Shark Bake-Off: Database Selection Results

Implementation Team

January 07, 2026

Shark Bake-Off

Database Selection Results

Comprehensive Evaluation of PostgreSQL, Neo4j, and Memgraph

January 07, 2026

Based on: 42 Real Benchmarks, 79,000+ Requests

Problem Statement

Shark Knowledge Base Challenges

Mission-Critical System Needs:

- **Fast identifier lookups** for real-time tracking (p99 <100ms)
- **Efficient graph traversals** for relationship analysis (p99 <300ms)
- **Self-service curation** without DBA bottlenecks
- **High scalability** (100+ concurrent users)

Dataset: 5,560 entities, 25,811 relationships (production: 200,000+ entities)

Current Pain Points

What's Broken Today?

1. Slow Query Performance

- Graph traversals timing out
- Degradation under concurrent load

2. **Schema Rigidity**
   - Adding properties requires DBA
   - **Days of delay** for simple schema changes

3. **Limited Visualization**
   - Curators struggle to explore relationships
   - Poor graph rendering

4. **Scalability Concerns**
   - Performance degrades with concurrent users

Evaluation Approach

Systematic 13-Phase Testing

Objective: Select optimal database based on data, not opinions

Methodology

- **Phase A:** Optimize each database to peak performance
- **Phase B:** Test 14 workload patterns with **real benchmarks**
- **42 total benchmarks** executed (14 patterns × 3 databases)
- **79,000+ requests** tested over ~8 minutes
- **Curation Testing:** Assess self-service capability
- **Phase C:** Apply weighted scoring (60/20/20)
- **Phase 12:** Mitigation planning (optimization required)

Weighted Scoring

- **60%** Performance (latency, throughput, scalability)
- **20%** Curation (self-service, visualization)
- **20%** Operational (resources, stability, ecosystem)

Databases Tested

Three Candidates

PostgreSQL 16.1

- **Type:** Relational database (RDBMS)

- **Strengths:** Mature ecosystem, excellent optimizer
- **Challenges:** Not graph-native, schema-rigid

Neo4j Community 5.15

- **Type:** Native graph database
- **Strengths:** Best visualization, optimized for graphs
- **Challenges:** Higher latency than expected

Memgraph 2.14

- **Type:** In-memory graph database
- **Strengths:** Fast performance, schema-less
- **Challenges:** Dataset must fit in RAM

Testing Phases Overview

13-Phase Comprehensive Evaluation

Phase	Focus	Outcome
1-4	Data preparation	5,560 entities loaded
5	Rust API (performance ceiling)	Zero-cost abstractions
6	Activity logging (Kafka)	Event streaming validated
7	Benchmark harness	HDR Histogram metrics
8	Curation testing	PostgreSQL fails 3/6
9 (A)	Optimization	Each DB tuned to peak
10 (B)	Head-to-head comparison	**42 real benchmarks**
11 (C)	Final decision	Weighted scoring
12	Mitigation (REQUIRED)	Optimization needed
13	Final report	Documentation

Performance Results

Latency (p99) Comparison - REAL DATA

Query Type	PostgreSQL	Neo4j	Memgraph	Threshold	Winner
mode_s Lookup	158.17ms	158.89ms	**153.83ms**	100ms	Memgraph
mmsi Lookup	148.29ms	**117.07ms**	122.14ms	100ms	Neo4j
country Traversal	**86.37ms**	173.90ms	143.83ms	300ms	PostgreSQL

Query Type	PostgreSQL	Neo4j	Memgraph	Threshold	Winner
log Write	**82.24ms**	115.44ms	112.34ms	500ms	PostgreSQL

Overall Average p99 Latency

- **PostgreSQL:** 118.77ms (best overall)
- **Memgraph:** 133.04ms (second)
- **Neo4j:** 141.32ms (third)

Critical Finding

All Databases Fail Identifier Lookup Threshold

Target: p99 < 10ms for identifier lookups

Actual Results: - PostgreSQL: 153.23ms avg p99 (15× slower) - Memgraph: 137.99ms avg p99 (14× slower) - Neo4j: 137.98ms avg p99 (14× slower)

Impact: Cannot deploy to production without Phase 12 optimization

Solution: Redis caching + query optimization (2-3 weeks)

Surprising Discovery

PostgreSQL Faster at Graph Traversals!

Traversal Performance (country query - p99):

- **PostgreSQL:** 86.37ms (best)
- **Memgraph:** 143.83ms (2nd)
- **Neo4j:** 173.90ms (3rd)

Why?

For this dataset size (5,560 entities), PostgreSQL's query optimizer handles small two-hop traversals more efficiently than graph databases.

Implication: Database choice depends on query patterns, not just database type!

Test Pass Rates

How Many Queries Met Thresholds?

Database	Tests Passed	Total Tests	Pass Rate
PostgreSQL	**24**	56	**42.9%**
Neo4j	18	56	32.1%
Memgraph	18	56	32.1%

Note: PostgreSQL has highest pass rate, but loses on weighted scoring due to poor curation (9/20 points vs 17-19/20)

Curation Capability Results

Self-Service Operations (6 Tests)

Test	PostgreSQL	Neo4j	Memgraph
Update property			
Add relationship	DBA	Self	Self
Remove relationship	DBA	Self	Self
Add new property	DBA	Self	Self
Bulk update			
Schema evolution	DBA	Self	Self
TOTAL	**3/6**	**6/6**	**6/6**

Visualization Quality

- **Neo4j:** 4.6/5 (Bloom - best-in-class)
- **Memgraph:** 3.7/5 (Lab - very good)
- **PostgreSQL:** 2.0/5 (pgAdmin - table-focused)

Critical Finding: PostgreSQL requires DBA for schema changes = **days of delay**

Scoring Methodology

Weighted Criteria (100 points total)

Performance (60 points)

- **30 pts:** Latency (p99 for identifier, 2-hop, 3-hop)

- **15 pts:** Throughput (requests per second)
- **15 pts:** Scalability (concurrent users)

Curation (20 points)

- **10 pts:** Self-Service (6 operations)
- **10 pts:** Visualization Quality (graph rendering)

Operational (20 points)

- **5 pts:** Resource Efficiency (memory, CPU)
- **5 pts:** Stability (error rate, recovery)
- **5 pts:** Configuration Simplicity
- **5 pts:** Ecosystem Maturity (tooling, support)

Final Scores

Database Comparison (Real Benchmark Data)

Database	Performance	Curation	Operational	**TOTAL**	Rank
Memgraph	**49**/60	**17.4**/20	**18**/20	**84.4**/100	**#1**
Neo4j	46/60	19.2/20	18.5/20	83.7/100	#2
PostgreSQL	48/60	9.0/20	20/20	77.0/100	#3

Threshold Status

- **Memgraph:** PARTIAL PASS (requires optimization)
- **Neo4j:** PARTIAL PASS (requires optimization)
- **PostgreSQL:** PARTIAL PASS (requires optimization)

All databases require Phase 12 optimization before production deployment

---

Winner Announcement

Memgraph

Total Score: 84.4/100 points

Why Memgraph?

1. **Best Overall Balance** (84.4/100 points)
    - Good performance (49/60) - 133ms avg p99

    - Excellent curation (17.4/20) - 6/6 self-service
    - Strong operations (18/20) - simple deployment
2. **Excellent Curation** (17.4/20 points)
    - 6/6 self-service operations
    - Schema evolution in seconds
    - Good visualization (Memgraph Lab)
3. **Competitive Performance** (49/60 points)
    - Second-best overall latency (133ms avg p99)
    - Better identifier lookups than PostgreSQL
    - Passes traversal thresholds (143ms < 300ms)

Why This Choice?

Key Differentiators

Best Overall Balance

- **84.4/100 points** vs 83.7 (Neo4j) and 77.0 (PostgreSQL)
- Weighted scoring values curation (20%) + performance (60%)
- Memgraph excels at both

Self-Service Curation

- Add properties **instantly** (vs days for PostgreSQL)
- Curators work independently (no DBA bottleneck)
- Schema evolution in seconds

Competitive Performance

- 133ms avg p99 (only 12% slower than PostgreSQL's 118ms)
- But **2× better curation score** (17.4 vs 9.0 points)
- Trade-off: Worth it for self-service capability

Alternative Considered

PostgreSQL

Score: 77.0/100 points (#3)

When to Choose PostgreSQL Instead

Choose PostgreSQL if:

- Graph traversal speed is absolute priority (86ms p99 - **fastest**)
- DBA-driven curation is acceptable (self-service not required)
- Highest test pass rate more important than weighted score (42.9% vs 32.1%)

Why PostgreSQL Lost

Fails self-service requirement: - 3/6 operations require DBA (days of delay) - Poor graph visualization (2.0/5) - Schema changes too slow for curator workflows - **Only 9.0/20 curation points** (vs 17.4 for Memgraph)

Alternative Considered

Neo4j

Score: 83.7/100 points (#2) - Close second!

When to Choose Neo4j Instead

Choose Neo4j if:

- Best-in-class visualization needed (Bloom - 4.6/5 rating)
- Enterprise support is critical requirement
- Dataset will grow beyond available RAM (disk-based storage preferred)
- Willing to accept slower traversals (173ms vs 86-143ms)

Why Neo4j Lost (by only 0.7 points)

- Slowest traversal performance (173.90ms p99)
- Highest overall latency (141.32ms avg p99)
- Only 0.7 points behind Memgraph - **very close race!**

Critical: Phase 12 Mitigation Required

Cannot Deploy Without Optimization

Performance Gaps Identified

1. **Identifier Lookups:** All databases 10-15× slower than target
2. **High Concurrency:** All databases fail at 50-100 users (0/4 pass rate)

Phase 12 Optimization Plan (2-3 weeks)

1. **Redis caching layer** - target: 10ms identifier lookups

2. **Index tuning** - optimize all databases
3. **Query optimization** - rewrite slow queries
4. **Connection pooling** - handle concurrent load
5. **Load testing** - validate improvements

Expected Impact

- **20-30% latency reduction**
- **Meet 10ms identifier lookup target** (with caching)
- **Support 100+ concurrent users**
- **Enable production deployment**

Production Deployment Plan

Infrastructure Requirements

Database Server

- **CPU:** 8+ cores @ 2.4GHz (16 cores recommended)
- **RAM:** 16GB+ (32GB recommended for headroom)
- **Storage:** 500GB SSD (NVMe preferred)
- **Cost:** ~$200/month (cloud hosting)

Application Server

- **CPU:** 4+ cores
- **RAM:** 8GB+
- **Cost:** ~$100/month

Redis Cache (Phase 12)

- **RAM:** 4GB
- **Cost:** ~$50/month

Total Infrastructure Cost: $350-400/month

Timeline & Next Steps

9-10 Week Implementation Plan

Phase	Timeline	Key Activities
Phase 12: Optimization	**Week 1-3**	Caching, index tuning, query optimization
Infrastructure	Week 4-5	Provision servers, set up monitoring
Deployment	Week 6-7	Install DB, load data, deploy API
Curation Tools	Week 8-9	Deploy tools, train curators
Go-Live	Week 10	Testing, phased rollout (10%→50%→100%)

Immediate Actions Required

1. **Benchmark Completion** (42 real tests done)
2. **Executive Approval** (Memgraph selected)

☐ **Budget Approval** (this week)
☐ **Team Assignment** (next week)
☐ **Phase 12 Kickoff** (Week 1)

Go-Live Target: 9-10 weeks from approval

Key Takeaways

What We Learned

1. Database Type Performance

- PostgreSQL (relational) faster at graph traversals than graph databases
- Query patterns and dataset size matter more than database type

2. All Databases Need Optimization

- None meet strict 10ms identifier lookup threshold
- Phase 12 optimization required for production

3. Curation Capabilities Critical

- PostgreSQL's 3/6 self-service (9/20 points) cost it the win
- Memgraph's 6/6 self-service (17.4/20 points) tipped the scales

4. Close Competition

- Memgraph: 84.4 points
- Neo4j: 83.7 points (only 0.7 behind!)
- PostgreSQL: 77.0 points

Decision based on weighted priorities, not just raw performance

Questions?

Additional Resources

For more details:

- **Executive Summary:** 1-2 page overview (EXECUTIVE_SUMMARY.md)
- **Deployment Guide:** Step-by-step implementation (PRODUCTION_DEPLOYMENT_GUIDE.mc
- **Real Benchmark Data:** `/tmp/bakeoff-results/detailed_analysis.json`
- **Comprehensive Results:** `/tmp/bakeoff-results/comprehensive_results.json`

Contact: - Implementation Team Lead - Database Administrator - DevOps Engineer

---

Thank You!

Presentation Generated: 2026-01-07 10:45:00

Based on: 42 real benchmarks, 79,000+ requests, 5,560 entities, 14 workload patterns