TETRA

TETRA CA PBC · BENCHMARK REPORT · OPENCYPHER

# By the Numbers

## Reads, writes, & concurrency: TETRAbase vs Neo4j

TETRAbase, the first Matrix Database, stores data as a matrix. The system uses OpenCypher over Bolt (Neo4j's binary protocol) or Connect RPC. This maximizes data system interoperability. This report measures the 21 LDBC Social Network Benchmark Interactive queries at five scale factors from 17 million to 1.78 billion edges, a matrix of 18 OpenCypher write workloads at 1 to 32 clients, and readers and writers running together at every scale factor. We compare to Neo4j on the smaller scale factors.

**66x**

**FASTER THAN NEO4J**

21 LDBC queries at SF10

**13x**

**LESS CPU**

the same queries, SF10

**10.3x**

**MORE READS PER SECOND**

8 concurrent readers, SF1

**7.9x**

**SMALLER STORE**

after 18 write workloads

SECTION 01

# At a glance

TETRAbase against Neo4j: the same server, the same data, the same OpenCypher queries.

---

## 66x

**FASTER THAN NEO4J ON THE LDBC GRAPH BENCHMARK**




Total time for the 21 LDBC queries at SF10, 177 million edges.

## 13x

**LESS CPU FOR THE SAME QUERIES**




Server CPU for one run of the 21 queries at SF10.

## 10.3x

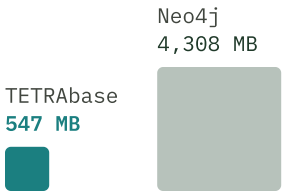
**MORE READS PER SECOND UNDER CONCURRENT LOAD**




8 concurrent LDBC readers on the SF1 graph.

## 7.9x

**SMALLER STORE AFTER THE WRITES**




Squares drawn by area. 18 OpenCypher write workloads at 200,000 rows.

## 1.78B edges

**ANSWERED FASTER THAN NEO4J ANSWERS 17 MILLION**




The 21 LDBC queries. SF100 has 103x the edges of SF1.

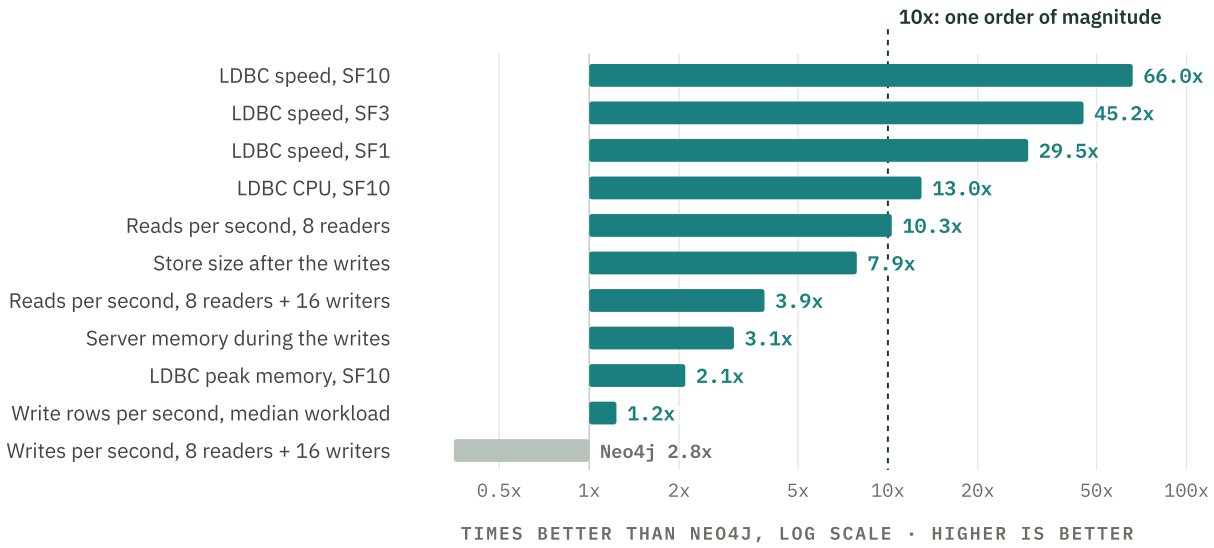
## 2.1x

**LESS MEMORY AT PEAK**




Largest server memory during the 21 queries at SF10.

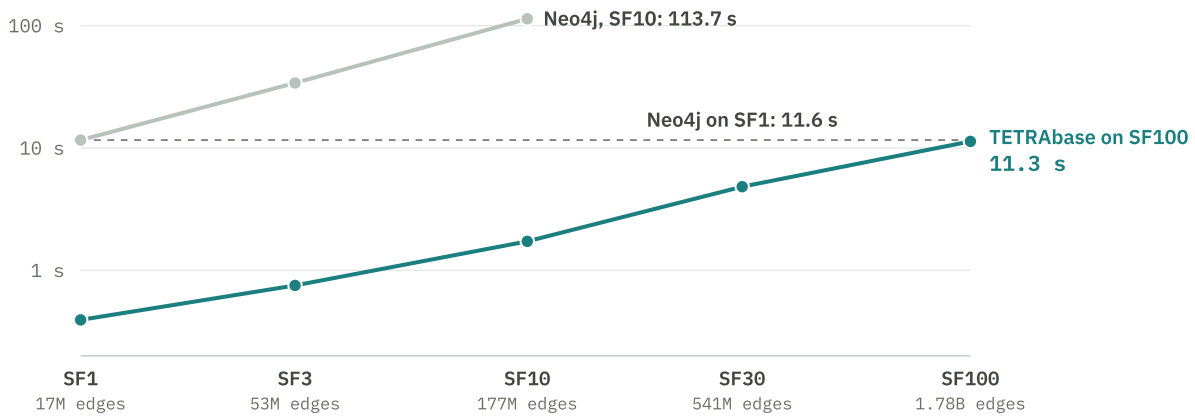
## Every headline measure, against Neo4j




Each bar: how many times better TETRAbase does than Neo4j. Faster, less CPU, more reads, a smaller store. Left of 1x, Neo4j leads.

## The largest graph against the smallest

total time for the 21 queries, log scale, lower is better




TETRAbase answers the 21 queries on 1.78 billion edges in less time than Neo4j takes on 17 million.

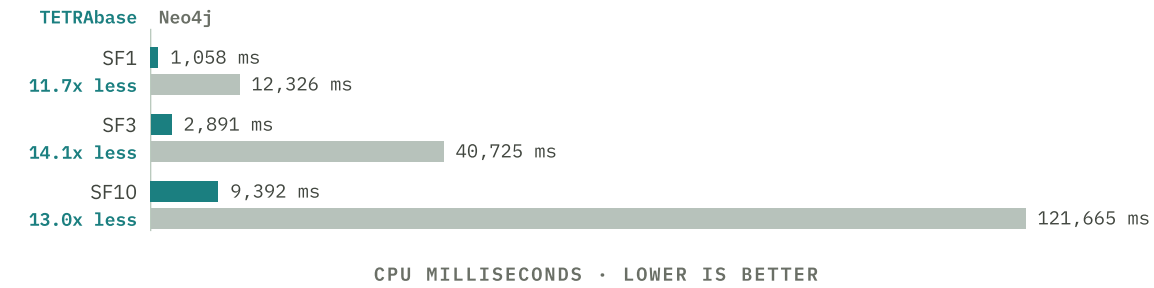
## Speed and resources at every size Neo4j ran

### Total time for the 21 LDBC queries



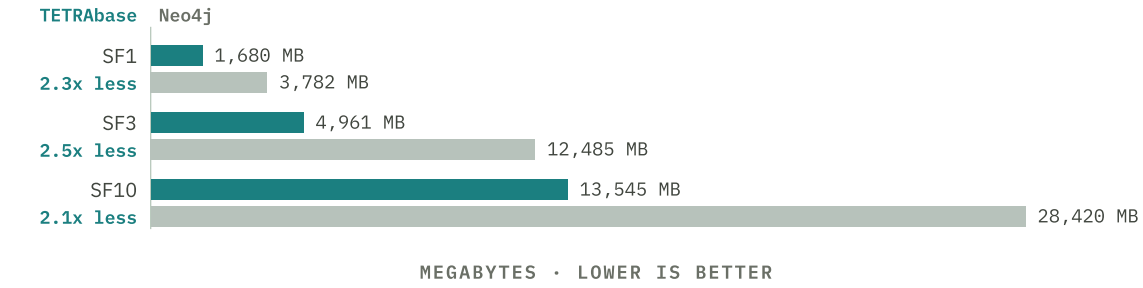

One run of the 21 queries, the sum of the medians.

### CPU for the same run



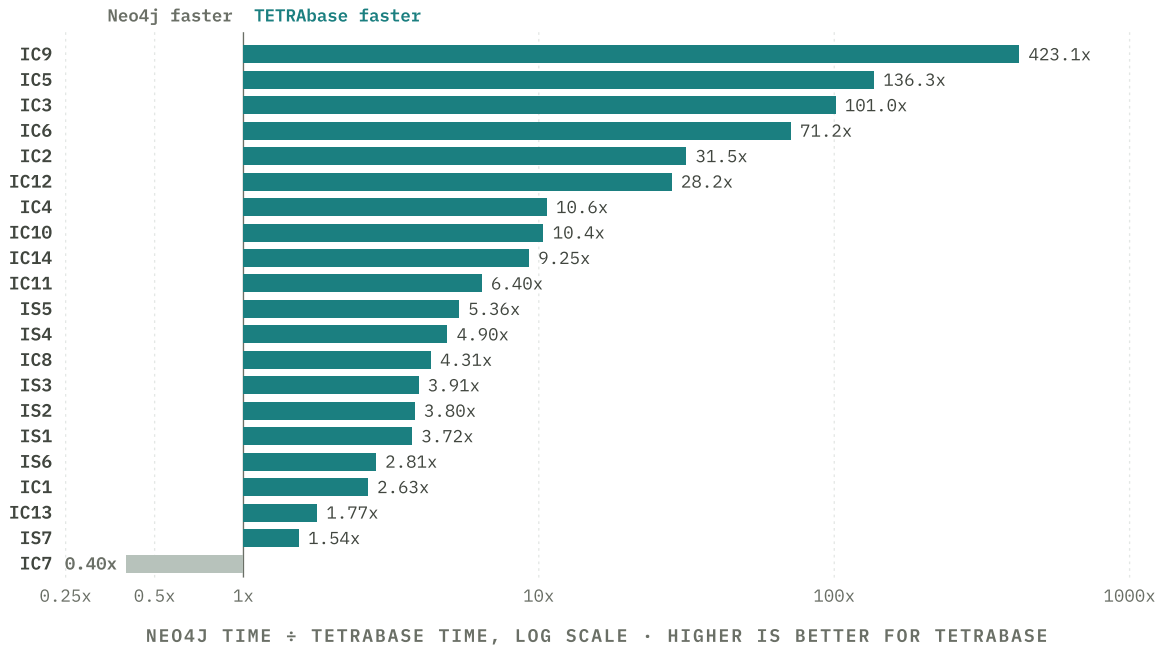

Server CPU, as the operating system reports it.

### Memory at peak




Largest server memory during the run.

## Query by query, SF10




Neo4j's median time divided by TETRAbase's, per query. Log scale. 20 of 21 queries run faster on TETRAbase; the largest gains come on queries that reach friends of friends.

SECTION 02

# Executive summary

TETRAbase vs Neo4j, OpenCypher over Bolt.

| TEST | TETRABASE AGAINST NEO4J |
|---|---|
| **LDBC total time, SF10** | 1.72 s against 113.7 s: 66x faster |
| **LDBC CPU per run, SF10** | 9,392 ms against 121,665 ms: 13x less |
| **LDBC, SF1 to SF100** | All 21 queries answered at every size, up to 1.78 billion edges |
| **LDBC row counts** | Same as Neo4j on all 21 queries |
| **Reads per second, 8 readers, SF1** | 130.2 against 12.6: 10.3x |
| **Reads per second, 8 readers + 16 writers** | 44.3 against 11.5: 3.9x |
| **Writes per second, 8 readers + 16 writers** | 3,116 against 8,844 |
| **Write workloads, 32 clients** | 13 of 18 as fast as or faster than Neo4j |
| **Store after the writes** | 547 MB against 4,308 MB: 7.9x smaller |
| **Wrong answers under load** | 0 of 41,445 reads |
| **Since 2026-10-12** | Total LDBC time down 20% to 33% at every size |

## TETRAbase

- The first Matrix Database: stores data as a matrix and runs each query as operations on it.
- Speaks OpenCypher over Bolt, Neo4j's binary protocol, or Connect RPC. The same drivers and queries run on both engines.
- No indexes for any read in this report.
- Store always encrypted at rest.

## Scope

- One run on one server, 2026-11-01.
- Neo4j 5.26 community, measured on the same server and stored.

SECTION 03

# Summary

The headline numbers, side by side.

| | TETRABASE | NEO4J |
|---|---|---|
| LDBC total time, SF1 | **392.7 ms** | 11,582.3 ms |
| LDBC total time, SF3 | **752.5 ms** | 33,989.8 ms |
| LDBC total time, SF10 | **1,722.5 ms** | 113,666.2 ms |
| LDBC CPU per run, SF10 | **9,392 ms** | 121,665 ms |
| LDBC peak memory, SF10 | **13,545 MB** | 28,420 MB |
| LDBC queries answered faster, SF10 | **20 of 21** | 1 of 21 |
| Write workloads as fast or faster, 32 clients | **13 of 18** | 5 of 18 |
| Point write, median latency, 1 client | **0.931 ms** | 0.982 ms |
| Store after the writes | **547 MB** | 4,308 MB |
| Reads per second, 8 readers, SF1 | **130.2** | 12.6 |
| Reads per second, 8 readers + 16 writers | **44.3** | 11.5 |
| Writes per second, 8 readers + 16 writers | 3,116 | **8,844** |
| Wrong answers under load | **0 of 41,445** | 0 of 3,750 |
| Indexes for the LDBC reads | **none** | 9 constraints, 8 indexes |

LDBC total time: the sum of the 21 query medians. Neo4j ran at SF1 to SF10 for reads, 200,000 rows for writes and SF1 for reads and writes together.

SECTION 04

# The LDBC Social Network Benchmark

What the datasets are, where they come from, and why we test with them.

| | |
|---|---|
| **Who publishes it** | The Linked Data Benchmark Council (LDBC), an independent non-profit of database vendors and researchers. ldbcouncil.org |
| **What it models** | A social network: people, who knows whom, forums, posts, comments, tags and likes, and where people live, study and work. |
| **How LDBC builds it** | A data generator with a fixed recipe. The same scale factor always produces the same graph, so anyone can rebuild our datasets. |
| **Why it looks real** | A few people have huge friend lists and most have small ones. Names and interests follow country. Activity spikes around events. |
| **What it tests** | 21 Interactive read queries, built to hit the hard parts of graph queries: multi-hop traversals, large fan-outs, top-k sorting, aggregation and shortest paths. |
| **Why we use it** | The public yardstick for graph databases. Open data, open queries and open reference implementations: every engine answers the same questions, and anyone can check the results. |

## Scale factors

- The scale factor (SF) sets the size of the generated data: SF1 is about 1 GB raw, SF100 about 100 GB.
- Each step adds more people and far more posts and comments. Tags stay fixed.
- Five sizes show how query time grows as the graph grows 103x.

| | SF1 | SF3 | SF10 | SF30 | SF100 |
|---|---|---|---|---|---|
| Raw data, approximately | 1 GB | 3 GB | 10 GB | 30 GB | 100 GB |
| People | 9,892 | 24,328 | 65,645 | 165,430 | 448,626 |
| Forums | 90,492 | 221,792 | 595,453 | 1,503,655 | 4,080,604 |
| Posts | 1,003,605 | 2,597,141 | 7,435,696 | 19,968,658 | 57,987,023 |
| Comments | 2,052,169 | 6,413,095 | 21,865,475 | 67,126,524 | 220,096,052 |
| Tags | 16,080 | 16,080 | 16,080 | 16,080 | 16,080 |
| Nodes, all types | **3,181,724** | **9,281,922** | **29,987,835** | **88,789,833** | **282,637,871** |
| Edges, all types | **17,256,038** | **52,695,734** | **176,623,433** | **540,915,383** | **1,775,513,741** |
| Edges against SF1 | 1.0x | 3.1x | 10.2x | 31.3x | 102.9x |

Counts from our census of the stores this run loaded.

## The 21 queries

- **IS1 to IS7, short reads:** look up one person or one message and its neighbours.
- **IC1 to IC14, complex reads:** walk friends and friends of friends, then sort or count what they find.
- We run the LDBC reference implementation of every query, unmodified.

| QUERY | WHAT IT ASKS |
|---|---|
| **IS1** | Profile of a person |
| **IS2** | A person's recent messages |
| **IS3** | A person's friends |
| **IS4** | Content of a message |
| **IS5** | Creator of a message |
| **IS6** | Forum of a message |
| **IS7** | Replies to a message |
| **IC1** | Friends up to three hops away with a given first name |
| **IC2** | Recent messages by your friends |
| **IC3** | Friends and friends of friends who have been to two given countries |
| **IC4** | New topics among your friends' posts |
| **IC5** | New groups your friends and friends of friends joined |
| **IC6** | Tags that appear together in friends' posts |
| **IC7** | Recent likers of your messages |
| **IC8** | Recent replies to your messages |
| **IC9** | Recent messages by friends and friends of friends |
| **IC10** | Friend recommendation |
| **IC11** | Job referral through friends |
| **IC12** | Expert search over friends' replies |
| **IC13** | Shortest path between two people |
| **IC14** | Trusted connection paths between two people |

SECTION 05

# What we measured

Four tests, one interface, one server.

| TEST | WHAT RUNS | SIZES | NEO4J RAN AT |
|---|---|---|---|
| **Reads** | The 21 LDBC queries, one client | SF1 to SF100 | SF1, SF3, SF10 |
| **Writes** | 18 OpenCypher write workloads, 1 to 32 clients | 20,000, 50,000 and 200,000 rows | 200,000 rows |
| **Reads and writes together** | 8 LDBC readers with 0, 1, 4 or 16 writers, 60 seconds per window | SF1 to SF100 | SF1 |
| **Scaling** | The 7 short LDBC queries at 1 to 32 clients | SF1 to SF100 | not run |

## Terms

| TERM | MEANING |
|---|---|
| Median | The middle time over repeated runs of a query, after warm-up runs. |
| Total time | The sum of the 21 query medians. |
| Point, block, range | A write that touches 1 row, 1,000 rows from a list, or a range of 1,000 keys. |
| Rows per second | Write throughput. A point statement counts 1 row; a block or range statement 1,000. |
| Read-back | After each write test, a new session reads the rows back and compares them with what the clients sent. |
| Wrong answer | A read under load whose rows differ from the same query's answer with no writers running. |
| Octave | One doubling of throughput. Perfect scaling from 1 client to 32 gains 5 octaves. |
| p99 | The time 99% of statements beat. |
| Cold load | Time from process start to the first query, with the page cache dropped. |

## The 18 write workloads

- Each workload writes its own label.
- Each client writes its own keys: clients never wait on each other's locks.
- Each test: 1 second of warm-up, 3 seconds timed, then read back every row written.
- Every pass writes new values: no statement rewrites a value already there.

| WORKLOAD | STATEMENT |
|---|---|
| **create-node.point.props** | `CREATE (:CN {id: $id, name: $name, age: $age, score: $score})` |
| **create-node.point.bare** | `CREATE (:CB {id: $id})` |
| **create-node.block.*** | `UNWIND $rows AS r CREATE (:CN {...})`, 1,000 rows |
| **create-edge.point.*** | `MATCH (a:CV {id: $a}), (b:CV {id: $b}) CREATE (a)-[:CR {w: $w}]->(b)` |
| **create-edge.block.*** | `UNWIND $pairs AS p MATCH (a:CV {id: p.a}), (b:CV {id: p.b}) CREATE ...` |
| **update.point.blind** | `MATCH (x:CU {id: $k}) SET x.score = $c` |
| **update.*.incr** | `SET x.hits = x.hits + 1` |
| **update.*.row** | `SET x.v = x.id * $c + $p` |
| **update.range.*** | `MATCH (x:CU) WHERE x.id >= $lo AND x.id < $hi SET ...` |
| **upsert.point** | `MERGE (x:CM {id: $k}) SET x.u = $u` |
| **upsert.block** | `UNWIND $rows AS r MERGE (x:CM {id: r.id}) SET x.u = r.u` |
| **delete.point** | `MATCH (x:CDP {id: $k}) DELETE x` |
| **delete.range** | `MATCH (x:CDB) WHERE x.id >= $lo AND x.id < $hi DELETE x` |

props: creates set properties. bare: creates set only the key.

## The writer cycle, reads and writes together

- Each writer runs seven OpenCypher statements in turn: create, relate, set, increment, set from row, upsert, delete.
- Writers never touch what the readers read.

SECTION 06

# Test conditions

Where the two setups differ, and which engine each difference helps.

| | TETRABASE | NEO4J | HELPS |
|---|---|---|---|
| Queries and parameters | same | same | neither |
| Indexes, LDBC reads | none | 9 constraints, 8 indexes | Neo4j |
| Constraints, writes | 6 unique | 6 unique | neither |
| Durability | journal fsync before confirming; concurrent commits share one fsync | transaction log flush on every commit | neither |
| Encryption at rest | always on | off | Neo4j |
| Memory | defaults | page cache and heap sized per graph size | Neo4j |
| Server | same machine | same machine | neither |
| Client | same machine, loopback | same machine, loopback | neither |
| Measured on | 2026-11-01 | 2026-09-24 reads; 2026-10-26 writes | neither |

## Neo4j memory settings

| | SF1 | SF3 | SF10 |
|---|---|---|---|
| Page cache | 4GB | 8GB | 24GB |
| Heap | 2GB to 8GB | 8GB | 16GB |

SECTION 07

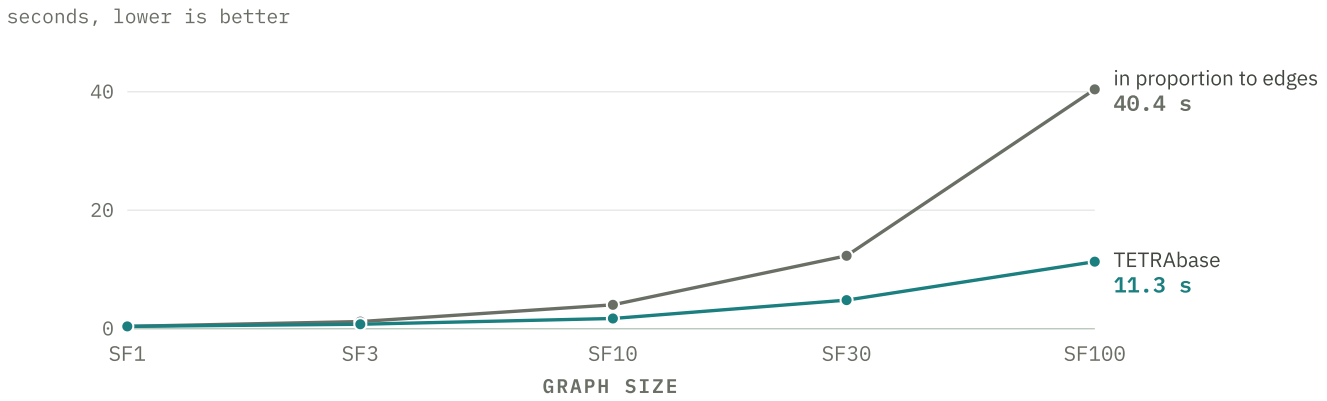
# Reads at five graph sizes

The 21 LDBC queries from 17 million to 1.78 billion edges.

| | SF1 | SF3 | SF10 | SF30 | SF100 |
|---|---|---|---|---|---|
| Edges | 17,256,038 | 52,695,734 | 176,623,433 | 540,915,383 | 1,775,513,741 |
| Store on disk | 0.34 GiB | 1.04 GiB | 3.61 GiB | 11.56 GiB | 38.88 GiB |
| Bytes per edge | 20.8 | 21.3 | 21.9 | 22.9 | 23.5 |
| Cold load | 1.1 s | 3.3 s | 10.3 s | 8.3 s | 31.6 s |
| Memory at ready | 777 MB | 2,168 MB | 6,927 MB | 9,957 MB | 34,162 MB |
| Memory at peak | 1,680 MB | 4,961 MB | 13,545 MB | 25,149 MB | 62,679 MB |
| Total time, 21 queries | **392.7 ms** | **752.5 ms** | **1,722.5 ms** | **4,822.7 ms** | **11,312.5 ms** |
| CPU per run | 1,058 ms | 2,891 ms | 9,392 ms | 29,593 ms | 100,545 ms |
| Queries answered | **21 of 21** | **21 of 21** | **21 of 21** | **21 of 21** | **21 of 21** |

Every store is encrypted and has no index.

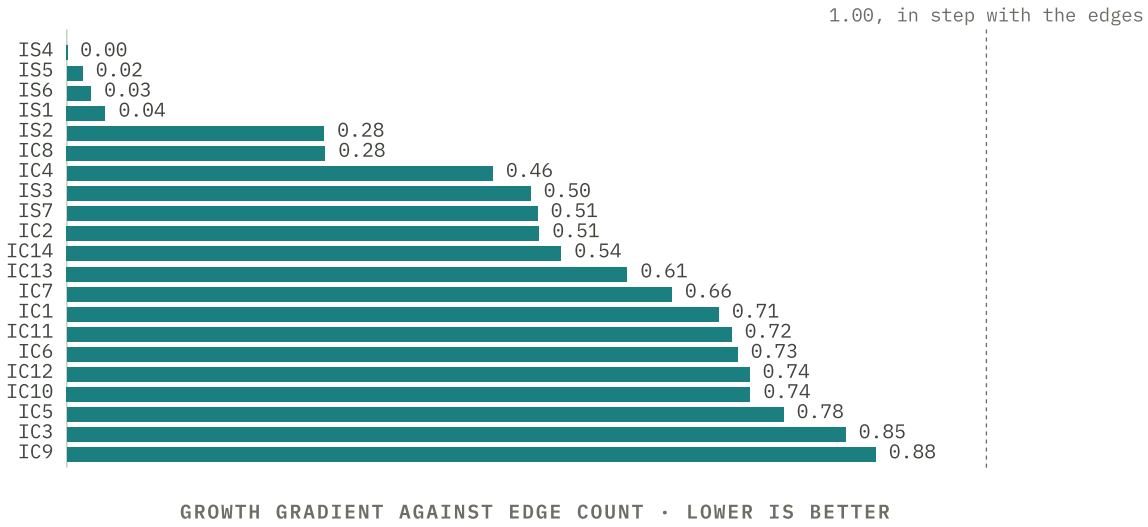
## Total time by graph size




Grey: the SF1 time grown in step with the edge count. Horizontal axis: edges, log scale.

## How each query grows with the graph

- Gradient 1.00: time grows in step with the edges.
- Gradient 0: time stays flat as the graph grows.
- All 21 queries sit below 1.00.




Slope of log median time against log edge count, SF1 to SF100.

## Since our 2026-10-12 report

| | SF1 | SF3 | SF10 | SF30 | SF100 |
|---|---|---|---|---|---|
| Total time, 2026-10-12 | 504.5 ms | 1,083.9 ms | 2,570.8 ms | 6,324.4 ms | 14,209.7 ms |
| Total time, 2026-11-01 | **392.7 ms** | **752.5 ms** | **1,722.5 ms** | **4,822.7 ms** | **11,312.5 ms** |
| Change | **-22%** | **-31%** | **-33%** | **-24%** | **-20%** |
| CPU per run, 2026-10-12 | 1,291 ms | 3,505 ms | 10,243 ms | 33,557 ms | 107,350 ms |
| CPU per run, 2026-11-01 | **1,058 ms** | **2,891 ms** | **9,392 ms** | **29,593 ms** | **100,545 ms** |

Same harness, parameters and server. Appendix C: each query.

SECTION 08

# TETRAbase vs Neo4j: reads

The 21 LDBC queries at SF1, SF3 and SF10, with the same parameters on both engines.

| | SF1 | SF3 | SF10 |
|---|---|---|---|
| Edges | 17,256,038 | 52,695,734 | 176,623,433 |
| TETRAbase total time | **392.7 ms** | **752.5 ms** | **1,722.5 ms** |
| Neo4j total time | 11,582.3 ms | 33,989.8 ms | 113,666.2 ms |
| TETRAbase faster by | **29.5x** | **45.2x** | **66.0x** |
| TETRAbase CPU per run | **1,058 ms** | **2,891 ms** | **9,392 ms** |
| Neo4j CPU per run | 12,326 ms | 40,725 ms | 121,665 ms |
| TETRAbase uses less CPU by | **11.7x** | **14.1x** | **13.0x** |
| TETRAbase peak memory | **1,680 MB** | **4,961 MB** | **13,545 MB** |
| Neo4j peak memory | 3,782 MB | 12,485 MB | 28,420 MB |
| Queries TETRAbase answers faster | **20 of 21** | **20 of 21** | **20 of 21** |
| Same row counts | **21 of 21** | **21 of 21** | **21 of 21** |

SECTION 09

# Writes

18 OpenCypher workloads at 200,000 rows, at 1 client and at 32.

**13/18**
**AS FAST OR FASTER AT 32 CLIENTS**
11 of 18 at 1 client

**3.93x**
**LARGEST LEAD**
block relationship creates, 1 client

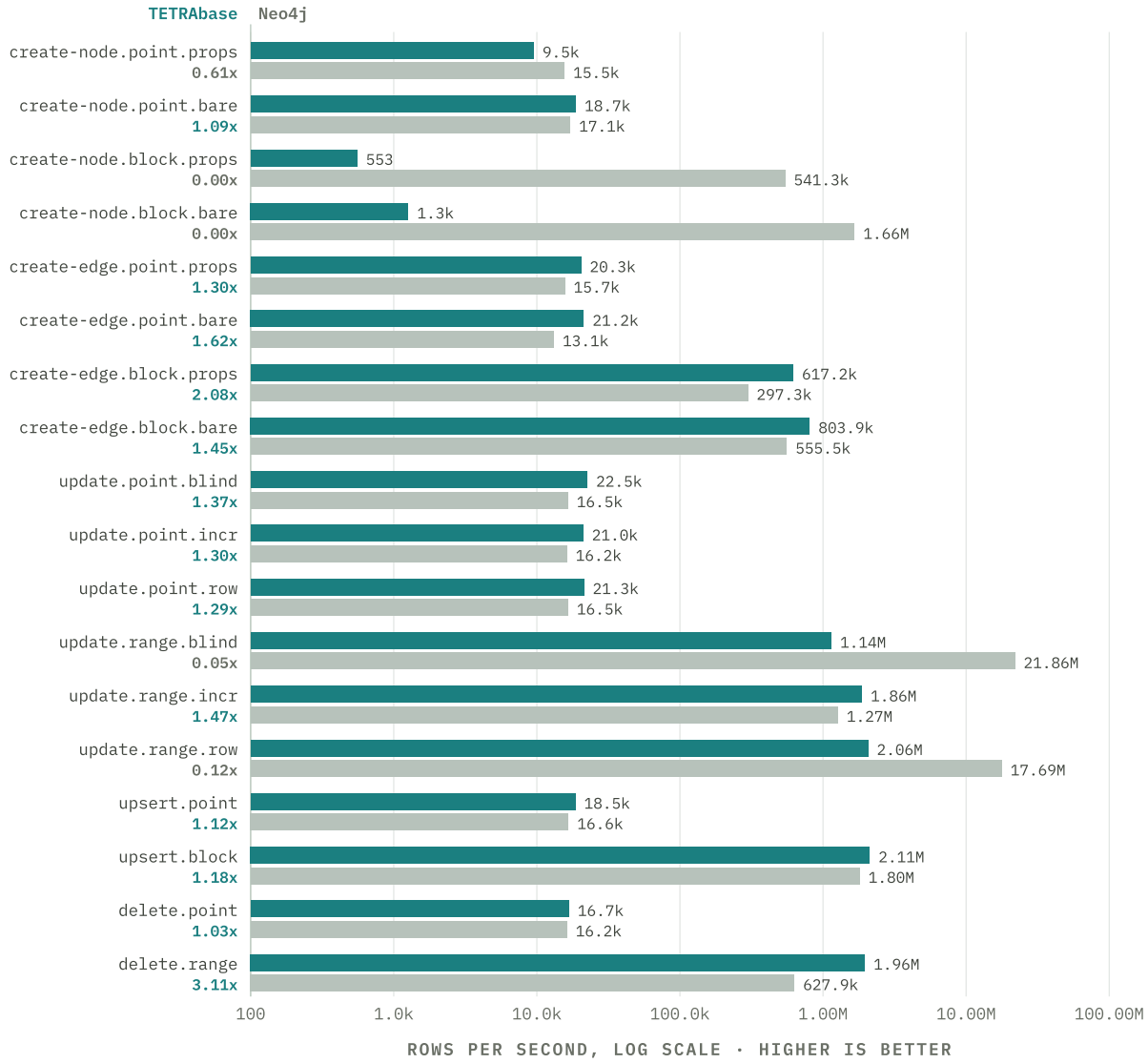
**7.9x**
**SMALLER STORE**
547 MB against 4,308 MB

**3.1x**
**LESS SERVER MEMORY**
4,031 MB against 12,301 MB

| WORKLOAD | T C=1 | N C=1 | T ÷ N | T C=32 | N C=32 | T ÷ N |
|---|---|---|---|---|---|---|
| create-node.point.props | **1.0k** | 932 | 1.10x | 9.5k | **15.5k** | 0.61x |
| create-node.point.bare | **1.0k** | 889 | 1.15x | **18.7k** | 17.1k | 1.09x |
| create-node.block.props | 531 | **76.1k** | 0.01x | 553 | **541.3k** | 0.00x |
| create-node.block.bare | 1.4k | **201.7k** | 0.01x | 1.3k | **1.66M** | 0.00x |
| create-edge.point.props | **1.0k** | 986 | 1.05x | **20.3k** | 15.7k | 1.30x |
| create-edge.point.bare | **1.4k** | 974 | 1.45x | **21.2k** | 13.1k | 1.62x |
| create-edge.block.props | **70.3k** | 38.2k | 1.84x | **617.2k** | 297.3k | 2.08x |
| create-edge.block.bare | **63.1k** | 16.1k | 3.93x | **803.9k** | 555.5k | 1.45x |
| update.point.blind | 1.2k | **1.5k** | 0.84x | **22.5k** | 16.5k | 1.37x |
| update.point.incr | **1.2k** | 1.1k | 1.10x | **21.0k** | 16.2k | 1.30x |
| update.point.row | **1.5k** | 1.1k | 1.43x | **21.3k** | 16.5k | 1.29x |
| update.range.blind | 280.7k | **1.08M** | 0.26x | 1.14M | **21.86M** | 0.05x |
| update.range.incr | **484.8k** | 176.2k | 2.75x | **1.86M** | 1.27M | 1.47x |
| update.range.row | 324.5k | **905.3k** | 0.36x | 2.06M | **17.69M** | 0.12x |
| upsert.point | 756 | **887** | 0.85x | **18.5k** | 16.6k | 1.12x |
| upsert.block | **177.4k** | 127.6k | 1.39x | **2.11M** | 1.80M | 1.18x |
| delete.point | 814 | **830** | 0.98x | **16.7k** | 16.2k | 1.03x |
| delete.range | **259.6k** | 110.5k | 2.35x | **1.96M** | 627.9k | 3.11x |

Rows per second. T: TETRAbase. N: Neo4j. Appendix D: every client count.

## Rows per second at 32 clients, 200,000 rows




Under each name: TETRAbase's rate divided by Neo4j's.

## Latency and CPU per statement

| | TETRABASE | NEO4J |
|---|---|---|
| Median point write, 1 client | **0.931 ms** | 0.982 ms |
| Median point write, 32 clients | **1.578 ms** | 1.856 ms |

| WORKLOAD | P50 C=1 T | N | P50 C=32 T | N | P99 C=32 T | N | CPU C=32 T | N |
|---|---|---|---|---|---|---|---|---|
| **create-node.point.props** | **0.82** | 0.95 | 3.25 | **1.93** | 9.81 | **3.33** | 0.70 | **0.43** |
| **create-node.point.bare** | **1.01** | 1.01 | **1.61** | 1.80 | 4.76 | **3.12** | 0.43 | **0.40** |
| **create-node.block.props** | 1,846 | **12.27** | 57,848 | **57.15** | 57,849 | **99.02** | 1,904 | **28.56** |
| **create-node.block.bare** | 891 | **4.89** | 25,349 | **18.91** | 25,349 | **47.25** | 863 | **8.46** |
| **create-edge.point.props** | **0.95** | 1.08 | **1.61** | 1.96 | 3.68 | **3.24** | **0.31** | 0.48 |
| **create-edge.point.bare** | **0.61** | 0.98 | **1.51** | 2.14 | **2.61** | 4.19 | **0.25** | 0.54 |
| **create-edge.block.props** | **13.54** | 24.79 | **49.20** | 78.75 | **139** | 364 | **23.84** | 66.70 |
| **create-edge.block.bare** | **15.44** | 48.28 | **37.71** | 56.46 | **62.92** | 97.17 | **21.60** | 43.09 |
| **update.point.blind** | 0.93 | **0.65** | **1.23** | 1.85 | **2.71** | 3.26 | **0.24** | 0.38 |
| **update.point.incr** | **0.89** | 0.98 | **1.14** | 1.86 | **2.76** | 3.34 | **0.27** | 0.38 |
| **update.point.row** | **0.59** | 0.97 | **1.18** | 1.81 | **2.73** | 3.44 | **0.27** | 0.38 |
| **update.range.blind** | 3.47 | **0.90** | 24.90 | **1.44** | 89.22 | **2.71** | 7.96 | **1.33** |
| **update.range.incr** | **2.00** | 5.60 | **13.27** | 27.07 | 71.50 | **34.51** | **3.69** | 5.80 |
| **update.range.row** | 3.03 | **1.09** | 13.80 | **1.79** | 54.20 | **3.05** | 3.91 | **1.68** |
| **upsert.point** | 1.24 | **1.01** | **1.58** | 1.83 | 4.40 | **3.31** | 0.56 | **0.38** |
| **upsert.block** | **5.30** | 6.38 | **11.86** | 16.99 | 103 | **21.71** | **6.27** | 17.06 |
| **delete.point** | 1.12 | **1.07** | **1.58** | 1.87 | **2.66** | 3.33 | **0.35** | 0.43 |
| **delete.range** | **3.87** | 7.93 | **15.48** | 35.65 | **16.23** | 50.76 | 12.19 | **8.12** |

Milliseconds per statement, 200,000 rows. T: TETRAbase. N: Neo4j. CPU: server CPU per statement. Median point write: the median over the eight single-row workloads.

## How writes scale with clients

| WORKLOAD | T ROWS/S C=1 | T OCTAVES | N ROWS/S C=1 | N OCTAVES |
|---|---|---|---|---|
| **create-node.point.props** | 1.0k | 3.21 | 932 | **4.06** |
| **create-node.point.bare** | 1.0k | 4.19 | 889 | **4.26** |
| **create-node.block.props** | 531 | 0.06 | 76.1k | **2.83** |
| **create-node.block.bare** | 1.4k | -0.10 | 201.7k | **3.04** |
| **create-edge.point.props** | 1.0k | **4.29** | 986 | 3.99 |
| **create-edge.point.bare** | 1.4k | **3.90** | 974 | 3.75 |
| **create-edge.block.props** | 70.3k | **3.13** | 38.2k | 2.96 |
| **create-edge.block.bare** | 63.1k | 3.67 | 16.1k | **5.11** |
| **update.point.blind** | 1.2k | **4.18** | 1.5k | 3.47 |
| **update.point.incr** | 1.2k | **4.13** | 1.1k | 3.89 |
| **update.point.row** | 1.5k | 3.80 | 1.1k | **3.95** |
| **update.range.blind** | 280.7k | 2.02 | 1.08M | **4.34** |
| **update.range.incr** | 484.8k | 1.94 | 176.2k | **2.85** |
| **update.range.row** | 324.5k | 2.67 | 905.3k | **4.29** |
| **upsert.point** | 756 | **4.61** | 887 | 4.22 |
| **upsert.block** | 177.4k | 3.57 | 127.6k | **3.82** |
| **delete.point** | 814 | **4.36** | 830 | 4.28 |
| **delete.range** | 259.6k | **2.91** | 110.5k | 2.51 |

T: TETRAbase. N: Neo4j. Octaves gained from 1 client to 32; perfect scaling: 5.00.

## Load and storage

| | ROWS LOADED | LOAD | STORE AFTER WRITES | SERVER PEAK |
|---|---|---|---|---|
| TETRAbase, 20,000 rows | 100,000 | 11.9 s | 499 MB | 2,999 MB |
| TETRAbase, 50,000 rows | 250,000 | 37.0 s | 550 MB | 3,444 MB |
| TETRAbase, 200,000 rows | 1,000,000 | 543.3 s | 547 MB | 4,031 MB |
| Neo4j, 200,000 rows | 1,000,000 | 3.7 s | 4,308 MB | 12,301 MB |

Each engine loads five labels at the test's row count before the writes start.

SECTION 10

# Write verification

We read back every write test's rows.

| | TESTS | READ BACK EXACTLY | MISMATCHED |
|---|---|---|---|
| TETRAbase, 20,000 rows | 108 | 102 | 6 |
| TETRAbase, 50,000 rows | 108 | 107 | 1 |
| TETRAbase, 200,000 rows | 108 | **108** | 0 |
| Neo4j, 200,000 rows | 108 | 107 | 1 |

## Mismatched tests

| ENGINE | ROWS | WORKLOAD | CLIENTS | READ BACK |
|---|---|---|---|---|
| TETRAbase | 20,000 | **create-edge.block.bare** | 16 | 1,907,000 of 2,052,000 relationships |
| TETRAbase | 20,000 | **create-edge.block.bare** | 32 | 2,916,000 of 3,688,000 relationships |
| TETRAbase | 20,000 | **update.range.blind** | 32 | 625 of 1,000 rows with the new value, one client |
| TETRAbase | 20,000 | **update.range.incr** | 32 | 6,432,125 of 8,081,000 increments |
| TETRAbase | 20,000 | **update.range.row** | 32 | 625 of 1,000 rows with the new value, one client |
| TETRAbase | 20,000 | **upsert.block** | 32 | 20,375 rows in the label, expected 20,000 |
| TETRAbase | 50,000 | **update.point.blind** | 2 | 3,139 of 3,209 rows with the new value, one client |
| Neo4j | 200,000 | **create-edge.block.props** | 32 | 3 statements deadlocked |

- Each mismatched test read back the same result three times, the last 5 seconds after its final write.
- The Neo4j test failed because three of its statements deadlocked.
- Every ratio in this report uses 200,000-row tests.

SECTION 11

# Reads and writes together

8 LDBC readers and up to 16 OpenCypher writers on the SF1 graph.

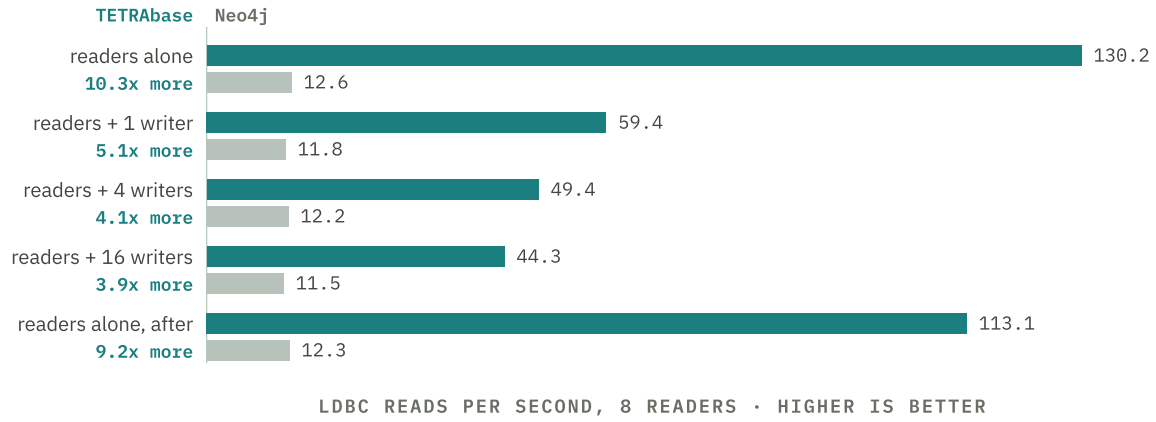
| WINDOW | READS/S T | N | READ TIME T | N | WRITES/S T | N | WRITE P99 T | N |
|---|---|---|---|---|---|---|---|---|
| readers alone | **130.2** | 12.6 | | | | | | |
| 1 writer alone | | | | | 326 | **1,067** | 16.4 | **1.7** |
| 4 writers alone | | | | | 1,076 | **2,916** | 24.4 | **2.6** |
| 16 writers alone | | | | | 3,017 | **9,219** | 28.1 | **3.0** |
| readers + 1 writer | **59.4** | 11.8 | +113.1% | +5.4% | 283 | **1,039** | 24.5 | **1.8** |
| readers + 4 writers | **49.4** | 12.2 | +147.9% | +2.2% | 934 | **2,859** | 25.9 | **2.6** |
| readers + 16 writers | **44.3** | 11.5 | +177.2% | +7.8% | 3,116 | **8,844** | 27.4 | **3.2** |
| readers alone, after | **113.1** | 12.3 | +13.0% | +0.0% | | | | |

T: TETRAbase. N: Neo4j. Read time: change in total read time against readers alone. Write p99 in milliseconds. 60 seconds per window.

## Correctness under load

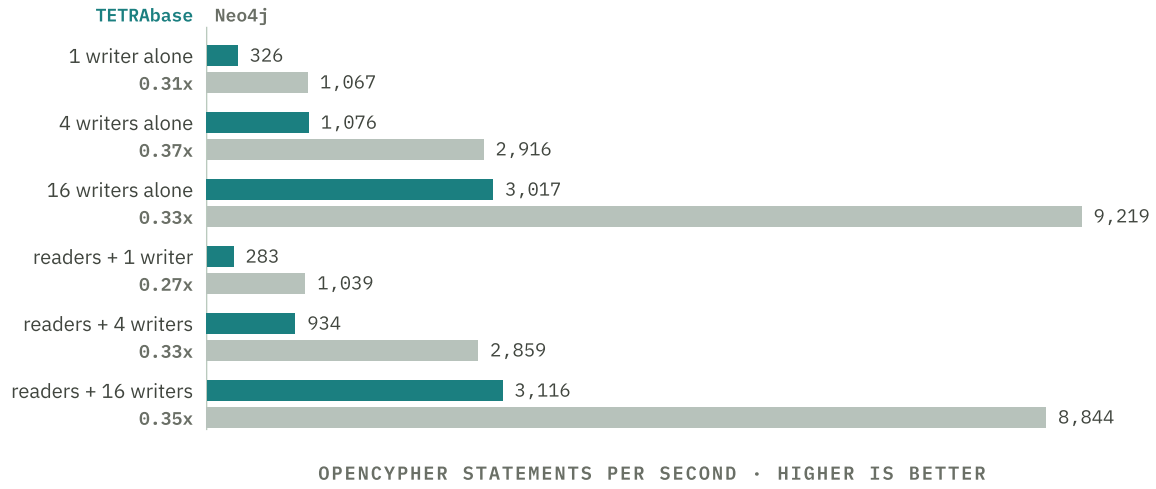
| | TETRABASE | NEO4J |
|---|---|---|
| Reads checked, SF1 | 24,000 | 3,750 |
| Wrong answers | **0** | 0 |
| Write windows that read back correctly | **6 of 6** | 6 of 6 |

## Reads per second at SF1




Runs of the 21 LDBC queries per second, across all eight readers.

## Writes per second at SF1




Statements per second from the seven-statement writer cycle. Under each window: TETRAbase's rate divided by Neo4j's.

## Write time by statement

| STATEMENT | P50 TETRABASE | P50 NEO4J | P99 TETRABASE | P99 NEO4J |
|---|---|---|---|---|
| create | 3.09 | **1.67** | 18.05 | **3.03** |
| relate | **1.64** | 1.72 | 14.51 | **3.21** |
| blind | 1.68 | **1.62** | 5.69 | **2.88** |
| incr | **1.61** | 1.67 | 5.95 | **2.87** |
| row | **1.59** | 1.65 | 13.18 | **2.85** |
| upsert | **1.64** | 1.66 | 5.46 | **2.85** |
| delete | 19.50 | **1.71** | 112.80 | **2.99** |

Milliseconds per statement, 16 writers alone, SF1.

## Reads under 16 writers

| READ | T ALONE | T + 16 | SLOWDOWN | N ALONE | N + 16 | SLOWDOWN | N ÷ T + 16 |
|---|---|---|---|---|---|---|---|
| **IC11** | 8.95 | 302.82 | 33.9x | 51.24 | 57.19 | 1.12x | **0.19x** |
| **IC6** | 64.14 | 823.01 | 12.8x | 1845.00 | 1955.38 | 1.06x | **2.38x** |
| **IS6** | 0.37 | 4.55 | 12.4x | 0.61 | 0.34 | 0.55x | **0.07x** |
| **IC12** | 49.13 | 454.58 | 9.25x | 144.93 | 176.84 | 1.22x | **0.39x** |
| **IC4** | 38.51 | 348.80 | 9.06x | 286.65 | 308.05 | 1.07x | **0.88x** |
| **IS7** | 0.65 | 5.01 | 7.69x | 0.93 | 0.69 | 0.74x | **0.14x** |
| **IC1** | 2.72 | 9.18 | 3.37x | 4.06 | 3.38 | 0.83x | **0.37x** |
| **IC8** | 8.33 | 19.16 | 2.30x | 17.26 | 17.25 | 1.00x | **0.90x** |

Median milliseconds at SF1, the eight reads that slowed most on TETRAbase. T: TETRAbase. N: Neo4j. Appendix F: all 21.

## TETRAbase at every graph size

| | READS/S ALONE | + 16 WRITERS | READ TIME | WRITES/S ALONE | + READERS | WRONG ANSWERS | READ-BACK | PEAK |
|---|---|---|---|---|---|---|---|---|
| SF1 | 130.2 | 44.3 | +177.2% | 3,017 | 3,116 | 0 of 24,000 | 6 of 6 | 6,867 MB |
| SF3 | 46.2 | 25.3 | +75.1% | 1,596 | 1,236 | 0 of 10,879 | 6 of 6 | 16,819 MB |
| SF10 | 16.3 | 10.9 | +41.4% | 519 | 445 | 0 of 4,406 | 6 of 6 | 44,052 MB |
| SF30 | 6.0 | 3.7 | +48.9% | 166 | 133 | 0 of 1,762 | 6 of 6 | 61,272 MB |
| SF100 | 0.8 | 0.8 | +2.0% | 46 | 30 | 0 of 398 | 6 of 6 | 137,876 MB |

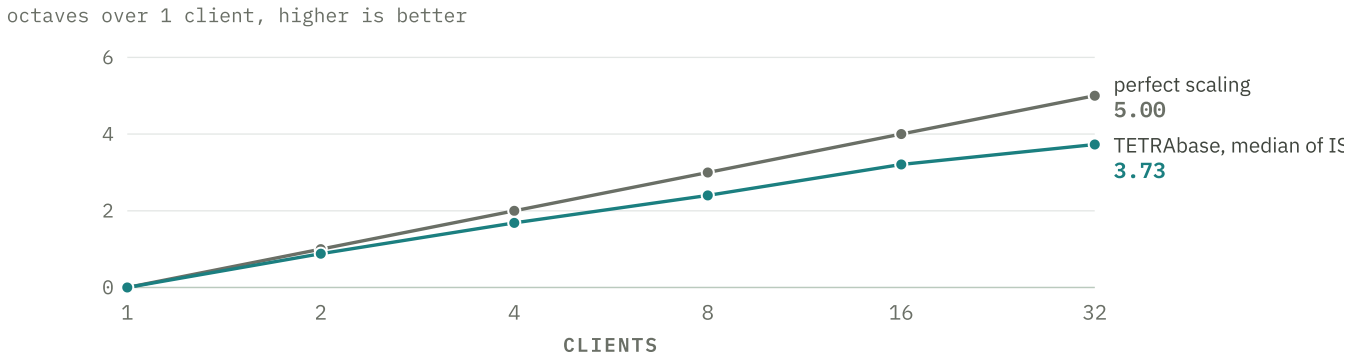
Neo4j did not run above SF1. Read time: change in total read time from readers alone to readers with 16 writers.
Writes: 16 writers. Peak: server memory across all windows.

SECTION 12

# Scaling with clients

Throughput gained as clients double, in octaves.

## Short reads at SF1




One octave: one doubling of throughput.

| QUERY | OCTAVES AT 32, SF1 | SF3 | SF10 | SF30 | SF100 | RUNS/S C=1 | C=32 |
|---|---|---|---|---|---|---|---|
| **IS1** | 3.73 | 3.70 | 3.62 | 3.72 | 3.07 | 4,713 | 62,541 |
| **IS2** | 4.47 | 4.29 | 4.28 | 4.51 | 2.93 | 564 | 12,504 |
| **IS3** | 2.86 | 3.05 | 2.80 | 3.34 | 3.71 | 328 | 2,376 |
| **IS4** | 3.58 | 3.62 | 3.89 | 3.72 | 3.71 | 7,115 | 85,078 |
| **IS5** | 3.83 | 3.97 | 3.98 | 4.29 | 3.80 | 6,182 | 88,218 |
| **IS6** | 3.76 | 4.00 | 4.52 | 3.53 | 3.62 | 3,061 | 41,564 |
| **IS7** | 3.55 | 2.69 | 3.47 | 2.69 | 2.99 | 2,192 | 25,751 |

Runs per second at SF1. 0 wrong answers at every size and client count.

SECTION 13

# Limitations

What these figures do not cover.

| | |
|---|---|
| **Stored Neo4j results** | Measured on 2026-09-24 and 2026-10-26, not alongside TETRAbase. |
| **Neo4j coverage** | SF1 to SF10 for reads, SF1 for reads and writes together, 200,000 rows for writes. |
| **One server** | No cluster, no replication. Clients on the same machine. |
| **Short write tests** | 3 timed seconds each. Effects that build over minutes do not show. |
| **Small samples at SF100 under load** | 2 to 6 runs per query per window. |
| **Medians** | Every latency is a median unless marked p99. |

SECTION 14

# Rig and reproduction

Server, software and harness.

```
commit      a51f2d6e
run         2026-11-01T22:46:52Z
server      rescue, AMD EPYC 7502P, 32 cores / 64 threads, Zen 2
harness     ops/epyc/bench/bench.sh
binaries    ldbc 11cc45f93635d360  tetra 30c8b6f14055d51d
            wladder ce56582bd04796ae
TETRAbase   no index for any read; store encrypted at rest
            journal fsync before confirming; concurrent commits share one fsync
LDBC        SNB Interactive v1 reference queries, unmodified
            SF1 3 10 30 100, up to 282,637,871 nodes and 1,775,513,741 edges
            the same parameters on both engines
writes      18 OpenCypher workloads x clients 1 2 4 8 16 32 x 20,000 50,000 200,000 rows
            1 s warm-up, 3 s timed, read-back on a new session
concurrency 8 LDBC readers, 0 1 4 16 writers, 60 s windows
scaling     IS1 to IS7 at clients 1 2 4 8 16 32, every graph size
neo4j       5.26.0 community, OpenJDK 17, page cache and heap per graph size
            LDBC reference setup: 9 constraints + 8 indexes
            transaction log flush on every commit
            measured once on this server, stored under ops/epyc/bench/reference/
raw         results/bench/a51f2d6e/results.json, schema tetra-bench/1
```

## Reproduction

- Raw results, stored Neo4j results, query parameters and the harness: available on request.
- This document reads every figure from those files.

APPENDIX A

# Reads by query

All 21 LDBC queries at all five graph sizes.

| QUERY | SF1 | SF3 | SF10 | SF30 | SF100 | GROWTH | ROWS |
|---|---|---|---|---|---|---|---|
| **IS1** | 0.19 | 0.27 | 0.24 | 0.24 | 0.25 | 0.04 | 1 |
| **IS2** | 1.66 | 2.02 | 2.72 | 3.70 | 6.19 | 0.28 | 10 |
| **IS3** | 2.08 | 2.82 | 4.62 | 11.48 | 19.18 | 0.50 | 2,409 |
| **IS4** | 0.14 | 0.14 | 0.17 | 0.14 | 0.13 | 0.00 | 1 |
| **IS5** | 0.18 | 0.19 | 0.15 | 0.19 | 0.20 | 0.02 | 1 |
| **IS6** | 0.28 | 0.26 | 0.32 | 0.31 | 0.30 | 0.03 | 1 |
| **IS7** | 0.45 | 0.63 | 0.78 | 1.71 | 5.32 | 0.51 | 10 |
| **IC1** | 2.41 | 4.73 | 10.80 | 25.39 | 63.41 | 0.71 | 20 |
| **IC2** | 13.09 | 21.93 | 39.03 | 66.15 | 147.87 | 0.51 | 20 |
| **IC3** | 53.95 | 110.90 | 311.02 | 1,104.40 | 2,317.38 | 0.85 | 20 |
| **IC4** | 29.09 | 46.53 | 78.23 | 134.45 | 250.63 | 0.46 | 10 |
| **IC5** | 37.35 | 82.13 | 193.94 | 479.57 | 1,414.62 | 0.78 | 20 |
| **IC6** | 52.92 | 90.81 | 218.39 | 493.34 | 1,552.48 | 0.73 | 9 |
| **IC7** | 29.79 | 38.48 | 55.96 | 197.37 | 595.77 | 0.66 | 20 |
| **IC8** | 4.94 | 5.27 | 4.27 | 10.89 | 17.56 | 0.28 | 20 |
| **IC9** | 10.71 | 24.39 | 67.69 | 194.22 | 620.78 | 0.88 | 20 |
| **IC10** | 97.20 | 225.95 | 576.50 | 1,386.73 | 2,916.17 | 0.74 | 10 |
| **IC11** | 8.33 | 16.89 | 36.50 | 97.34 | 229.13 | 0.72 | 10 |
| **IC12** | 42.01 | 68.96 | 112.88 | 594.76 | 1,062.09 | 0.74 | 18 |
| **IC13** | 0.24 | 0.31 | 0.73 | 2.61 | 2.89 | 0.61 | 1 |
| **IC14** | 5.63 | 8.93 | 7.53 | 17.74 | 90.15 | 0.54 | 90 |
| **Total** | **392.7** | **752.5** | **1,722.5** | **4,822.7** | **11,312.5** | 0.74 | |

Median milliseconds. Growth: gradient against edge count. Rows: at SF100.

APPENDIX B

# Reads, TETRAbase vs Neo4j

Median milliseconds at SF1, SF3 and SF10.

| QUERY | SF1 T | N | SF3 T | N | SF10 T | N | SF10 N ÷ T |
|---|---|---|---|---|---|---|---|
| IS1 | **0.19** | 0.95 | **0.27** | 0.90 | **0.24** | 0.90 | **3.72x** |
| IS2 | **1.66** | 7.60 | **2.02** | 7.45 | **2.72** | 10.34 | **3.80x** |
| IS3 | **2.08** | 9.60 | **2.82** | 14.67 | **4.62** | 18.09 | **3.91x** |
| IS4 | **0.14** | 0.87 | **0.14** | 0.88 | **0.17** | 0.82 | **4.90x** |
| IS5 | **0.18** | 0.86 | **0.19** | 0.84 | **0.15** | 0.82 | **5.36x** |
| IS6 | **0.28** | 0.91 | **0.26** | 0.86 | **0.32** | 0.89 | **2.81x** |
| IS7 | **0.45** | 1.22 | **0.63** | 1.16 | **0.78** | 1.21 | **1.54x** |
| IC1 | **2.41** | 4.90 | **4.73** | 22.50 | **10.80** | 28.45 | **2.63x** |
| IC2 | **13.09** | 388.92 | **21.93** | 703.81 | **39.03** | 1,231.36 | **31.5x** |
| IC3 | **53.95** | 2,955.69 | **110.90** | 9,212.05 | **311.02** | 31,415.47 | **101.0x** |
| IC4 | **29.09** | 255.00 | **46.53** | 471.62 | **78.23** | 832.01 | **10.6x** |
| IC5 | **37.35** | 2,199.58 | **82.13** | 6,793.62 | **193.94** | 26,440.72 | **136.3x** |
| IC6 | **52.92** | 1,612.66 | **90.81** | 4,875.91 | **218.39** | 15,553.50 | **71.2x** |
| IC7 | 29.79 | **16.49** | 38.48 | **20.65** | 55.96 | **22.48** | 0.40x |
| IC8 | **4.94** | 16.41 | **5.27** | 15.95 | **4.27** | 18.39 | **4.31x** |
| IC9 | **10.71** | 3,008.47 | **24.39** | 9,018.39 | **67.69** | 28,638.62 | **423.1x** |
| IC10 | **97.20** | 792.92 | **225.95** | 2,176.62 | **576.50** | 5,970.12 | **10.4x** |
| IC11 | **8.33** | 47.68 | **16.89** | 112.43 | **36.50** | 233.46 | **6.40x** |
| IC12 | **42.01** | 138.70 | **68.96** | 141.55 | **112.88** | 3,177.62 | **28.2x** |
| IC13 | **0.24** | 1.08 | **0.31** | 1.06 | **0.73** | 1.30 | **1.77x** |
| IC14 | **5.63** | 121.78 | **8.93** | 396.85 | **7.53** | 69.61 | **9.25x** |

T: TETRAbase. N: Neo4j.

APPENDIX C

# Change by query since 2026-10-12

Median milliseconds at SF10 and SF100.

| QUERY | SF10 OCT 12 | NOV 1 | CHANGE | SF100 OCT 12 | NOV 1 | CHANGE |
|---|---|---|---|---|---|---|
| **IS1** | 0.20 | 0.24 | +21% | 0.24 | 0.25 | +6% |
| **IS2** | 2.83 | 2.72 | -4% | 6.44 | 6.19 | -4% |
| **IS3** | 4.00 | 4.62 | +16% | 19.12 | 19.18 | 0% |
| **IS4** | 0.11 | 0.17 | +52% | 0.14 | 0.13 | -4% |
| **IS5** | 0.18 | 0.15 | -15% | 0.18 | 0.20 | +14% |
| **IS6** | 0.22 | 0.32 | +45% | 0.24 | 0.30 | +24% |
| **IS7** | 0.80 | 0.78 | -2% | 5.00 | 5.32 | +6% |
| **IC1** | 10.23 | 10.80 | +6% | 58.20 | 63.41 | +9% |
| **IC2** | 206.77 | 39.03 | -81% | 140.37 | 147.87 | +5% |
| **IC3** | 439.80 | 311.02 | -29% | 2,591.96 | 2,317.38 | -11% |
| **IC4** | 75.62 | 78.23 | +3% | 399.24 | 250.63 | -37% |
| **IC5** | 190.62 | 193.94 | +2% | 1,289.13 | 1,414.62 | +10% |
| **IC6** | 228.48 | 218.39 | -4% | 1,668.02 | 1,552.48 | -7% |
| **IC7** | 59.16 | 55.96 | -5% | 622.49 | 595.77 | -4% |
| **IC8** | 5.22 | 4.27 | -18% | 16.23 | 17.56 | +8% |
| **IC9** | 131.95 | 67.69 | -49% | 645.42 | 620.78 | -4% |
| **IC10** | 1,056.71 | 576.50 | -45% | 5,672.66 | 2,916.17 | -49% |
| **IC11** | 43.15 | 36.50 | -15% | 258.90 | 229.13 | -11% |
| **IC12** | 108.17 | 112.88 | +4% | 714.33 | 1,062.09 | +49% |
| **IC13** | 0.89 | 0.73 | -18% | 3.73 | 2.89 | -23% |
| **IC14** | 5.69 | 7.53 | +32% | 97.63 | 90.15 | -8% |
| **Total** | 2,570.8 | **1,722.5** | -33% | 14,209.7 | **11,312.5** | -20% |

Same harness, parameters and server.

APPENDIX D

# Writes at 200,000 rows

Rows per second at every client count.

### TETRABASE

| WORKLOAD | C=1 | C=2 | C=4 | C=8 | C=16 | C=32 |
|---|---|---|---|---|---|---|
| **create-node.point.props** | 1.0k | 1.7k | 2.8k | 5.2k | 8.7k | 9.5k |
| **create-node.point.bare** | 1.0k | 1.5k | 3.3k | 7.0k | 9.0k | 18.7k |
| **create-node.block.props** | 531 | 538 | 506 | 526 | 569 | 553 |
| **create-node.block.bare** | 1.4k | 1.6k | 1.3k | 1.3k | 1.3k | 1.3k |
| **create-edge.point.props** | 1.0k | 1.8k | 3.3k | 6.3k | 11.4k | 20.3k |
| **create-edge.point.bare** | 1.4k | 1.7k | 2.6k | 5.8k | 12.9k | 21.2k |
| **create-edge.block.props** | 70.3k | 133.9k | 262.6k | 412.9k | 598.8k | 617.2k |
| **create-edge.block.bare** | 63.1k | 126.2k | 251.0k | 420.6k | 581.4k | 803.9k |
| **update.point.blind** | 1.2k | 1.9k | 2.4k | 6.5k | 11.2k | 22.5k |
| **update.point.incr** | 1.2k | 1.2k | 3.7k | 7.2k | 10.0k | 21.0k |
| **update.point.row** | 1.5k | 1.1k | 3.4k | 6.2k | 13.0k | 21.3k |
| **update.range.blind** | 280.7k | 507.5k | 885.0k | 1.47M | 1.45M | 1.14M |
| **update.range.incr** | 484.8k | 824.2k | 1.41M | 2.33M | 3.05M | 1.86M |
| **update.range.row** | 324.5k | 605.0k | 1.02M | 1.85M | 2.59M | 2.06M |
| **upsert.point** | 756 | 1.9k | 2.2k | 5.0k | 9.7k | 18.5k |
| **upsert.block** | 177.4k | 350.8k | 651.9k | 1.06M | 1.88M | 2.11M |
| **delete.point** | 814 | 2.2k | 2.3k | 7.6k | 9.1k | 16.7k |
| **delete.range** | 259.6k | 486.3k | 797.6k | 1.68M | 1.61M | 1.96M |

## Neo4j, 200,000 rows

| WORKLOAD | C=1 | C=2 | C=4 | C=8 | C=16 | C=32 |
|---|---|---|---|---|---|---|
| **create-node.point.props** | 932 | 1.7k | 2.8k | 5.2k | 9.2k | 15.5k |
| **create-node.point.bare** | 889 | 2.0k | 3.2k | 5.8k | 9.7k | 17.1k |
| **create-node.block.props** | 76.1k | 134.3k | 223.4k | 323.6k | 478.5k | 541.3k |
| **create-node.block.bare** | 201.7k | 262.2k | 442.5k | 744.3k | 1.23M | 1.66M |
| **create-edge.point.props** | 986 | 1.5k | 2.7k | 4.7k | 8.5k | 15.7k |
| **create-edge.point.bare** | 974 | 1.4k | 2.4k | 4.2k | 8.0k | 13.1k |
| **create-edge.block.props** | 38.2k | 70.1k | 128.8k | 233.0k | 395.2k | 297.3k ✗ |
| **create-edge.block.bare** | 16.1k | 39.3k | 95.4k | 234.7k | 436.2k | 555.5k |
| **update.point.blind** | 1.5k | 1.1k | 3.0k | 5.6k | 9.7k | 16.5k |
| **update.point.incr** | 1.1k | 1.2k | 3.0k | 5.7k | 9.8k | 16.2k |
| **update.point.row** | 1.1k | 1.5k | 3.0k | 6.0k | 9.8k | 16.5k |
| **update.range.blind** | 1.08M | 1.22M | 3.96M | 8.10M | 13.93M | 21.86M |
| **update.range.incr** | 176.2k | 337.9k | 656.3k | 869.1k | 1.09M | 1.27M |
| **update.range.row** | 905.3k | 1.76M | 3.52M | 6.61M | 11.32M | 17.69M |
| **upsert.point** | 887 | 2.0k | 3.0k | 5.3k | 9.9k | 16.6k |
| **upsert.block** | 127.6k | 287.9k | 541.9k | 934.2k | 1.27M | 1.80M |
| **delete.point** | 830 | 1.8k | 2.8k | 5.3k | 9.1k | 16.2k |
| **delete.range** | 110.5k | 227.2k | 327.3k | 427.1k | 469.3k | 627.9k |

×: three statements deadlocked.

APPENDIX E

# Writes at 20,000 and 50,000 rows

TETRAbase, rows per second at every client count.

### 20,000 ROWS

| WORKLOAD | C=1 | C=2 | C=4 | C=8 | C=16 | C=32 |
|---|---|---|---|---|---|---|
| **create-node.point.props** | 1.1k | 1.5k | 2.8k | 4.7k | 8.5k | 9.5k |
| **create-node.point.bare** | 857 | 2.0k | 2.5k | 5.7k | 10.6k | 16.4k |
| **create-node.block.props** | 3.9k | 3.5k | 3.8k | 3.3k | 2.7k | 1.9k |
| **create-node.block.bare** | 4.3k | 4.7k | 4.3k | 3.4k | 3.4k | 3.5k |
| **create-edge.point.props** | 1.4k | 1.6k | 3.2k | 5.1k | 11.3k | 21.8k |
| **create-edge.point.bare** | 880 | 1.9k | 2.4k | 5.9k | 13.1k | 24.3k |
| **create-edge.block.props** | 68.0k | 134.5k | 251.8k | 412.0k | 540.7k | 660.6k |
| **create-edge.block.bare** | 62.0k | 116.2k | 201.5k | 359.3k | 470.3k ✗ | 790.8k ✗ |
| **update.point.blind** | 1.1k | 2.0k | 2.8k | 4.5k | 11.9k | 25.3k |
| **update.point.incr** | 1.4k | 1.1k | 3.8k | 6.2k | 13.0k | 21.1k |
| **update.point.row** | 1.3k | 1.6k | 2.3k | 6.9k | 12.8k | 24.2k |
| **update.range.blind** | 403.9k | 252.9k | 267.5k | 1.73M | 2.02M | 2.18M ✗ |
| **update.range.incr** | 775.5k | 1.25M | 1.87M | 2.92M | 2.62M | 2.18M ✗ |
| **update.range.row** | 469.8k | 1.01M | 1.65M | 2.20M | 2.88M | 1.83M ✗ |
| **upsert.point** | 1.4k | 1.2k | 3.5k | 5.7k | 12.5k | 22.5k |
| **upsert.block** | 204.2k | 396.6k | 646.7k | 1.36M | 1.99M | 2.18M ✗ |
| **delete.point** | 1.4k | 1.8k | 2.4k | 4.6k | 9.0k | 18.4k |
| **delete.range** | 561.0k | 917.7k | 0 | 0 | 0 | 0 |

×: read-back mismatched. 0: a delete test that emptied its slice during warm-up.

## 50,000 rows

| WORKLOAD | C=1 | C=2 | C=4 | C=8 | C=16 | C=32 |
|---|---|---|---|---|---|---|
| **create-node.point.props** | 1.0k | 1.5k | 2.5k | 5.2k | 9.2k | 9.5k |
| **create-node.point.bare** | 1.5k | 1.1k | 3.8k | 4.5k | 10.4k | 21.0k |
| **create-node.block.props** | 2.1k | 1.7k | 1.7k | 1.9k | 1.6k | 1.8k |
| **create-node.block.bare** | 3.6k | 3.4k | 3.0k | 3.5k | 2.6k | 2.7k |
| **create-edge.point.props** | 1.2k | 1.8k | 2.9k | 6.3k | 11.5k | 18.4k |
| **create-edge.point.bare** | 1.2k | 1.9k | 3.5k | 5.4k | 8.8k | 18.0k |
| **create-edge.block.props** | 70.9k | 140.0k | 266.1k | 404.1k | 565.1k | 577.7k |
| **create-edge.block.bare** | 70.2k | 130.1k | 219.1k | 373.3k | 574.8k | 806.2k |
| **update.point.blind** | 1.2k | 1.2k ✗ | 2.2k | 6.1k | 13.1k | 22.6k |
| **update.point.incr** | 932 | 2.3k | 3.5k | 4.7k | 8.8k | 16.9k |
| **update.point.row** | 947 | 2.0k | 2.7k | 4.5k | 8.9k | 18.4k |
| **update.range.blind** | 360.0k | 733.4k | 1.16M | 1.80M | 1.88M | 1.86M |
| **update.range.incr** | 730.0k | 1.10M | 1.90M | 2.64M | 2.68M | 2.25M |
| **update.range.row** | 407.1k | 873.5k | 1.34M | 2.25M | 2.68M | 1.47M |
| **upsert.point** | 797 | 2.0k | 2.7k | 4.3k | 9.4k | 18.6k |
| **upsert.block** | 181.6k | 378.2k | 695.3k | 1.24M | 1.95M | 2.36M |
| **delete.point** | 847 | 1.2k | 3.2k | 10.5k | 12.9k | 16.6k |
| **delete.range** | 266.7k | 415.0k | 640.9k | 1.38M | 0 | 0 |

×: read-back mismatched. 0: a delete test that emptied its slice during warm-up.

APPENDIX F

# Reads under writes by query

Median milliseconds at SF1, readers alone and with 16 writers.

| READ | T ALONE | T + 16 | SLOWDOWN | N ALONE | N + 16 | SLOWDOWN | N ÷ T + 16 |
|---|---|---|---|---|---|---|---|
| IS1 | **0.43** | 0.62 | 1.43x | 1.26 | 0.45 | 0.36x | **0.73x** |
| IS2 | 12.63 | 23.56 | 1.87x | **8.77** | 8.17 | 0.93x | **0.35x** |
| IS3 | **4.91** | 6.15 | 1.25x | 11.38 | 11.08 | 0.97x | **1.80x** |
| IS4 | **0.28** | 0.35 | 1.27x | 0.93 | 0.45 | 0.48x | **1.28x** |
| IS5 | **0.24** | 0.42 | 1.74x | 0.62 | 0.35 | 0.57x | **0.83x** |
| IS6 | **0.37** | 4.55 | 12.4x | 0.61 | 0.34 | 0.55x | **0.07x** |
| IS7 | **0.65** | 5.01 | 7.69x | 0.93 | 0.69 | 0.74x | **0.14x** |
| IC1 | **2.72** | 9.18 | 3.37x | 4.06 | 3.38 | 0.83x | **0.37x** |
| IC2 | **190.10** | 421.46 | 2.22x | 434.07 | 468.85 | 1.08x | **1.11x** |
| IC3 | **65.08** | 109.25 | 1.68x | 3,309.20 | 3,488.34 | 1.05x | **31.9x** |
| IC4 | **38.51** | 348.80 | 9.06x | 286.65 | 308.05 | 1.07x | **0.88x** |
| IC5 | **43.74** | 51.64 | 1.18x | 2,330.94 | 2,517.94 | 1.08x | **48.8x** |
| IC6 | **64.14** | 823.01 | 12.8x | 1,845.00 | 1,955.38 | 1.06x | **2.38x** |
| IC7 | 43.06 | 61.89 | 1.44x | **20.91** | 18.57 | 0.89x | **0.30x** |
| IC8 | **8.33** | 19.16 | 2.30x | 17.26 | 17.25 | 1.00x | **0.90x** |
| IC9 | **587.10** | 639.93 | 1.09x | 3,374.93 | 3,568.46 | 1.06x | **5.58x** |
| IC10 | **113.91** | 150.05 | 1.32x | 869.02 | 1,086.79 | 1.25x | **7.24x** |
| IC11 | **8.95** | 302.82 | 33.9x | 51.24 | 57.19 | 1.12x | **0.19x** |
| IC12 | **49.13** | 454.58 | 9.25x | 144.93 | 176.84 | 1.22x | **0.39x** |
| IC13 | **0.51** | 0.72 | 1.42x | 1.60 | 0.73 | 0.46x | **1.01x** |
| IC14 | **6.25** | 7.08 | 1.13x | 124.17 | 152.08 | 1.22x | **21.5x** |

T: TETRAbase. N: Neo4j.

EARLY ACCESS

# Test TETRAbase on your own data

Our early-access program: a small group of teams running TETRAbase on their own workloads. Don't take our word for it.

SCHEDULE A TRIAL

INQUIRIES@TETRA-CA.COM

---

RAW RESULTS, STORED NEO4J RESULTS, QUERY PARAMETERS AND THE HARNESS: AVAILABLE ON REQUEST · TETRA-CA.COM