VFAST Transactions on Software Engineering
https://vfast.org/journals/index.php/VTSE@ 2026, ISSN(e): 2309-3978, ISSN(p): 2411-6246
Volume 14, Number 1, January-March 2026 *pp: 335-349*

Vector+SQL Retrieval with Selectivity Workloads: Measuring Tail Latency and Quality under Filtered Top-K

Sadaf Bibi[1*], **Faheem Akhter Rajput**[1], **Muhammad Younis**[1], **Sadia Bibi**[2], **Basit Raza**[3]

[1]Department of Artificial Intelligence, Aror University of Art, Architecture, Design and Heritage, Sukkur, Sindh, Pakistan; [2]Department of Computer Science, Sukkur IBA University, Sukkur, Sindh, Pakistan; [3]Institute of Computer Science, Shah Abdul Latif University, Khairpur Mirs, Sindh, Pakistan

Keywords: *hybrid retrieval, vector search, SQL filtering, tail latency, selectivity, approximate nearest neighbors,*

Journal Info:
Submitted:
February 10, 2026
Accepted:
March 25, 2026
Published:
March 31, 2026

Abstract We study Vector+SQL hybrid retrieval under structured filtering conditions and propose a reproducible Top-K evaluation framework for uniformly analyzing the quality and tail latency of different retrieval strategies. We construct controllable selectivity workloads on three text domains: arXiv, news, and movies, and compute exact ground truth within the same filter candidate set to ensure fair comparison. Experiments uniformly evaluate exact prefilter, ANN post-filter, two-stage probing, partition-aligned retrieval, and adaptive routing, and report Recall@20 and p50/p95/p99 latency. The results show that filter selectivity and predicate structure significantly affect the quality–latency trade-off in hybrid retrieval: exact prefilter has the highest accuracy but the largest tail latency; ANN post-filter is faster, but recall drops significantly under strict filtering; adaptive routing achieves more robust overall performance on the evaluated workloads. These results demonstrate that reporting only average latency is insufficient to reflect the true system cost of hybrid retrieval. Our report provides a systematic framework for evaluating, designing, and deploying hybrid retrieval strategies under filtering conditions, accounting for correctness, tail latency, and reproducibility.

***Correspondence author email address:** sadaf.bibi.dev@gmail.com
DOI: 10.21015/vtse.v14i1.2353

1 Introduction

Many practical applications, such as recommender systems, news retrieval, Q&A forums, and academic literature search, require both semantic similarity matching and structured constraint filtering. In other words, a system must perform semantic retrieval over vector representations while also applying SQL-style predicates, such as category, time, author, price range, or language, to support joint filtering. Therefore, Vector+SQL hybrid retrieval is no longer merely a research topic in vector databases; it is a crucial capability of production-grade data systems [1, 12–14].

The core system tension of this problem lies in the ongoing conflict between quality and cost. On the one hand, performing structured filtering first, followed by exact ranking within the filtered candidate set, typically provides more reliable results; however, when the corpus is large, query load fluctuates significantly, or the candidate set remains large, tail latency (especially p95/p99) often deteriorates rapidly. On the other hand, Approximate Nearest Neighbor (ANN) mmethods such as HNSW, FAISS, DiskANN, and SPANN can significantly reduce query latency, but under strict filtering or low selectivity conditions, post-filtering failures, insufficient

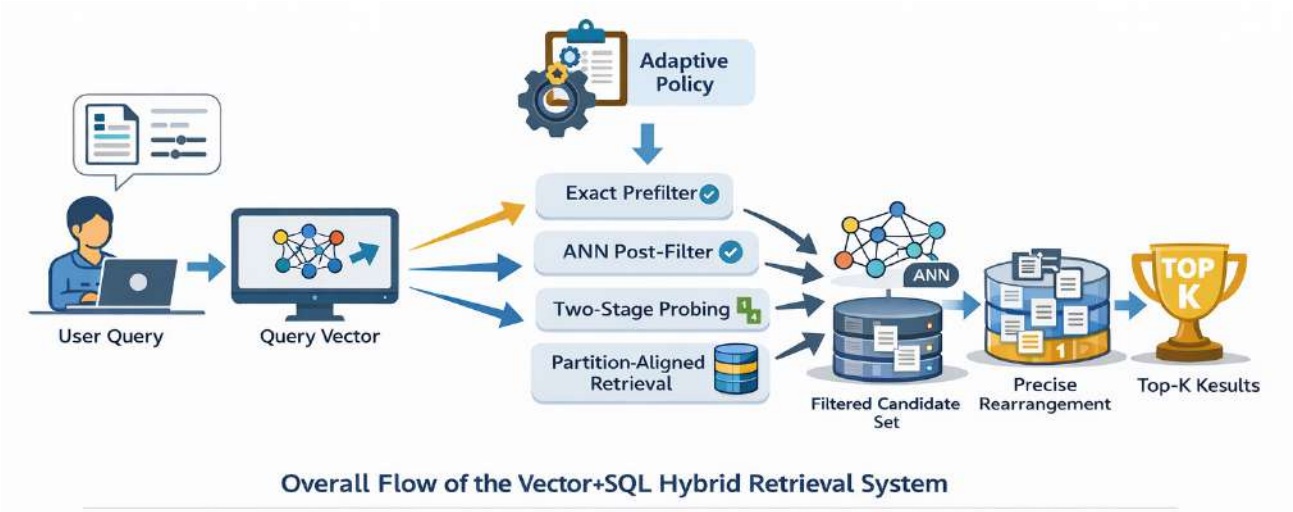

Figure 1. Vector+SQL Overall workflow of the Vector+SQL hybrid retrieval framework.

effective candidates, and improper path selection can all lead to decreased recall. Therefore, compared to unfiltered semantic retrieval, hybrid retrieval with structured conditions more closely approximates the complex decision-making scenarios in real-world system deployments, as illustrated in Figure, 1.

Although vector retrieval and ANNs have been extensively studied, existing literature often focuses on unfiltered vector search, vector database system design, or approximate retrieval algorithms themselves. A unified, systematic, and fair evaluation framework for filtered Top-K behavior under SQL-style filtering conditions remains lacking. Especially in filtered retrieval scenarios, if the ground truth is not defined within the same filtered candidate set, quality comparisons between different methods may be distorted. Accordingly, we address the following question: how do the retrieval quality and tail latency of different strategies change under different selectivity conditions when semantic vector retrieval and structured predicate filtering coexist? To answer this question, we construct a unified evaluation framework for filtered Top-K and compare the behavior of exact prefilter, ANN post-filter, two-stage probing, partition-aligned retrieval, and adaptive routing within the same filtering context.

The main contributions of this study are as follows:

1. We propose a correctness-safe filtered Top-K evaluation protocol: for any query and filtering predicate, a corresponding candidate set is first constructed, and the exact ground truth is then computed only within this candidate set, thereby avoiding evaluation bias caused by comparison with the globally unfiltered top-K.
2. We construct a Vector+SQL hybrid retrieval workload with controllable selectivity to analyze how filtering intensity affects retrieval quality and tail latency.
3. We compare several representative strategies within a unified framework, including exact prefilter, ANN post-filter, two-stage probing, partition-aligned retrieval, and adaptive routing, and analyze their quality–latency trade-offs in filtered Top-K scenarios.
4. We emphasize tail latency rather than average latency alone and report p50/p95/p99 to better characterize system performance in production-like environments [17].
5. We provide reproducible experimental outputs, including per-query logs, summary tables, latency analysis under fixed recall targets, scalability curves, and cost profiles, thereby providing a verifiable experimental foundation for future research.

2 Background and Related Work

In modern vector database management systems (VDBMSs), Vector+SQL hybrid retrieval has emerged as

a central systems problem [1–3, 12–16]. Among these, ANN methods based on graph structures and quantization techniques form the main algorithmic foundation for large-scale scalable vector retrieval [4–10]. Furthermore, tail-aware, benchmark-driven evaluation is also essential for characterizing performance behavior in real-world systems [11, 17, 18].

2.1 Vector Search and ANN Indexes

Vector similarity retrieval is a core problem in modern information retrieval and data management. In recent years, graph-structured indexes have performed particularly well in ANN retrieval. HNSW has achieved a good balance between high recall and low latency through hierarchical navigable small-world graphs and has become a representative graph indexing method [4]. FAISS has promoted the popularization of large-scale vector retrieval primitives and engineering implementations, and supports brute-force search, quantized indexing, and high-performance top-K selection in GPU environments [5]. In terms of vector compression, Product Quantization (PQ) provides a classic path for storage compression and approximate distance calculation of high-dimensional vectors [6]. Furthermore, DiskANN and SPANN have promoted the low-cost deployment of billion-level vector retrieval through two routes: disk-friendly graph indexing and memory-disk hybrid indexing, respectively [7, 8]. Overall, these methods collectively lay the foundation for modern ANN systems to balance throughput, recall, and memory costs.

2.2 Hybrid Retrieval and Filtered ANN

Unlike pure vector search, queries in real-world applications often involve both semantic similarity requirements and structured constraints, such as "finding similar items, but specifying `category = X` and `time > Y`". These hybrid queries require the system to process both vector operators and relational predicates, making them more challenging than traditional ANN retrieval. The most direct approach is to first return candidates from the global ANN index and then apply structured filtering to them; however, under low selectivity conditions, this path often suffers from insufficient effective candidates, leading to a decline in the quality of the filtered Top-K. Existing research on this issue mainly follows three directions: first, partitioned or routed indexes, which narrow the search scope by splitting data according to filter keys; second, two-stage retrieval, which balances efficiency and quality by progressively expanding the candidate budget and combining exact reranking; and third, more tightly integrating vector operators into the database system, enabling them to work collaboratively with SQL filtering, joins, and full-text search. Recent reviews of vector databases further point out that hybrid queries involving attribute constraints and vector retrieval remain a core research challenge in VDBMS design.

2.3 Tail Latency, Benchmarking, and Reproducibility

In system evaluation, reporting only average latency is often insufficient to characterize true service quality, as user experience is often dominated by high-quantile tail latency, especially at p95 and p99. This problem is even more pronounced in hybrid retrieval: the filtering selectivity, candidate set size, and reordering cost all vary with the query, resulting in significant heterogeneity between queries. On the other hand, work such as ANN-Benchmarks has promoted standardized evaluations of approximate nearest neighbor algorithms, enabling comparisons of latency-quality relationships between different methods under a unified interface. However, for hybrid retrieval with structured filtering conditions, there is still a lack of a systematic benchmark framework that defines ground truth within the same filtered candidate set, reports both quality and tail latency, and unifies the comparison of multiple strategies.

Therefore, fair evaluation of filtered Top-K algorithms, tail latency characterization, and experimental reproducibility constitute the direct background for this work. Beyond the algorithms and systems themselves, the reproducibility practices in dataset releases are also valuable for benchmark-oriented research. Recent public dataset works demonstrate that providing machine-readable metadata, summary tables, visualizations, and reproducible generation code helps improve data auditability, experimental transparency, and the efficiency of subsequent benchmark reuse [19]. Although this type of work does not directly address Vector+SQL hybrid retrieval, it provides methodological insight for building filtered retrieval benchmarks and designing evaluation protocols.

2.4 Gap Statement

In summary, existing research has made significant progress in ANN algorithms, large-scale vector database systems, and a comprehensive overview of vector databases. However, for hybrid retrieval with filtering conditions, a unified evaluation framework that simultaneously meets the following requirements is still lacking: 1) the ground truth is defined within the filtered candidate set; 2) the workload has a controllable selectivity; 3) the evaluation covers both quality and tail latency; and 4) it can uniformly compare multiple candidate generation and execution paths on the same query set. Against this backdrop, we aim to characterize the behavioral boundaries of Vector+SQL hybrid retrieval under different filtering loads from a systems evaluation perspective.

3 Problem Setup and Research Questions

3.1 System Model

We model Vector+SQL hybrid retrieval as a filtered Top-K retrieval problem with structured predicate constraints. Let the corpus be represented as a table-structured collection D, where each record corresponds to a document or object, containing a unique identifier `rid`, a data domain label `dataset`, a text field `text`, and structured attributes available in some datasets (such as `category` and `year`). For any query, the system simultaneously receives two types of input: one is the text query, mapped to a query vector v_q via the embedding function $f(\cdot)$; the other is the structured predicate P, such as `category IN (...)` or `year >= ...`. Therefore, the retrieval objective can be expressed as: among records satisfying the predicate P, return the top-K results that are closest to the query vector v_q.

Under this setting, the predicate P first defines a filter candidate set:

$$C = \{d_i \in D : P(d_i) = \text{true}\}$$

The system then performs ranking only within C and returns the top-K results. This modeling form directly characterizes the key challenge of hybrid retrieval: the size of the candidate set $|C|$ corresponding to different queries may differ significantly, thus causing the same retrieval strategy to perform inconsistently under different filtering intensities. To ensure the interpretability and reproducibility of subsequent comparisons. All strategies are evaluated under a unified problem definition.

3.2 Evaluation Metrics

We characterize system behavior from three dimensions: quality, latency, and workload structure.

3.2.1 Retrieval Quality under Filtering Conditions

We use Recall@K as the primary metric in the filtered Top-K setting to measure the consistency between the results returned by a certain strategy and the filtered ground truth. For a given query q and predicate P, the ground truth is not calculated on the global corpus, but only within the same filtered candidate set C based on normalized vector similarity to obtain the exact top-K results. Accordingly, Recall@K is defined as:

$$\text{Recall@}K = \frac{|\text{GT@}K \cap \text{Pred@}K|}{K}$$

where $\text{GT@}K$ denotes the exact top-K results on the candidate set C, and $\text{Pred@}K$ denotes the top-K results returned by a given strategy.. This definition avoids evaluation bias caused by comparing the filtered prediction results with the global unfiltered ground truth.

3.2.2 Latency Metrics, Especially Tail Latency

Considering that the execution cost of hybrid retrieval varies with filtering intensity and candidate size, we report $p50$, $p95$, and $p99$ latencies simultaneously to reflect typical query experience and high-quantile tail behavior, respectively. This setting is more suitable for characterizing the stability and cost of the system in real-world service scenarios compared to reporting only the average latency.

3.2.3 Workload Attributes

In addition to output metrics, we also record workload attributes directly related to explaining retrieval behavior, including filter selectivity, candidate set size $|C|$, and the routing proportion of the sub-strategies under the adaptive policy. This information is directly helpful in understanding the quality–latency trade-offs of different methods.

3.3 Research Questions

Based on the above-mentioned questions, we focus on answering the following four research questions:

RQ1. In hybrid retrieval with SQL-style filtering conditions, what is the trade-off between Recall@K and tail latency (p95/p99)?

RQ2. Compared to the exact ranking baseline, how do ANN-like strategies perform under different selectivity conditions, especially the differences in behavior between strict and loose filtering scenarios?

RQ3. Can adaptive hybrid strategies reduce tail latency compared to exact prefilter while maintaining robust retrieval quality under evaluated workloads? We focus on workload-conditioned gains, rather than a conclusion of "optimal in all scenarios."

RQ4. When the corpus size grows under controlled conditions, how do the system's latency and quality trends change? Does the relevant behavior remain stable, or does it deteriorate with increasing scale?

3.4 Correctness-Safe Evaluation Scope

In the filtered retrieval scenario, we define the ground truth as the exact top-K results within the same filtered candidate set. Specifically, for any query q and structured predicate P, the system first generates a candidate set C through SQL filtering. Then, it calculates the exact ranking result based on normalized embedding similarity only on this set and uses it as the ground truth shared by all comparison strategies. If the candidate set for some queries is too large and requires uniform upper limit control, all strategies are compared on the same candidate universe to ensure evaluation fairness. This design directly avoids the evaluation bias common in filtered retrieval, which involves comparing the filtered prediction results with the globally unfiltered ground truth.

To improve experimental transparency, we report the main workload and evaluation parameters under uniform settings. The current experiments cover three data domains: arXiv, news, and movies; he retrieval cutoff is fixed at $K = 20$; the filter selectivity ranges from 0.1% to 80%; the HNSW `efSearch` values are {16, 32, 64, 128, 256}; the base candidate budget for the ANN post-filter is set to 200; two-stage probing uses a progressively expanding probe ladder [1,2,4]; when the candidate set is too large, a uniform candidate upper bound is used to support fair and reproducible exact evaluation. Table 1 summarizes the main symbols and terms used in this study.

4 Datasets and Workloads

4.1 Datasets

We construct a cross-domain Vector+SQL hybrid retrieval benchmark using three publicly available text datasets, corresponding to academic literature, news text, and movie metadata scenarios, respectively. These three datasets collectively cover long text summaries, short news articles, and highly structured content descriptions, facilitating the observation of strategy behavior differences under different corpus sizes and filtering structures. The unified corpus in the current experiment contains 190,730 arXiv documents, 171,500 news documents, and 43,323 movie documents. Since these actual sizes did not exceed the single dataset limits set in the code (arXiv 200k, news 180k, movies 60k), and the total corpus size of 405,553 documents was lower than the global limit of 520,000, no additional global random down sampling was triggered in the current Q2 experiment, as illustrated in Figure, 2.

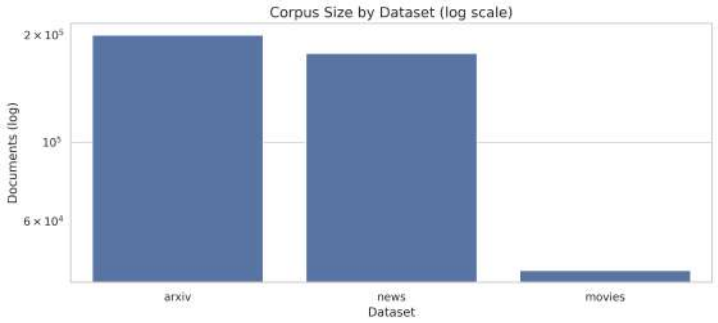

Figure 2. Corpus size by dataset (log scale). The three benchmark datasets differ substantially in scale, which affects latency and scalability behavior.

Specifically, the arXiv dataset uses concatenated titles and abstracts as retrieval text, and extracts `category` and `year` information from the category field and version time; the news dataset uses `headline` and `short_description` as the text body, while retaining the parsable year from the news category and date; the movies dataset uses movie titles and synopses to construct text fields, and extracts the main category from `genres` and the year from `release_date`. All three

Table 1. Explanation of the main symbols and terms used in this study.

Symbol / Term	Meaning	Notes / Example
D	Full corpus (all documents)	Union of all datasets
N	Total number of documents in a corpus/scope	$N = \|D\|$
d_i	The i-th document	Stored as a row in the corpus table
`rid`	Unique record/document identifier	Integer ID for each document
`dataset`	Dataset/domain label of a document	e.g., `arxiv`, `news`, `movies`
`text`	Document text field used for embedding	title + abstract / overview / body, etc.
`category`	Optional categorical attribute	Used for filters / partitions (if available)
`year`	Optional temporal attribute	Used for range filters (if available)
$f(\cdot)$	Embedding function	Maps text to a vector (e.g., SBERT)
$x_i \in \mathbb{R}^d$	Embedding of document d_i	$x_i = f(\texttt{text}_i)$
d	Embedding dimension	e.g., $d = 384$ for MiniLM
$\hat{x}_i$	Normalized document embedding	$\hat{x}_i = x_i / \|x_i\|$
q	A query (text input)	Query text is embedded to a vector
$v_q \in \mathbb{R}^d$	Query embedding vector	$v_q = f(q)$
P	SQL predicate / filter condition	e.g., `category IN (...)`, `year >= 2020`
C	Filtered candidate set	$C = \{d_i \in D : P(d_i) = \text{true}\}$
$\|C\|$	Size of the filtered candidate set	Number of documents satisfying predicate P
s	Selectivity of the filter	$s = \|C\|/N$
K	Retrieval cutoff (top-K)	Commonly $K = 10$ or $K = 20$
$\mathrm{sim}(a, b)$	Similarity function	Often cosine similarity or inner product
$\mathrm{TopK}(C, v_q)$	Exact top-K over candidate set C	Used as ground truth within C
$\mathrm{GT@}K$	Ground truth top-K results	Exact top-K within C
$\mathrm{Pred@}K$	Retrieved top-K results by a strategy	Output list of K `rid`s
Recall@K	Fraction of $\mathrm{GT@}K$ retrieved in $\mathrm{Pred@}K$	$\frac{\|\mathrm{GT@}K \cap \mathrm{Pred@}K\|}{K}$
Latency (ms)	Time per query for a strategy	Reported as p50 / p95 / p99
p50 / p95 / p99	Latency percentiles	Median / tail latencies
ANN	Approximate Nearest Neighbor search	e.g., HNSW, IVF-PQ, DiskANN
HNSW	Graph-based ANN index	Key knob: `efSearch`
`efSearch`	HNSW search parameter	Higher $\rightarrow$ better recall, higher latency
`ann_probe`	Number of ANN candidates retrieved before filtering / rerank	Used in post-filter / two-stage retrieval
Strategy mix	Share of queries handled by each sub-strategy	For adaptive policies
Embedding bytes	Storage size of embeddings	Approx.: $N \times d \times 4$ for float32
Index bytes	Approximate index storage footprint	e.g., serialized FAISS index size
Build time	Time to compute embeddings and build indexes	Report separately for reproducibility

datasets can provide both semantic text and structured attributes, making them suitable for evaluating mixed retrieval behavior with filtering conditions.

To avoid comparability issues caused by different processing logics from different data sources, we map three types of data to the same table structure:

$$(\texttt{rid}, \texttt{dataset}, \texttt{text}, \texttt{category}, \texttt{year})$$

where `rid` is a globally unique record identifier, `dataset` denotes the dataset/domain label, `text` is the text field used for embedding generation, and `category` and `year` are optional structured attributes. The preprocessing process remains lightweight and consistent: first, text content is constructed according to the field rules of each dataset; then, empty text or samples that are too short are removed (records with `text` lengths not exceeding 10 are discarded); finally, a contiguous global `rid` is reassigned to all records. The current experiments primarily construct the filtering workload around the 'category' predicate, while the 'year' field is retained in the unified corpus, mainly to maintain pattern integrity and support future expansion.

4.2 Workload Design

We employ a selectivity-controlled filtering workload to characterize the quality–latency behavior under different filtering intensities. For each dataset, the experiments generate $Q = 160$ query instances; all randomization processes use a fixed random seed $SEED = 42$. The core filtering conditions of the evaluated workloads consist of category predicates. Specifically, the system first randomly samples an interval from eight target selectivity intervals. The selectivity ranges are defined as follows: 0.1%-0.5%, 0.5%-1.0%, 1.0%-2.0%, 2.0%-5.0%, 5%-10%, 10%-20%, 20%-40%, and 40%-80%.

Then, it samples the target proportions within this interval and selects one or more categories based on the empirical proportion of each category in the dataset, constructing a filtering predicate of the form `category IN (...)`. The resulting candidate set is denoted as C, and its selectivity is defined as $s = |C|/N$.

To avoid instability in evaluation caused by degenerate queries, if the size of the candidate set corresponding to a predicate is less than $\max(50, 3K)$ (60 in this experiment), the system falls back to a broader filter within the dataset. Subsequently, a document is randomly selected from the candidate set C as the query source, and its normalized embedding vector is directly used as the query representation. This design ensures that the query has at least semantically relevant neighbors within the filtered candidate set, thus making the filtered Top-K quality metric more interpretable.

4.3 Evaluation-Facing Workload Parameters

To maintain consistency with subsequent strategy comparisons, we perform a uniform scan of `efSearch` on each dataset, with values of $\{16, 32, 64, 128, 256\}$. Under each dataset-parameter combination, all strategies are compared on the same batch of generated query instances. The base candidate budget for the ANN post-filter is set to 200, and two-stage probing uses a progressively expanding probe ladder $[1, 2, 4]$. When the size of a filter candidate set exceeds 80,000, the system performs uniform sampling without replacement on this candidate set and computes the exact ground truth for all strategies within the same candidate universe to ensure tractability, fairness, and reproducibility. This setup allows us to systematically compare the behavior of exact prefilter, ANN post-filter, two-stage retrieval, and adaptive routing within a unified filtered Top-K context.

5 Methods: Strategies Compared

5.1 Compared Retrieval Strategies

We compare four basic retrieval paths and a rule-based adaptive hybrid strategy under a unified filtered Top-K setting. All methods share the same goal: to return the top-K results that are closest to the query vector among records satisfying the structured predicate P. The strategies differ mainly in whether filtering is applied before or after retrieval, whether approximate indexes are used for candidate generation, and whether execution paths are selected according to query conditions, as illustrated in Figure, 3.

5.1.1 Exact prefilter baseline (oracle).

Exact prefilter first executes the structured predicate P to obtain a filtered candidate set C, and then performs exact similarity calculation and top-K ranking only on C. Because this method directly returns the exact top-K results within the filtered candidate set, it is used as a correctness reference in this study. Its limitations are also quite clear: when $|C|$ is large, each query requires exact scoring on a large filter set, which easily leads to higher query costs and worse tail latency.

5.1.2 ANN post-filter.

ANN post-filter first performs approximate retrieval on the global HNSW index, and then applies structured filtering to the returned candidates. In its implementation, the system first retrieves a fixed number of candidates from the global index (currently set to `ann_probe`=200), then retains only records that satisfy the predicate P, and re-ranks the retained candidates based on exact similarity to output the top-K. This path usually has lower latency, but under strict filtering conditions, the initial ANN candidates may lack sufficient effective hits, thus the quality of the filtered Top-K is more likely to degrade.

5.1.3 Two-stage retrieval (probe ladder + rerank).

Two-stage retrieval introduces a progressively expanding candidate budget on top of ANN post-filter to alleviate the candidate shortage problem under strict filtering. In implementation, the system progressively increases the search scale on the global HNSW index according to a probe ladder ([1,2,4]), using candidate budgets of `ann_probe`, $2 \times$ `ann_probe`, and $4 \times$ `ann_probe` sequentially, constrained by a global limit of 6000. Each step first performs an ANN search, then applies the filtering conditions; once the number of valid candidates reaches the top-K requirement, the system stops expanding and performs exact reranking on the remaining candidates. Compared to a fixed-budget post-filter, this strategy is generally more robust under strict filtering,

but its additional ANN calls may increase tail latency for some queries.

5.1.4 Partition-aligned retrieval.

When the filtering predicate is aligned with an explicit category key, our study further considers partition-aligned retrieval. Specifically, the system constructs local partitioned indexes based on datasets and categories. The current implementation only builds partitioned indexes for categories with at least 800 documents per dataset, limiting this to the top 60 most frequent categories. For queries of the form '`category = X`' or single-element '`category IN (...)`', if a corresponding partitioned index exists, the system performs ANN retrieval only within that partition. This path reduces the proportion of invalid candidates by narrowing the search space, thus offering practical systems advantages when the filter key and partition key are highly consistent. However, its effectiveness depends on the consistency of the predicate structure and index organization; if the filter conditions do not correspond to the constructed partitions, this path is not applicable. Furthermore, the current implementation directly returns the ANN top-K results from the partitioned index without performing additional exact reranking.

5.1.5 Adaptive hybrid policy.

In addition to the fixed strategy, we also evaluate a rule-based adaptive hybrid policy. This strategy does not attempt to process all queries with a single path, but instead routes between existing strategies based on the candidate set size and predicate structure. The current implementation uses the following decision rules: when the size of the candidate set being filtered satisfies $|C| \leq 25{,}000$, exact prefilter is directly used; if the predicate can be parsed into a single-class condition and the corresponding partition index exists, partition-aligned retrieval is selected; otherwise, it falls back to two-stage retrieval. Therefore, the role of the adaptive policy is not to propose new underlying retrieval operators, but to evaluate an interpretable, low-complexity strategy routing mechanism within a unified evaluation framework and examine its overall robustness on the evaluated workloads.

5.2 Execution View

From the execution flow perspective, the system first generates a vector representation based on the query text and determines the filtering candidate set or its equivalent constraints based on the structured predicate P. Subsequently, different strategies generate candidate results on paths such as "filtering before sorting", "approximate retrieval before filtering", or "direct retrieval by partition". For exact prefilter, sorting is done directly within the filtering set; for ANN post-filter and two-stage retrieval, the system performs exact similarity re-ranking on valid candidates after filtering; for partition-aligned retrieval, the ANN top-K results are directly returned within the corresponding partition index. The adaptive policy then performs rule-based selection among the above paths. This design enables to directly compare different candidate generation mechanisms and their quality-delay behavior within a unified filtered Top-K context, as shown in Figure, 3.

6 Evaluation Protocol

6.1 Filtered Ground Truth and Fair Comparison

To ensure the correctness and comparability of comparisons between different strategies, we uniformly define ground truth under the filtered Top-K setting. For any query q and its structured predicate P, a candidate set is first obtained by SQL filtering:

$$C(q, P) = \{d_i \in D \mid P(d_i) = \text{true}\}$$

Subsequently, the exact top-K is calculated only within this candidate set based on normalized vector similarity, and this is defined as the shared ground truth:

$$\text{GT@}K(q, P) = \text{TopK}(C(q, P), v_q)$$

Therefore, the quality of any strategy is evaluated within the same filtering context, rather than compared with the top-K results on the globally unfiltered corpus. Accordingly, this study adopts

the following as the primary quality metric:

$$\text{Recall@}K = \frac{|\text{GT@}K \cap \text{Pred@}K|}{K}$$

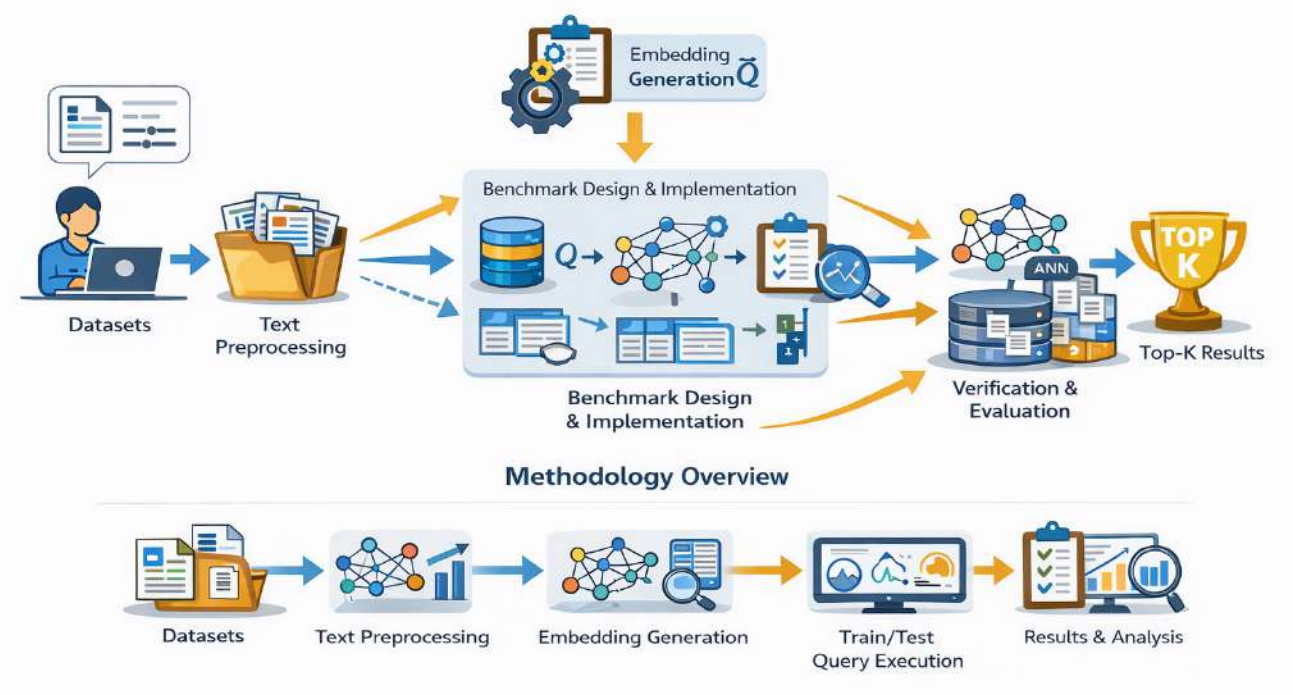

Figure 3. End-to-end pipeline including corpus construction, embedding generation, index building, workload generation, filtered retrieval, and evaluation.

When the filter candidate set for certain queries is too large and direct exact ranking is too costly, we impose a uniform upper bound on the candidate set. Specifically, when $|C| > 80{,}000$, uniform sampling without replacement is performed from C to obtain a shared candidate universe $\tilde{C} \subseteq C$, and the exact ground truth is computed on $\tilde{C}$. All comparison strategies are evaluated on the same $\tilde{C}$, thus avoiding evaluation bias caused by different methods facing different candidate spaces.

6.2 Controlled Experimental Setting

We perform a uniform comparison of all strategies on a fixed query set. The current experiment is set up with 160 queries per dataset, a fixed retrieval truncation of K=20, and repeated runs of different strategies and parameter configurations on identical query instances. For HNSW, we uniformly scan `efSearch` over {16, 32, 64, 128, 256} to observe the relationship between recall improvement and latency growth. For ANN post-filter, the base candidate budget is set to `ann_probe` = 200; for two-stage retrieval, a probe ladder is used, with a global cap of 6000 on the expanded ANN candidate set size to avoid unfair gains through unconstrained search. This setup ensures that latency-quality comparisons between different strategies are based on consistent and reproducible experimental conditions.

6.3 Implementation and Reproducibility

All experiments were performed in the Kaggle runtime environment. Structured filtering was performed using DuckDB, ANN retrieval was implemented based on FAISS's HNSW index, and text embeddings were generated by SentenceTransformer. To reduce redundant computational overhead and improve experimental reproducibility, the system cached the metadata table, embedding representation, and index files, and saved a per-query log and summary result table for each run. Therefore, the experimental results can not only be used to generate the final charts but also be subsequently checked and verified.

6.4 Reporting Protocol

We report results from three levels: quality, latency, and workload structure. The quality metric is primarily filtered Recall@20; latency metrics are uniformly reported at p50, p95, and p99 to characterize typical query performance and tail behavior. The results are presented from two main perspectives: first, an overall

dataset-wise comparison to analyze strategy differences across corpus domains; second, a fine-grained comparison by selectivity binning to characterize the impact of changes in filtering intensity on system behavior. Building upon this foundation, we further report on latency analysis, scalability testing, and cost profiling under a fixed recall target. Bootstrap confidence intervals are added to relevant charts, if necessary, to reflect statistical fluctuations. Overall, this evaluation protocol, while maintaining filtering accuracy and comparative fairness, can systematically reveal the quality–latency trade-off of hybrid retrieval under real-world filtering loads.

7 Results

7.1 Overall Quality–Latency Trade-off

Figure, 4 summarizes the relationship between p95 latency and Recall@20 for different strategy families on three datasets. Overall, hybrid retrieval under filtering conditions exhibits obvious workload-conditioned behavior, with no single optimal point among different policies. The exact prefilter consistently maintains a Recall@20 = 1.00 on all three datasets, thus it can be considered an oracle baseline for filtering Top-K; however, it also consistently exhibits the highest tail latency. In contrast, the ANN post-filter has the lowest p95 latency, but its quality degrades significantly under strict filtering conditions. Two-stage retrieval can alleviate the candidate shortage problem of fixed-budget post-filters in some scenarios, but its benefits are not stable. Overall, adaptive routing shows stronger overall robustness on the evaluated workloads.

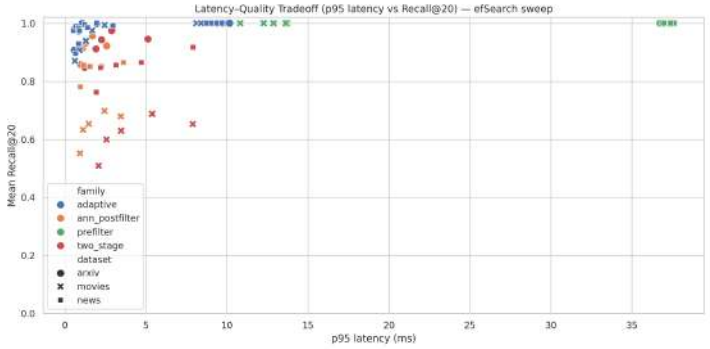

Figure 4. Relationship between p95 latency and Recall@20 for different .

Taking the results from the fixed recall target analysis as an example, the p95 latency of the exact prefilter on arXiv, news, and movies is 36.74 ms, 36.88 ms, and 10.82 ms, respectively, and the Recall@20 is always 1.00. In contrast, with a Recall@20 of approximately 0.95, the p95 latency of adaptive routing is only 0.63 ms, 0.51 ms, and 1.67 ms, respectively, showing significantly lower tail overhead. The speed advantage of ANN post-filter is also significant, but its quality performance is more affected by filtering conditions, especially in news and movies scenarios where it is difficult to consistently achieve high recall, as shown in Figure, 5.

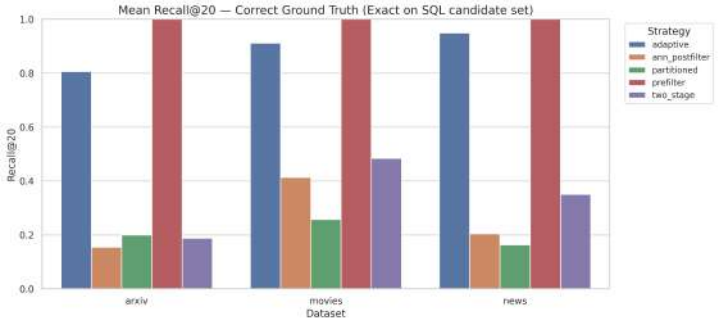

Figure 5. Average Recall@20 for different strategies.

7.2 Latency under Fixed Recall Targets

To more directly compare the engineering costs of different strategies under similar quality requirements, we further analyze the minimum p95 latency under a fixed Recall@20 target, as shown in Figure, 6. This analysis avoids the bias caused by simply comparing "faster" or "more accurate," and is closer to the decision-making logic of real systems under service level constraints.

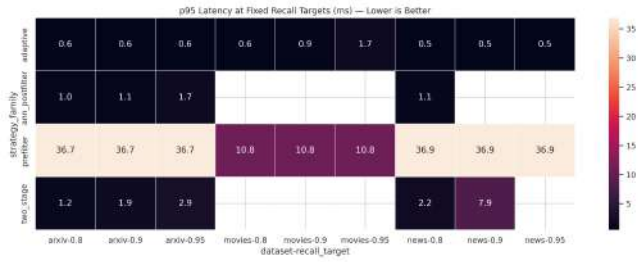

Figure 6. p95 latency for a fixed Recall@20 target.

The results show that adaptive routing achieves a high recall operating point on all three datasets with a significantly lower tail latency than the exact prefilter.

Taking a Recall@20 target of 0.95 as an example, the p95 latency of adaptive routing on arXiv, news, and movies is 0.63 ms, 0.51 ms, and 1.67 ms, respectively, while the exact prefilter latency is 36.74 ms, 36.88 ms, and 10.82 ms, respectively. In comparison, while ANN post-filters still achieve an average recall of over 0.95 on arXiv, they fail to consistently achieve higher, fixed recall targets on news and movies datasets. Similarly, two-stage retrieval on movies also fails to meet the fixed recall requirements of 0.80, 0.90, and 0.95. This indicates that in scenarios with stricter filtering or more complex candidate structures, a single-path global ANN search struggles to consistently provide high-quality results.

7.3 Effect of Filter Selectivity on Retrieval Quality

The filter selectivity directly impacts retrieval quality. Based on actual workload statistics of the Q2 query set, the average selectivity for the three datasets is approximately 0.172 for arXiv, 0.203 for news, and 0.211 for movies. This demonstrates that the current experiment is not an approximately unfiltered setup, but rather an evaluation of system behavior under substantial filtering pressure. Figure, 7 shows the variation of Recall@20 across different selectivity ranges.

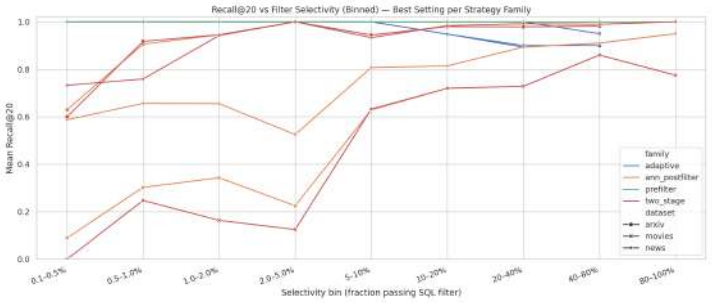

Figure 7. Recall@20 under different selectivity bins.

Among all the comparison methods, ANN post-filter is the most sensitive to the selectivity. This is because this method first searches for nearest neighbors in the global index and then performs structured filtering on the returned results; when the filtering conditions are strict, the initial ANN candidates often lack sufficient effective hits, resulting in a significant decrease in the quality of the filtered Top-K. Taking movies as an example, in the strictest 0.1%–0.5% interval, the Recall@20 of ANN post-filter is only about 0.13; while in the more lenient 40%–80% interval, this value can rise to about 0.97. In contrast, exact prefilter remains stable across all selectivity ranges, while adaptive routing exhibits a smoother overall quality curve by switching between different paths.

7.4 Tail Latency and Strategy Behavior

Figure, 8 and Figure, 9 illustrate the tail latency performance of different strategies under filtering workloads. The results show that reporting only the average latency significantly underestimates the true system cost, especially for exact prefilter and strategies that require incrementally expanding the candidate budget, where there is a significant difference between high-quantile latency and the average.

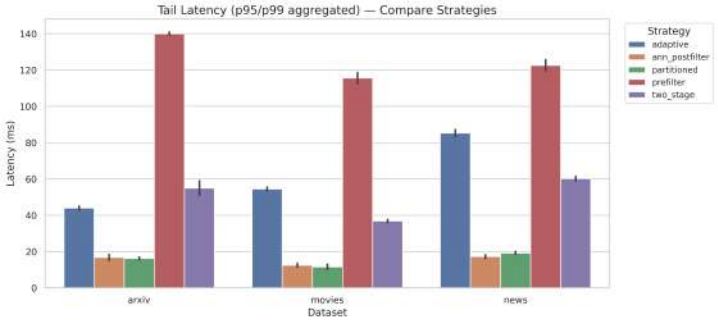

Figure 8. Comparison of tail latency for different strategies.

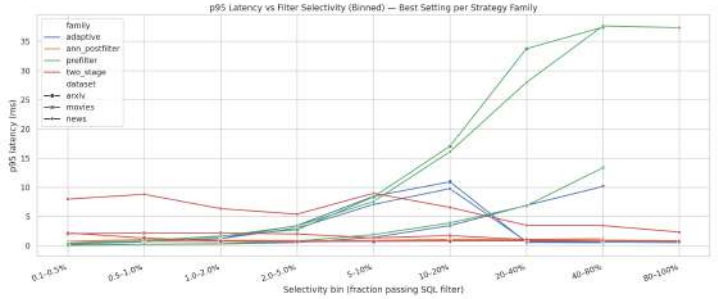

Figure 9. p95 latency under different selectivity binning.

The high tail latency of exact prefilter comes from the computational cost of performing exact scoring and ranking on the filtered candidate set, and this cost is further amplified when the candidate set is still large after SQL filtering. ANN post-filter is significantly faster, but this advantage comes at the cost of recall risk under strict filtering conditions. Two-stage retrieval

can alleviate the problem of insufficient effective candidates in post-filter in some scenarios by gradually increasing the candidate budget, but its benefits are dataset-dependent. For example, in movies, two-stage does not show a stable advantage under high recall targets. This demonstrates that the quality–latency trade-off in hybrid retrieval is highly dependent on the filtering structure, rather than simply determined by the underlying ANN path.

7.5 Behavior of Adaptive Routing

Adaptive routing is better understood as a rule-based routing mechanism rather than a single retrieval algorithm. Its advantage does not stem from the dominance of a particular sub-path in all scenarios, but rather from selecting a more suitable execution path based on candidate size and predicate structure. Figure, 10 shows the composition of adaptive routing on different datasets.

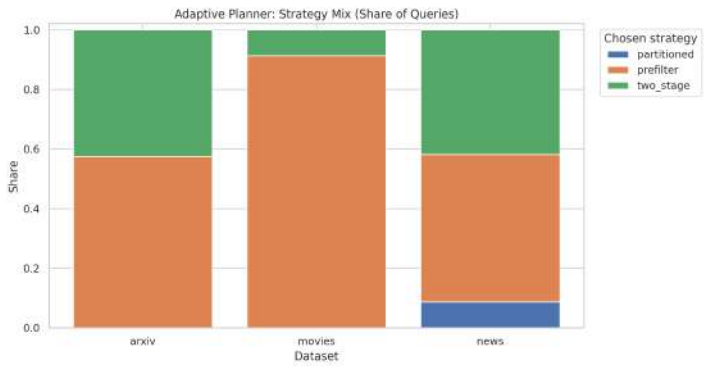

Figure 10. The strategy composition of Adaptive routing.

From the current results, the behavior of adaptive routing shows clear differences across datasets. On arXiv, approximately 62.5% of queries in the adaptive route use exact prefilter, and approximately 37.5% use two-stage. For movies, these percentages are approximately 90.75% and 9.25% respectively, indicating that most queries fall within a candidate size range suitable for exact filtering. The distribution for news is relatively more balanced: approximately 55.0% use prefilter, 37.38% use two-stage, and only about 7.63% use partitioned paths. This result demonstrates that the robustness of adaptive routing primarily stems from path allocation based on query structure, rather than relying on a fixed sub-strategy.

7.6 Scalability and Cost Analysis

Figure, 11 illustrates the scalability experiments on arXiv. As the corpus size increased from 50k to the complete subset corresponding to the current capped size, the system's p95 latency only increased from approximately 8.31 ms to 9.12 ms, while the average recall remained above 0.998. Meanwhile, index building time and storage overhead increased with the corpus size: building time increased from approximately 7.87 s to approximately 51.13 s, and the serialized index size also increased accordingly. It should be noted that the two target sizes of 200k and 400k under the current data cap actually fall within the same complete corpus range, therefore their index sizes are basically the same; the latter mainly reflects the repeated measurement results under the capped conditions. The cost profile further

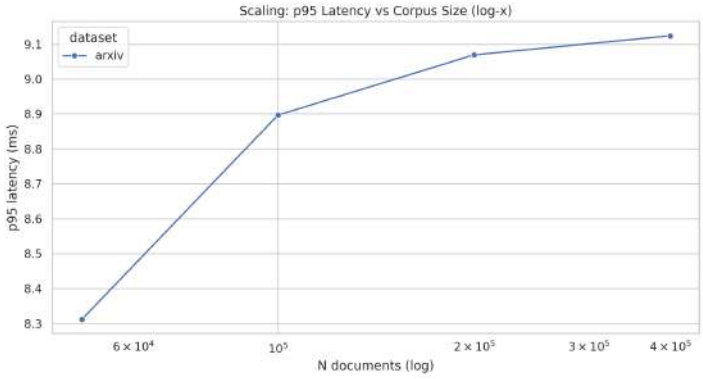

Figure 11. p95 latency under corpus size changes.

shows that embedding storage overhead increases linearly with the number of documents, while global ANN indexes and partitioned indexes introduce additional construction and storage costs. On the current unified corpus, the serialization size of the global HNSW index is approximately 940 MB, and the construction time is approximately 274 s; the total size of the partitioned index is approximately 555 MB, and the construction time is approximately 24.70 s. Therefore, from a system deployment perspective, the evaluation of hybrid retrieval methods should not only focus on the quality or latency of a single query, but also comprehensively consider the index construction cost, storage overhead, and the matching relationship between the filtering structure and the index organization (see Figure. 12).

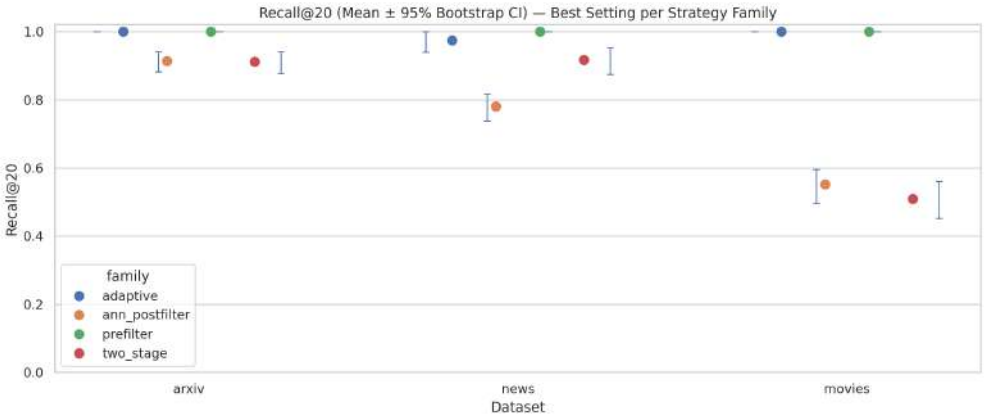

Figure 12. 95% confidence interval for Recall@20 under optimal settings.

8 Discussion

The results show that the performance of Vector+SQL hybrid retrieval is inherently workload-conditioned, with behavior shaped by filter selectivity, candidate set size, and predicate structure. Exact prefilter provides the most reliable correctness reference but incurs the highest tail latency cost; ANN post-filter offers a significant speed advantage but is more prone to quality degradation under strict filtering conditions; two-stage retrieval can alleviate the candidate shortage problem in some scenarios, but its benefits are dataset-dependent; adaptive routing, by selecting the execution path based on query conditions, exhibits a more robust overall balance under the . Therefore, hybrid retrieval with filtering conditions should not be handled using a single fixed path, but rather a workload-oriented strategy should be chosen between correctness, tail latency, and execution cost.

However, this study still has several limitations. Current experiments primarily focus on constructing workloads around category predicates, and have not yet systematically covered multi-attribute combination filtering and more complex SQL structures. The adaptive policy is currently rule-driven, rather than a data-driven learning planner. Furthermore, our evaluation mainly focuses on single query behavior and has not yet incorporated systemic factors such as high concurrency, cache contention, and mixed transaction loads. Accordingly, this work is better understood as a correctness-safe, tail-aware evaluation and analysis framework, rather than a final conclusion for all real-world database scenarios.

9 Conclusion

We study Vector+SQL hybrid retrieval under structured filtering conditions and develop a unified evaluation framework that balances correctness, fairness, tail latency, and reproducibility. The framework defines ground truth within the same filtered candidate set and systematically compares exact prefilter, ANN post-filter, two-stage retrieval, partition-aligned retrieval, and adaptive routing across the arXiv, news, and movies domains. Experimental results show that exact prefilter is the most stable correctness baseline but has the highest tail latency; ANN post-filter is faster but degrades under strict filtering; two-stage retrieval provides conditional improvements; and adaptive routing exhibits stronger overall robustness on the evaluated workloads. Overall, the study shows that filter selectivity and predicate structure significantly reshape the quality–latency trade-off in hybrid retrieval, and that reporting average latency alone is insufficient to characterize real system cost.

Author Contributions

Sadaf Bibi: Conceptualization, Methodology. **Faheem Akhter Rajput**: Supervision. **Muhammad Younis**: Visualization, Investigation. **Sadia Bibi**: Data curation, Writing- Original draft preparation. **Basit Raza**: Software, Writing- Reviewing and Editing.

Compliance with Ethical Standards

The authors declare that there are no conflicts of interest. We use publicly available secondary datasets, and

the research content pertains to retrieval systems and benchmark evaluations. It does not involve human subjects, animal experiments, or any personally identifiable data collected by the authors; therefore, informed consent is not required.

AI Assistance Disclosure

The authors declare that the AI tools are used only for language polishing, formatting, and technical expression optimization. No AI tools are used for research data generation, experimental result generation, result interpretation, or citation generation. All AI-assisted text content has been reviewed and verified item by item by the authors, who bear full responsibility for the final content of the paper.

Funding Information

No funding was received for this study.

References

[1] J. J. Pan, J. Wang, and G. Li, "Survey of Vector Database Management Systems," *The VLDB Journal*, vol. 33, no. 5, pp. 1591–1615, 2024.

[2] T. Taipalus, "Vector Database Management Systems: Fundamental Concepts, Use-Cases, and Current Challenges," *Cognitive Systems Research*, vol. 85, Art. no. 101216, 2024.

[3] J. J. Pan, J. Wang, and G. Li, "Vector Database Management Techniques and Systems," in *Companion of the 2024 International Conference on Management of Data (SIGMOD-Companion '24)*, Santiago, Chile, 2024, pp. 597–604.

[4] Y. A. Malkov and D. A. Yashunin, "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824–836, 2020.

[5] J. Johnson, M. Douze, and H. Jégou, "Billion-Scale Similarity Search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021.

[6] H. Jégou, M. Douze, and C. Schmid, "Product Quantization for Nearest Neighbor Search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2011.

[7] S. J. Subramanya, F. Devvrit, H. V. Simhadri, R. Krishnawamy, and R. Kadekodi, "DiskANN: Fast Accurate Billion-Point Nearest Neighbor Search on a Single Node," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[8] Q. Chen, B. Zhao, H. Wang, M. Li, C. Liu, Z. Li, M. Yang, and J. Wang, "SPANN: Highly-Efficient Billion-Scale Approximate Nearest Neighbor Search," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 5199–5212.

[9] C. Fu, C. Xiang, C. Wang, and D. Cai, "Fast Approximate Nearest Neighbor Search With the Navigating Spreading-out Graphs," *Proceedings of the VLDB Endowment*, vol. 12, no. 5, pp. 461–474, 2019.

[10] R. Guo, P. Sun, E. Lindgren, Q. Geng, D. Simcha, F. Chern, and S. Kumar, "Accelerating Large-Scale Inference with Anisotropic Vector Quantization," in *Proceedings of the 37th International Conference on Machine Learning*, *Proceedings of Machine Learning Research*, vol. 119, 2020, pp. 3887–3896.

[11] M. Aumüller, E. Bernhardsson, and A. Faithfull, "ANN-Benchmarks: A Benchmarking Tool for Approximate Nearest Neighbor Algorithms," *Information Systems*, vol. 87, Art. no. 101374, 2020.

[12] J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu, K. Yu, Y. Yuan, Y. Zou, J. Long, Y. Cai, Z. Li, Z. Zhang, Y. Mo, J. Gu, R. Jiang, Y. Wei, and C. Xie, "Milvus: A Purpose-Built Vector Data Management System," in *Proceedings of the 2021 International Conference on Management of Data*, New York, NY, USA: Association for Computing Machinery, 2021, pp. 2614–2627.

[13] R. Guo, X. Luan, L. Xiang, X. Yan, X. Yi, J. Luo, Q. Cheng, W. Xu, J. Luo, F. Liu, Z. Cao, Y. Qiao, T. Wang, B. Tang, and C. Xie, "Manu: A Cloud Native Vector Database Management System," *Proceedings of the VLDB Endowment*, vol. 15, no. 12, pp. 3548–3561, 2022.

[14] C. Chen, C. Jin, Y. Zhang, S. Podolsky, C. Wu, S.-P. Wang, E. Hanson, Z. Sun, R. Walzer, and J. Wang, "SingleStore-V: An Integrated Vector Database System in SingleStore," *Proceedings of the VLDB Endowment*, vol. 17, no. 12, pp. 3772–3785, 2024.

[15] S. Gong, H. Sun, Z. Fang, L. Liu, L. Chen, and Y. Gao, "VStream: A Distributed Streaming Vector Search System," *Proceedings of the VLDB Endowment*, vol. 18, no. 6, pp. 1593–1606, 2025.

[16] J. Sun, G. Li, J. Pan, J. Wang, Y. Xie, R. Liu, and W. Nie, "GaussDB-Vector: A Large-Scale Persistent Real-Time Vector Database for LLM Applications," *Proceedings of the VLDB Endowment*, vol. 18, no. 12, pp. 4951–4963, 2025.

[17] J. Dean and L. A. Barroso, "The Tail at Scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[18] D. Lemire, O. Kaser, N. Kurz, L. Deri, C. O'Hara, F. Saint-Jacques, and G. Ssi-Yan-Kai, "Roaring Bitmaps: Implementation of an Optimized Software Library," *Software: Practice and Experience*, vol. 48, no. 4, pp. 867–895, 2018.

[19] B. Raza, S. Bibi, S. Bibi, and A. Nawaz, "SADA COLOR DATASET (SCD): 9 Paper Colors $\times$ 4 Illumination Conditions for Robust Color Vision Evaluation," *Spectrum of Engineering Sciences*, vol. 4, no. 2, pp. 871–887, 2026, doi: 10.5281/zenodo.18844499.