OurSQL FOUNDATION



I'll be speaking at:

DuckDB × MySQL Day.

November 17 * Online Conference

Session: ProxySQL Welcomes DuckDB

René Cannaò

CEO at ProxySQL

A report in 3 minutes 52 seconds

MySQL

3 min 52 s

DuckDB on a DBTrail copy

0.9 s

Join and aggregation by district and month

Published by Daniel Guzmán Burgos / DBTrail, September 2026. DuckDB queried an analytical copy.

Another published comparison

MySQL

5 min 32 s

DuckDB on a DBTrail copy

1.7 s

Best-selling-items report under transactional load

Published by Vadim Tkachenko / Percona, October 2026. DuckDB queried an analytical copy.

A faster answer changes the workflow

While the question is still in your head.

Change the filter. Inspect the answer. Continue investigating.

How does an application use that speed?

Faster execution creates an application opportunity.

Data freshness and request routing decide whether the application can use it.

One dataset, different work

Order lookup

Find a few records.
Complete the transaction.

Historical report

Read many records.
Return a small aggregate.

Existing option: improve MySQL

Improve the access path

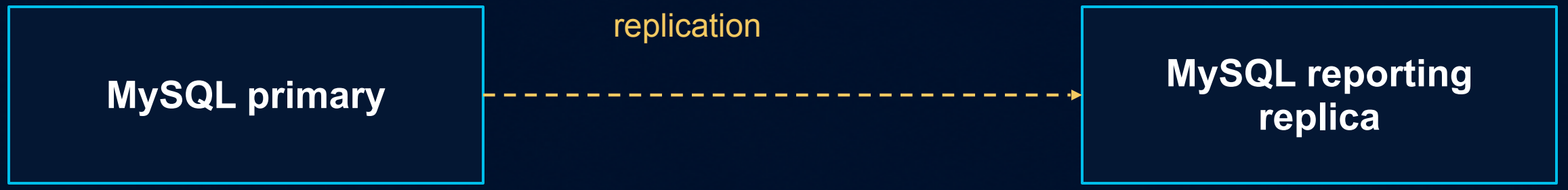
Remove unnecessary processing.
Match the query to useful indexes.

Maintain a summary

Make a repeated report cheap.
Maintain the totals as data changes.

A new question may need a different summary.

Existing option: a reporting replica



Application transactions

Report execution

Isolation protects the primary. The report still uses the MySQL engine.

Existing option: a periodic analytical copy



The answer describes a defined snapshot.

Fresher copies need an update path



The copy needs update and recovery behavior.

HTAP

Hybrid Transactional and Analytical Processing

Transactions and analytics on recent operational data.

A useful composition for existing MySQL

Keep the transactional role.

Add an analytical path with defined guarantees.

Another engine adds application decisions

Application A

Own connection pools
Own engine-selection policy

Application B

Own connection pools
Own engine-selection policy

The same decisions repeat across clients.

ProxySQL already manages database access



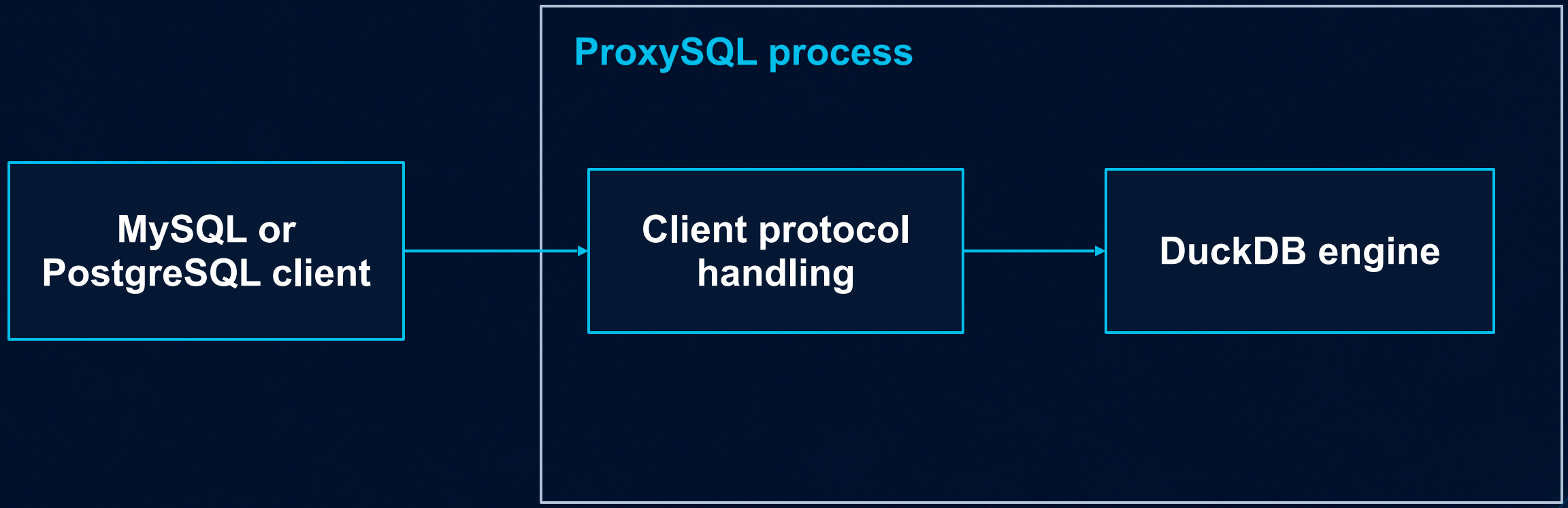
A shared place to express request policy.

One known report is a useful starting point



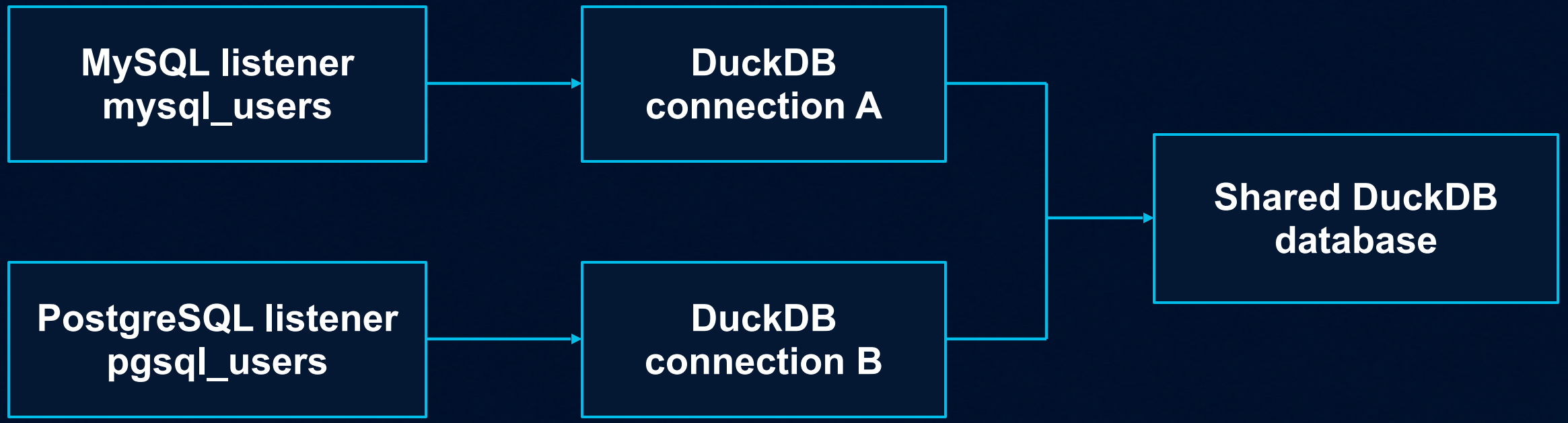
A narrow rollout gives evidence for one query family.

DuckDB runs inside the ProxySQL process



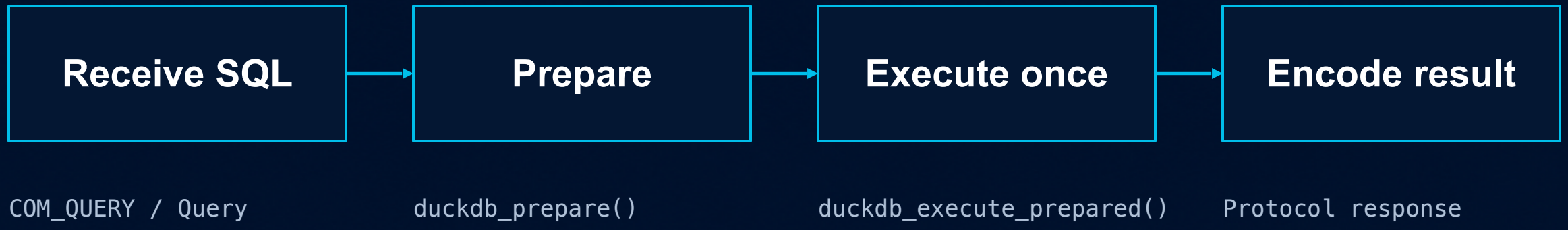
One shared database, with a DuckDB connection per client.

Two protocols, one embedded database



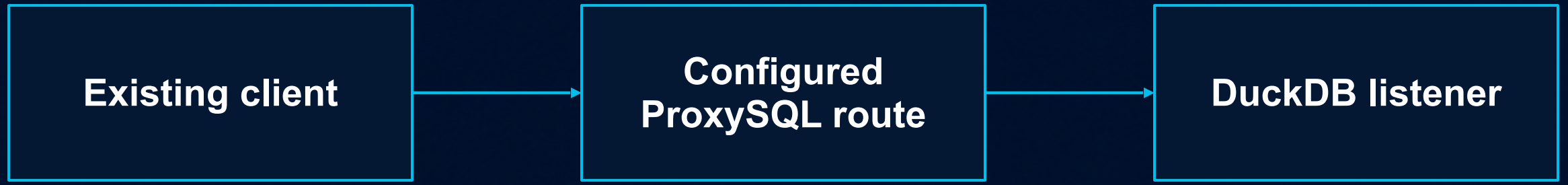
Per-client session state; shared database contents

Inside the plugin query path



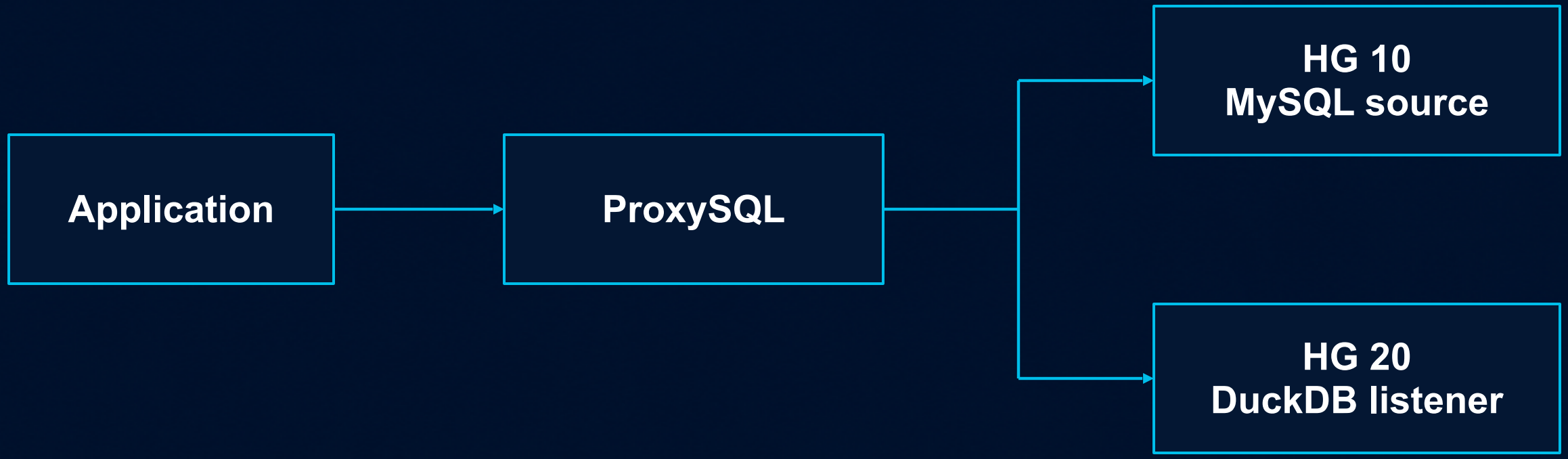
Result adaptation is decided before execution.

The analytical destination gains an access path



The ordinary proxy path can target the analytical interface.

A DuckDB listener can be a hostgroup backend



HG 20 uses the normal backend connection path, even on the same host.

A concrete routing rule for an analytical client

```
INSERT INTO mysql_query_rules
  (rule_id, active, username, destination_hostgroup, apply)
VALUES (100, 1, 'analytics_report', 20, 1);

LOAD MYSQL QUERY RULES TO RUNTIME;
SAVE MYSQL QUERY RULES TO DISK;
```

Dedicated reporting client; HG 20 and credentials configured separately

A familiar connection still needs compatible SQL

Familiar client connection	Compatibility still matters
A supported wire protocol	DuckDB SQL
A reachable destination	Supported client execution mode
A printed result	Driver-visible meaning

An analytical query through a familiar client

```
SELECT region, SUM(amount) AS total
FROM (VALUES ('North', 24), ('West', 68),
             ('North', 31), ('West', 17))
     AS orders(region, amount)
GROUP BY region ORDER BY region;
```

Expected result: North 55 / West 85

One DuckDB statement sent with a simple-query client API

A printed value is only part of the result

A terminal prints

123.45

A driver receives

A value and column metadata

Precision, binary values, nulls and time interpretation matter.

Typed results across both client protocols

DuckDB value	MySQL metadata	PostgreSQL metadata
DECIMAL(12,2)	NEWDECIMAL	NUMERIC(12,2)
BOOLEAN	TINYINT(1)	BOOLEAN
BLOB	LONG_BLOB	BYTEA

Native type metadata; values still use text transfer

Embedding shares a process boundary

Access and analytical execution share one host process.

Resource use and process lifetime remain part of the deployment choice.

Operating DuckDB through ProxySQL Admin

```
UPDATE global_variables SET variable_value='2GB'
WHERE variable_name='duckdb-memory_limit';
UPDATE global_variables SET variable_value='4'
WHERE variable_name='duckdb-threads';
UPDATE global_variables SET variable_value='32'
WHERE variable_name='duckdb-max_connections';

LOAD DUCKDB VARIABLES TO RUNTIME;
SAVE DUCKDB VARIABLES TO DISK;
```

inspect runtime_global_variables after applying

Admission, persistence and external access

Control	Implementation consequence
mysql_users / pgsql_users	Admission to one shared database
duckdb-database_path	:memory: lifetime or persistent file
duckdb-enable_external_access	External access defaults to false

Authentication, persistence and external access answer different questions.

A reachable engine needs a useful dataset



An experiment can begin from a defined dataset.

1. Define the MySQL source and its credentials

```
INSERT INTO duckdb_replica_sources
  (source_id, hostname, port,
   username, password, server_id, enabled)
VALUES ('oltp', 'mysql-source.example', 3306,
        'duckdb_repl', '<source-password>', 100, 1);
```

oltp = source name / 100 = replication consumer ID

The credentials belong to an existing account on source MySQL.

Source account: SELECT on chosen tables + REPLICATION CLIENT / SLAVE + RELOAD

2. Select tables, save configuration and start

Use the source_id configured on the preceding slide

```
INSERT INTO duckdb_replica_tables
  (source_id, schema_name, table_name, active)
VALUES ('oltp', 'app', 'orders', 1);

SAVE DUCKDB REPLICA TO DISK;
START DUCKDB REPLICA;
```

Prerequisites: configured source; writable file-backed DuckDB database

The snapshot must meet the change stream

Starting snapshot	Subsequent changes
Consistent source state	Committed source transactions
Matching source coordinates	Apply after that boundary

The two parts must agree on their place in source history.

Rows and their checkpoint advance together

Applied rows

Their matching source position

One DuckDB commit

Binlog rows and the DuckDB checkpoint



INSERT / UPDATE / DELETE + proxysql_replica.position

Recovery makes the copy interpretable

A durable boundary connects stored rows to source history.

A fast answer is useful when its data can be interpreted correctly.

Native replica: current operating boundaries

Implemented direction

Selected-table snapshot
Committed-change apply
Durable restart position

Remaining boundaries

Source and type restrictions
Schema-change handling
Automatic live reconnect

Inspecting replica state and handling an interruption

```
SELECT * FROM stats_duckdb_replica;

STOP DUCKDB REPLICA;
START DUCKDB REPLICA;
```

Restart resumes a checkpoint. RESET removes the replica copy.

seconds_behind_source = age of the last applied event

Verifying the route with ProxySQL statistics

```
SELECT rule_id, hits
FROM stats_mysql_query_rules;

SELECT hostgroup, digest_text, count_star
FROM stats_mysql_query_digest;

SELECT * FROM stats_mysql_connection_pool;
```

Rule matches / query destinations / backend activity

ProxySQL's place in the change

Make faster analytics usable by existing MySQL applications.

The access layer can evolve with the available engines.