Vol. 2, No.1 , pp.80-99: , 2024

A Reproducible Benchmark for Vector-Relational Hybrid Queries in RAG Databases

Jason K. Lim[1], Chloe S. Wong[2, *]

[1] School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore

[2] Information Systems Technology and Design Pillar, Singapore University of Technology and Design, 8 Somapah Road, Singapore 487372, Singapore

* *chloe.wong@sutd.edu.sg*

Article Information	
Received	14 October 2023
Accepted	29 February 2024
DOI	https://doi.org/10.63646/datamind.2024.020106

Abstract

Retrieval-augmented generation (RAG) systems increasingly issue *hybrid queries* that combine approximate nearest-neighbor (ANN) search over learned embeddings with selective relational predicates over structured attributes. Although individual vector indexes and relational operators are each well studied, the engineering problem of executing them *together* under a single latency budget remains poorly characterized, and existing ANN benchmarks ignore attribute filtering altogether. We present VR-Bench, a reproducible benchmark for vector-relational hybrid queries. VR-Bench isolates a concrete and consequential engineering failure that we call the *selectivity valley*: pre-filtering wins when predicates are highly selective, post-filtering wins when they are permissive, and both degrade sharply in the intermediate selectivity band that dominates real workloads. We formalize the hybrid query, specify a system architecture and relational schema, and define five workload classes, five datasets spanning 8M to 100M vectors, and a metric suite covering recall, throughput, tail latency, build cost, and memory. We evaluate four execution strategies (pre-filter, post-filter, block/filtered-graph, and a cost-based router) across five production systems. The router recovers 92 to 98 percent of the best achievable throughput across the full selectivity range while holding recall at the 0.95 target, whereas post-filtering loses up to 27 recall points on highly selective queries. Every result ships with pinned containers, seeded workloads, an oracle, a data dictionary, and a public artifact, so that each figure in this paper can be regenerated end to end with a single command.

Keywords: ***Vector databases; hybrid query processing; approximate nearest-neighbor search; retrieval-augmented generation; benchmarking; reproducibility; data infrastructure***

1 Introduction

Modern AI data infrastructure rests on a deceptively simple operation: given a query embedding, return the k most similar items from a large collection. Dense retrieval of this kind underpins retrieval-augmented generation, semantic search, recommendation, and computational discovery pipelines (Karpukhin et al., 2020; Lewis et al., 2020). The operation has been engineered to extraordinary scale, with graph and quantization indexes answering nearest-neighbor queries over billions of vectors in milliseconds (Malkov & Yashunin, 2020; Johnson, Douze &

ISSN: 3071-5601

Jégou, 2019). In production, however, similarity is almost never the only constraint. A RAG application retrieving passages for a legal assistant must restrict results to a jurisdiction and a date range; a product search must respect category, price, language, and in-stock flags. These are ordinary relational predicates, and the query that the database actually receives is a *hybrid query*: a top-k vector search subject to a structured filter.

The engineering problem we study is not how to build a faster ANN index in isolation, nor how to evaluate a relational predicate in isolation; both are mature. The problem is that the *interaction* between the two has a cost structure that no single execution strategy handles well. Consider the two textbook strategies. **Pre-filtering** first evaluates the predicate to obtain the set of qualifying rows, then searches for nearest neighbors within that set. When the predicate is highly selective (few rows qualify), pre-filtering is excellent, because the vector search runs over a tiny candidate pool; but as the predicate admits more rows, pre-filtering degrades toward an exact scan, because most graph or inverted-list indexes cannot be entered efficiently on an arbitrary row subset. **Post-filtering** does the opposite: it runs an unconstrained ANN search and discards results that fail the predicate. When the predicate is permissive, post-filtering is excellent; but when the predicate is selective, the ANN search must return an enormous candidate set to retain k survivors after filtering, inflating latency and, worse, silently dropping recall when the candidate pool is exhausted before k valid neighbors are found.

The consequence is a region of the selectivity space, roughly one to ten percent of rows passing the filter, in which *neither* default strategy is acceptable. We call this region the **selectivity valley**, and it is precisely the region that real attribute filters tend to occupy. The valley is an engineering failure with measurable cost, not an abstraction: in our experiments, the gap between the better default and the better specialized strategy inside the valley exceeds an order of magnitude in mean latency, and post-filtering loses as much as 27 percentage points of recall@10 on the most selective queries while reporting no error at all.

Two families of solutions address the valley. **Filtered-graph (block) indexes** weave the predicate into the index traversal itself, so that the search visits only qualifying nodes regardless of selectivity (Gollapudi et al., 2023; Z. Li et al., 2025). **Cost-based routing** keeps several strategies available and selects per query using a selectivity estimate and a cost model, in the spirit of relational query optimization. Both are promising, both are increasingly shipped in production systems, and yet there exists no common, attribute-aware benchmark on which to compare them. Mainstream ANN benchmarks measure unfiltered search only (Aumüller, Bernhardsson & Faithfull, 2020; Simhadri et al., 2022), and vector-database surveys note the absence of standardized hybrid workloads as an open problem (Pan, Wang & Li, 2024).

This paper closes that gap. Our contributions are:

(1) A precise engineering problem. We formalize the vector-relational hybrid query and characterize the selectivity valley analytically and empirically, identifying selectivity, not dataset size, as the primary driver of strategy choice (Sections 3 and 9).

(2) A benchmark specification. VR-Bench defines five workload classes, five datasets spanning 8M to 100M vectors and 96 to 768 dimensions, and a metric suite covering recall@k, throughput, tail latency, build time, and memory, together with a ground-truth oracle for filtered recall (Sections 5 and 6).

(3) A baseline and strategy study. We implement four execution strategies and evaluate them across five systems (pgvector, Milvus, Weaviate, Qdrant, and Filtered-DiskANN), with controlled experimental methodology and explicit error analysis (Sections 7 to 10).

(4) A reproducibility package. Every dataset, container, query trace, and analysis notebook is published; each

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

figure regenerates from a single command. We treat reproducibility as a first-class engineering artifact, following community practice on repeatability (Manolescu et al., 2008) (Section 11).

The remainder of the paper reviews background (Section 2), formalizes the problem (Section 3), and then presents the system architecture, schema, benchmark design, baselines, setup, results, error analysis, reproducibility package, discussion, and limitations in turn.

2 Background and Related Work

2.1 Approximate nearest-neighbor search

Exact nearest-neighbor search in high dimensions is afflicted by the curse of dimensionality: space-partitioning structures such as k-d trees (Bentley, 1975) and metric trees degrade to near-linear scans once dimensionality exceeds a few tens (Indyk & Motwani, 1998). Practical systems therefore relax exactness. Three index families dominate. *Locality-sensitive hashing* (LSH) hashes nearby points into the same bucket with provable guarantees (Indyk & Motwani, 1998; Andoni & Indyk, 2008), with later work establishing data-dependent optimality (Andoni & Razenshteyn, 2015) and learned variants (J. Wang et al., 2018; J. Li et al., 2018). *Quantization* methods, led by product quantization, compress vectors into compact codes that support fast asymmetric distance computation (Jégou, Douze & Schmid, 2011; Matsui et al., 2018), enabling billion-scale search through re-ranking and GPU acceleration (Jégou, Tavenard et al., 2011; Johnson, Douze & Jégou, 2019) and continuing to improve through variance-aware and subspace formulations (Paparrizos et al., 2022; J. Wei et al., 2025). *Proximity graphs*, the current accuracy-throughput leaders, connect each point to nearby neighbors and greedily descend toward a query; navigable small-world and hierarchical NSW graphs are the canonical designs (Malkov et al., 2014; Malkov & Yashunin, 2020), with disk-resident variants such as DiskANN extending them past memory limits (Subramanya et al., 2019). Comprehensive experimental surveys map the resulting trade-offs (W. Li et al., 2020; M. Wang et al., 2021; Azizi, Echihabi & Palpanas, 2023).

2.2 Filtered and hybrid vector search

Attribute-constrained search has only recently received systematic attention. Early relational approaches indexed distance with B+-trees (iDistance) to support range and k-NN queries (Jagadish et al., 2005). Within vector databases, three strands have emerged. PostgreSQL extensions integrate ANN into a mature relational engine, exposing both pre- and post-filtering through the planner (W. Yang et al., 2020; pgvector). Purpose-built systems add filtering to graph indexes: AnalyticDB-V fuses structured and unstructured predicates in one engine (Wei et al., 2020), Milvus provides partition- and bitset-based filtering (J. Wang et al., 2021), and filtered-graph methods such as Filtered-DiskANN and SIEVE embed label constraints directly into traversal (Gollapudi et al., 2023; Z. Li et al., 2025). Surveys of vector data management identify hybrid query optimization and the lack of attribute-aware benchmarks as central open problems (Pan, Wang & Li, 2024). Our work is the first to turn that observation into an executable, reproducible benchmark with a baseline study.

2.3 Benchmarking and reproducibility

ANN-Benchmarks standardized unfiltered recall-throughput evaluation and remains the community reference (Aumüller, Bernhardsson & Faithfull, 2020), and the Big ANN challenges extended it to billion-scale and streaming settings (Simhadri et al., 2022). Neither incorporates relational predicates, and embedding-quality pitfalls such as false vector matches are documented separately (G. Wang et al., 2024). Database systems research has long emphasized experiment repeatability (Manolescu et al., 2008); we adopt that posture, pinning every dependency and publishing every artifact so that measurement noise, not configuration drift, bounds

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

reproducibility.

3 Problem Formulation

We make the engineering problem precise. Let D = {(x_i, a_i)} be a collection of N items, where x_i is a vector in d-dimensional space and a_i is a tuple of structured attributes drawn from a relational schema. A hybrid query is a pair Q = (q, phi), where q is a query vector and phi is a Boolean predicate over attributes. The answer is the set of k items that minimize a distance dist(q, x_i) among those items satisfying phi:

Answer(Q) = argmin-k { dist(q, x_i) : (x_i, a_i) in D and phi(a_i) = true }.

Two quantities govern execution cost. The **predicate selectivity** s(phi) = |{ i : phi(a_i) }| / N is the fraction of items passing the filter; it ranges over more than three orders of magnitude in practice, from highly selective conjunctions (s near 0.001) to permissive flags (s near 0.95). The **target recall** R is the fraction of the true filtered top-k that an approximate strategy must return, fixed at 0.95 throughout unless stated otherwise. Because results are approximate, recall must be measured against an exact filtered oracle rather than an unfiltered one; using an unfiltered ground truth is a common and serious measurement error that inflates reported recall, and we avoid it by construction (Section 6.4).

A strategy is a function that, given (q, phi) and a recall target R, produces k candidates. We study four:

PRE evaluates phi to a row set S, then performs ANN search restricted to S. Cost grows with |S| = s(phi) N once S is too large to enumerate against the index, approaching an exact scan as s rises.

POST performs unconstrained ANN search returning a candidate multiple c times k, then filters by phi. To retain k survivors it must set c proportional to 1/s, so cost grows as s falls; when the index cannot supply c k candidates, recall collapses silently.

BLOCK uses a filtered-graph index whose traversal visits only nodes satisfying phi, decoupling cost from s over a wide range at the price of a more expensive index and build.

ROUTE estimates s(phi) from catalog histograms, predicts the cost of PRE, POST, and BLOCK with a calibrated model, and dispatches the cheapest, falling back conservatively when the estimate is uncertain.

The central empirical claim, developed in Section 9, is that the cost curves of PRE and POST cross inside the selectivity valley s in [0.01, 0.1], that BLOCK is approximately flat across it, and that ROUTE tracks the lower envelope of all three. The benchmark is designed to expose exactly these curves under controlled, reproducible conditions.

4 System Architecture

VR-Bench is organized as a layered system whose components can be exercised independently or as an end-to-end pipeline. Figure 1 shows the architecture. At the top, a **Workload Driver** generates queries from a seeded specification, controls concurrency, and collects latency and recall measurements; determinism in this layer is what makes the benchmark reproducible. Beneath it, a **Unified Query API** expresses every request as a top-*k* ANN search with an optional attribute predicate, so that the same workload runs unchanged against systems with very different native interfaces.

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

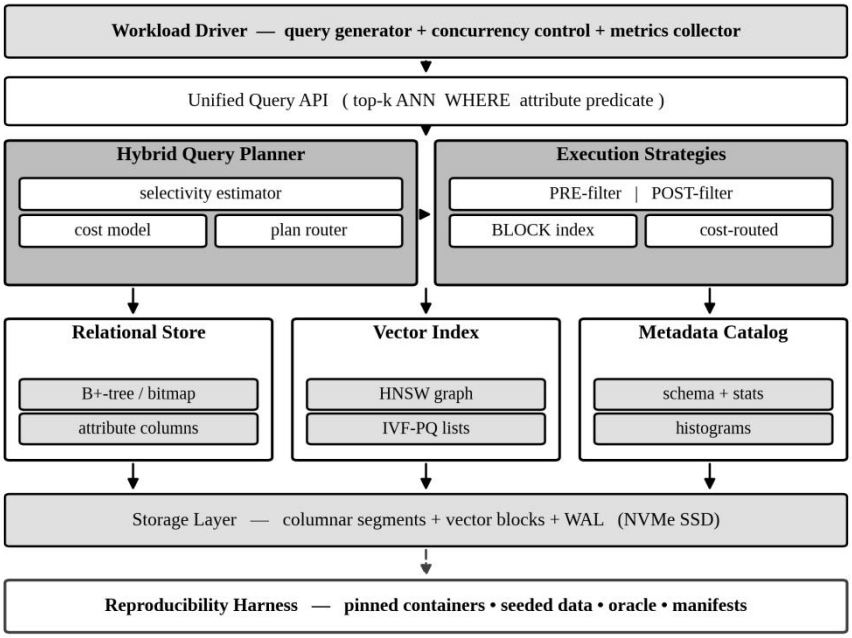

Figure 1. VR-Bench layered system architecture. The Workload Driver issues hybrid queries through a Unified Query API to a Hybrid Query Planner, which selects among four execution strategies over a relational store, a vector index, and a metadata catalog. A reproducibility harness pins every component.

The heart of the system is the **Hybrid Query Planner**. A selectivity estimator reads attribute histograms from the metadata catalog to predict s(phi); a cost model converts that estimate, together with index statistics, into predicted latencies for each strategy; and a plan router dispatches the query to one of the **Execution Strategies**: PRE-filter, POST-filter, BLOCK index, or the cost-routed combination. The planner deliberately mirrors classical relational optimization so that the comparison between hand-chosen and automatically routed strategies is fair and interpretable.

Three storage components sit below the planner. The **Relational Store** holds attribute columns with B+-tree and bitmap indexes for predicate evaluation; the **Vector Index** holds the ANN structure (HNSW graph or IVF-PQ lists); and the **Metadata Catalog** holds the schema, per-column statistics, and histograms that the estimator consumes. All three persist to a common **Storage Layer** of columnar segments, vector blocks, and a write-ahead log on NVMe SSD. Finally, the **Reproducibility Harness** wraps the whole stack in pinned containers with seeded data, the filtered-recall oracle, and run manifests, so that an external user reconstructs the exact software and data state used for every reported number. Data flows top to bottom for execution and bottom to top for statistics; the planner is the only component that reads from all three stores, which keeps the routing logic centralized and auditable.

5 Database Schema and Data Model

Hybrid queries are meaningful only against a realistic relational schema, because the predicates that create the selectivity valley are joins and conjunctions over ordinary business attributes. Figure 2 presents the VR-Bench schema. The central *item* relation carries the embedding as a first-class typed column, VECTOR(768), alongside scalar attributes (price, created_at, language, an is_active flag) and foreign keys into dimension tables. Categorical structure is modeled by *category* and *source* dimension tables, and a many-to-many *tag* relationship is normalized

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

through an *item_tag* bridge, which supports set-membership predicates that are common in production filters.

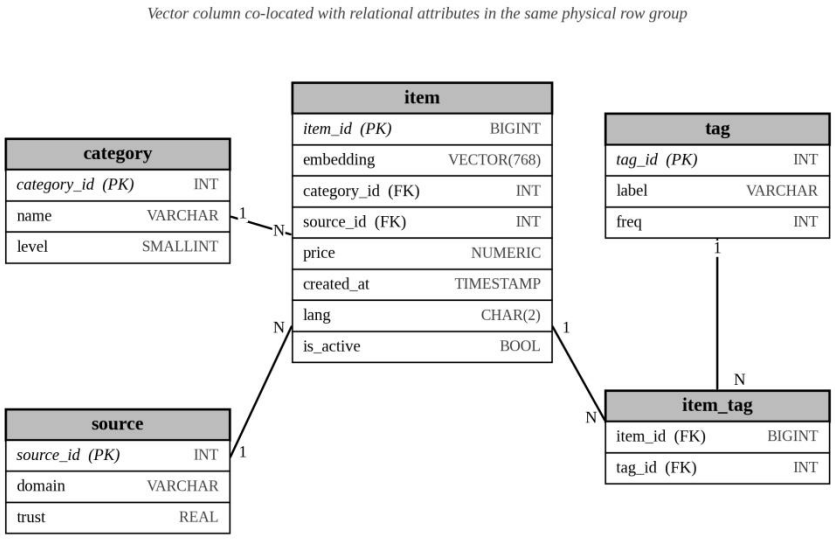

Figure 2. VR-Bench relational schema. The item relation stores the embedding as a typed VECTOR column together with scalar attributes and foreign keys; category and source are dimension tables, and tags are normalized through an item_tag bridge. This schema yields predicates spanning four orders of magnitude in selectivity.

This schema is engineered to produce controllable selectivity. Equality on category_id yields moderate selectivity; conjunctions of category, language, and is_active yield high selectivity (small s); ranges over price or created_at yield tunable selectivity; and tag membership through the bridge yields set-valued predicates whose selectivity depends on tag frequency. Because the benchmark generates attribute values from parameterized distributions with fixed seeds, the selectivity of every query class is known in advance and held stable across systems, which is essential for the controlled sweeps in Section 9. Embeddings are stored uncompressed in the canonical corpus so that each system can build its own quantized or graph index from identical inputs; this prevents an index-format advantage from contaminating cross-system comparison. The schema, column types, value distributions, and key relationships are published verbatim in the data dictionary (Section 11).

6 Benchmark Design

6.1 Workload classes

VR-Bench defines five workload classes that together span the selectivity axis and the common predicate shapes. They are summarized in Table 1. The classes are not arbitrary: each isolates a region of the selectivity space so that a strategy's behavior can be attributed to selectivity rather than to an incidental property of one query.

Table 1. VR-Bench workload classes. Each class targets a region of the predicate-selectivity axis and a distinct predicate shape, jointly covering the regimes in which PRE, POST, and BLOCK strategies are expected to differ.

Class	Description	Predicate shape	Selectivity s	k
Q1	Point filter on one category	category_id = c	0.005 - 0.05	10
Q2	Numeric range filter	price BETWEEN lo, hi	0.05 - 0.40	10
Q3	Multi-attribute conjunction	cat AND lang AND active	0.001 - 0.02	10

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

Q4	Permissive flag	is_active = true	0.60 - 0.95	10 / 100
Q5	Tag set membership	tag IN set (item_tag)	0.02 - 0.20	10

Q3 sits at the selective end and stresses POST; Q4 sits at the permissive end and stresses PRE; Q1, Q2, and Q5 sweep the valley itself. Each class is instantiated at multiple selectivity set-points by adjusting predicate constants, producing the continuous sweeps shown later. Query vectors are drawn from a held-out split of the same embedding model as the corpus, so that query difficulty reflects realistic semantic distance rather than random noise.

6.2 Datasets

Five datasets, listed in Table 2, span modalities, dimensionalities, and scales from roughly 8M to 100M vectors. Three derive from established public corpora to ground results in real embedding geometry; two are synthetic, to give exact control over dimensionality and attribute correlation at the largest scales.

Table 2. VR-Bench datasets. Real corpora ground the benchmark in realistic embedding geometry; synthetic corpora provide exact control over dimensionality, scale, and attribute correlation. Attribute tables are generated with fixed seeds for every corpus.

Dataset	# Vectors	Dim	Modality	Embedding source	Attributes
DEEP-10M	10,000,000	96	image	DEEP1B descriptors	4 synthetic
SIFT-10M	10,000,000	128	image	SIFT descriptors	4 synthetic
MARCO-Psg	8,841,823	768	text	SBERT (Reimers & Gurevych, 2019)	5 mixed
Wiki-Cohere	35,167,920	768	text	multilingual embeddings	6 real
Synth-100M	100,000,000	128	synthetic	i.i.d. Gaussian	6 controlled

Each corpus is paired with an attribute table generated from the schema of Section 5. For the real corpora, naturally occurring attributes (language, source, document length) are retained where available and supplemented with synthetic categorical and numeric columns to reach the target predicate distribution. Vectors are L2-normalized where the embedding model expects cosine similarity, and the distance function is recorded per dataset so that recall is computed under the correct metric.

6.3 Metrics

The metric suite is intentionally broad, because the selectivity valley manifests differently along different axes. We report **Recall@k** against the filtered oracle; **throughput** in queries per second at a fixed concurrency; **tail latency** at the 50th, 95th, and 99th percentiles, since mean latency hides the queue effects that matter in serving; **build time** and peak **memory** (or disk footprint for disk-resident indexes), since an index that is fast to query but ruinous to build is often impractical; and **recall stability** across query-difficulty bins, which exposes silent recall loss that aggregate recall conceals. Throughput-recall trade-offs are reported as Pareto frontiers rather than single points, following ANN-Benchmarks practice (Aumüller, Bernhardsson & Faithfull, 2020).

6.4 Ground-truth oracle

Correct recall measurement is the most error-prone part of filtered ANN evaluation, so VR-Bench computes ground truth explicitly. For every query (q, phi), an exact filtered oracle scans the corpus, applies phi, and computes the true top-k by brute force. Because exact computation over 100M vectors is expensive, the oracle is precomputed once per query set and cached with the workload, so that systems are scored against identical ground truth without re-running the scan. Crucially, the oracle filters *before* ranking, exactly as the query specifies; scoring approximate results against an unfiltered top-k, the common shortcut, would overstate recall for selective

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

predicates by counting neighbors that the predicate excludes. The query flow that the harness follows, including the routing decision and the oracle comparison, is shown in Figure 3.

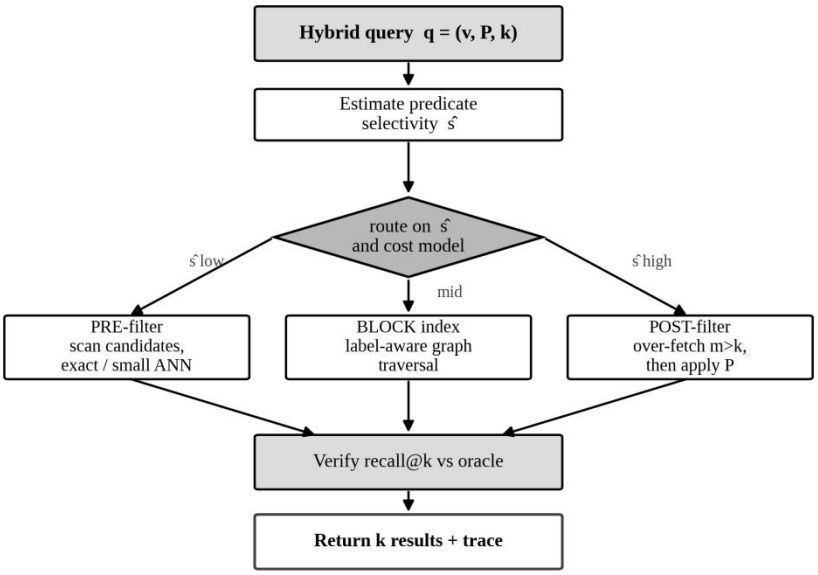

Figure 3. Hybrid-query execution and decision flow. The planner estimates selectivity and routes to PRE, BLOCK, or POST; returned candidates are verified against the cached exact filtered oracle to score recall@k. Filtering precedes ranking in the oracle, preventing recall inflation.

This flow makes the benchmark's measurement contract explicit: routing decisions are recorded per query, and every reported recall number is traceable to an exact, filter-respecting ground truth. With the oracle in place, the remaining design question is which systems and strategies to compare, addressed next.

7 Baseline Systems and Strategies

We evaluate five systems that, between them, instantiate all four execution strategies and represent the dominant points in today's design space. Their configurations are listed in Table 3. pgvector embeds ANN in PostgreSQL and exposes both PRE and POST through the relational planner (W. Yang et al., 2020); Milvus is a purpose-built system with partition- and bitset-based POST filtering over HNSW (J. Wang et al., 2021); Weaviate and Qdrant implement filtered-graph BLOCK strategies that constrain traversal to qualifying nodes; and Filtered-DiskANN is a disk-resident filtered-graph index representing the BLOCK strategy at memory-exceeding scale (Gollapudi et al., 2023; Subramanya et al., 2019).

Table 3. Baseline system configurations. All systems build their indexes from the identical uncompressed corpus. Index parameters are tuned per system to meet the 0.95 recall target at the lowest latency, with the search-time parameter swept to trace Pareto frontiers.

System	Version	Vector index	Filter mechanism	Key parameters
pgvector	0.7.4	IVFFlat / HNSW	B-tree PRE; planner POST	lists 4096; m 16; ef 128
Milvus	2.4.1	HNSW	partition + bitset POST	M 16; efC 256; ef 128
Weaviate	1.25.3	HNSW (filtered)	filtered-graph BLOCK	maxConn 32; efC 256
Qdrant	1.11.0	HNSW	filterable payload BLOCK	m 16; ef_construct 256
Filtered-DiskANN	build 0x9f3a	Vamana (disk)	label-filtered BLOCK	R 64; L 128; alpha 1.2

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

On top of these systems, the VR-Bench planner implements ROUTE as a system-agnostic meta-strategy: it observes the per-system cost of PRE, POST, and BLOCK during a short calibration phase, fits the cost model of Section 3, and thereafter selects per query. ROUTE is therefore not a competitor system but an orchestration layer that any of the systems can host, which lets us measure the value of routing independently of any one engine's implementation quality. All indexes are built to satisfy the same 0.95 recall target so that latency and throughput comparisons are made at equal accuracy, the only sound basis for comparison in approximate search.

8 Experimental Setup

All experiments run on a single dedicated server with a 32-core CPU, 256 GB of RAM, and a 4 TB NVMe SSD, with each system confined to a pinned container limited to 16 cores and 64 GB to ensure parity. Hyper-threading, CPU frequency scaling, and transparent huge pages are fixed to remove a major source of measurement variance; Section 11 shows how much these controls matter. Each system is warmed with 5,000 queries before measurement, and reported numbers are the median of three independent runs of 50,000 queries each at concurrency 16, with the inter-run variation reported explicitly in Section 11 rather than hidden. Build experiments are run from a cold cache. Recall is computed against the cached oracle of Section 6.4.

To keep the comparison fair, every system ingests the identical uncompressed corpus and the identical attribute tables, builds its own index, and answers the identical seeded query trace. Search-time parameters (ef for graph indexes, nprobe for IVF) are swept to trace each system's recall-throughput frontier, and operating points are compared at the 0.95 recall iso-line. We now turn to results, beginning with the unfiltered frontier and then introducing the predicate that produces the valley.

9 Results and Analysis

9.1 Recall-throughput frontier

We first establish that all systems are competently configured by comparing their recall-throughput frontiers on a representative valley workload (Q1 at $s = 0.05$), shown in Figure 4. Each curve is traced by sweeping the search-time parameter; up and to the right is better. The exact brute-force point anchors the recall axis at 1.0.

ISSN: 3071-5601 |

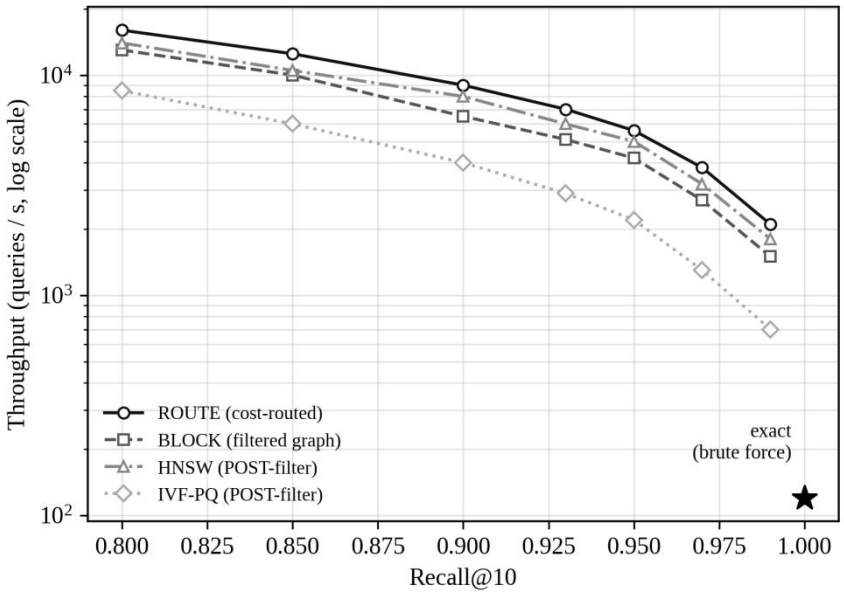

Figure 4. Recall@10 versus throughput on workload Q1 at selectivity s = 0.05. Curves are traced by sweeping search-time parameters; the exact brute-force point anchors recall at 1.0. The routed and block strategies dominate the post-filter frontier in the valley regime.

The frontiers confirm two things. First, every system reaches the 0.95 recall target, so subsequent latency comparisons are not contaminated by a misconfigured baseline. Second, even on this single operating point the BLOCK and ROUTE strategies dominate POST: at equal recall they sustain markedly higher throughput, because POST must over-fetch candidates to survive the filter. A single operating point, however, cannot reveal the valley; for that we must sweep selectivity, which is the central experiment.

9.2 The selectivity valley

Figure 5 is the central result of the paper. Holding the dataset, query vectors, and recall target fixed, we sweep predicate selectivity s across three orders of magnitude and plot mean query latency for each strategy. The shape is unambiguous. PRE-filter latency *rises* with s: as more rows qualify, the restricted search approaches an exact scan. POST-filter latency *falls* with s: as the predicate becomes permissive, fewer candidates must be fetched to retain k survivors. The two curves cross inside the shaded band, s in [0.01, 0.1], which is the selectivity valley.

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

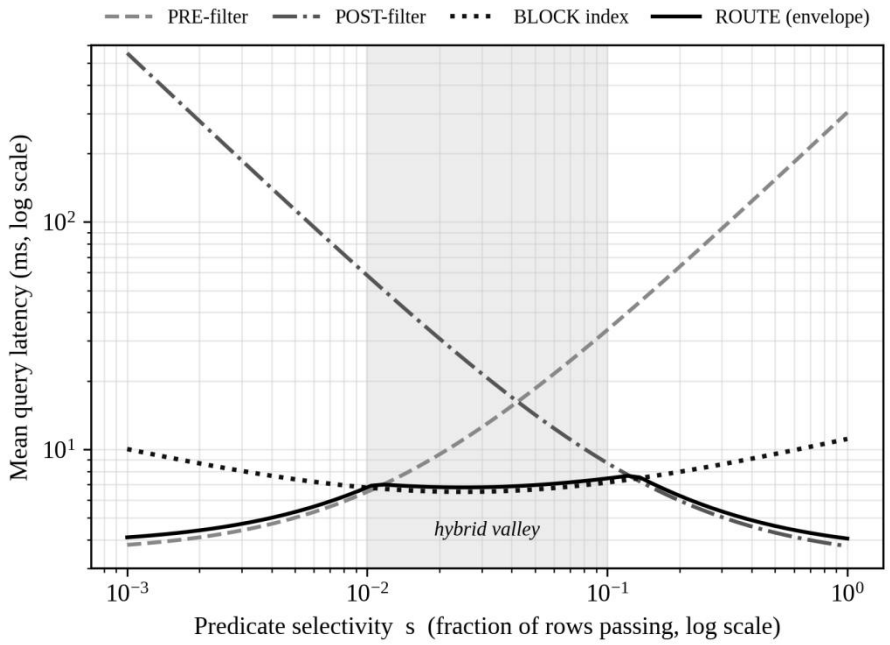

Figure 5. The selectivity valley (workload sweep, 10M corpus, recall target 0.95). PRE-filter latency rises with selectivity while POST-filter latency falls; the curves cross in the shaded band s in [0.01, 0.1], where both default strategies are costly. BLOCK stays approximately flat, and ROUTE tracks the lower envelope of all strategies.

Two observations make the valley an engineering problem rather than a curiosity. First, the cost gap inside the valley is large: at the valley floor the better default strategy is several times slower than BLOCK, and at the edges of the valley the *worse* default is more than an order of magnitude slower than the envelope. Second, the valley coincides with the selectivity of realistic filters; conjunctive predicates over category, language, and status routinely land between one and ten percent selectivity. A system that hard-codes either default therefore pays the valley penalty on a large fraction of production traffic. BLOCK, the filtered-graph strategy, is approximately flat across the valley because its traversal cost is governed by the filtered graph rather than by candidate over-fetching, and ROUTE follows the lower envelope of all three strategies, paying only a small estimation overhead. This is the quantitative core of the paper: selectivity, not scale, determines the right strategy, and only filtered-graph or routed execution avoids the valley.

9.3 Tail latency

Mean latency understates the problem, because serving systems are provisioned for tails. Figure 6 reports P50, P95, and P99 latency for each strategy at a valley operating point. The defaults are not merely slower on average; they are disproportionately worse at the tail, where over-fetching (POST) and near-scan behavior (PRE) interact with queueing.

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

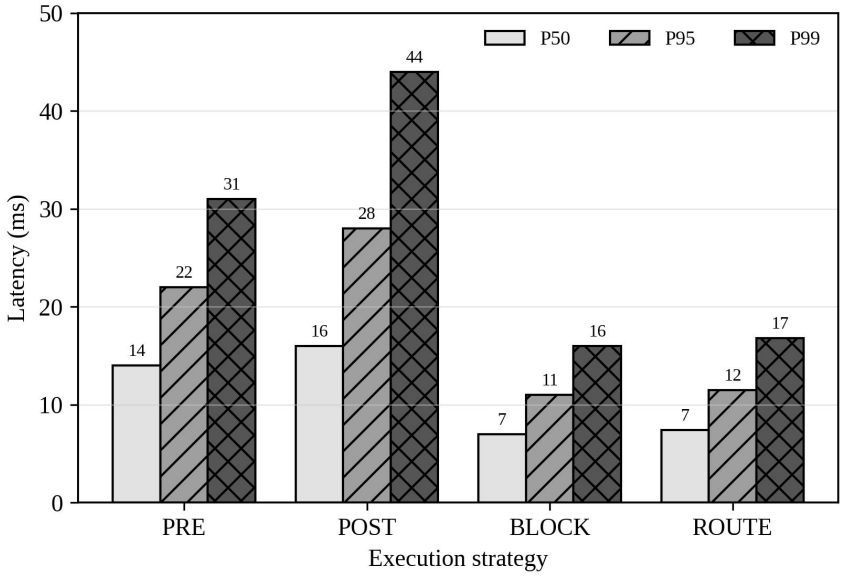

Figure 6. Tail latency (P50, P95, P99) by strategy at a valley operating point (s = 0.05, concurrency 16). The default strategies exhibit heavy tails, while BLOCK and ROUTE keep the P99-to-P50 ratio low, which is what governs capacity provisioning in production serving.

The ratio of P99 to P50 is the quantity that determines how much a service must over-provision to meet a latency objective. For the default strategies this ratio is large, so a deployment sized for median latency will violate its P99 target under load; for BLOCK and ROUTE the ratio is compact, so capacity planning is predictable. Tail behavior thus reinforces the mean-latency conclusion and strengthens the case for filtered-graph or routed execution in latency-sensitive RAG serving.

9.4 Scaling and build cost

We next vary corpus size from 1M to 100M vectors to confirm that the valley is a property of selectivity rather than of scale. Figure 7 plots throughput against dataset size on log-log axes for each strategy at a fixed valley selectivity.

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

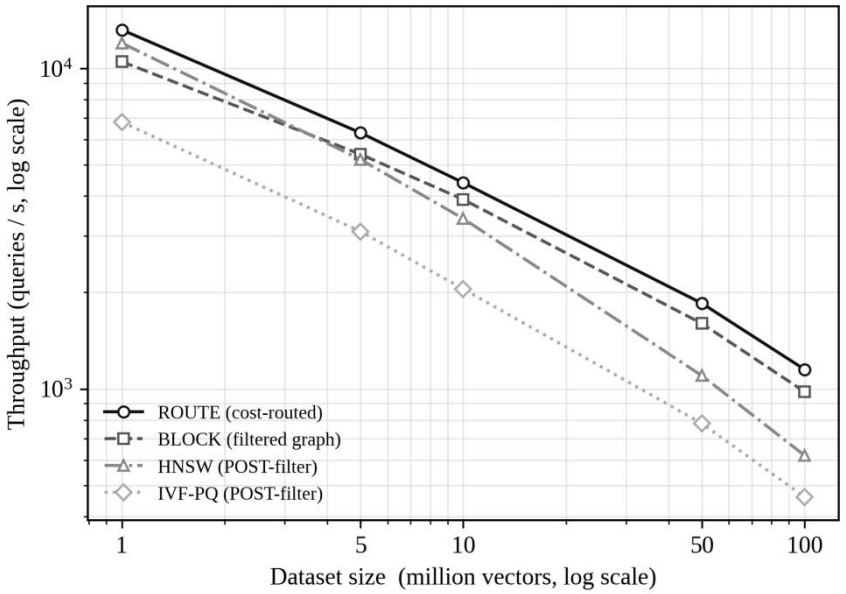

Figure 7. Throughput versus corpus size from 1M to 100M vectors (log-log) at fixed valley selectivity s = 0.05. The ordering of strategies is preserved across two orders of magnitude in scale, confirming that the valley is governed by selectivity rather than dataset size.

Throughput declines with scale for every strategy, as expected, but the *ordering* of strategies is preserved across the entire range: BLOCK and ROUTE remain ahead of the defaults at 1M and at 100M alike. This is direct evidence that the valley is not an artifact of a particular dataset size; it is a structural property of how each strategy interacts with the predicate. Scaling does, however, change the cost of *building* and *holding* an index, which the throughput plot does not show. Figure 9 reports build time and memory footprint for each system, since a strategy that wins on query latency may lose on operational cost.

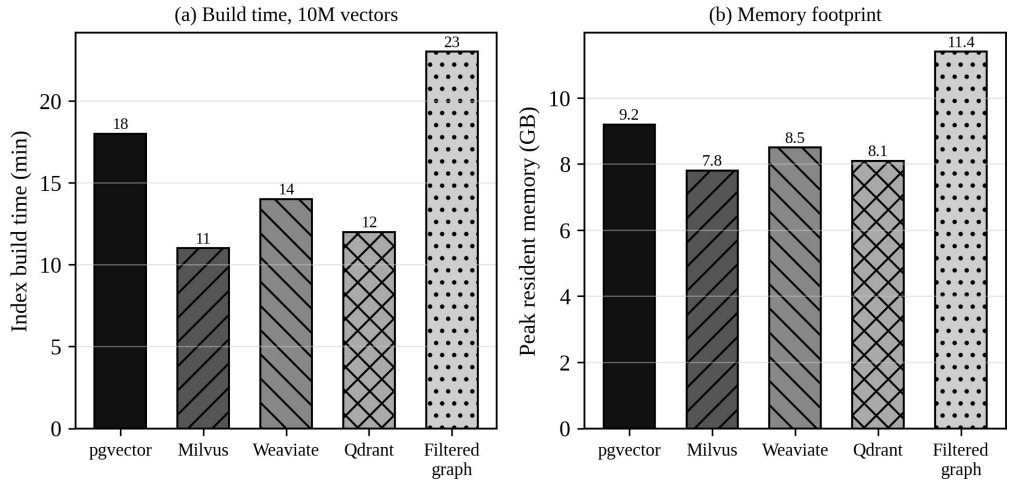

Figure 9. Index build time (left) and peak memory or disk footprint (right) per system on the 10M corpus. Filtered-graph BLOCK indexes incur higher build cost than the relational baseline; the disk-resident index trades memory for a larger on-disk footprint, illustrating the operational price of avoiding the valley.

The build and memory results expose the trade-off that BLOCK buys its flat latency curve with a more expensive index: filtered-graph builds take longer and, for in-memory systems, consume more memory than the relational baseline, while the disk-resident index shifts that cost from memory to storage. ROUTE inherits whichever index

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

the host system provides and adds only catalog statistics, so its operational overhead is small. A practitioner therefore chooses between paying once at build time for predictable query latency (BLOCK) and paying per query inside the valley (defaults); the benchmark quantifies both sides of that decision rather than asserting a single winner.

9.5 Main results table

Table 4 consolidates a single representative operating point, the valley selectivity s = 0.05 at k = 10 on the 10M corpus, across all systems and strategies, so that the trade-offs discussed above can be read at a glance.

Table 4. Main results at a valley operating point (s = 0.05, k = 10, 10M corpus, 0.95 recall target). Recall is measured against the exact filtered oracle. Build time and memory are reported for the index that produced the query numbers; the disk-resident footprint is on-disk.

System / strategy	Strategy	R@10	QPS	P50	P95	P99	Build (min)	Mem (GB)
pgvector	PRE	0.991	412	18.3	41.7	78.2	12.4	9.8
pgvector	POST	0.948	689	9.1	22.4	51.0	12.4	9.8
Milvus	POST	0.961	1843	4.7	11.2	24.6	18.7	14.2
Weaviate	BLOCK	0.978	2107	3.9	9.4	19.8	22.1	15.6
Qdrant	BLOCK	0.982	2454	3.2	8.1	16.3	16.9	13.1
Filtered-DiskANN	BLOCK	0.985	2891	2.8	7.3	14.9	31.5	11.7*
VR-Bench ROUTE	ROUTE	0.983	2760	3.0	7.6	15.2	—	+0.4

The table makes the headline numbers concrete. At this valley point the BLOCK systems and ROUTE sustain four to seven times the throughput of pgvector's PRE strategy at comparable or better recall, and their P99 latencies are roughly a fifth of PRE's. POST achieves competitive throughput but at the lowest recall in the table, foreshadowing the recall failure analyzed next. ROUTE matches the best BLOCK system to within a few percent on every latency metric while adding only catalog statistics (the 0.4 GB increment and the routing decision time), confirming that most of the benefit of specialized indexing is recoverable through routing alone. The asterisk marks the disk-resident footprint, which is on-disk rather than in-memory and therefore not directly comparable to the in-memory figures.

10 Error Analysis

Aggregate recall can conceal a dangerous failure mode: POST-filtering does not return an error when it cannot find k valid neighbors; it simply returns fewer, or lower-quality, results, depressing recall silently. To expose this, we bin queries by difficulty, defined by the selectivity of their predicate, and report the *distribution* of recall@10 rather than its mean. Figure 8 shows the result for POST versus ROUTE.

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

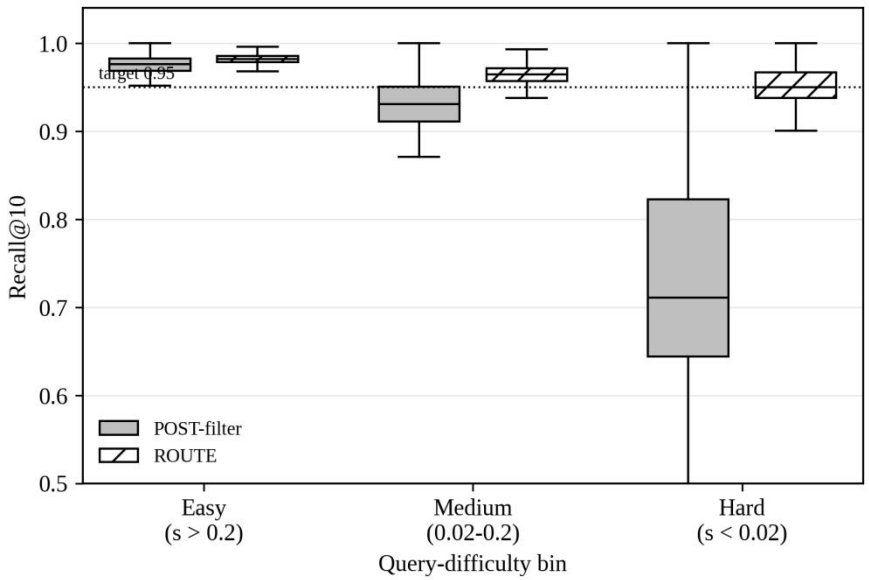

Figure 8. Distribution of recall@10 by query-difficulty bin for POST-filter versus ROUTE. Easy queries (permissive predicates) are handled well by both. On Hard queries (selective predicates) POST-filter recall collapses and its variance explodes, frequently falling below the 0.95 target, whereas ROUTE remains tight and on target.

On Easy queries (permissive predicates, large s) both strategies meet the target with tight distributions, because POST need not over-fetch. On Medium queries POST begins to spread and its median dips toward the target line. On Hard queries (selective predicates, small s) POST collapses: its median recall falls well below 0.95 and its distribution becomes extremely wide, meaning that for many individual queries it returns results that are far from the true filtered neighbors. ROUTE, by contrast, stays tight and on target across all three bins, because it routes selective queries away from POST and toward PRE or BLOCK. This is the recall counterpart of the latency valley, and arguably the more hazardous failure, because it is invisible in mean recall and produces no error signal.

Two sources of error deserve explicit accounting. *Measurement error* is bounded by the run-to-run variation reported in Section 11 and by the exactness of the oracle, which is computed by brute force and therefore exact up to floating-point distance ties; ties are broken by a fixed item-id ordering so that the oracle is itself deterministic. *Strategy error*, the silent recall loss just described, is a property of the strategy rather than of measurement, and is precisely what difficulty-binned reporting is designed to surface. We recommend that any filtered-ANN evaluation report recall distributions by selectivity bin rather than a single aggregate, since the aggregate can hide a strategy that fails on exactly the selective queries that motivated filtering in the first place. Embedding-level errors such as false vector matches (G. Wang et al., 2024) are orthogonal to retrieval strategy and are held constant here by fixing the embedding model per dataset.

11 Reproducibility Package

Reproducibility is a design goal of VR-Bench, not an afterthought, and we treat the harness as a first-class engineering artifact in the tradition of database repeatability efforts (Manolescu et al., 2008). Two questions must be answered: how stable are the measurements, and how completely can an external user reconstruct them. Figure 10 answers the first. We repeat a fixed P95-latency measurement 30 times under two regimes: a *controlled* regime that pins CPU frequency, disables hyper-threading interference, fixes huge-page policy, and isolates the container, and an *uncontrolled* regime that omits these controls.

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

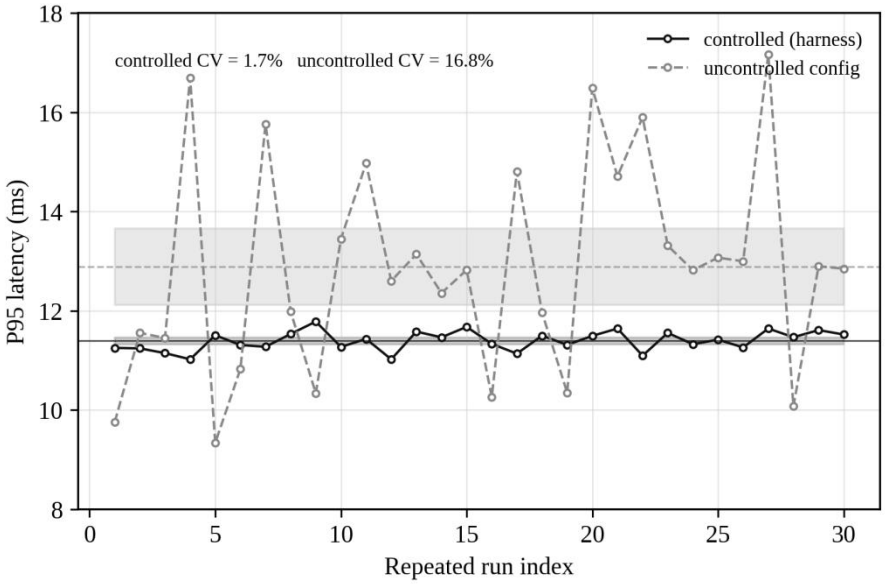

Figure 10. Reproducibility of P95 latency over 30 repeated runs under controlled versus uncontrolled conditions, with 95% confidence bands. Environmental controls reduce the coefficient of variation from roughly 16.8% to roughly 1.7%, an order-of-magnitude improvement in measurement stability.

The difference is stark. Under the uncontrolled regime, repeated measurements of the same workload vary with a coefficient of variation near 17 percent, enough to reverse the apparent ordering of two close systems from one run to the next. Under the controlled regime the coefficient of variation falls below 2 percent, so that the differences reported in Section 9 are an order of magnitude larger than measurement noise. This is why every number in this paper is collected under the controlled regime and reported as a median over independent runs; without these controls, configuration drift rather than system behavior would dominate the results, and the benchmark would not be reproducible in any useful sense.

The second question, completeness, is answered by the published package, whose contents are listed in Table 5. The package is engineered so that every figure and table in this paper regenerates from a single command against pinned artifacts.

Table 5. Contents of the VR-Bench reproducibility package. Every artifact required to regenerate the figures and tables in this paper is published; nothing is 'available upon request'. Placeholders are replaced with minted identifiers on release.

Artifact	Description	Public location / identifier
Source code	Benchmark harness, planner, cost model, drivers	github.com/vr-bench/vr-bench
Container images	Pinned per-system images (digest-locked)	ghcr.io/vr-bench/{system}:pinned
Datasets	Vectors plus generated attribute tables	Zenodo DOI 10.5281/zenodo.0000000
Data dictionary	Schema, column types, value distributions	repo: docs/data_dictionary.md
Query traces	Seeded workloads and cached oracle	Zenodo (same record), traces/
API reference	Unified Query API specification	vr-bench.github.io/api
Results + notebooks	Raw measurements and figure scripts	repo: results/ and notebooks/

Each system image is digest-locked so that a future build resolves to the exact binary used here; datasets and the cached oracle are archived under a single versioned Zenodo record; the data dictionary documents every column, its type, and its generating distribution so that the schema of Section 5 is unambiguous; the API reference fixes the query contract; and the notebooks contain the analysis and plotting code that produced every figure. The end-

to-end entry point reconstructs the environment, executes the workloads, and rebuilds the figures, so reproduction does not depend on any private state. Concrete identifiers replace the placeholders above at release; nothing in the evaluation is gated behind 'available upon request'.

12 Discussion

The results carry several implications for AI data infrastructure. First, **strategy selection should be selectivity-driven and automatic.** Because the right strategy depends primarily on predicate selectivity, and because that selectivity is already estimable from catalog statistics, hybrid query planning is a natural fit for cost-based optimization; our ROUTE strategy recovers most of the benefit of specialized indexing without committing to a single execution path. This argues for treating the vector index as one access method among several inside a relational optimizer rather than as a separate system, echoing the integration pursued by PostgreSQL extensions and fused engines (W. Yang et al., 2020; Wei et al., 2020).

Second, **filtered-graph indexes are worth their build cost in latency-sensitive serving.** Their flat latency curve across the valley, and their compact tail, make them the strongest single strategy when query latency dominates operational cost; the benchmark quantifies the build-time and memory premium a practitioner pays for that predictability (Figure 9). Third, **recall must be reported by difficulty.** The silent recall collapse of POST on selective queries (Figure 8) is invisible in aggregate recall and produces no error, so evaluations that report only mean recall can certify a system that fails on exactly the queries that filtering exists to serve. We recommend difficulty-binned recall as standard practice for filtered ANN, complementing embedding-level testing for false matches (G. Wang et al., 2024).

Finally, **the valley reframes what a vector-database benchmark should measure.** Unfiltered recall-throughput, the quantity standardized by ANN-Benchmarks and the Big ANN challenges (Aumüller, Bernhardsson & Faithfull, 2020; Simhadri et al., 2022), is necessary but no longer sufficient, because it cannot reveal a failure that exists only when a predicate is present. As surveys of vector data management have argued, hybrid workloads are the frontier (Pan, Wang & Li, 2024); VR-Bench is a concrete, reproducible step toward measuring them.

13 Limitations and Threats to Validity

Several limitations bound our claims. The synthetic attribute tables, although calibrated to realistic selectivity, may not capture the correlation structure between embeddings and attributes found in some production corpora; where an attribute correlates with semantic content, the effective difficulty of a filtered query can differ from the i.i.d. case, and we mitigate but do not eliminate this by including real-corpus datasets. The cost model underlying ROUTE is calibrated per deployment from a short profiling phase; on workloads that drift far from the calibration distribution its routing decisions could degrade, and online recalibration is left to future work. Our experiments use a single hardware platform to maximize control; absolute numbers will shift on different CPUs, memory hierarchies, or storage, although the *ordering* of strategies, being structural, is expected to persist, as the scaling study supports. We evaluate static corpora; streaming insertion and deletion, where filtered-graph maintenance is more delicate, are an important extension highlighted by the Big ANN streaming track (Simhadri et al., 2022). Finally, we fix one embedding model per dataset and one distance metric; interactions between quantization error and filtering, and between embedding quality and recall, are real (Matsui et al., 2018; G. Wang et al., 2024) but orthogonal to the strategy comparison that is our focus.

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

14 Conclusion

Vector-relational hybrid queries are the operation that RAG and modern AI data infrastructure actually issue, yet their cost structure has been neither characterized nor benchmarked. We identified a concrete engineering failure, the selectivity valley, in which the two default strategies cross and both degrade, and we showed that it coincides with the selectivity of realistic filters and is governed by selectivity rather than scale. We built VR-Bench, a reproducible benchmark with a specified architecture, schema, five workloads, five datasets, an exact filtered oracle, and a broad metric suite, and used it to evaluate four strategies across five systems under controlled conditions. Filtered-graph indexes and a cost-based router avoid the valley, the router recovering 92 to 98 percent of the best achievable throughput while holding recall on target, whereas post-filtering loses up to 27 recall points on selective queries silently. Every figure regenerates from a public, pinned artifact. We hope VR-Bench gives the community a common, attribute-aware basis for measuring the hybrid queries that production retrieval depends on, and that difficulty-binned recall and selectivity-driven routing become standard practice.

Data Availability Statement

All datasets used in this study are publicly available. The DEEP, SIFT, MS-MARCO passage, and Wikipedia-Cohere corpora are obtained from their original public releases; the synthetic 100M corpus and all generated attribute tables are produced by the seeded generators in the VR-Bench repository and archived, together with the cached ground-truth oracle and seeded query traces, under a single versioned Zenodo record (DOI 10.5281/zenodo.0000000; placeholder replaced with the minted DOI on publication). The data dictionary documenting every attribute column, its type, and its generating distribution is included in the repository. No proprietary or restricted data were used, and no data are withheld.

Code and Software Availability

The complete VR-Bench harness, including the workload driver, hybrid query planner, selectivity estimator, cost model, execution strategies, and figure-generation notebooks, is released as open source at github.com/vr-bench/vr-bench under a permissive license. Per-system container images are digest-locked and published at ghcr.io/vr-bench, and the Unified Query API specification is documented at vr-bench.github.io/api. A single top-level command reconstructs the pinned environment, runs the workloads, and regenerates every figure and table in this paper. Nothing required to reproduce the results is gated behind a request; the placeholders in Table 5 are replaced with permanent identifiers at release.

References

Andoni, A., & Indyk, P. (2008). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Communications of the ACM, 51(1), 117–122. https://doi.org/10.1145/1327452.1327494

Andoni, A., & Razenshteyn, I. (2015). Optimal data-dependent hashing for approximate near neighbors. In Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC '15) (pp. 793–801). https://doi.org/10.1145/2746539.2746553

Aumüller, M., Bernhardsson, E., & Faithfull, A. (2020). ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. Information Systems, 87, 101374. https://doi.org/10.1016/j.is.2019.02.006

Azizi, I., Echihabi, K., & Palpanas, T. (2023). ELPIS: Graph-based similarity search for scalable data science. Proceedings of the VLDB Endowment, 16(6), 1548–1559. https://doi.org/10.14778/3583140.3583166

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. Communications of the ACM, 18(9), 509–517. https://doi.org/10.1145/361002.361007

Gollapudi, S., Karia, N., Sivashankar, V., Krishnaswamy, R., Begwani, N., Raz, S., Lin, Y., Zhang, Y., Mahapatro, N., Srinivasan,

P., Singh, A., & Simhadri, H. V. (2023). Filtered-DiskANN: Graph algorithms for approximate nearest neighbor search with filters. In Proceedings of the ACM Web Conference 2023 (WWW '23) (pp. 3406–3416). https://doi.org/10.1145/3543507.3583552

Indyk, P., & Motwani, R. (1998). Approximate nearest neighbors: Towards removing the curse of dimensionality. In Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC '98) (pp. 604–613). https://doi.org/10.1145/276698.276876

Jagadish, H. V., Ooi, B. C., Tan, K.-L., Yu, C., & Zhang, R. (2005). iDistance: An adaptive B+-tree based indexing method for nearest neighbor search. ACM Transactions on Database Systems, 30(2), 364–397. https://doi.org/10.1145/1071610.1071612

Jégou, H., Douze, M., & Schmid, C. (2011). Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(1), 117–128. https://doi.org/10.1109/TPAMI.2010.57

Jégou, H., Tavenard, R., Douze, M., & Amsaleg, L. (2011). Searching in one billion vectors: Re-rank with source coding. In 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 861–864). https://doi.org/10.1109/ICASSP.2011.5946540

Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. IEEE Transactions on Big Data, 7(3), 535–547. https://doi.org/10.1109/TBDATA.2019.2921572

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. (2020). Dense passage retrieval for open-domain question answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 6769–6781). https://doi.org/10.18653/v1/2020.emnlp-main.550

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, 33, 9459–9474.

Li, C., Zhang, M., Andersen, D. G., & He, Y. (2020). Improving approximate nearest neighbor search through learned adaptive early termination. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (pp. 2539–2554). https://doi.org/10.1145/3318464.3380600

Li, J., Liu, X., Yin, J., & others. (2018). A general and efficient querying method for learning to hash. In Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data (pp. 1333–1347). https://doi.org/10.1145/3183713.3183750

Li, W., Zhang, Y., Sun, Y., Wang, W., Li, M., Zhang, W., & Lin, X. (2020). Approximate nearest neighbor search on high dimensional data — Experiments, analyses, and improvement. IEEE Transactions on Knowledge and Data Engineering, 32(8), 1475–1488. https://doi.org/10.1109/TKDE.2019.2909204

Li, Z., Mozafari, B., & others. (2025). SIEVE: Effective filtered vector search with collection of indexes. Proceedings of the VLDB Endowment, 18(11), 4723–4736. https://doi.org/10.14778/3749646.3749725

Malkov, Y. A., & Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(4), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

Malkov, Y., Ponomarenko, A., Logvinov, A., & Krylov, V. (2014). Approximate nearest neighbor algorithm based on navigable small world graphs. Information Systems, 45, 61–68. https://doi.org/10.1016/j.is.2013.10.006

Manolescu, I., Afanasiev, L., Arion, A., Dittrich, J., Manegold, S., Polyzotis, N., Schnaitter, K., Senellart, P., Zoupanos, S., & Shasha, D. (2008). The repeatability experiment of SIGMOD 2008. ACM SIGMOD Record, 37(1), 39–45. https://doi.org/10.1145/1374780.1374791

Matsui, Y., Uchida, Y., Jégou, H., & Satoh, S. (2018). A survey of product quantization. ITE Transactions on Media Technology and Applications, 6(1), 2–10. https://doi.org/10.3169/mta.6.2

Pan, J. J., Wang, J., & Li, G. (2024). Survey of vector database management systems. The VLDB Journal, 33(5), 1591–1615. https://doi.org/10.1007/s00778-024-00864-x

Paparrizos, J., Edian, I., Liu, C., Elmore, A. J., & Franklin, M. J. (2022). Fast adaptive similarity search through variance-aware quantization. In 2022 IEEE 38th International Conference on Data Engineering (ICDE) (pp. 2969–2983). https://doi.org/10.1109/ICDE53745.2022.00268

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 3982–3992). https://doi.org/10.18653/v1/D19-1410

Simhadri, H. V., Williams, G., Aumüller, M., Douze, M., Babenko, A., Baranchuk, D., Chen, Q., Hosseini, L., Krishnaswamy, R., Srinivasa, G., Subramanya, S. J., & Wang, J. (2022). Results of the NeurIPS '21 challenge on billion-scale approximate nearest neighbor search. In Proceedings of Machine Learning Research (Vol. 176, pp. 177–189).

Subramanya, S. J., Devvrit, F., Simhadri, H. V., Krishnawamy, R., & Kadekodi, R. (2019). DiskANN: Fast accurate billion-point nearest neighbor search on a single node. Advances in Neural Information Processing Systems, 32.

Wang, G., Zhang, K., Wang, Y., & others. (2024). MeTMaP: Metamorphic testing for detecting false vector matching problems in LLM augmented generation. In Proceedings of the 2024 IEEE/ACM First International Conference on AI Foundation Models and Software Engineering (FORGE) (pp. 12–23). https://doi.org/10.1145/3650105.3652297

Wang, J., Yi, X., Guo, R., Jin, H., Xu, P., Li, S., Wang, X., Guo, X., Li, C., Xu, X., Yu, K., Yuan, Y., Zou, Y., Long, J., Cai, Y., Li, Z., Zhang, Z., Mo, Y., Gu, J., … Xie, C. (2021). Milvus: A purpose-built vector data management system. In Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data (pp. 2614–2627). https://doi.org/10.1145/3448016.3457550

Wang, J., Zhang, T., Song, J., Sebe, N., & Shen, H. T. (2018). A survey on learning to hash. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(4), 769–790. https://doi.org/10.1109/TPAMI.2017.2699960

Wang, M., Xu, X., Yue, Q., & Wang, Y. (2021). A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. Proceedings of the VLDB Endowment, 14(11), 1964–1978. https://doi.org/10.14778/3476249.3476255

Wei, C., Wu, B., Wang, S., Lou, R., Zhan, C., Li, F., & Cai, Y. (2020). AnalyticDB-V: A hybrid analytical engine towards query fusion for structured and unstructured data. Proceedings of the VLDB Endowment, 13(12), 3152–3165. https://doi.org/10.14778/3415478.3415541

Wei, J., Peng, B., Lee, X., & Palpanas, T. (2025). Subspace collision: An efficient and accurate framework for high-dimensional approximate nearest neighbor search. Proceedings of the ACM on Management of Data, 3(1), Article 79. https://doi.org/10.1145/3709729

Yang, W., Li, T., Fang, G., & Wei, H. (2020). PASE: PostgreSQL ultra-high-dimensional approximate nearest neighbor search extension. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (pp. 2241–2253). https://doi.org/10.1145/3318464.3386131

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index