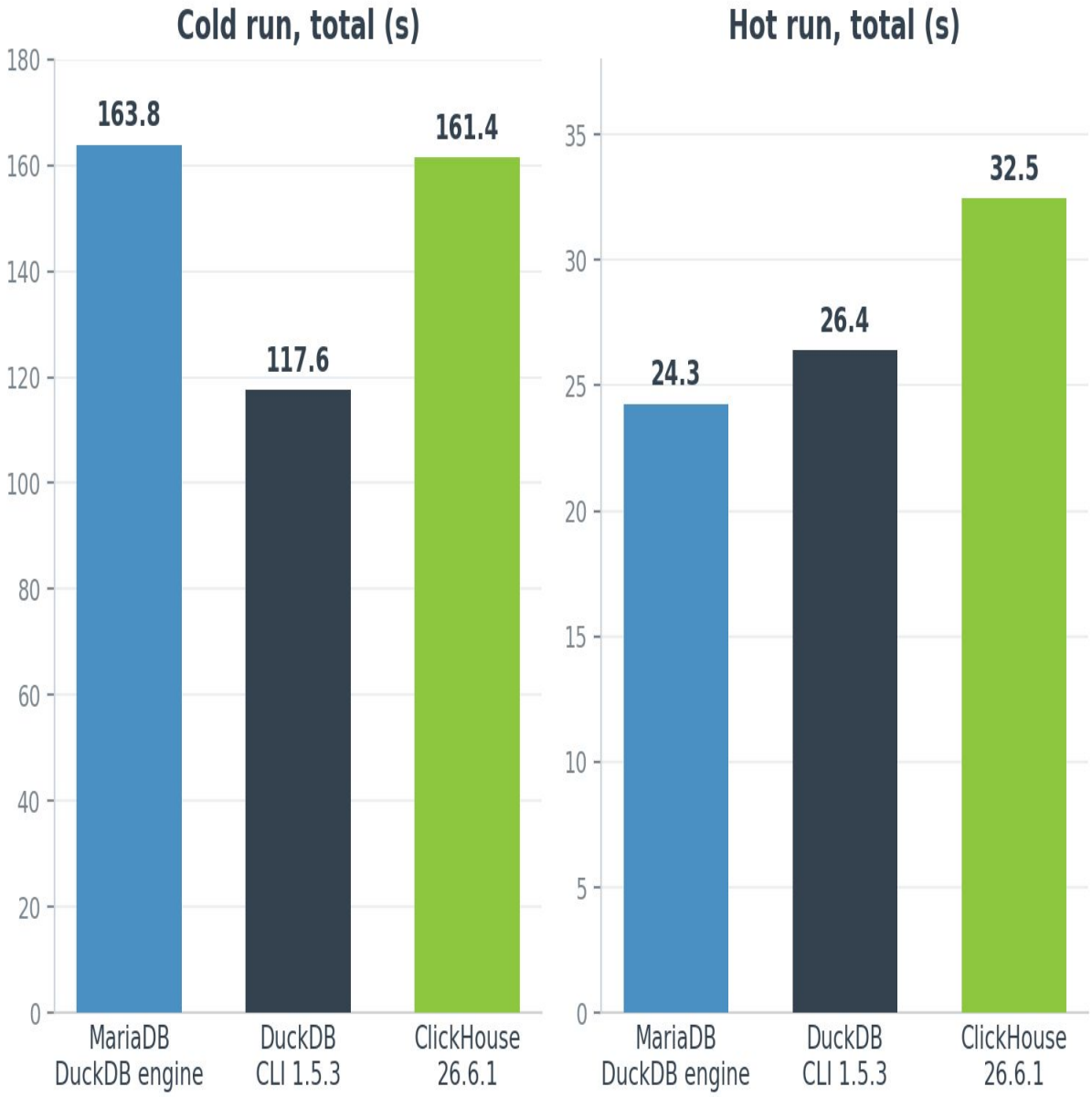
MariaDB



ClickBench: Load Time and Size

- Engine and DuckDB CLI loads from a single parquet file whilst CH uses partitions parquet
- Load time: DuckDB CLI 123 s, engine 147 s, ClickHouse 345 s. ClickHouse is roughly 2.3x slower to load than the engine.
- On-disk size: ClickHouse is densest at 15.3 GB, DuckDB CLI 20.5 GB, the engine 26.8 GB. ClickHouse trades load time for stronger compression.
- The engine carries some storage overhead versus the bare DuckDB file, but stays in the same order of magnitude.

MariaDB

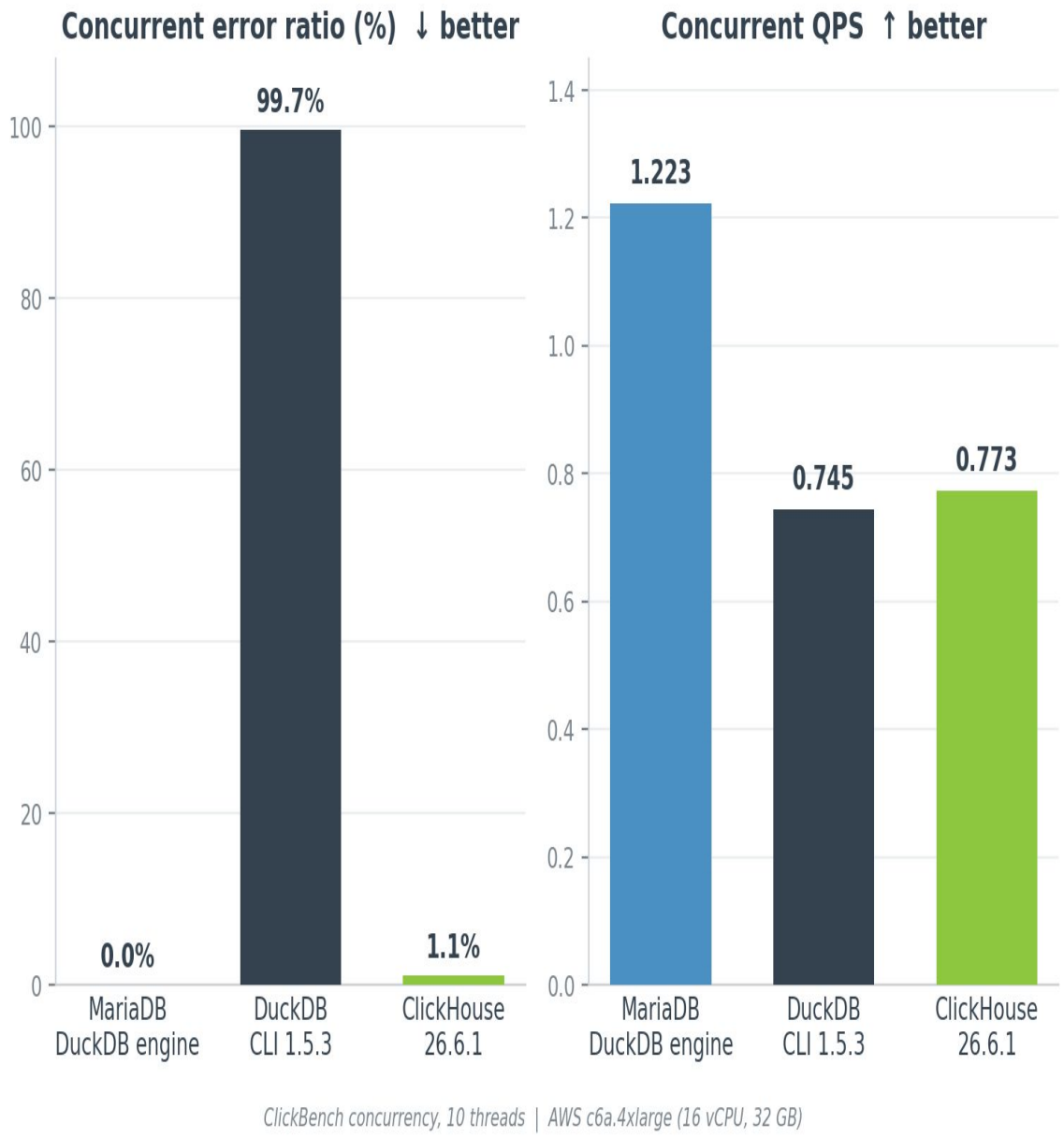


ClickBench | AWS c6a.4xlarge (16 vCPU, 32 GB) | 43 queries | DuckDB v1.5.3 | ClickHouse 26.6.1.1080

ClickBench: Single-Terminal Queries

- Hot run (43 queries, summed): the engine is fastest at 24.3 s, ahead of DuckDB CLI 1.5.3 at 26.4 s and ClickHouse at 32.5 s.
- Cold run tells the opposite story: the CLI leads at 117.6 s, while the engine and ClickHouse trail near 162 s. The engine pays more on first touch, then wins once warm.
- Hot = best of two warm runs per query. The engine matches native DuckDB on warm analytics and edges ahead on this workload.

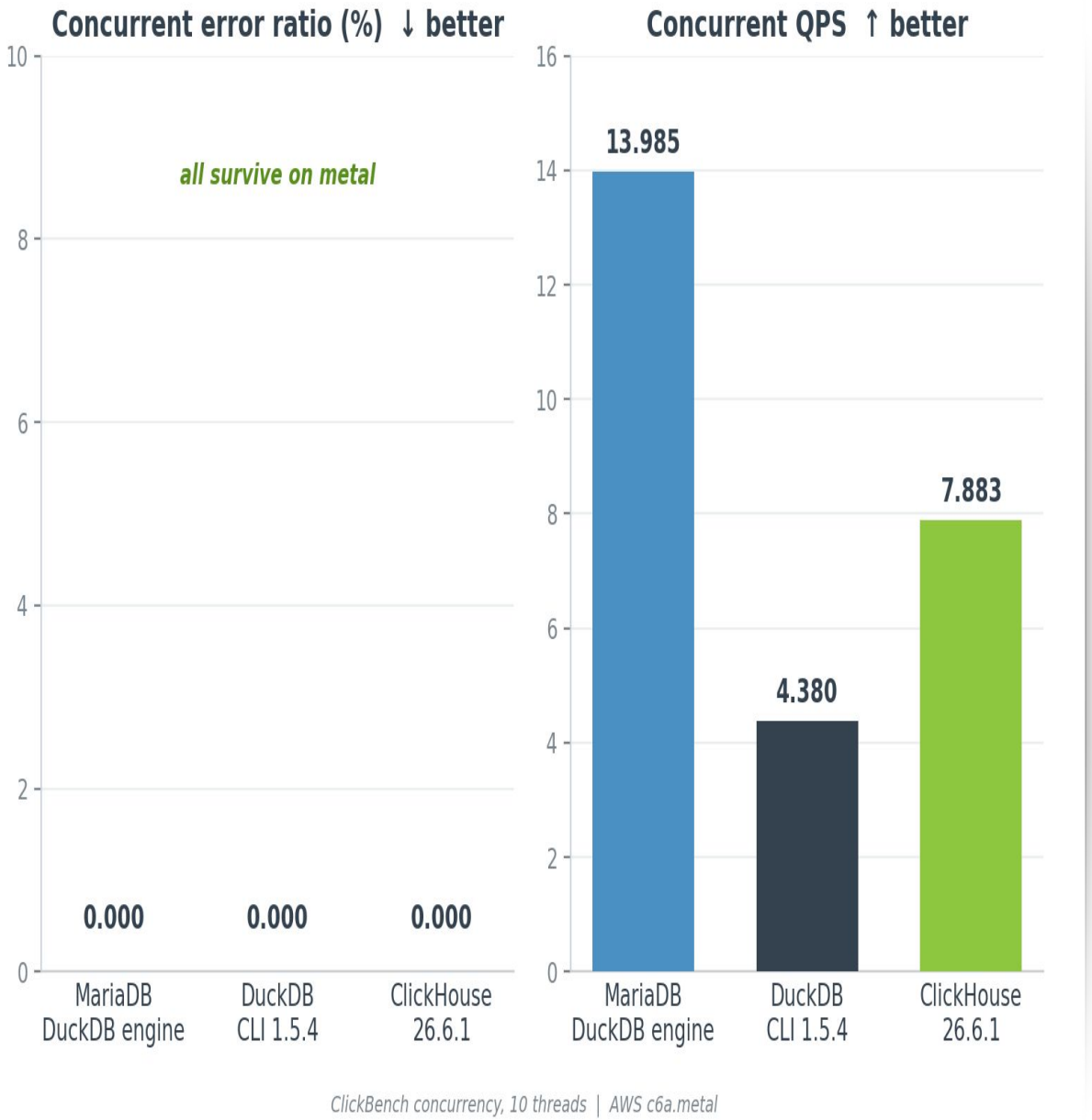
MariaDB



ClickBench Concurrency: 10 Threads

- The headline: the engine survives concurrency with a 0.000 error ratio, while native DuckDB CLI collapses at 0.997 (99.7% of queries fail). ClickHouse is steady at 0.011.
- Throughput: the engine leads at 1.223 QPS, ahead of ClickHouse 0.773 and DuckDB CLI 0.745. It is both the most reliable and the fastest under load.
- This is the multi-tenant point in numbers: many MariaDB connections share one DuckDB instance and keep serving, where the standalone CLI cannot.

MariaDB



Concurrency on c6a.metal

- On bare metal all three survive with a 0.000 error ratio. The CLI enjoys –readonly flag the way it did not on c6a.4xlarge.
- Throughput is the differentiator: the engine leads at 13.985 QPS, ahead of ClickHouse 7.883 and DuckDB CLI 4.380. That is about 1.8x ClickHouse and 3.2x the CLI.
- Many MariaDB connections share one DuckDB instance and scale on a big box, turning multi-tenant from a checkbox into real throughput.

MariaDB

So how does Sea Lion quack after the first lesson…?

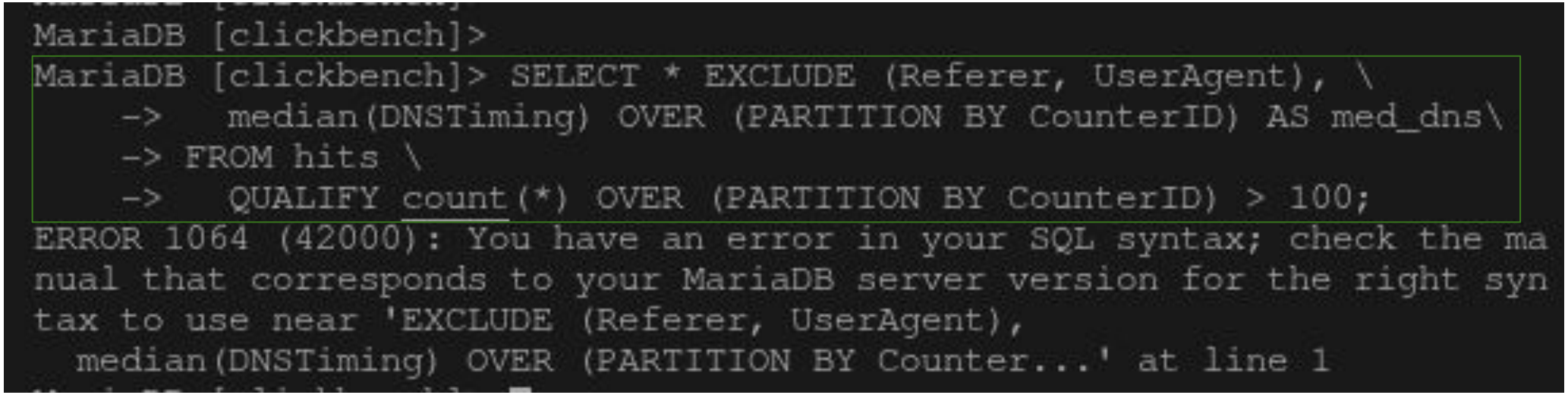
So how does Sea Lion quack after the first lesson…?

MariaDB

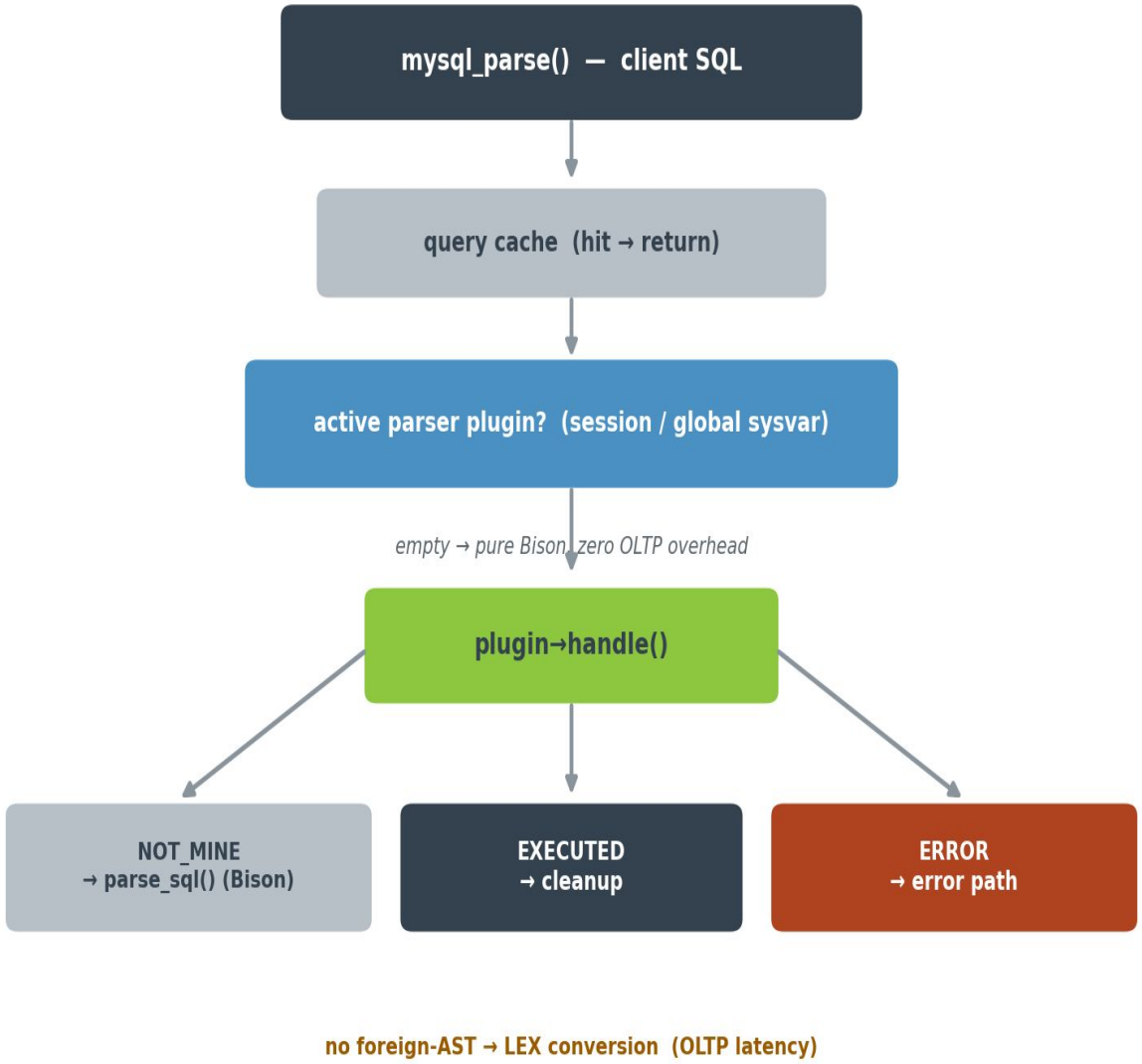
The Future of Sea Lion's Quacking

Embedding Isn’t Enough

- Even with DuckDB embedded, every client query still enters through MariaDB’s Bison parser.
- DuckDB-native syntax — EXCLUDE, QUALIFY, list / struct types, PIVOT, DuckDB extensions - are rejected before it ever reaches DuckDB.
- The engine can run it; the front door can’t. Embedding the engine isn’t enough on its



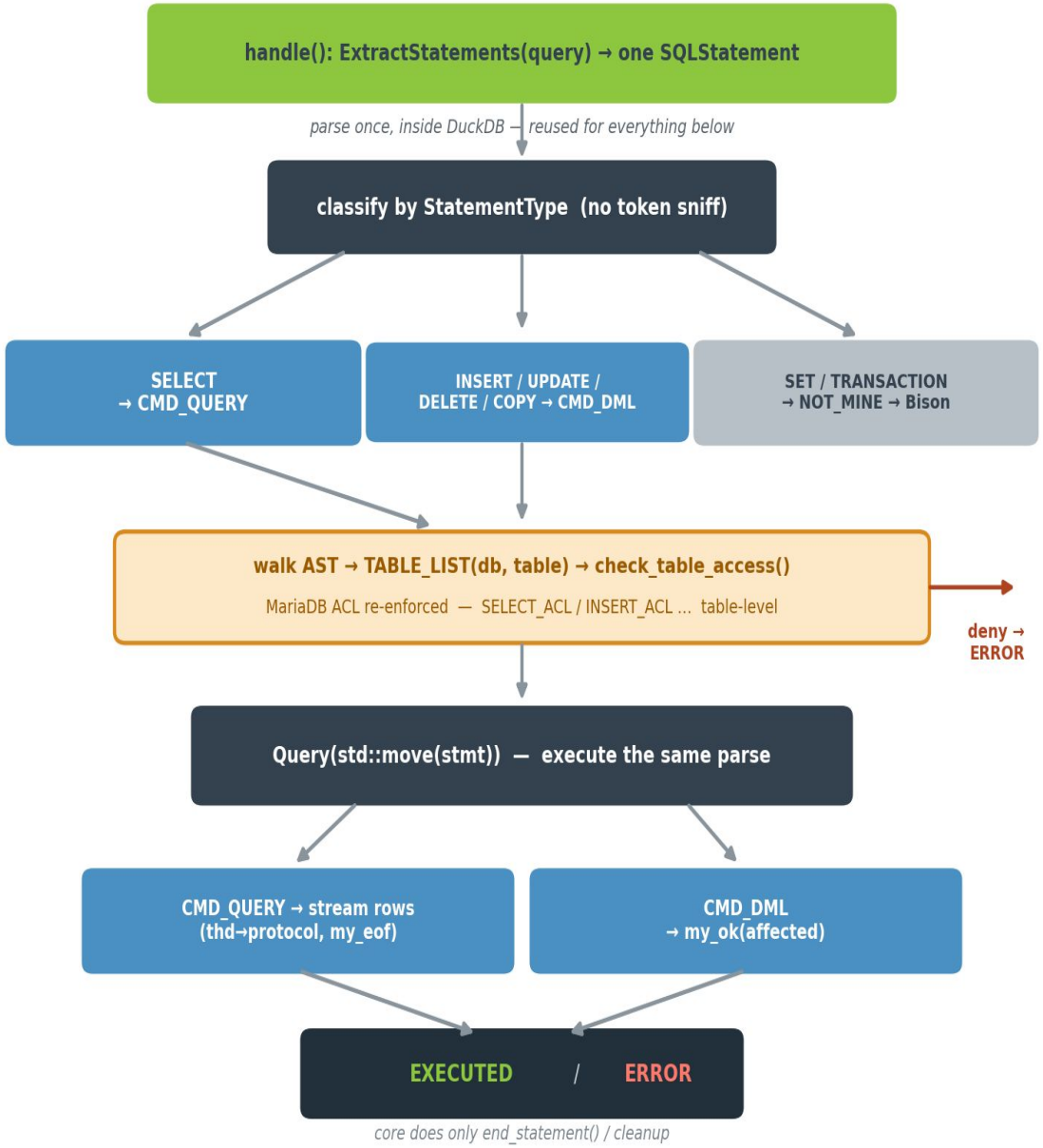
MariaDB



A Pluggable Parser plugin for MariaDB

- A new parser-plugin type lets a plugin take over SQL parsing for a connection, chosen by a session / global system variable.
- Empty value → pure Bison, zero OLTP overhead. The hook sits in mysql_parse(), after the query cache and before parse_sql().
- handle() returns NOT_MINE (fall back to Bison), EXECUTED (plugin ran it) or ERROR — no foreign-AST → LEX conversion, so OLTP latency stays low.

MariaDB



The Duck Parser Plugin

- handle() parses the query once with DuckDB (ExtractStatements → SQLStatement) and reuses that parse for classification, privilege checks and execution — no token sniff, no AST → LEX conversion.
- Classify by StatementType: SELECT streams rows; INSERT / UPDATE / DELETE / COPY return affected rows; SET and transactions fall back to Bison (native MariaDB).
- Privileges stay in MariaDB: since PARSER_EXECUTED skips normal execution, the plugin walks the parsed AST for base tables and calls check_table_access() — table-level GRANTs — before executing.

MariaDB

Full Duck over MariaDB client Connection

- Full DuckDB SQL in a plain MariaDB connection — no new client, driver, or wire protocol.
- EXCLUDE / REPLACE, QUALIFY, GROUP BY ALL, list / struct / map, lambdas, PIVOT / UNPIVOT, friendly SQL.
- Concurrency made DuckDB production-ready; the parser makes it native — the real “quack.”

MariaDB

Acknowledgement

- None of this would be possible without remarkable work of DuckDB Community and DuckLabs. Thank you for the Duck!
- MariaDB Foundation/Corporation for the Sea Lion!
- Alibaba and their engineering team for opensourcing AliSQL!
- Personally to Monty for suggesting Quacking narrative for this speech

MariaDB

https://github.com/MariaDB/server/tree/11.4/storage/duckdb

The engine. Where and when?

- Merged into upstream MariaDB 11.4,11.8,12.3,13.x
- MariaDB server packages with DuckDB storage engine will be available with the next maintenance release
  - scheduled data end of July
- As of now MariaDB 11.4,11.8,12.3, 13.x will force you to reduce plugin maturity
- GNU GPL v2

MariaDB

Thank you