arXiv:2602.11443v1 [cs.DB] 11 Feb 2026

向量数据库中的过滤近似最近邻搜索：系统设计与性能分析

Abylay Amanbayev、Brian Tsan、Tri Dang、Florin Rusu{amanbayev、btsan、tridang、frusu}@ucmerced.edu 加利福尼亚大学默塞德分校

2026年2月

摘要

检索增强生成（RAG）应用日益依赖过滤近似最近邻搜索（FANNS），以结合语义检索与元数据约束。尽管已有针对 FANNS 的算法创新，但人们仍不了解通用过滤策略在向量数据库中的表现。在本研究中，我们系统梳理了过滤策略的分类体系，并评估了其在 FAISS、Milvus 和 pgvector 中的集成。为提供稳健的基准测试框架，我们引入了新的关系型数据集 *MoReVec*，该数据集由两个表组成，包含 768 维文本嵌入以及丰富的元数据属性模式。我们进一步提出 全局-局部选择性（*GLS*）相关性度量，用于量化过滤器与查询向量之间的关系。

我们的实验表明，引擎中的算法自适应往往会盖过索引的原始性能。具体而言，我们发现：(1) *Milvus* 通过近似/精确混合执行实现了更稳定的召回率；(2)*pgvector*的基于成本的查询优化器经常选择次优的执行计划，即使精确顺序扫描能够以相近的延迟实现完美召回率，它仍倾向于采用近似索引扫描；(3) 对于低选择性查询，基于分区的索引（IVFFlat）优于基于图的索引（HNSW）。为便于开展这项分析，我们扩展了广泛使用的*ANN-Benchmarks*，使其支持过滤向量搜索，并在线开放使用。最后，我们将研究发现归纳为一系列实用指南，帮助用户为混合搜索工作负载选择索引类型并配置查询优化器。

1 引言

向量数据库已成为现代 AI 技术栈的重要组成部分，可充当大语言模型（LLM）的长期记忆 [25]。具体而言，它们通过向检索增强生成（RAG）应用提供来自外部知识库的相关上下文，减少幻觉 [22, 29]。这一过程通过**近似最近邻搜索（ANNS）**算法，从以向量嵌入形式存储的非结构化数据（如文本、图像和音频）中进行检索。尽管标准近似最近邻搜索是一个经过深入研究的问题，拥有高效算法[35, 15, 36, 20] 和成熟的基准测试 [2, 6, 55], ，但现实世界的部署表明，仅凭语义相似性通常不足以检索上下文。

1.1 动机

随着向量搜索系统从原型走向生产环境，人们期望它们能够处理混合查询，即通过针对结构化元数据 [60, 30] 的过滤器排除不相关的搜索结果。这一需求在各个领域普遍存在：医疗 RAG 系统必须找到症状相似（向量）且年龄处于特定范围内（过滤器）的患者；法律搜索引擎必须在特定司法管辖区（过滤器）内找到相关案例（向量）；电子商务平台则必须推荐视觉上相似（向量）且评分较高（过滤器）的产品 [58, 10]。

尽管这类查询具有普适性，但结构化过滤与向量索引遍历之间的交互仍是当前研究中的一个关键盲点。尽管近期的算法研究提出了专用融合

方法（例如 ACORN [45], Filtered-DiskANN [15], HQANN [61]），但这些方法通常会牺牲灵活性，要求针对特定属性或工作负载对索引进行调优。相比之下，通用型向量数据库旨在不依赖模式和数据，因此需要采用诸如预过滤 和后过滤之类的通用策略。

这造成了算法理论与实际应用之间的鸿沟。首先，现有文献缺乏用于描述这些策略的统一分类体系，导致不同系统中的定义模糊、实现不一致。其次，尽管通用策略具有灵活性，但其性能局限及其与系统组件之间的相互作用仍缺乏充分研究。

此外，业界缺乏能够对这些场景进行恰当基准测试的工具。现有基准测试无法真实地对这些系统施加压力；它们大多是“扁平”的，依赖基于图像的数据集（如 SIFT 或 GIST），或纯文本语料库（如 BEIR [54]），并使用合成或有限的元数据。关键在于，它们缺少可靠的相关性度量，无法区分性能下降究竟是由系统架构导致，还是源于过滤器与向量分布本身的“难度”。若不对这些因素进行严格评估，就难以明确向量数据库的架构选择如何影响过滤工作负载下召回率与查询延迟之间的权衡。

1.2 贡献

为应对这些挑战，我们对向量数据库系统中的通用过滤策略进行了全面评估。我们的具体贡献如下：

• **分类体系与系统分析**。我们对过滤策略（预过滤、后过滤和运行时过滤）的分类体系进行系统化梳理，并分析它们在三个广泛使用的系统中的实现：*FAISS*、*Milvus*和*pgvector*。我们着重说明，在生产环境中，诸如自适应查询执行和查询规划等系统级架构决策往往会压过原始算法特性。

• **全局-局部选择性（GLS）度量**。我们提出全局-局部选择性（*GLS*）相关性度量，以严格量化元数据过滤器与向量邻域之间的独立性。与以往基于距离的度量不同，GLS 将局部过滤器出现率相对于全局选择性进行归一化，从而为索引决策提供稳健信号。

• **FANNS 数据集与工作负载**。我们在一个新的关系型数据集*MoReVec*上开展实验，该数据集包含两张表：电影和评论。该数据集具有 768 维稠密文本嵌入，以及由标量和类别型元数据构成的丰富模式，从而能够评估过滤式近似最近邻查询。此外，该关系模式旨在支持未来对带有连接谓词的 ANNS 查询进行基准测试。

• **开源基准测试框架**。 我们发布了业界标准 *ANN-Benchmarks*的扩展版本，该版本能够执行具有动态选择性目标的过滤查询，为未来研究混合搜索性能提供便利。扩展后的框架、MoReVec 数据集和 GLS 相关性分析工具均已公开发布 [3]。

• **系统行为的实证洞察**。 通过广泛的实验，我们揭示了现代向量数据库中有悖直觉的性能特征：

  – 算法自适应：*Milvus* 通过采用混合执行模型，在“Dual-Pool”图遍历与自适应暴力搜索回退之间动态切换，因此在低选择率场景下，其召回率稳定性优于标准实现。– 优化器局限性： 我们发现，*pgvector*的基于成本的优化器经常会错误估算执行成本，即使精确顺序扫描能够以相近的延迟实现完美召回率，它仍倾向于选择近似索引扫描。– 索引能力： 与标准近似最近邻搜索的普遍认知相反，我们证明，在低选择率场景下，基于分区的索引（IVFFlat）往往优于基于图的索引（HNSW），这是因为聚类剪枝效率更高。

– **实用指南**。 我们将研究结果归纳为一系列可执行的指南，涵盖索引选择、参数调优和查询计划验证，为从业者提供在生产环境中优化混合搜索工作负载的路线图。

组织。 本文其余部分的组织结构如下。我们首先在第2节回顾该领域的近期研究，随后在第3节介绍近似最近邻搜索（ANNS）所需的背景知识。第4节介绍我们提出的全局-局部选择性（GLS）相关性指标。第5节系统梳理过滤策略的分类体系，并详细说明所评估系统的具体架构。第6节概述我们的基准测试框架，介绍 MoReVec 数据集和工作负载。第7节呈现实验结果，分析选择性和系统架构对性能的影响。最后，第8节总结我们的研究发现并展望潜在的未来研究方向。

2 相关工作

过滤近似最近邻搜索（FANNS）的系统化研究借鉴了高维索引、混合搜索算法以及现代向量数据库不断演进的架构。

表 1：本研究与近期其他过滤近似最近邻搜索基准测试的比较。

论文	描述
研究目标与框架	
本研究	**系统层面分析**：聚焦生产级向量数据库（FAISS、Milvus、pgvector）。对通用过滤（预处理/后处理/运行时）进行系统化梳理，并研究架构选择（算法适应性、查询规划）如何凌驾于原始算法性能之上。
*Shi*等人 [51]	**统一基准测试**：聚焦鲁棒性和公平性。建立标准化调优用于缓解实现偏差的策略类别分类（先过滤、先搜索、混合）。
Li 等人 [31]	**组件分类体系**：侧重于内部索引调控因素。通过剪枝启发式方法、入口点选择和边过滤成本等具体组件分析性能变化通过剪枝启发式方法、入口点选择和边过滤器等特定组件边过滤成本。
Iff 等人 [19]	**算法综述**：聚焦于专用／融合方法。针对基于 Transformer 的嵌入，对最先进算法进行基准测试（ACORN、Filtered-DiskANN）。
数据集模型	
本研究	**关系型（IMDb）**：包含两个表（电影、评论），其中有 768 维文本嵌入以及若干真实世界属性。 表之间具有一对多外键关系，可用于近似最近邻搜索并采用连接过滤。
*Shi*等人 [51]	**真实世界精选**：精选多样化的数据集，并整合到统一的测试框架中进行测试在不同数据分布下的整体鲁棒性。
Li 等人 [31]	**合成 +真实**：4 个数据集（最多包含 1000 万个向量），结合真实世界属性与合成属性，以实现对选择率扫描（0.1%–100%）的精确控制。
Iff 等人 [19]	**Flat（arXiv）**：单个表包含 270 万个向量和 11 个真实世界属性。大型文本数据集旨在测试融合算法。
关键见解	
本研究	**架构起主导作用**：Milvus 的算法适应性和暴力回退可稳住召回率；pgvector 的优化器常选择次优的近似扫描。引入 **GLS 相关性**，用于量化过滤器独立性。

续见下页

表 1 – 接上页

论文	描述
*Shi*等人 [51]	**调优敏感性**: 标准化调优会显著改变策略排名; “先过滤”与“先搜索”的权衡对实现细节高度敏感。
Li 等人 [31]	**剪枝驱动因素**: 性能取决于剪枝粒度和边遍历开销; 为每种属性类型提供实用的调优指南。
Iff 等人 [19]	**规模扩展失败**: 融合方法（如 ACORN）前景可期，但在索引方面可能遇到困难大规模场景下的规模与构建时间; 在所有选择率下没有普遍最优方案。

2.1 向量数据库架构

将近似最近邻搜索集成到数据库系统中，主要有两种路径：**专用向量数据库** 和**关系型扩展**。**Milvus** [57]和 **Pinecone** [48]等系统采用分布式架构，以优化水平扩展和数据分段。相比之下，**pgvector** [46] 和 **AnalyticDB-V** [60] 等扩展将向量类型嵌入关系核心，并利用现有的基于成本的优化器（CBO）。此外，PostgreSQL 生态系统中的其他方案则尝试将向量搜索引擎解耦，以绕过基于页的设计固有的开销 [23]。一个重要的研究空白在于：这些架构选择（而非仅仅是底层算法）如何影响性能，目前尚未得到充分理解。近期比较关系数据库管理系统与专用向量数据库的研究[66] 表明，性能差距正在缩小; 然而，对于高维向量场景下查询优化器的具体失效模式，文献尚未充分记录。

2.2 过滤式近似最近邻算法

近期的算法工作试图将元数据约束直接整合到索引结构中，以避免与后过滤相关的“召回率悬崖”。**Filtered-DiskANN** [15] 提出了一种过滤感知图构建策略，而**HQANN** [61] 则采用了向量与元数据的联合量化。最近，**ACORN** [45] 提出使用分层可导航小世界图，以便即使在低选择性谓词下也能保持连通性。尽管这些“融合”方法效率很高，但它们通常要求索引针对特定属性进行专门化，从而限制了其在必须支持任意模式上临时查询的通用数据库中的实用性。

2.3 ANNS基准测试

近似最近邻搜索的基础建立在四大算法支柱之上：**基于图** （例如，HNSW [35], Vamana [21]）、**基于分区**（例如，IVFFlat、LSH [20]）、**基于量化** （例如，PQ [26]）以及**基于树** 方法（例如，Annoy[5]）。这些算法的性能通常通过*ANN-Benchmarks* 框架 [6], 进行评估，该框架将召回率与吞吐量权衡形式化。然而，传统基准测试几乎只关注“扁平”向量搜索，使用的是缺乏结构化元数据的SIFT和GIST等数据集。因此，选择谓词的影响在标准文献中仍未得到充分探索。

2.4 与近期 FANNS 基准测试的区别

近期研究已开始探讨过滤近似最近邻搜索的复杂性。值得注意的是，*Shi*等人 [51], *Li* 等人 [31] 和 *Iff* 等人 [19]为混合向量搜索提供了有价值的基准测试。然而，我们的工作在三个关键方面不同于这些研究，详见表 1。

系统架构与算法融合。近期的基准测试主要评估专用算法，包括将元数据与索引结构紧密耦合的基于融合的方法（例如 ACORN、UNG）; 而我们关注通用过滤策略（预过滤、后过滤和运行时过滤）的系统级集成。我们认为，融合方法虽然在理论上效率很高，但缺乏通用型向量数据库所需的模式无关性。我们的结果表明，系统架构方面的选择，例如*Milvus*的算法适应性或*pgvector*的查询优化器成本模型，往往比原始算法性能更具决定性。

关系型与扁平数据模型。不同于以往依赖简单基于标签的元数据的"扁平"数据集基准测试 [19, 31], ，我们引入了一个关系型数据集，其中包含通过外键连接的电影表和评论表，从而能够评估真实数据库工作负载中固有的复杂查询计划（例如连接）。

查询-过滤器关系研究。我们提出全局-局部选择性（*GLS*）相关性，这是一种归一化指标，能够明确区分过滤器是丰富还是耗减语义邻域，从而为索引选择提供更清晰的信号。据我们所知，现有基准测试尚未明确量化筛选谓词与查询向量邻域之间的这种相关性。

3 预备知识

本节将概述向量相似性搜索的基本概念、本文评估的索引结构，以及用于量化筛选条件与查询的相关性的现有指标。

3.1 向量表示与嵌入

现代信息检索高度依赖于稠密向量嵌入 [27]——对非结构化数据（文本、图像、音频）的固定维度表示，其中语义相似性通过几何邻近性得以保留 [37]。不同于传统的稀疏方法（例如 BM25 [50], TF-IDF），稠密嵌入能够捕捉语义细微差异，从而检索出概念相关但可能不含完全相同关键词的项目。

当前项文本嵌入的最新技术采用基于 Transformer 的架构（例如 GPT [42, 8], BERT[11]）和对比学习 [33, 65]。这些模型将可变长度的文本映射为固定大小的向量 $v \in \mathbb{R}^d$（通常为 $d = 768$或 $d = 1024$）。

MTEB 排行榜。大规模文本嵌入基准（*MTEB*） [39, 40] 已成为评估不同任务中嵌入质量的标准，包括检索 [54], 聚类和分类。它根据模型的综合性能进行排序，为生产系统中的嵌入模型选择提供指导。

3.2 相似性度量

向量相似性度量用于量化查询向量 q 与数据库向量 v之间的语义关系。这些度量通常分为两类：基于距离的度量，用于衡量向量空间中的几何距离；以及基于角度的度量，用于衡量方向对齐程度。尽管近期研究提出了将向量相似度与元数据差异性进行聚合的融合距离 [61], 通用向量数据库主要依赖三种标准度量：

L_p-范数。该度量推广了标准的欧几里得（L_2）和曼哈顿（L_1）距离。其定义为 $d(v,\ q) = ||v - q||_p = (\sum_{i=1}^{n} |v_i - q_i|^p)^{1/p}$。欧几里得距离（$L_2$）是最常用的度量。然而，它容易受到"维度灾难"的影响。随着维度增加，空间体积呈指数级增长，导致最近点与最远点之间的距离比趋近于 1。这种对比度的损失会使基于距离的区分在极高维稀疏空间中变得困难，不过，现代稠密嵌入（例如 768 维）通常仍能保留足够的结构局部性。

内积。定义为 $d(v,\ q) = \langle v,\ q\rangle = \sum_{i=1}^{n} v_i q_i$，该度量将 v投影到q的模长缩放后的 q 上。它的计算效率很高，但对向量模长敏感；例如，两个相同的大型向量会得到比两个相同的小型向量更高的相似度分数，而这未必适用于所有语义任务。

余弦相似度。此度量通过测量两个向量之间夹角的余弦值来实现，实际上相当于对内积进行归一化： $d(v、\ q) = \frac{\langle v、\ q\rangle}{||v||||q||}$。这在数学上等同于对经过 L_2 归一化的向量[1]求内积。文本检索广泛采用此度量，因为它能够捕捉语义方向，而不受文档长度（向量模长）的影响；因此，它已成为在 MTEB 排行榜上评估的大多数嵌入模型的标准度量 [40]。

表 2：记号和关键参数。

符号	描述
数据集与查询	
$\mathcal{D}, N$	向量数据集及其基数（$\|\mathcal{D}\| = N$）
$\mathcal{Q}$	查询向量集
d	向量维度（在我们的实验中为 768）
k	要检索的最近邻数量
过滤与选择性	
$\phi(v)$	过滤谓词函数：若有效则返回 1，否则返回 0
σ_g	**全局选择性**：满足 ϕ的 $\mathcal{D}$ 所占比例
σ_l	**局部选择性**：满足 ϕ的 k-NN 所占比例
r	**选择性比率**：σ_l/σ_g（如 [62]中所提出）
ρ_q	**GLS 相关性**：逐查询的过滤器向量独立性
$\bar{\rho}$	跨 $\mathcal{Q}$的平均 GLS 相关性
索引参数	
M	HNSW：每个节点的最大边数
ef_{const}	HNSW：构建期间的候选队列大小
ef_{search}	HNSW：搜索期间的候选队列大小
n_{probe}	IVFFlat：要搜索的聚类数量（质心）

3.3 索引与搜索质量

k-近邻问题需要遍历整个向量集合，同时记录迄今为止找到的 k 个最近邻。此类穷举搜索需要计算所有 N 向量与查询向量之间的距离，对于包含 N 个向量的数据集，其时间复杂度为 $O(Nd)$。尽管这种方法的时间复杂度是多项式级别，但它需要大量计算资源，尤其是在工作负载较高且向量集合较大的系统中。为此，人们开发了近似最近邻搜索算法，以牺牲少量准确率来显著提升速度 [32]。

这类索引策略种类繁多，并由成熟的评估框架 [2] 提供支持。它们通常分为四类：基于图，例如 HNSW、Vamana；基于分区，例如倒排文件、局部敏感哈希；基于量化，例如乘积量化；以及基于树，例如 Annoy、k-d树算法 [21, 26,20, 7, 41]。某些算法可能会将多种方法结合起来。下面我们将讨论一些常用的近似最近邻搜索算法的详细信息：倒排文件索引，这是一种基于分区的算法；以及 HNSW，这是一种基于图的算法。

倒排文件索引（IVF）是一种用于信息检索系统的数据结构，能够高效定位包含特定术语的文档。它建立了术语 t_i 与包含这些术语的文档集合 D_i 之间的关联。对于近似最近邻搜索，倒排文件索引通常与沃罗诺伊图或 k均值聚类结合使用，根据与一组点或对象的接近程度，将空间划分为不同区域 [26, 53]。通过将每个向量与沃罗诺伊图中的点或簇质心关联起来，查询可以高效地定位相关文档，从而优化检索过程。这利用了文档之间的空间关系，同时也缩小了近似最近邻搜索问题的搜索空间。当关联基于纯距离（L2距离或余弦）时，该算法称为*IVFFlat*。

分层可导航小世界 (HNSW) 是一种基于图的索引算法，它扩展了可导航小世界 (NSW) 模型，使高维向量搜索具备对数级可扩展性 [35]。标准 NSW 算法通过迭代将每个向量与其最近邻连接起来，并添加一些连接向量空间不同区域的长距离链接，从而构建图。

HNSW 的核心思想是按连接长度进行区分：将长程连接（捷径）置于上层，以便在向量空间中快速遍历；而短程连接则保留在下层，用于进行细粒度的局部搜索。为此，向量会依次插入图结构中。对于每个插入的元素，都会以指数递减的概率随机选择一个整数最大层 l 。这种概率分配

确保所有向量都存在于最下层，而只有一小部分“枢纽”顶点分布在上层，充当通往图的高速通道。

搜索过程从最高层开始，并自上而下进行。算法在当前层中贪心地遍历图元素，直到到达一个最近邻，此时再下降到下方的下一层。这种“逐步聚焦”策略使算法能够忽略绝大多数距离计算。到达基础层后，会维护一个候选优先队列，以跟踪已找到的最近邻。该队列的大小由参数 ef_{search} 控制，用作搜索延迟与召回质量之间可调节的权衡。

Recall@k 是评估近似搜索准确率的主要度量。它衡量索引成功检索到的真实最近邻所占的比例，并定义为：

$$\text{Recall@}k = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{k} \tag{1}$$

.

在此，真正例 (TP) 表示相关项的数量，而 假负例 (FN) 表示被近似方法遗漏的真实邻居。由于 TP +FN $= k$，这可简化为已找到的前k 个真实值所占的比例。

还有另一个重要特性，即查询处理速度，以每秒处理的查询数量或 **每秒查询数** （每秒查询数）来衡量。索引方法通常以图的形式进行评估，其中每秒查询数是召回率 [6]的函数。

3.4 过滤器类型

向量搜索中的过滤谓词通常根据约束的性质分为两大类：基于元数据的过滤器和基于距离的过滤器。

基于元数据的过滤器用于对与向量嵌入相关联的结构化属性（例如标签、时间戳或唯一标识符）施加约束。这相当于关系型 SQL 中的 WHERE 子句，如今已成为现代向量数据库的标准功能 [38, 48, 46, 49]。它们可细分为：

- 类别过滤器：对离散标签进行精确匹配（例如，类别为 = “电子产品”），或检查集合成员关系（例如，标签 CONTAINS “紧急”）。有专门针对这类过滤器的基准测试，并有许多算法致力于解决这一问题 [52, 36, 24]。
- 标量过滤器：对连续值执行数值范围查询（例如，价格 $>$ 50.0 或年份 BETWEEN 2020 AND 2024）。
- 字符串过滤器：使用标准子串搜索（例如 LIKE）或正则表达式执行模式匹配操作。

基于距离的过滤器通常被视为一个独立问题，即范围搜索，它利用向量之间的距离来剪枝结果。这些过滤器并非检索固定的 k个最近邻，而是返回落在查询指定半径内的所有向量（例如，余弦相似度 $>$ 0.8） [9]。

第三种新兴范式是语义过滤，它在这些传统类别之外运作。在此，约束由自然语言谓词而非结构化模式字段定义。这些系统通常采用二次大语言模型步骤来评估需要语义推理的条件（例如，“查找与向量数据库相关的论文”，其中主题是推断得出的，而非显式标注） [44]。

在本研究中，我们仅关注**基于元数据的过滤器**——具体而言是标量不等式——因为它们代表了生产级向量数据库中混合搜索的标准集成点（见第6.4节）。

在本研究中，我们采用标准的数据库定义，其中选择性 σ 表示满足筛选谓词的有效行的比例 。因此，我们使用术语低选择率 来表示严格过滤器（其中 $\sigma \to 0$，例如1%），并使用术语高选择率 来表示宽松过滤器（其中 $\sigma \to 1$，例如50%）。

3.5 查询-过滤器相关性：基于距离的方法

在 FANNS 中，将过滤器与向量相似度结合起来，会带来如何平衡召回率和延迟的挑战。当过滤属性与向量嵌入的空间分布存在相关性（或负相关）时，这些挑战会进一步加剧。

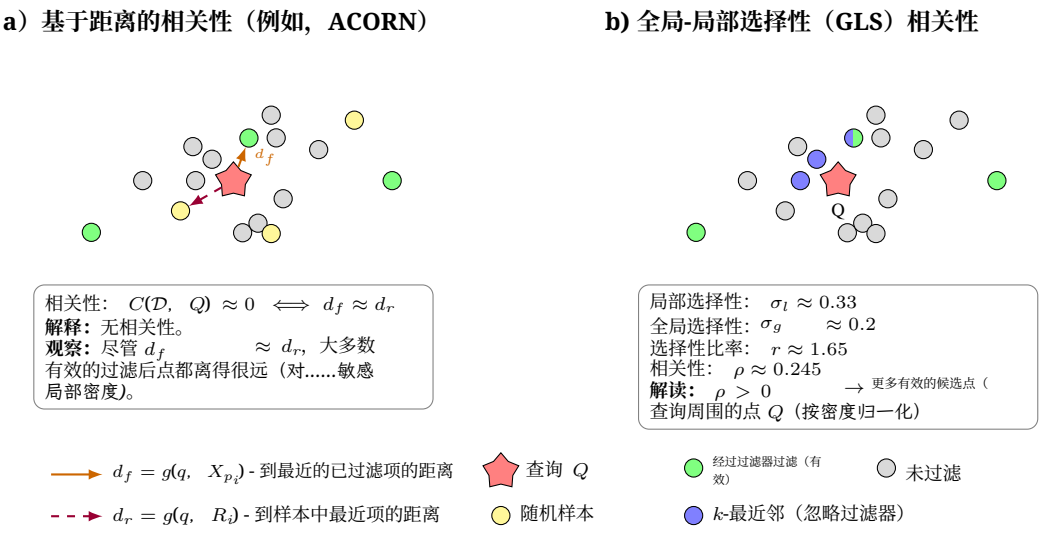


图1：相关性度量的比较。(a) 基于距离的相关性 [45]对几何密度较为敏感，可能会将密集簇误判为不相关。(b) 所提出的 GLS 相关性通过将局部选择性 σ_l 与全局基线 σ_g进行比较，对密度进行归一化。

ACORN 等最先进的方法 [45]，使用基于距离的方法量化这种关系。具体而言，它们会将过滤后结果的距离分布与随机基线进行比较，并将数据集 $\mathcal{D}$ 上某个工作负载 $\mathcal{Q}$ 的查询-过滤器相关性 $C(\mathcal{D},\ \mathcal{Q})$ 定义为查询向量与 $\mathcal{D}$ 中大小相同的过滤随机子集之间最小距离的期望差值：

$$C(\mathcal{D}, \mathcal{Q}) = \mathbb{E}_{(x_i,p_i)\in\mathcal{Q}} \left[\mathbb{E}_{R_i}[g(x_i, R_i)] - g(x_i, X_{p_i})\right], \tag{2}$$

其中：

- X_{p_i} 是满足过滤谓词 p_i 的向量子集。
- R_i 是从 $\mathcal{D}$中均匀抽取的随机子集，且 $|R_i| = |X_{p_i}|$。
- $g(x,\ S) = \min_{y\in S} \text{dist}(x,\ y)$ 是查询 x 到集合 S 中最近向量的距离。

直观而言，正的 $C(\mathcal{D},\ \mathcal{Q})$ 表示已过滤向量在空间上聚集于查询附近（比随机情况下更近），这意味着局部密度较高。相反，负值则表明过滤后的子集比随机样本更稀疏或距离更远。虽然该度量能够有效反映几何上的“难度”，但它容易受到局部邻域密度的影响，即每个查询 q 到其邻居的平均距离各不相同。通过对最小距离取平均，该度量混淆了全局聚类与逐查询局部性，无法区分富集（局部选择性高）与仅仅是位于密集区域附近。此外，为查询优化估计此类相关性度量仍是一项尚待解决的挑战。

4 全局-局部选择性相关性

我们引入了**全局-局部选择性（GLS）相关性**这一度量，用于量化过滤器的全局选择性与其在查询的局部邻域内的选择性之间的关系。它明确揭示了过滤器在每个查询层面上，是否会增加或减少查询向量周围相关邻域中的匹配项。

考虑一个由 N 个向量 $v \in \mathbb{R}^d$组成的数据集 $\mathcal{D}$。应用过滤器 $\phi: \mathcal{D} \to 0,1$，以选出满足谓词的向量，并假设过滤后的子集非空（即 $|v \in \mathcal{D} \mid \phi(v) = 1| > 0$；空过滤器

在分析中会忽略）。全局过滤器选择率，记为 σ_g，用于量化该过滤器在整个数据集中的基准出现率：

$$\sigma_g = \frac{|\{v \in \mathcal{D} \mid \phi(v) = 1\}|}{N} \in (0, 1] \tag{3}$$

这可作为语料库级先验，与任何特定查询无关。

对于一组查询向量 $\mathcal{Q}$，其中每个 $q \in \mathcal{Q} \subseteq \mathbb{R}^d$，我们将局部邻域 $\mathcal{N}_q$ 定义为 q 在 $\mathcal{D}$ 中的 k 个最近邻，其中 $|\mathcal{N}_q| = k$。局部过滤器选择率 σ_l 随后用于衡量该邻域中通过过滤器的比例：

$$\sigma_l = \frac{|\{v \in \mathcal{N}_q \mid \phi(v) = 1\}|}{k} \in [0, 1] \tag{4}$$

这里，$\sigma_l > \sigma_g$ 表示局部富集（过滤属性与相似向量聚类），而 $\sigma_l < \sigma_g$ 则表示耗减（负相关）。

为根据这些选择率推导出方向性相关性，我们计算原始的选择性比率 ：

$$r = \frac{\sigma_l}{\sigma_g} \in [0, \infty), \tag{5}$$

该比率将局部出现频率相对于全局基线进行归一化：$r = 1$ 表示中性，$r > 1$ 表示富集，$0 < r < 1$ 表示耗减。我们注意到，近期研究也独立发现，该比率是估算经过过滤的近似最近邻搜索计算成本的关键因素 [62]。然而，r 无界这一特性在两方面限制了它的实用性。首先，这使得**聚合**变得复杂：简单的均值对离群值十分敏感，罕见的全局过滤器若出现在局部邻域中，可能产生指数级增大的比率，从而扭曲数据集级统计量。其次，在**优化**方面，成本模型通常需要归一化输入（例如 $[-1, 1]$ 或 $[0, 1]$），以便为计划切换设定稳定的阈值。无界度量难以定义一致的启发式方法，来决定何时优先采用特定索引策略。因此，我们采用双线性（M¨obius）变换，得到每个查询对应的过滤器-向量选择率相关性 ρ_q：

$$\rho_q = \frac{r - 1}{r + 1} \in [-1, 1) \tag{6}$$

这会将比率对称地映射到 $[-1, 1)$，其中 $\rho_q = 0$ 表示中性，$\rho_q > 0$ 表示正相关（富集强度），$\rho_q < 0$表示负相关（耗减强度）。该变换是单调且可微的，其导数为 $f'(r) = {}^{2}{}_{(r+1)_2} > 0$。用于反归一化的逆变换为 $r = (1 + \rho_q)/(1 - \rho_q)$，其中 $\rho_q \in [-1, 1)$。

最后，为了跨查询进行聚合并评估系统性纠缠程度，我们定义平均过滤器-向量选择率相关性

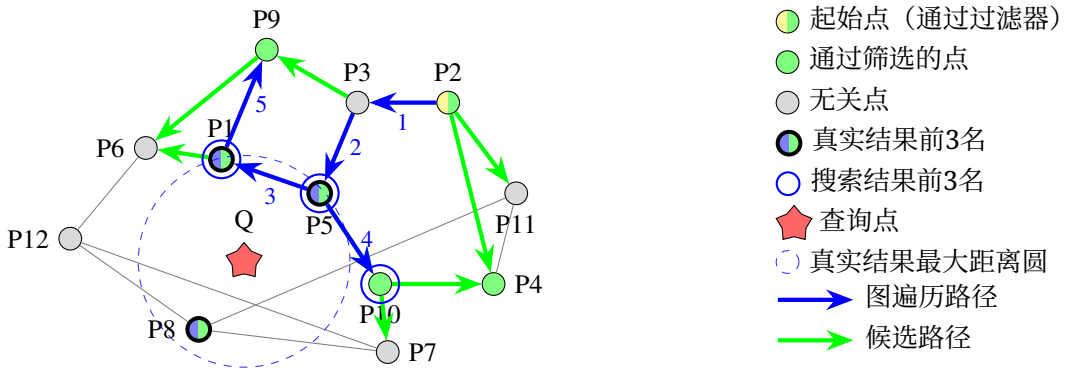
$$\bar{\rho} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \rho_q \in [-1, 1) \tag{7}$$

我们还可以通过计算近似 k-最近邻的选择率（使用 ANN-index *(近似局部选择率)* ）以及从我们的数据集中均匀抽取的样本的选择率（近似全局选择率），来获得该度量的一个较为准确的估计。

5 TA过滤策略和系统分类

5.1 FANNS策略

当返回结果集还必须满足某些元数据（或结构化数据）约束时，这类查询通常称为混合查询、过滤近似最近邻搜索或过滤向量搜索。在这些情况下，我们需要对满足给定条件的元素子集执行选择操作，这通常称为过滤。在关系数据库管理系统（RDBMS）中，这一操作称为应用选择谓词。



Step	当前项	结果	邻居	候选项	已访问
0	P2	[]	P3、P4、P11	P3、P4、P11	P2、P3、P4、P11
1	P3	P5, P9, P2	P2, P5, P9	**P5**, P9, P4	P2, P3, P4, P5, P9, P11
2	P5	P1, P5, P10	P1, P3, P10	**P1**, P10, P9	P1, P2, P3, P4, P5, P9, P10, P11
3	P1	P1, P5, P10	P5, P6, P9	**P10**, P9 *	P1, P2, P3, P4, P5, P6, P9, P10, P11
4	P10	P1, P5, P10	P4, P5, P7	**P9** *	P1, P2, P3, P4, P5, P6, P7, P9, P10, P11
5	P9	P1、P5、P10	P1、P3、P6	[] *	P1、P2、P3、P4、P5、P6、P7、P9、P10、P11

* 由于没有邻居点，候选项不会更新 $n \in Neighbors$，使得 $d(n,\ q) < max(d(Results,\ q))$ 或 $|Results| < ef_{搜索}$

图 2：在 1 层 HNSW 索引中，使用预过滤 策略进行图遍历， $m = 3$且搜索参数为$k = 3$、 $ef_{\text{search}} = 3$。

在本研究中，我们将选择谓词和过滤器这两个术语互换使用，并假设它们在不同系统中含义等同。

我们参考了近期关于向量数据库的综述 [17, 43] ，概述过滤近似最近邻搜索的主要策略，并给出其精确定义。混合查询中的过滤通常采用三种主要方法：预过滤、运行时过滤和后过滤。这些术语看似直观明了，但文献显示，目前尚缺乏系统性的理解，导致不同研究中的定义不一、概念相互重叠 [43,34, 58, 51], ，尤其是在预过滤方面。每种方法的具体实现取决于底层索引类型（例如，基于图或基于分区）。

我们不讨论基于融合的 方法 [45, 61], ，这类方法会将元数据值融合到索引构建或距离计算中。本研究主要不纳入这些方法，是因为它们缺乏模式无关性。通过将元数据与向量索引紧密耦合，这些方法会损害系统灵活性和可扩展性。融合方法难以处理稀疏或高维属性数据；如果许多向量缺少某些属性，或者属性总数很大，将所有属性融合到单一、紧凑的向量空间或度量中，就会成为复杂且非平凡的问题。此外，融合方法可能需要预先了解查询工作负载，才能将元数据有效地整合到索引或距离度量中，这限制了它们对任意模式或临时查询的适用性。基于这些原因，这里不将其视为现有近似最近邻搜索索引的通用过滤策略。

5.1.1 预过滤 +近似最近邻搜索

预先评估过滤谓词，以生成一个位集，标明哪些向量符合过滤条件。随后使用该位集引导向量搜索，理想情况下通过屏蔽无关点来缩小有效搜索空间。然而，在基于图的索引中，情况未必如此。

对于基于图的索引，遍历过程中仍需计算距离，以确保图的连通性并朝查询向量的方向导航。系统会查询位集，以确定是否应将已访问的向量加入候选项优先队列。

对于基于分区的索引（例如 IVFFlat），位集可以在计算每个向量的距离之前检查过滤器。这种方法能够及早缩小搜索空间，避免对不符合条件的点进行不必要的距离计算。

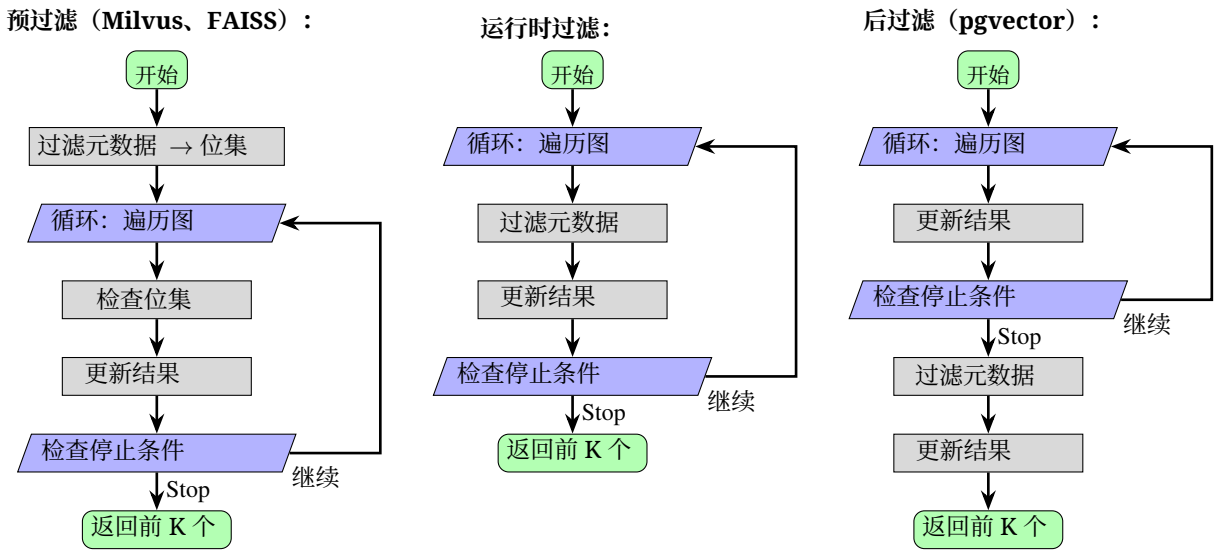


图 3：近似最近邻搜索的过滤策略。

关于预过滤 + k 近邻搜索的说明。有时，对于选择率较低的查询，向量搜索引擎可以应用过滤器，并在筛选后保留的点中进行穷举搜索。需要将其与预过滤 + 近似最近邻搜索区分开来，因为前者能够精确求解问题，而非近似求解。

5.1.2 运行时过滤

在这种方法中，属性检索和谓词评估采用惰性方式执行，仅在搜索过程中访问候选向量时触发。尽管这种按需模式可能因随机 I/O 访问而引入延迟，但它能显著减少谓词评估的总次数。这种权衡在内存受限的环境中，或向量过滤器计算开销较大时尤其有利，例如涉及复杂正则表达式、字符串模式匹配或跨多个关系的连接条件。

在 基于图的索引 中，会检索图导航过程中遍历的向量的属性。为减轻这些重复检索带来的 I/O 开销，可以采用批处理策略，例如遍历多个跳数，并一次性检索所有已访问点的属性。

在基于分区的索引中，仅获取搜索空间内向量的属性（即 IVFFlat 中相关聚类的属性）。这种定向获取通常不会产生超出标准聚类访问之外的额外 I/O 开销，因为目标聚类中所有点的属性都会一并加载。

我们未将运行时过滤纳入主要评估，因为所评估的内存中系统（FAISS、pgvector）的标准 API 并未原生提供此功能。虽然运行时过滤对于 DiskANN [15] 等基于磁盘的系统至关重要，可用于掩盖 I/O 延迟，但要将其应用于内存中的 HNSW，就需要自定义修改索引，这偏离了本文旨在系统化研究的“开箱即用”系统行为。

5.1.3 后过滤

在向量搜索检索出一组候选向量后，再应用过滤器。这种做法将过滤与搜索解耦，但可能会纳入错误候选项（即相似但不符合过滤条件的向量），并在搜索后将其丢弃。它通常具有较低的搜索延迟，但召回率可能会降低。这种方法常见于关系型数据库管理系统，例如 PostgreSQL 中的 pgvector，也可能用于 Oracle Database 等专有系统。

在基于图的索引中，过滤只会针对候选优先队列中有限数量的向量进行（例如，在 HNSW 中，增大队列大小会增加遍历延迟）。对于与向量相似度不相关的低选择性谓词

，这可能导致 top-k 结果不完整（幸存者少于 k 个）。

在基于分区的索引中，搜索参数通常允许探测更大的候选集（例如，在 IVFFlat 中设置 $nprobe$以检查多个聚类），从而降低结果不完整的风险；与图索引相比，尽管在选择性极高时仍存在这一问题。

5.1.4 FANNS 的查询优化

将向量搜索集成到关系型数据库管理系统中，可以编写更复杂的查询，其中涉及关系运算符，例如与其他表连接，或对连接后的关系应用过滤器。这带来了查询优化中的一个根本难题：是先进行向量搜索，再执行查询中的“关系型”部分，还是先完成查询中的“关系型”部分，以确定相关向量的集合，然后再进行相似性搜索。

查询优化器依靠基数估计在不同执行计划之间进行选择。然而，若要优先执行向量搜索，优化器就需要准确的选择性估计来确定扩展因子——根据k/选择性计算候选项，以确保过滤后仍有 k 结果。然而，如果缺乏可靠的估计，系统就只能先完成关系连接或过滤，以生成符合条件的 ID 子集。随后，这个子集会作为约束传递给向量引擎，从而确保准确率，但可能会增加延迟。

5.2 向量搜索系统

所有为向量相似性搜索设计的系统可分为三大类：关系数据库向量扩展（例如 PG-Vector [46], PASE [63], VBASE [64]）、专用向量数据库（例如 Milvus [57],Pinecone [48], Weaviate [59]）以及向量搜索库（例如 FAISS [12], ParlayANN）。我们将测试每一类中的代表性系统。

系统	标量	类别	字符串匹配	过滤策略
扩展				
PG-Vector	✓	✓	✓	基于成本的（后置/前置 + 近似最近邻搜索）
PASE	✓	✓	✓	与 PG 集成的 IVFFlat / HNSW
VBASE	✓	✓	✓	松弛单调性（内核集成）
专用数据库				
Milvus	✓	✓	✓（正则表达式）	预过滤 + 近似最近邻搜索
Qdrant	✓	✓	✓ (精确匹配/子串)	预过滤（负载索引）
Weaviate	✓	✓	✓ (BM25)	预过滤（倒排索引）
Pinecone	✓	✓	✓ (前缀)	预过滤（元数据索引）
Vespa	✓	✓	✓ (ling)	多阶段（预过滤/排序）
Elasticsearch	✓	✓	✓ (全文)	混合（预处理/后处理/重新评分）
库				
FAISS（使用 NumPy）	✓	×	×	预过滤 + 近似最近邻搜索（IDSelector）
HNSWlib	×	×	×	后过滤（遍历期间）
ParlayANN	×	×	×	并行图搜索

表 3：向量搜索系统比较。

5.2.1 向量扩展

向量扩展将向量数据类型直接集成到关系数据库列中，从而保留核心关系型能力，例如 ACID 合规性，以及通过 SQL 语法执行带有连接和 GROUP BY 谓词的查询。一个关键优势是可在现有关系型基础设施中无缝采用，无需进行数据迁移。尽管如此，与专用解决方案相比，这些扩展可能存在功能限制。我们选择 **PG-Vector** [46]，是因为其作为 PostgreSQL 扩展 [47]得到积极维护并被广泛采用。它支持全面的

过滤器范围，基本上涵盖 PostgreSQL 中可用的所有谓词（例如，通过 WHERE子句进行的标量、类别型和字符串操作）。查询优化器的基于代价的模型允许在过滤策略之间切换，但它并不总是选择最优的过滤式近似最近邻查询，如第7节所示。

其他值得关注的候选项包括 **VBASE** [64], ，这是一个基于 PostgreSQL 构建的系统，通过修改数据库内核来支持“松弛单调性”，以实现更智能的搜索终止检查；以及**PASE**[63], ，这是阿里巴巴开发的 PostgreSQL 扩展，支持 IVFFlat 和 HNSW 索引。

5.2.2 专用向量数据库

专用向量数据库 摒弃传统关系模式，通常不支持类似 SQL 的语法。它们的形态各异，从无模式的键值存储到强制采用预定义模式的数据库都有，后者用于向量集合，并且通常原生支持嵌入生成。与向量扩展相比，专用向量数据库在索引和查询方面具有性能优势，不过这些差异通常可以缩小到微不足道的程度 [66]。我们选择**Milvus** [57, 38] ，因为它是开源的，且维护完善。它支持多种过滤器，包括标量和类别型谓词，以及通过正则表达式进行字符串匹配。*Knowhere* 是 Milvus 的向量搜索引擎，基于 FAISS 构建并进行了定制化优化。Milvus 主要采用预过滤 + 近似最近邻搜索策略，并使用基于逻辑的查询优化器，该优化器在较新版本中仍在不断演进。

此类别中的其他代表性系统包括：

• **Pinecone** [48]: 一种专有的闭源云数据库。虽然其实现细节不透明，但据称采用了自定义的基于图的索引，将元数据过滤直接整合到索引结构中。此方法旨在避免在标准 HNSW 图上进行激进预过滤时通常会出现的图不连通问题。

• **Qdrant** [49]: 一种开源引擎（Rust），采用“基于分段”的架构。它为元数据维护独立的数据结构（例如，用于关键词的哈希表和用于数值范围的 B树）。在过滤搜索期间，这些元数据索引会生成一个位集，用作掩码，以便在检索向量时识别符合条件的点。

• **Weaviate** [59]: 一种开源引擎（Go），将 HNSW 向量索引与用于标量数据的传统倒排索引（倒排列表）相结合。对于过滤查询，系统会先查询倒排索引，获取对象 ID 的允许列表（或位图），然后在 HNSW 图搜索过程中用它来过滤候选项。

• **Vespa** [56]: 一个开源张量计算引擎。与纯向量存储不同，Vespa 将查询作为张量运算执行。它还可以自动切换到精确搜索，以处理带有低选择率过滤器的查询。

• **Elasticsearch** [13]: 基于 Apache Lucene 构建，采用了 Lucene 的 HNSW 实现。系统会评估过滤选择性，以决定采用近似搜索（在图遍历期间检查过滤器），还是对经过过滤的文档集进行精确暴力扫描。

5.2.3 向量搜索库

向量搜索库（如 FAISS [12, 14], ）是轻量级、模式无关的工具，专为向量索引和检索而优化。与完整的数据库不同，它们不负责持久化存储，而是对预加载向量进行索引。过滤式近似最近邻查询需要借助外部位集，标明符合过滤条件的向量。我们采用 **FAISS** ，因为它是开源软件，且在该领域应用广泛。

这一类别中的其他知名库包括 :

• **HNSWlib** [35]: 一个仅包含头文件的 C++ 库，实现了 HNSW 图算法。它是多个向量数据库（如 Chroma）的核心索引引擎。

• **ScaNN** [16]: 由 Google 开发，这个库引入了各向异性向量量化。与最小化欧几里得重构误差的标准乘积量化（PQ）不同，ScaNN 优化了一种优先考虑方向准确性的损失函数，这对于内积搜索至关重要。

• **ParlayANN** [36]：一个专注于近似最近邻搜索并行算法的研究库，涵盖图索引（例如 HNSW、Vamana）。其架构旨在最大限度地减少线程争用，并确保在高并发多核系统上的行为具有确定性。

6 基准测试框架

6.1 ANN-Benchmarks

由于硬件差异和实现细节各不相同，评估近似最近邻搜索算法向来十分困难。ANN-Benchmarks [6] 通过为每种算法提供容器化测试环境来解决这一问题，并附带 YAML 配置文件，用于定义构建说明、超参数和索引构建流程。通过验证后，这些算法会在一组固定的公开数据集上运行，以评估召回率（检索到的真实最近邻所占比例）与吞吐量（每秒查询数，即 QPS）之间的权衡；测试通常会受到单线程 CPU 执行等资源限制。

该基准测试采用经过筛选的多样化高维向量数据集，涵盖图像和文本等领域，包括 GloVe-100 等词嵌入（角距离，100 维）和 SIFT-128 等图像特征（欧几里得距离，128 维），以及 Kosarak 等稀疏数据集（杰卡德距离，最高 27,983 维），具体汇总见表4。评估在 AWS r6i 实例等标准化硬件上进行，以确保可复现性；性能指标以对数刻度绘制，展示召回率与吞吐量之间的权衡，同时呈现索引构建时间和内存占用等次要因素。该框架已对 30 多种算法进行了基准测试，其中包括 FAISS、Milvus、PG-Vector、HNSWlib 和 Annoy 等知名库；社区成员也持续贡献，确保结果保持最新。

6.2 现有数据集的局限性

数据集名称	Dim.	基数	查询	距离	领域	嵌入类型	$\lvert V\rvert^*$	$\lvert A\rvert^{**}$
SIFT-1M	128	1,000,000	10,000	L2	图像	人工构造	1	0
GIST-1M	960	1,000,000	1,000	L2	图像	人工构造	1	0
GLOVE-100	100	1,183,514	10,000	余弦	Text	基于计数	1	0
MNIST-784	784	60,000	10,000	L2	图像	原始像素	1	0
NYTimes-256-ang	256	290,000	10,000	余弦	Text	Word2Vec嵌入	1	0
Kosarak	27,983	74,962	500	杰卡德相似度	Web	稀疏二值	1	0
LAION-5B***	768	5.85×10^9	–	余弦	多模态	CLIP（对比式）	2	6
ArXiv***	768	2,700,000	10,000	余弦	Text	基于 Transformer	1	10
Wiki-22-12-en	768	35,200,000	–	余弦	Text	基于 Transformer	1	5
YFCC***	4,096	100,000,000	–	L2	图像	深度卷积神经网络	1	5
YouTube 音频***	128	6,100,000	–	L2	音频	人工构造	1	2
YouTube 视频***	1,024	6,100,000	–	L2	视频	人工构造	1	2

表4：用于近似最近邻搜索基准测试的数据集 [4]。* $|V|$ – 每条记录的向量数量。 ** $|A|$ – 每条记录的元数据属性数量。 *** 未纳入 ANN-Benchmarks；此处添加以便与新兴的面向文本的数据集进行比较。

尽管现有近似最近邻数据集十分严谨，但仍存在一个关键不足：它们主要源自非文本嵌入（例如图像嵌入或通用词向量），因此与稠密文本检索等现代自然语言处理任务的相关性有限。LAION-5B 数据集是一种广受欢迎的大规模替代方案，通过数十亿图文对缓解了规模问题，但其属性仅限于基本的类别标签（例如美学评分、内容标记），缺少进行细致评估所需的连续型或灵活多样的元数据。

ArXiv 数据集是一个显著的例外，其中包含来自超过 270 万篇 arXiv 摘要的基于 Transformer 的嵌入，以及 11 个结构化属性（例如数值型引用次数、类别型领域和时间元数据）。同样，*Wikipedia-22-12-en-embeddings* 提供了从维基百科衍生出的 3500 万个基于 Transformer 的向量，规模庞大。虽然它包含有价值的结构化元数据，却也带来了显著的结构复杂性：与

标准的一对一映射不同，该数据集为每个逻辑实体（文章）分配数量不定的向量，因为每个段落都会生成一个单独的向量。这为实体检索带来了“可变基数”问题，使基准测试变得更加复杂，因为必须区分向量级（段落）相似性与条目级（文章）相关性。

然而，这些数据集存在不足，因为它们无法支持复杂连接。其范围有限，主要包含数值型和类别型字段，缺乏更深入的关系结构，因此无法扩展到诸如带类别层级的时间范围连接等查询组合。此外，作者数量、类别列表和版本历史等属性虽具有学术性质，但对于更广泛的检索任务（而非小众的学术过滤）价值有限。为求完整，本文纳入用于多模态比较的 *YFCC100M* 和 *YouTube* 音频/视频 数据集，但显然它们并不适用：YFCC 依赖以图像为中心的深度卷积神经网络嵌入，而 YouTube 数据集使用人工构造的特征，且元数据极少。

6.3 数据集设计

为解决这些局限性，我们推出 **MoReVec**（电影与评论向量），这是一个旨在支持针对向量嵌入和结构化属性进行优化的数据集。

我们借鉴用于构建关系数据库管理系统中查询优化器评估的连接排序基准（JOB） [28]的丰富 IMDb 数据生态，将电影简介和评论的文本嵌入与可获取的网络来源元数据合并。具体而言，我们使用 *gte-base-en-v1.5* 模型 [33],生成 768 维向量；该模型是大规模文本嵌入基准（MTEB）排行榜上的高性能嵌入方法，应用于电影简介和用户评论等文本字段。所有嵌入均经过 L_2归一化，确保在所有受评估系统中，余弦相似度等同于内积。

生成的数据集遵循表 5 中详述的可扩展模式，从而支持高级混合查询，将语义相似性搜索与基于属性的筛选和可扩展的连接操作相结合。为开展可扩展性研究，我们将每个数据集实例化为三种基数：

- 小型： $|Movies| = 9,\ 999$； $|Reviews| = 247,\ 286$；
- 中型： $|Movies| = 99,\ 560$； $|Reviews| = 1,\ 496,\ 493$；
- 大型： $|Movies| = 551,\ 155$， $|Reviews| = 2,\ 598,\ 267$；

字段	描述	数据类型
mid	唯一 字母数字母数字标识符电影	可变长字符串
片名	电影片名	可变长字符串
year	电影上映年份已发布	整数
类型	电影类型	可变长字符串
平均评分	平均评分 of电影	浮点数
投票数	电影的投票数电影	整数
电影向量	电影简介 em-嵌入	768维向量

（a）电影表

字段	描述	数据类型
rid	评论的唯一字母数字标识符评论	可变长字符串
uid	用户的字母数字标识符留下评论的用户	可变长字符串
mid	电影表中 mid 的外键	可变长字符串
评论向量	评论的向量嵌入，指向一部电影	768维向量
电影评分	用户对电影给出的评分	整数
总投票数	对该评论的总投票数	整数
正面投票占比	评论正面投票的占比view	浮点数
质量	根据评论的总投票数和正面投票占比评估	可变长字符串

(b) 评论表

表 5：所提出的数据集的模式。

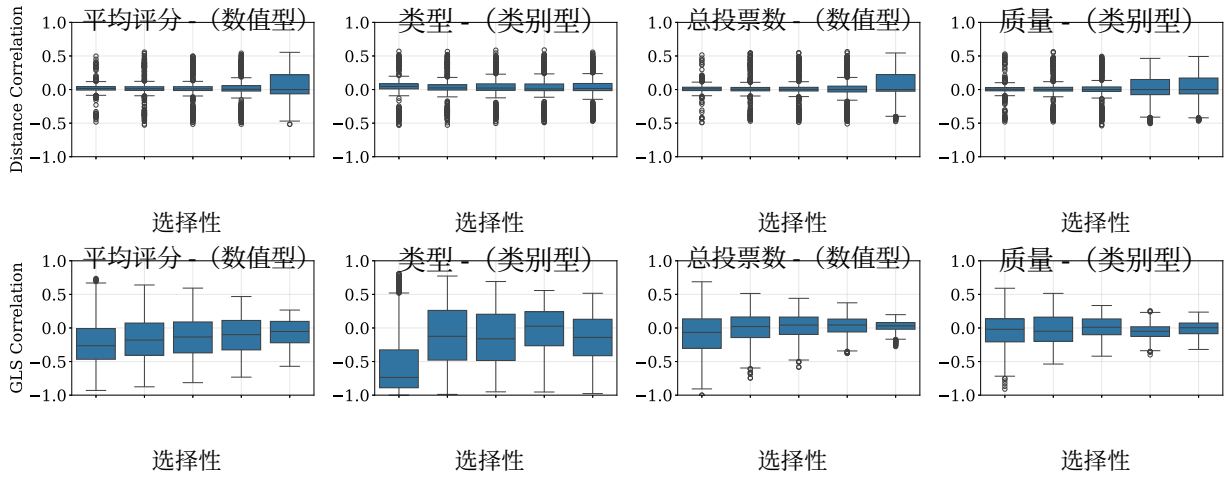


图4：不同目标选择率下，元数据属性与向量嵌入之间逐查询相关性值的分布 ρ_q

6.4 工作负载

过滤器。图4显示，在两个数据集的整个选择性范围内，对于所有考察的属性，向量与元数据属性之间的平均*GLS* 相关性 $\bar{\rho}$ 都趋近于 0。在低选择性水平（$\sigma_g \leq 0.1$）下，我们观察到某些属性的相关性明显偏向负值。这在一定程度上可归因于有限邻域大小（$|\mathcal{N}_q| = 2048$）带来的采样伪影；当全局选择性足够低时，固定邻域内有效邻居的预期数量较少。因此，查询得到的有效局部邻居往往少于预期值（$\sigma_l < \sigma_g$），使 ρ_q 趋向于 -1，并导致均值偏斜。

考虑到这一现象，结果证实，元数据属性大多独立于嵌入空间，因此我们可以研究过滤器选择性对过滤近似最近邻性能的单独影响。因此，我们对平均评分列（电影数据集）和总票数列（评论数据集）应用数值/标量过滤器。我们选择过滤阈值，使预期选择性 σ_g（即通过过滤器的数据集占比）大致符合以下目标值：

- 小型数据集： $\sigma_g \approx \{0.01;\ 0.03;\ 0.05;\ 0.1;\ 0.2;\ 0.5\}$
- 中型数据集： $\sigma_g \approx \{0.0003;\ 0.001;\ 0.01;\ 0.03;\ 0.05;\ 0.1;\ 0.2\}$
- 大型数据集： $\sigma_g \approx \{0.0003;\ 0.001;\ 0.01;\ 0.03;\ 0.05;\ 0.1\}$

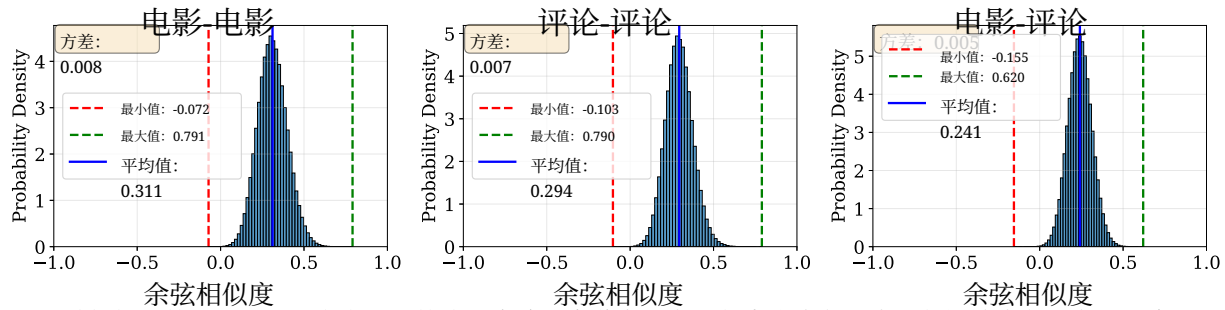


图 5：随机抽取的 1,000,000 个向量对的成对余弦距离分布。左：仅电影嵌入；中：仅评论嵌入；右：混合对，每对各包含一个电影嵌入和一个评论嵌入。

查询。由于语义搜索查询与数据使用相同的嵌入模型生成，我们假设电影和评论 表中的向量分布相似。图 5 展示了各表及其混合数据中向量对的成对余弦距离分布，为这一假设提供了支持。

因此，我们从每个数据集中随机抽取 1,000 个向量作为查询集。每个查询在语义上对应于"查找最接近查询向量 q 的前 k 个向量，其中标量属性的值大于或等于阈值 t"。我们测试 k 的四种不同取值：$k \in \{1, 10, 40, 100\}$。每个 1,000 个查询向量都会针对每个过滤条件和全部四种 k 执行，从而为每个数据集生成 $1000 \times 7 \times 4 = 28{,}000$ 个经过过滤的 k-近似最近邻查询（包括不使用过滤器的运行）。

我们的查询工作负载未包含多表 JOIN 谓词。虽然 pgvector 在语法上支持此类谓词，但其查询优化器对于这类查询始终会退回到顺序扫描，从而完全绕过向量索引。相反，Milvus 和 FAISS 等模式无关系统缺少原生执行连接所需的关系运算符。

7 实证评估

我们对 *FAISS*、*Milvus* 和 *pgvector* 中 FANNS 的性能进行了广泛的实证比较。我们重点关注查询吞吐量与召回率之间的权衡，以及各系统采用的底层查询规划和执行策略。我们的主要目标是确定通用过滤策略（预过滤和后过滤）在真实世界的向量数据库中的表现，并考察系统级架构选择在多大程度上会改变底层索引结构的预期算法行为。为此，我们的评估将探讨以下问题：

- 过滤选择性如何影响召回率与吞吐量的权衡？我们考察低选择性过滤器是否总会降低性能，还是某些索引类型和过滤策略能更好地适应稀疏结果集。

- 在过滤搜索中，*HNSW* 传统上相较于 *IVFFlat* 的优势是否依然成立？我们测试在不同选择性范围内，HNSW 基于图的遍历是否仍优于 IVFFlat 的簇剪枝机制。

- 系统层面的架构决策会如何影响性能？我们探究算法适应性、数据分段（Milvus）和基于成本的查询优化（pgvector）等因素，对过滤近似最近邻搜索性能的影响是否大于索引算法本身的选择。

- 关系型向量扩展中的查询优化器能否为混合查询有效选择执行计划？我们分析 pgvector 的基于成本的优化器能否可靠地选择兼顾召回率和延迟的执行计划。

为便于复现和进一步研究，我们扩展的 过滤版 *ANN-Benchmarks* 框架、MoReVec 关系型数据集以及 GLS 相关性分析工具均可在 GitHub 上获取 [3]。

表 6：实验中使用的软件版本。

系统	服务器/库版本	备注
Milvus	v2.6.6	pymilvus 2.6.3
FAISS	1.12.0 (faiss_hnsw)	CPU 版本
pgvector	0.8.1 (git 主分支)	PostgreSQL 16

7.1 实验设置

索引。 我们评估了两种广泛应用的近似最近邻索引方法：*HNSW* 和*IVFFlat*。对于**HNSW** 索引，我们探索以下超参数配置：

- Construction parameters: $(M, ef_{\text{construction}}) \in \{(5, 25), (10, 50), (15, 75)\}$

• 搜索参数：$ef_{\text{搜索}} \in \{1, 10, 40, 100, 200, 500, 1000\}$

对于 **IVFFlat** 索引，我们通过调整质心数量来控制数据集大小带来的权衡：

• 构建参数：$clusters = \{small$：(100,500) $medium$：(300, 1200) $large$：(750, 1600)$\}$。这些范围大致遵循常见的 $C \approx \sqrt{N}$ 启发式规则，其中 C 表示聚类数量，N 表示数据集大小。

• 搜索参数：$n_{\text{探测数}} = \{1, 5, 10, 20, 50, 150, 300\}$

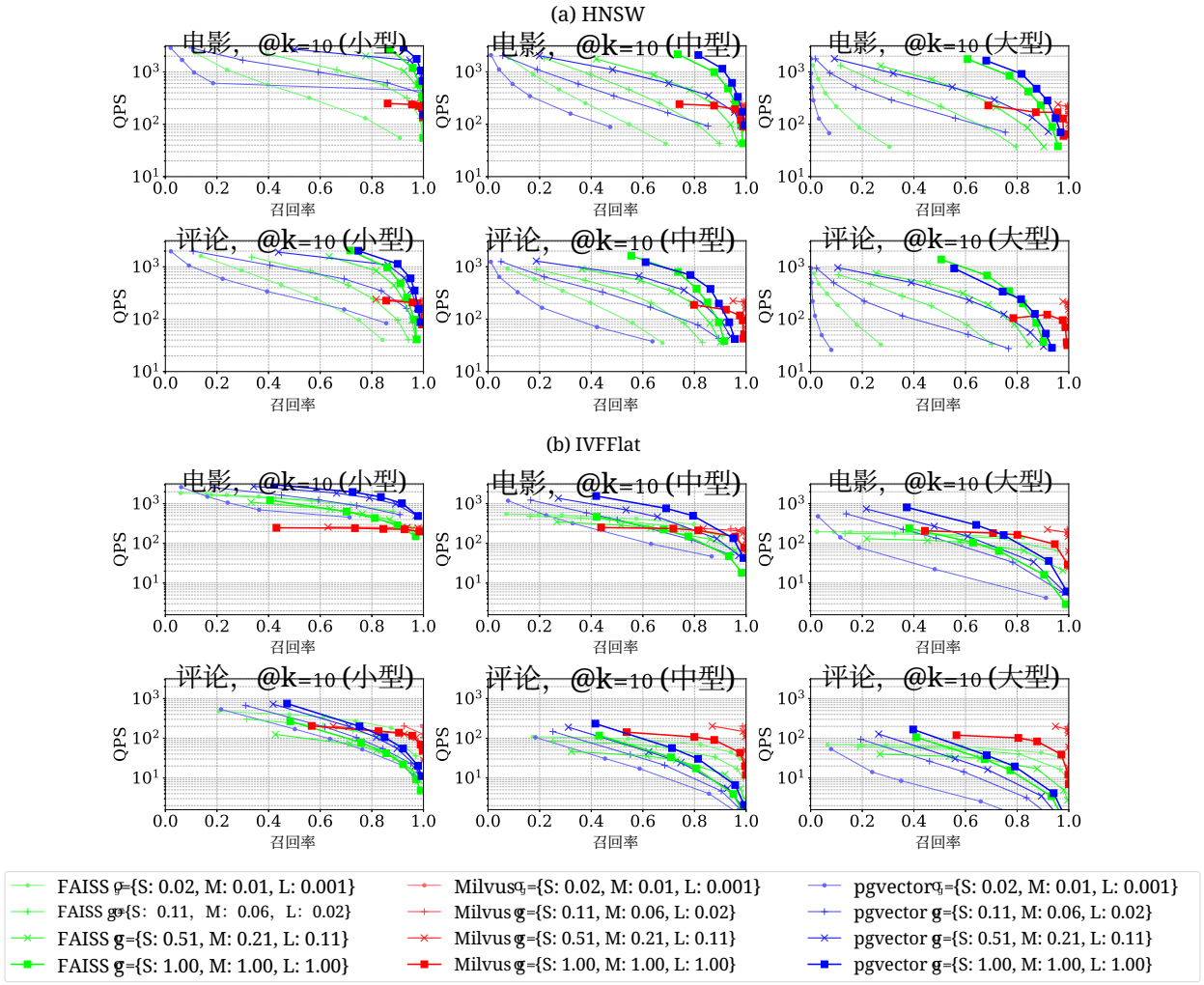


图 6：不同选择性水平下的每秒查询数–召回率曲线。

ANN-Benchmarks 实现方面的变更。我们扩展了 ANN-Benchmarks 的实验循环，使其能够动态生成 YAML 配置：在预处理阶段，每个索引参数集对应一个“运行组”。在查询阶段，我们在同一个已构建索引上测试过滤器/k/索引/搜索参数组合，无需重新构建；过滤器以目录层级的形式编码在 HDF5 中，并将结果汇总到一份全面的 CSV 文件中（每个过滤器/k/查询/索引/搜索参数对应的运行时间和召回率）。

执行协议。按照 ANN-Benchmarks 的方法，所有查询均以单线程方式逐个执行，每次处理一个查询。这种设计将各系统的算法与架构性能特征从

并发效应中隔离出来，从而能够进行受控比较。所有受评估的系统均以内存中 模式运行；在开始查询执行前，索引和数据集会全部加载到随机存取存储器中。我们不评估基于磁盘或流式处理的工作负载。测量阶段内，每个查询仅执行一次，不重复执行；吞吐量（每秒查询数）按测得的单查询延迟的倒数计算。不采用批量查询执行——每个查询均单独提交并完成。

过滤召回率的真实值。 对于过滤查询，真实值的计算方式是：先将过滤谓词应用于整个数据集，然后在过滤后的子集上执行精确 k-近似最近邻搜索（蛮力距离计算）。形式上，对于查询向量 q 和过滤谓词 ϕ，我们将真实值定义为 $\{v \in \mathcal{D} \mid \phi(v) = 1\}$中与 q距离最小的 k 个向量。

硬件和软件配置。 与 ANN-Benchmarks [6], 一样，我们在 Docker 容器中运行所有实验，其中包含受评估系统所需的全部依赖项。我们使用一台运行 Ubuntu 24.04 LTS 的机器，配备 56 核 Intel Xeon E5-2660（2.00 GHz）处理器和 256 GB 随机存取存储器。表 6 列出了所有受评估系统的确切版本。为确保可复现性，所有实验均使用指定版本。

范围。 我们的实验仅关注标量不等式过滤器 （例如，评分 >= 阈值）。类别过滤器、字符串模式匹配和多属性合取均不在实验工作负载范围内。此外，我们仅评估内存中执行；基于磁盘的执行和运行时过滤策略不在考察范围内。

7.2 结果

7.2.1 过滤选择性的影响

如图 6 所示，过滤选择性提高时，HNSW 和 IVFFlat 实现的召回率都会持续受到负面影响。无论采用预过滤还是后过滤策略，都会出现这种性能下降，但其作用机制有所不同。在预过滤中，低选择率过滤器会使真实最近邻的分布变得稀疏，包括查询向量周围的区域。相反，在后过滤中，索引扫描后有效候选池会大幅缩小；如果初始搜索未能检索到足够多满足谓词的候选项，最终结果集就不可避免地会被截断。*Milvus* 是个明显的例外，它凭借混合架构在低选择率下仍能保持较高的召回率（见第 7.2.7 节）。

尽管总体而言，IVFFlat 在较低选择性水平下的召回率略有下降，但它可能表现出更优的每秒查询数性能，如图 15 中升高的每秒查询数-召回率曲线所示。这一结果符合预期，因为 IVFFlat 的簇剪枝机制降低了距离计算的计算开销。

然而，基于图的索引中的系统吞吐量（每秒查询数）基本不受选择性变化的影响，因为无论是否存在过滤器，底层图结构都需要进行相当的遍历。这种稳定性表明，与图遍历期间所需的高开销距离计算相比，过滤的额外开销可以忽略不计，尤其是在大型数据集上。

FAISS 中的谓词评估。为满足位集要求，我们实现了一个 FAISS 索引包装器，使用 NumPy 处理谓词评估。对于每个查询，我们首先计算一个布尔掩码，用于指示哪些向量符合过滤器条件，然后将该掩码转换为 FAISS IDSelector。此过程会即时执行，开销可以忽略不计（在大型数据集上为 < 1 毫秒）。图 7 总结了各代表性工作负载增加的延迟。

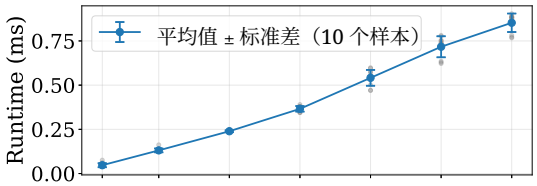


图 7：FAISS 中的位集生成运行时间。

要点。 过滤选择性是召回率的一项普遍约束。然而，其对吞吐量的影响因架构而异：基于图的索引表现出每秒查询数不变的特性，这证实遍历成本掩盖了过滤开销；而基于分区的索引（IVFFlat）则利用剪枝，在低选择性场景下实现了更高的吞吐量。因此，过滤近似最近邻搜索的性能，与其说取决于元数据评估成本，不如说更多取决于所选向量索引的具体机制——需要在遍历稳健性与剪枝效率之间取得平衡。

7.2.2 k的影响

图 14 分析了请求的最近邻数量（k）与系统召回率之间的关系。在搜索参数（例如 efSearch 或 nprobe）保持不变的情况下，索引生成的候选池原始大小保持静态。随着 k 增大，系统必须从这一固定大小的候选集中提取更多有效结果。这会导致召回率系统性下降，因为固定集合包含k 有效邻居的概率会降低。

这种影响在采用后过滤的 *pgvector* HNSW 实现中尤为明显，但在选择率较低时，召回率跃升至 1.0 的情况除外（这是由查询优化器切换到顺序扫描所触发的）。由于过滤器仅应用于由 efSearch 限定的最终最近邻集合，因此，原始向量搜索检索到的大多数候选项不满足谓词的统计可能性增加。结果是，许多真正的邻居从未被检查，因为它们不在未过滤向量空间中的前 efSearch 个候选项之内，从而导致召回率急剧下降。

要点。 为了在增加检索规模 k 时保持较高的召回率，必须按比例增大搜索参数（例如增加 efSearch）。然而，这需要进行权衡，因为更大的搜索参数会带来更高的计算成本和更高的查询延迟。

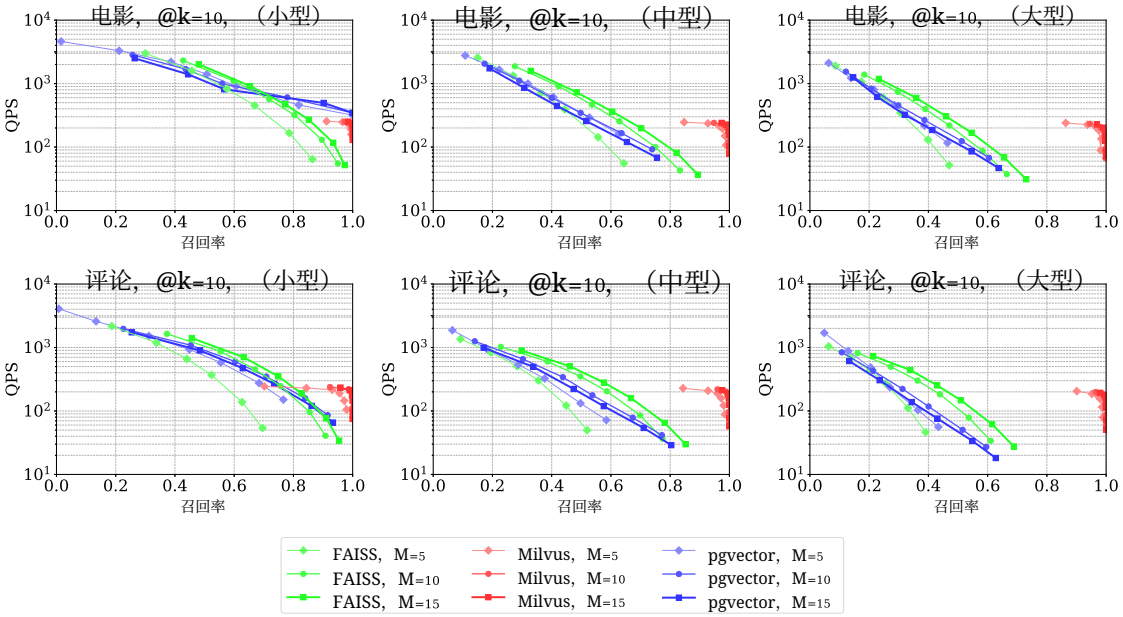


图 8：HNSW 在不同 M 值下的 QPS–召回率曲线。

7.2.3 HNSW 中的过滤策略

后过滤方法在不受谓词引导的情况下遍历 HNSW 图。这样会纳入许多最终被丢弃的候选项，从而降低在 efSearch 预算内可实现的有效召回率。因此，低选择性查询的召回率预计会较低。不过，就运行时间而言，对于未过滤或高选择性查询，*pgvector* 中的后过滤实现与*FAISS* 中的预过滤实现性能相当，甚至略胜一筹。我们认为，这是因为*pgvector* 避免了较长的图遍历，并减少了提前进行过滤评估的开销。

尽管*Milvus*的 Knowhere 引擎基于*FAISS*构建，但由于其混合架构（参见第 7.2.7 节），两者的性能特征存在显著差异。

要点。 虽然通常认为预过滤能带来更高的召回率，但对于高选择率的过滤器，后过滤可能是最佳选择。

7.2.4 HNSW 图密度

如图 9 所示，提高 HNSW 构建参数 M （以及隐含的 efConstruction）会显著增加*pgvector* 和 *FAISS*中的索引构建时间。尽管*Milvus* 表现出对构建参数的不变性，这同样是其混合架构（第 7.2.7 节）所致。HNSW 的构建成本本就会随着 M 值增大而上升，因为用于节点插入的内部向量搜索会变得计算开销高昂。图密度的增加仅能带来微不足道的召回率提升，并且对所有系统的搜索延迟影响甚微（甚至会产生负面影响）（图 8），除非图过于稀疏（例如， $M = 5$）。因此，当优先考虑快速构建索引时，较小 M 值的“轻量级”图更可取。

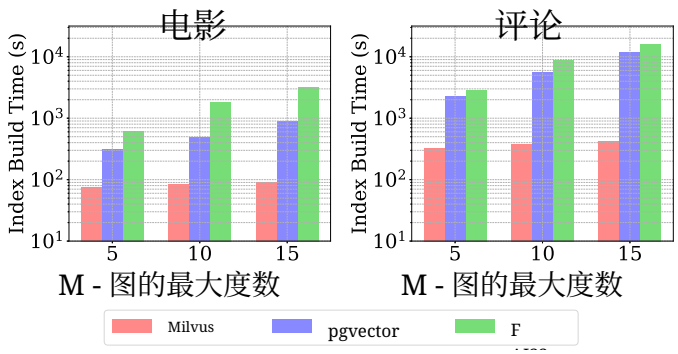


图 9：HNSW 的索引构建时间。

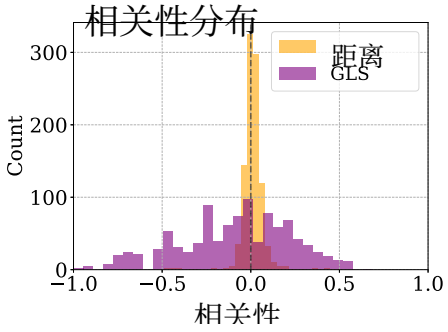


图 10：GLS 相关性与基于距离的相关性值的分布。

7.2.5 IVFFlat 索引

传统上认为，IVFFlat 在通用近似最近邻搜索任务中的表现不如 HNSW，因为其召回率与吞吐量的效率较低。然而，我们的结果表明，在某些特殊情况下——尤其是使用低选择率过滤器时——IVFFlat 的表现优于 HNSW。这可能是因为 IVFFlat 采用聚类剪枝，其适应向量空间稀疏区域的能力可能优于 HNSW 的图遍历机制。

尽管 HNSW 在标准基准测试中占据优势，但它在过滤搜索中的优势并非绝对。我们观察到，对于每个系统，都存在特定的选择性阈值，使得 HNSW 和 IVFFlat 的每秒查询数-召回率曲线相交（图 11）。

要点。 在低选择性场景中，有效邻居稀疏，HNSW 因而需要遍历较远的节点，这对其构成挑战。由于候选集缺少更新，这可能导致搜索过早终止（见图 2）。在这种情况下，IVFFlat 的基于分区的方法提供了一种稳健的替代方案，这表明：1）采用工作负载感知索引选择有助于提升查询性能；2）为了在整个选择性范围内实现最佳性能，可能需要混合索引方法。这可以包括根据查询谓词在图方法和分区方法之间切换，或者根据估计的选择性采用相应的搜索参数。

7.2.6 查询-过滤器相关性对性能的影响

尽管过滤选择性（σ_g）是影响性能的主要因素，但我们对全局-局部选择性（GLS）相关性的分析揭示了元数据与召回率之间更为微妙的关系。如图 12 所示，随着 GLS 相关性从低转高，检索准确率显著提升，而查询延迟（或每秒查询数）在三种相关性场景下均保持不变。这种关系之所以成立，是因为高 GLS 相关性意味着有效邻居在空间上聚集于查询附近，因此索引结构更容易找到它们。反之，低或负的 GLS 相关性表明，有效邻居被“推”至查询自然邻域的外围，这给基于图和基于分区的遍历都带来了巨大挑战。

最后，图 10 展示了该度量的灵敏度：基于距离的相关性只能捕捉范围较窄的交互（约为 $[-0.3, 0.3]$），而 GLS 相关性几乎涵盖了从 -1 到 1 的整个范围。这证实 GLS 是识别“困难”查询的一种表达能力更强的工具。通过将查询划分为低

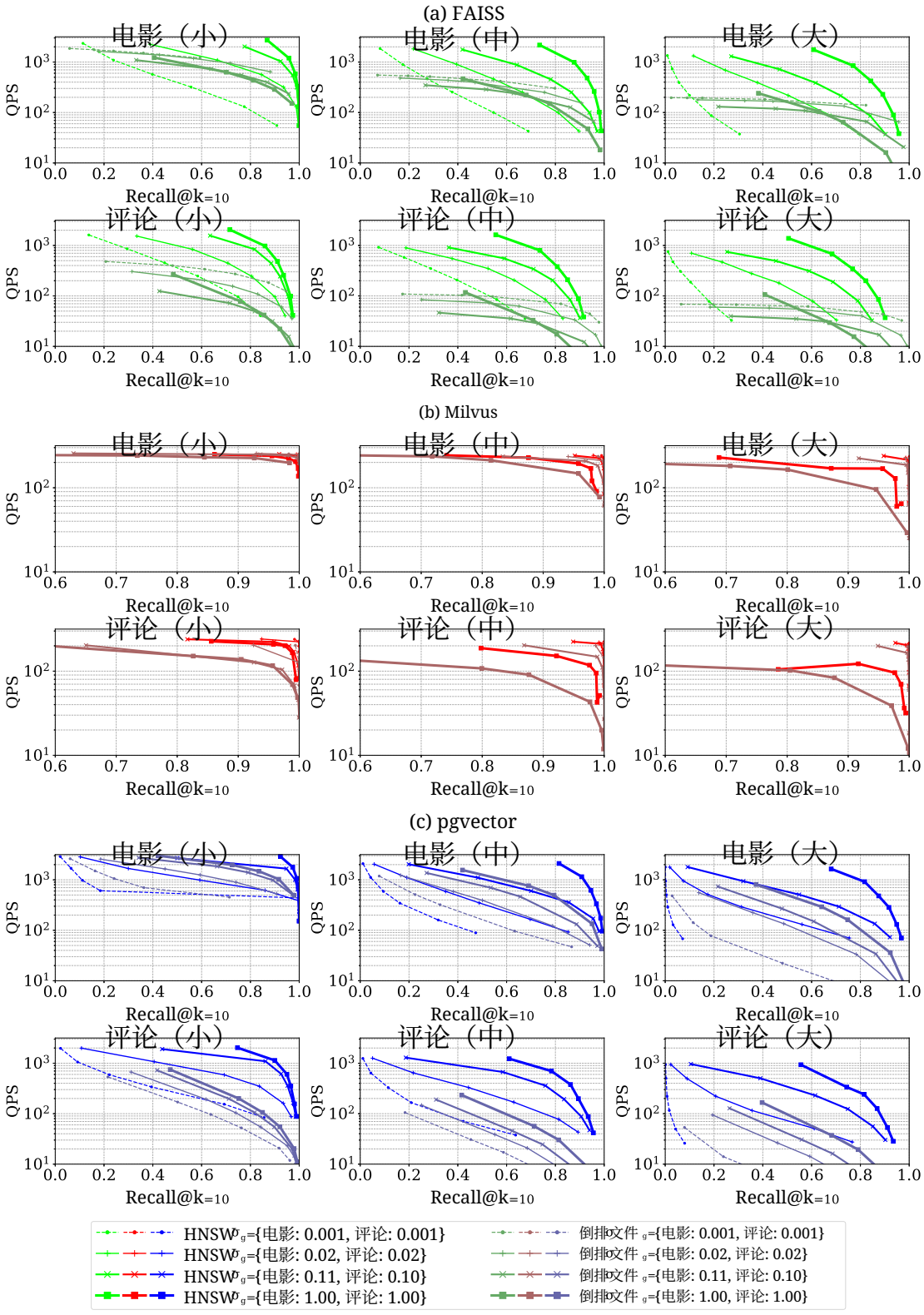


图 11：不同选择性水平下 HNSW 与 IVFFlat 索引的 QPS–召回率曲线。

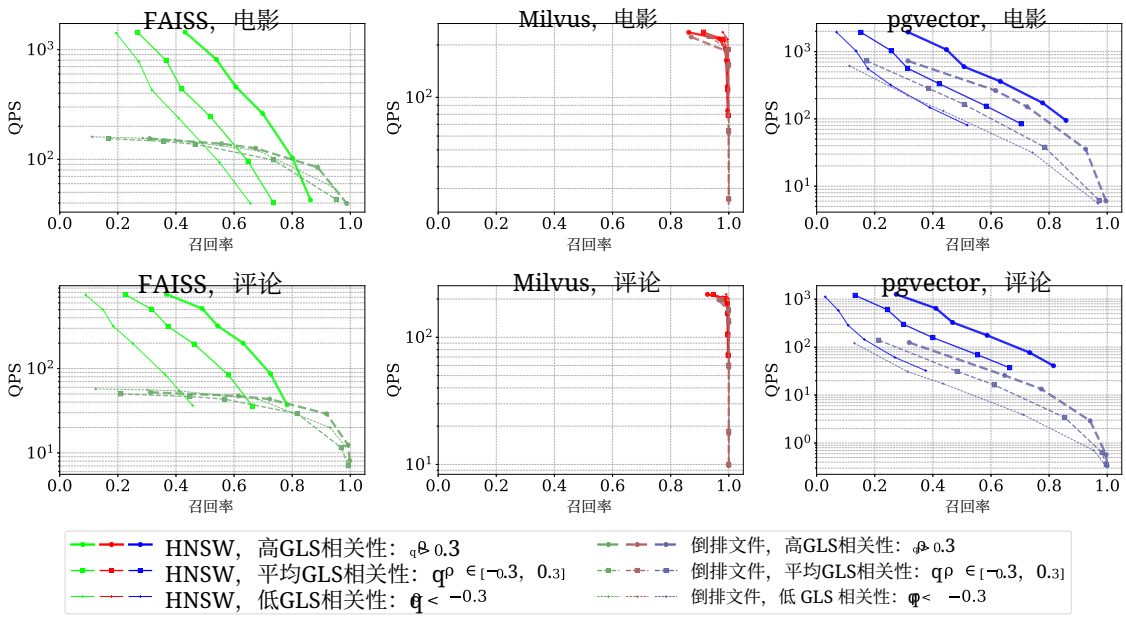


图 12：不同 GLS 相关性水平下的每秒查询数–召回率曲线。

（$\rho_q < -0.3$）、中型（$-0.3 \le \rho_q \le 0.3$）和高（$\rho_q > 0.3$）GLS 区间的情况下，我们发现，在所有测试系统中，较低的 GLS 始终会导致较低的召回率。这与我们先前关于选择性可作为系统级规划中独立变量进行研究的论断并不矛盾；相反，这表明，虽然σ_g 决定了系统的平均行为，但 GLS 相关性解释了各个用户可能遇到的单次查询差异。

7.2.7 Milvus：混合架构与自适应搜索

Milvus 向量数据库运行于名为*Knowhere*的专用引擎之上。该引擎不同于标准 FAISS 包装器，以确保过滤搜索期间的稳定性。*Knowhere* 使用 FAISS 构建图，但所有搜索操作均通过*hnswlib* [18]的深度修改分支执行。与我们的初步假设相反，该系统较高的召回率并非主要由数据分段或“散射-收集”机制驱动。消融研究表明，无论数据被划分为许多 1GB 分段，还是存储在单个 16GB 整体中，*Milvus* 都能保持近乎完美的召回率（图 13）。相反，这种鲁棒性源于搜索引擎内部采用的分层算法策略。

双池图遍历。 最关键的差异在于 Knowhere 在图遍历期间对候选队列的处理方式。标准 FAISS HNSW 实现使用单个优先队列，使已过滤（无效）节点与有效节点竞争队列位置。在过滤率较高时（例如过滤掉 90% 的节点），搜索束会被无效的导航节点占满，导致召回率骤降。Knowhere 通过实现*Dual-Pool* 策略（NeighborSetDoublePopList）解决了这一问题。在遍历期间，它维护两个独立的优先队列：一个用于有效结果，另一个用于仅供导航使用的无效节点。这样可以确保搜索预算（ef）不会被已过滤向量蚕食，从而让 *Milvus* 即使在中等选择率下（例如 10% – –5%），也能保持近乎完美的召回率，而标准 HNSW 实现的性能会出现下降。

自适应回退机制。 为处理限制性极强的过滤器，*Milvus* 采用确定性回退策略，而非基于成本的优化器。引擎会在搜索前检查过滤选择性；如果已过滤向量的比例超过特定阈值（代码库中设为 93%），就会完全绕过 HNSW 索引，执行暴力扫描（预过滤 + kNNS）。由于搜索空间缩小，这项操作的计算成本仍然很低（典型分段为 1–5 毫秒），同时可保证 100% 的召回率。此外，如果图遍历未能返回所请求的 k 邻居，搜索后还会触发一道“安全网”回退，以确保即使在稀疏向量区域中也能保持一致性。

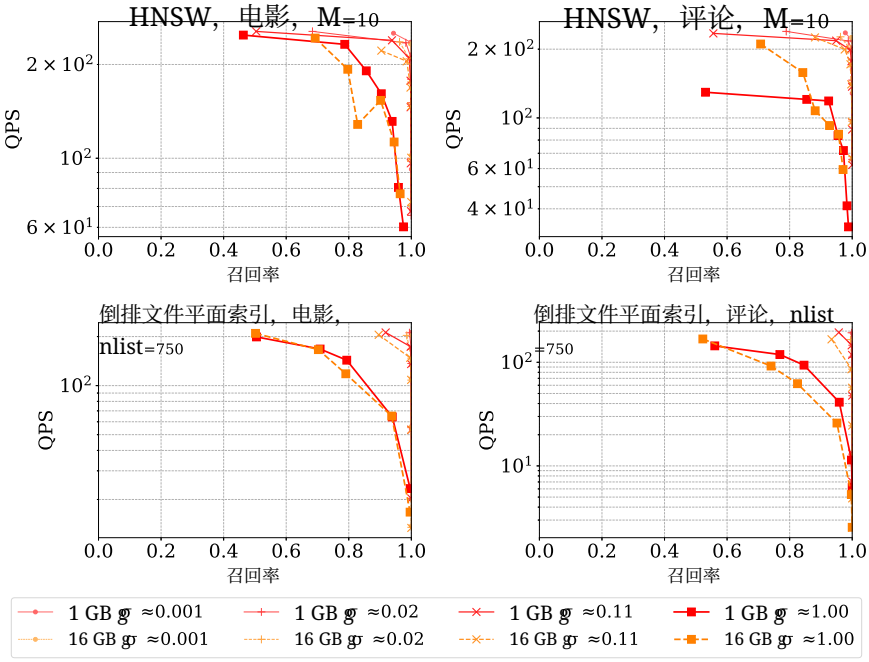


图13：Milvus 中不同分段大小的每秒查询数–召回率曲线。

这种自适应行为也适用于*IVFFlat* 索引，这解释了为何*Milvus* 即使在 $nprobe = 1$ 严格过滤器下也能达到 1.0 的召回率。在这种情况下，标准倒排索引会失效，因为少数匹配向量分散在尚未访问的聚类中；而 Knowhere 会检测到高选择率，并切换为对有效向量进行线性扫描，实际上将该查询视为小型数据问题。因此，无论底层索引类型如何，对于经过严格过滤的查询，*Milvus* 都会表现出“精确搜索”行为。

构建效率。索引构建的鲁棒性得益于批量并行构建流水线（使用 Folly 持久线程池），而非简单的分段并行化。Knowhere 还会执行构建后图修复步骤，重新连接在并发插入期间被隔离的向量，这也解释了为何其基础召回率高于原版 FAISS。

属性索引。 *Milvus* 的性能对是否存在属性索引不太敏感（图16）。这与*pgvector*形成鲜明对比，后者可能会根据配置更改查询执行计划。我们的分析证实，*Milvus* 并未采用传统关系型意义上的基于成本的查询优化器，而是依赖这种静态预过滤和逐段回退机制。

要点。 Milvus 通过复杂的“混合”引擎设计优先保证召回率稳定性。其性能特征是算法专门化混合设计的副产品：*Dual-Pool* 遍历可防止在中等强度过滤条件下出现搜索饥饿，而自适应暴力搜索回退则确保在 HNSW 和 IVFFlat 上使用严格过滤条件时的准确率。这种设计使系统的行为如同精确搜索引擎，能够针对选择率高且难以找到邻居的查询实现完美召回率。

7.2.8 PG-Vector：过滤近似最近邻搜索的查询优化

回退到 k近邻。 PostgreSQL 查询优化器旨在选择估计总成本最低的执行计划。为集成向量相似性搜索，*pgvector* 为每种受支持的向量索引类型引入了特定的成本函数。这些计算严重依赖基数估计——这是关系型数据库管理系统优化器中的标准做法——以预测近似最近邻搜索操作涉及的元组数量。整体成本函数取决于参数 k 以及应用选择谓词 S_σ后数据集的估计基数。我们观察到两种不同的查询计划：

• 近似最近邻搜索 +后过滤：在此计划中，首先执行向量索引搜索。对于 HNSW，这涉及填充一个大小为 ef_{search} 的队列；对于 IVFFlat，则涉及探测 n_p 个聚类。随后，对索引扫描返回的元组应用选择谓词。以下是一个使用 HNSW 索引、包含$r = 9.5$ 和 $k = 20$的查询的示例计划：

限制 -> 使用 vector_idx 扫描 dataset 按以下条件排序：
(vector <=> $0) 过滤器：(attribute > 8.5)

• 预过滤 + *k* 近邻搜索：此处首先通过顺序扫描应用选择谓词，然后计算查询向量与经过过滤的元组之间的距离，执行精确 k 近邻搜索。这种回退到暴力搜索的方式虽然能确保理论上完美的召回率（精确 k 近邻搜索），但对于较大的子集而言，计算效率可能较低。以下是使用 $r = 9.5$ 和 $k = 21$进行查询的执行计划：

限制 -> 排序 排序键：((dataset.vector <=> $0))" 排序方法：前
N 项堆排序 内存：XYZ kB" -> 对 dataset 进行顺序扫描 过滤器：
(attribute > 8.5)"

随后，我们在用于过滤的元数据属性上启用 B 树索引，并重复了该实验。在此配置下，查询优化器会获得一条额外的执行路径：

• **属性索引扫描 + k 近邻搜索**：此计划利用 averagerating 上的索引高效地过滤数据集，随后通过严格计算与查询向量之间的距离，找出 k 个最近邻。以下是使用 $r = 9.5$ 和 $k = 10$进行查询的执行计划：

限制 -> 排序 排序键：((dataset.vector <=> $0)) 排序方法：前 N 项堆排序 -> 位图堆
扫描 ondataset 重新检查条件：(attribute > 8.5) -> 位图索引扫描 on idx_attribute
索引条件：(attribute > 8.5::double precision)

如图17所示，属性索引扫描 +*k* 近邻搜索执行计划始终能达到完美召回率。然而，*pgvector* 的自动执行计划选择似乎并不理想；对于许多低选择性查询，查询优化器倾向于选择基于向量索引的向量搜索，而忽略顺序扫描执行计划。我们的数据表明，在某些情况下，强制采用属性索引扫描 + *k* 近邻搜索（或预过滤 + *k* 近邻搜索）执行计划，可以在延迟与所选近似执行计划相当的情况下实现完美召回率。

要点：在*pgvector* 中为元数据建立索引，会显著改变查询执行计划，但也暴露出优化器成本模型的局限性。具体而言，即使在先进行过滤、再精确计算距离能够以相近或更低的计算成本保证召回率的情况下，优化器仍常常优先选择近似向量索引搜索。这表明，高维向量运算与标量索引扫描之间的成本权重还需要进一步调整，以避免选择召回率不理想的执行计划。

7.3 总结

我们对多个开源系统进行了广泛的实证评估，包括 *FAISS*、*Milvus* 和 *pgvector*，由此得出若干关于过滤器选择性、索引拓扑与系统架构之间相互作用的重要见解：

选择性-效率权衡： 我们观察到，不同索引类型的表现各不相同。基于图的索引在选择率较低时，无论采用何种过滤策略，其每秒查询数-召回率曲线都会变差；而基于分区的索引（IVFFlat）在某些情况下则能显著提升效率。因此，在 FAISS 中，预过滤位集可以让搜索跳过对探测到的聚类中无效向量进行开销高昂的距离计算。尽管在固定的 n_{probe} 下，受候选池限制，绝对召回率可能会下降，但计算开销的减少会大幅提升吞吐量（每秒查询数）。因此，与选择性更高的情形相比，IVFFlat 的每秒查询数-召回率权衡曲线表现更优。

索引反转： 我们证明了，在低选择性场景中，HNSW 传统上的主导地位正受到挑战。在此类场景下，基于分区的索引（IVFFlat）凭借高效的聚类剪枝机制，能够提供更强的鲁棒性。

算法鲁棒性与延迟下限： 我们对 *Milvus*的分析表明，通过对图索引和基于分区的索引两者进行算法专门化，系统实现了召回率稳定性。系统采用自适应暴力搜索回退机制和“Dual-Pool”遍历，以使 HNSW 和 IVFFlat 都能实现高召回率。然而，这种鲁棒性实际上设定了一个“延迟下限”：为在严格过滤条件下维持精确搜索准确率，系统牺牲了峰值吞吐量。

优化器脆弱性： 我们揭示了*pgvector*基于成本的优化器存在一些局限性：当顺序扫描或属性索引扫描能够以有竞争力的延迟实现完美召回率时，优化器却常常无法切换到这些扫描方式。这凸显了关系型向量扩展需要更好的基数和成本估算模型。

查询-过滤器相关性：我们的研究结果表明，尽管平均而言元数据与向量分布的相关性较弱（平均 GLS ≈ 0），但单次查询性能对局部相关性变化高度敏感。通过 GLS 指标，我们证明，较高的 GLS 相关性可预测更高的召回率，因为这意味着有效邻居存在空间聚类。然而，数据集中的高方差和接近零的均值表明，尽管相关性能够解释单查询召回率波动，但其变化过于剧烈，不适合用于全局成本估算。因此，查询优化器可以可靠地利用全局选择性进行执行计划选择，但必须考虑到相关性较低的“困难”查询可能需要更激进的搜索参数，才能维持目标召回率水平。

7.4 过滤向量搜索实用指南

基于我们的实验结果，我们为部署过滤向量搜索系统的从业者提炼出以下可操作的推荐：

1. **根据预期过滤选择性选择索引类型**。不要假设 HNSW 普遍优于 IVFFlat。对于预期选择性为 $\sigma_g \lesssim 5\%$的工作负载，可考虑 IVFFlat；对于 $\sigma_g \gtrsim 20\%$或以未过滤查询为主的情况，HNSW 仍然更可取。对于混合工作负载，应同时维护两种索引类型，并在可能时按查询进行选择。

2. **为过滤条件下的召回率下降做好规划**。随着选择性降低，过滤近似最近邻搜索会固有地降低召回率——无论系统或索引如何。应将召回率损失视为一项纳入预算的资源。对于关键查询，应提供搜索参数覆盖，或在检测到候选项耗尽时允许回退到精确搜索。

3. **将搜索参数随 k 进行缩放**。由于候选池大小由搜索参数（efSearch、nprobe）固定，在不缩放这些参数的情况下增大 k 会系统性降低召回率。当召回率重要时，缩放 efSearch $\propto k$（或 nprobe $\propto k$）。

4. **为保证鲁棒性，需要接受吞吐量惩罚**。在过滤条件下保证高召回率的引擎（例如 *Milvus*）不会出现标准库中常见的“快速失败”行为。虽然这能确保准确率，但也无法达到允许召回率下降的系统所实现的超高每秒查询数（例如 *FAISS*、*pgvector*）。迁移到鲁棒型引擎的实践者应为这一延迟下限做好预算，因为系统不会返回速度快但质量低的结果。

5. **避免过度调优图密度**。 在过滤条件下，增加 HNSW 构建参数（M、efConstruction）的收益会逐渐递减：召回率仅略有提升，构建时间却显著增加。建议采用更稀疏的图，并将调优预算分配给搜索时参数。

6. **积极为元数据属性建立索引（pgvector）**。 在关系型向量扩展中，元数据索引可启用优化器能够利用的执行计划，以处理过滤查询。若没有这些索引，系统往往会采用降低召回率的后过滤策略。

7. **不要盲目信任查询优化器**。对于关键工作负载，请手动验证查询计划（EXPLAIN ANALYZE）。即使精确扫描可以在相近的延迟下实现完美召回率，优化器也可能优先选择近似索引扫描。对于选择性非常高的过滤器，可以考虑强制使用属性索引扫描或顺序扫描计划。

8. **使用全局选择性来选择执行计划，但要监控相关性**。由于元数据属性通常与嵌入空间的相关性较弱（平均 GLS ≈ 0），系统吞吐量主要由全局选择性（σ_g）决定。根据 σ_g 和 k选择主要计划：σ_g 较低时，优先采用精确搜索或预过滤搜索；σ_g较高时，ANN优先计划较为稳妥。不过请注意，GLS 相关性较低的查询（有效邻居被“推”离查询点）会导致召回率降低。对于召回率关键型应用，可考虑使用 GLS 作为诊断工具，识别这些“困难”的查询区域，并动态增加搜索深度（例如， ef_{search}）。

8 结论与未来工作

在本研究中，我们应对了 FANNS 中算法理论与系统级实现之间日益扩大的差距。首先，我们通过将基于图的索引和基于分区的索引的预过滤、后过滤和运行时过滤形式化为分类体系，系统梳理了 FANNS 中模糊不清的研究图景。其次，我们为处理基于谓词的查询的现代向量数据库建立了严格的评估框架。为此，我们引入了 MoReVec 这一新数据集，它包含两个关系：电影 和 评论。这些关系通过外键相连，并同时具备基于 Transformer 的嵌入与结构化元数据属性。最后，我们提出了全局-局部选择性（GLS）度量，用以量化元数据属性与向量嵌入之间的独立性——这对于将选择性的影响与语义聚类区分开来至关重要。

Li限性。 我们承认，有若干因素可能影响本研究发现的可推广性：

内部效度。 我们的结论对所探索的特定超参数范围（efSearch、nprobe、 M）较为敏感。不同的参数扫描可能在索引类型之间产生不同的交叉点。所有实验均在内存中进行，数据集完全加载到随机存取存储器中；在内存压力或基于磁盘的执行下，结果可能会有所不同。我们的评估沿用了 ANN-Benchmarks 的单线程执行模型，该模型隔离了算法行为，但无法捕捉生产环境并发效应。

外部效度。 结果可能无法推广到以下情况：(1) 其他嵌入模型（如 CLIP、E5、OpenAI 嵌入），这些模型会产生不同的向量分布；(2) 在未归一化向量上采用其他距离度量（内积、 L_2）；(3) 元数据高度相关的工作负载，其中过滤属性在语义上与嵌入空间相对应；以及 (4) 分布式或多节点部署，其中网络延迟和数据分区会引入额外变量。我们的结论主要适用于单节点向量数据库中的通用、模式无关过滤策略，而不适用于融合方法（如 ACORN、HQANN）或工作负载感知索引。

未来工作。 将向量搜索集成到关系系统中，仍是数据库研究的前沿领域。尽管 pgvector 等当前系统在语法层面支持混合查询，但其优化器往往缺乏足够的复杂性，无法在多种工作负载下维持性能。我们将开发适用于过滤近似最近邻搜索的选择性感知查询优化器确定为一个主要方向。此类优化器必须超越简单的启发式方法，执行以下操作：

- 最优索引与参数选择：在运行时根据特定的筛选约束和预算，动态选择基于图（例如 HNSW）或基于分区（例如倒排文件）的索引，并调整搜索参数（例如 ef_{search}、 n_{probe}）。

- 相关性感知估计：研究估计的广义最小二乘相关性，以预测通过过滤器筛选的邻居的“可达性”。这使引擎能够考虑向量空间中的属性分布聚类不均匀的情况，因为这会影响识别有效候选项所需的搜索开销。

- 复杂关系运算互操作：将优化逻辑扩展到连接 + 向量搜索场景中，此时无法立即进行属性过滤。在这种情况下，优化器不仅必须评估相互竞争的执行计划的成本效益，还必须估算其召回率：(1) 先连接，即先执行关系型

连接以缩小候选池，然后再进行向量搜索；或 (2) 近似最近邻搜索优先，即先执行向量搜索，获取一个预估为过大的候选集，随后在后续阶段执行连接，以处理剩余的关系约束。

此外，我们计划研究混合索引策略，将多种算法方法（例如图结构分区）整合到一个统一的设计中。在此框架下，实时选择率估计不会触发不同索引之间的切换，而是用于指导内部遍历逻辑和参数。此类索引旨在面对严格的筛选约束以及不断演变的关系型数据集所常见的分布偏移时，仍能保持稳健。

致谢

本研究得到美国国家科学基金会项目资助，奖项编号为 2008815。

参考文献

[1] C. C. Aggarwal, A. Hinneburg, 以及 D. A. Keim. 论距离度量在高维空间中的惊人表现. 载于国际数据库理论会议第*8*届会议论文集，ICDT ’01，第420–434页，柏林、海德堡，2001年。施普林格出版社。[2] ANN-Benchmarks：近似最近邻算法的基准测试工具。https://ann-benchmark s.com/。访问日期：2026年1月20日。[3] 用于过滤搜索的 ANN 基准测试扩展。https://github.com/aabylay/ANN-benchmark-HQ。访问日期：2026年1月20日。[4] 近似最近邻评测：大型数据集。https://corpus-texmex.irisa.fr/。访问日期：2026年1月20日。[5] Annoy：近似最近邻，没错。https://github.com/spotify/annoy。访问日期：2026年1月20日。[6] M. Aum¨uller、E. Bernhardsson 和 A. Faithfull。Ann-benchmarks：近似最近邻算法的基准测试工具。*CoRR*，arXiv:1807.05614，2018年。[7] J. L. Bentley。用于关联搜索的多维二叉搜索树。《*ACM* 通讯》，18(9):509–517，1975年9月。[8] T. B. Brown、B. Mann、N. Ryder、M. Subbiah、J. Kaplan、P. Dhariwal、A. Neelakantan、P. Shyam、G. Sastry、A. Askell、S. Agar- wal、A. Herbert-Voss、G. Krueger、T. Henighan、R. Child、A. Ramesh、D. M. Ziegler、J. Wu、C. Winter、C. Hesse、M. Chen、E. Sigler、M. Litwin、S. Gray、B. Chess、J. Clark、C. Berner、S. McCandlish、A. Radford、I. Sutskever 和 D. Amodei。语言模型是少样本学习器。载于第*34*届神经信息处理系统国际会议论文集，NIPS ’20，美国纽约州雷德胡克，2020年。Curran Associates Inc.[9] E. Ch´avez、G. Navarro、R. Baeza-Yates 和 J. L. Marroqu´ın。在度量空间中搜索。《*ACM*计算综述》，33(3):273–321，2001年9月。[10] P. Covington、J. Adams 和 E. Sargin。用于 YouTube 推荐的深度神经网络。载于第*10*届美国计算机协会推荐系统会议论文集，RecSys ’16，第191–198页，美国纽约，2016年。计算机协会。[11] J. Devlin、M.-W. Chang、K. Lee 和 K. Toutanova。BERT：用于语言理解的深度双向 Transformer 预训练。*CoRR*，arXiv:1810.04805，2019年。[12] M. Douze、A. Guzhva、C. Deng、J. Johnson、G. Szilvasy、P.-E. Mazar´e、M. Lomeli、L. Hosseini 和 H. J´egou。FAISS 库。《*IEEE* 大数据汇刊》，第1–17页，2025年。[13] Elasticsearch：免费、开源、分布式的 RESTful 搜索引擎。https://github.com/elastic/elast icsearch。访问日期：2026年1月20日。

[14] FAISS：高效相似性搜索库。https://github.com/facebookresearch/faiss。访问日期：2026年1月20日。[15] S. Gollapudi、N. Karia、V. Sivashankar、R. Krishnaswamy、N. Begwani、S. Raz、Y. Lin、Y. Zhang、N. Mahapatro、P. Srinivasan、A. Singh 和 H. V. Simhadri。Filtered-DiskANN：带过滤器的近似最近邻搜索图算法。载于2023年ACM Web会议论文集，WWW '23，第3406–3416页，美国纽约，2023年。计算机协会。[16] R. Guo、P. Sun、E. Lindgren、Q. Geng、D. Simcha、F. Chern 和 S. Kumar。利用各向异性向量量化加速大规模推理。载于第37届国际机器学习会议论文集，ICML'20。JMLR.org，2020年。[17] Y. Han、C. Liu 和 P. Wang。向量数据库综述：存储与检索技术及挑战。CoRR，arXiv:2310.11703，2023年。[18] 用于快速近似最近邻搜索的仅头文件 C++/Python 库。https://github.com/nmslib/hnswlib。访问日期：2026年2月9日。[19] P. Iff、P. Bruegger、M. Chrapek、M. Besta 和 T. Hoefler。基于 Transformer 的嵌入向量上的过滤近似最近邻搜索算法基准测试。CoRR，arXiv:2507.21989，2025年。[20] O. Jafari、P. Maurya、P. Nagarkar、K. M. Islam 和 C. Crushev。局部敏感哈希算法及其应用综述。CoRR，arXiv:2102.08942，2021年。[21] S. Jayaram Subramanya、F. Devvrit、H. V. Simhadri、R. Krishnawamy 和 R. Kadekodi。DiskANN：单节点上的快速、精确十亿级最近邻搜索。载于 H. Wallach、H. Larochelle、A. Beygelzimer、F. d'Alch´e-Buc、E. Fox 和 R. Garnett 编，《神经信息处理系统进展》，第32卷。Curran Associates, Inc.，2019年。[22] Z. Ji、N. Lee、R. Frieske、T. Yu、D. Su、Y. Xu、E. Ishii、Y. J. Bang、A. Madotto 和 P. Fung。自然语言生成中的幻觉综述。《ACM计算综述》，55(12)，2023年3月。[23] C. Jin、Y. Zhang、J. Liu 和 J. Wang。PostgreSQL 中的快速向量搜索：一种解耦方法。载于创新数据系统研究会议（CIDR），2026年。[24] Y. Jin、Y. Wu、W. Hu、B. M. Maggs、J. Yang、X. Zhang 和 D. Zhuo。Curator：利用低选择性过滤器实现高效向量搜索。CoRR，arXiv:2601.01291，2026年。[25] Z. Jing、Y. Su、Y. Han、B. Yuan、H. Xu、C. Liu、K. Chen 和 M. Zhang。当大语言模型遇上向量数据库：综述。CoRR，arXiv:2402.01763，2024年。[26] H. J´egou、M. Douze 和 C. Schmid。用于最近邻搜索的乘积量化。《IEEE模式分析与机器智能汇刊》，33(1):117–128，2011年。[27] V. Karpukhin、B. Oguz、S. Min、P. Lewis、L. Wu、S. Edunov、D. Chen 和 W.-t. Yih。用于开放领域问答的稠密段落检索。载于 B. Webber、T. Cohn、Y. He 和 Y. Liu 编，《2020年自然语言处理经验方法会议论文集》（EMNLP），第6769–6781页，线上，2020年11月。计算语言学协会。[28] V. Leis、A. Gubichev、A. Mirchev、P. Boncz、A. Kemper 和 T. Neumann。查询优化器究竟有多好？《VLDB学报》，9(3):204–215，2015年11月。[29] P. Lewis、E. Perez、A. Piktus、F. Petroni、V. Karpukhin、N. Goyal、H. K¨uttler、M. Lewis、W. tau Yih、T. Rockt¨aschel、S. Riedel 和 D. Kiela。面向知识密集型自然语言处理任务的检索增强生成。CoRR，arXiv:2005.11401，2021年。[30] J. Li、H. Liu、C. Gui、J. Chen、Z. Ni、N. Wang 和 Y. Chen。京东电商平台实时视觉搜索系统的设计与实现。载于第19届国际中间件会议（工业界）论文集，Middleware '18，第9–16页，美国纽约，2018年。计算机协会。[31] M. Li、X. Yan、B. Lu、Y. Zhang、J. Cheng 和 C. Ma。近似最近邻搜索中的属性过滤：深入实验研究。CoRR，arXiv:2508.16263，2025年。[32] W. Li、Y. Zhang、H. Wang、K. Lu 和 M. Kudo。高维数据上的近似最近邻搜索：实验、分析与改进。《IEEE知识与数据工程汇刊》，32(8):1475–1488，2020年。

[33] Z. Li、X. Zhang、Y. Zhang、D. Long、P. Xie 和 M. Zhang。通过多阶段对比学习实现通用文本嵌入。*CoRR*，arXiv:2308.03281，2023年。[34] Y. Lin、K. Zhang、Z. He、Y. Jing 和 X. S. Wang。向量-标量混合数据上的过滤近似最近邻搜索综述。*CoRR*，arXiv:2505.06501，2025年。[35] Y. A. Malkov 和 D. A. Yashunin。使用分层可导航小世界图实现高效且稳健的近似最近邻搜索。*CoRR*，arXiv:1603.09320，2018年。[36] M. D. Manohar、Z. Shen、G. E. Blelloch、L. Dhulipala、Y. Gu、H. V. Simhadri 和 Y. Sun。ParlayANN：可扩展且确定性的基于并行图的近似最近邻搜索算法。*CoRR*，arXiv:2305.04359，2024年。[37] T. Mikolov、I. Sutskever、K. Chen、G. Corrado 和 J. Dean。词语和短语的分布式表示及其组合性。载于神经信息处理系统国际会议第*27*届会议论文集——第*2*卷，NIPS'13，第3111–3119页，纽约州雷德胡克，美国，2013年。Curran Associates Inc。[38] Milvus：云原生向量数据库。https://milvus.io/。访问日期：2026年1月20日。[39] 大规模文本嵌入基准排行榜。https://huggingface.co/spaces/mteb/leaderboard。访问日期：2026年1月20日。[40] N. Muennighoff、N. Tazi、L. Magne 和 N. Reimers。MTEB：大规模文本嵌入基准。载于 A. Vlachos 和 I. Augenstein 编，计算语言学协会欧洲分会第*17*届会议论文集，第2014–2037页，克罗地亚杜布罗夫尼克，2023年5月。计算语言学协会。[41] M. Muja 和 D. G. Lowe。面向高维数据的可扩展最近邻算法。*IEEE*模式分析与机器智能汇刊，36(11)：2227–2240，2014年。[42] OpenAI。GPT-4技术报告。*CoRR*，arXiv:2303.08774，2023年。[43] J. J. Pan、J. Wang 和 G. Li。向量数据库管理系统综述。《*VLDB*期刊》，33(5)：1591–1615，2024年7月。[44] L. Patel、S. Jha、M. Pan、H. Gupta、P. Asawa、C. Guestrin 和 M. Zaharia。语义算子：用于丰富的基于人工智能的数据处理的声明式模型。*CoRR*，arXiv:2407.11418，2025年。[45] L. Patel、P. Kraft、C. Guestrin 和 M. Zaharia。ACORN：针对向量嵌入和结构化数据的高性能、谓词无关搜索。*CoRR*，arXiv:2403.04871，2024年。[46] pgvector：用于 PostgreSQL 的开源向量相似性搜索。https://github.com/pgvector/pgvector。访问日期：2026年1月20日。[47] 借助 Amazon Aurora PostgreSQL 上的 pgvector 0.8.0，大幅提升向量搜索的性能和相关性。https: // aws.amazon.com/blogs/database/supercharging-vector-search-performance-and-relev ance-with-pgvector-0-8-0-on-amazon-aurora-postgresql/。访问日期：2026年1月20日。[48] Pinecone：面向机器学习的向量数据库。https://www.pinecone.io/。访问日期：2026年1月20日。[49] Qdrant：向量相似性搜索引擎和向量数据库。https://github.com/qdrant/qdrant。访问日期：2026年1月20日。[50] S. E. Robertson 和 S. Walker。对概率加权检索的二泊松模型的一些简单有效近似。载于第*17*届 *ACM* 信息检索研究与发展国际会议论文集，SIGIR '94，第232–241页，柏林、海德堡，1994年。施普林格出版社。[51] J. Shi、Y. Cai 和 W. Zheng。过滤近似最近邻搜索：统一基准测试与系统性实验研究 [实验、分析与基准测试]。*CoRR*，arXiv:2509.07789，2025年。[52] H. V. Simhadri、M. Aum¨uller、A. Ingber、M. Douze、G. Williams、M. D. Manohar、D. Baranchuk、E. Liberty、F. Liu、B. Lan-drum、M. Karjikar、L. Dhulipala、M. Chen、Y. Chen、R. Ma、K. Zhang、Y. Cai、J. Shi、Y. Chen、W. Zheng、Z. Wan、J. Yin 和 B. Huang。Big ANN：NeurIPS'23竞赛结果。*CoRR*，arXiv:2409.17424，2024年。

[53] Sivic 和 Zisserman。Video Google：一种用于视频中对象匹配的文本检索方法。载于第九届 *IEEE* 国际计算机视觉会议论文集，第1470–1477页，第2卷，2003年。[54] N. Thakur、N. Reimers、A. R¨uckl´e、A. Srivastava 和 I. Gurevych。BEIR：用于零样本评估信息检索模型的异构基准测试。载于 J. Vanschoren 和 S. Yeung 编，神经信息处理系统数据集与基准专题论文集，第1卷，2021年。[55] 向量数据库基准测试。https://zilliz.com/vdbbench-leaderboard。访问日期：2026年1月20日。[56] Vespa：开放式大数据服务引擎。https://vespa.ai/。访问日期：2026年1月20日。[57] 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 和 C. Xie。Milvus：专为向量数据管理打造的系统。载于*2021*年国际数据管理会议论文集，SIGMOD '21，第2614–2627页，纽约，美国，2021年。计算机协会。[58] M. Wang、L. Lv、X. Xu、Y. Wang、Q. Yue 和 J. Ni。由可导航邻近图驱动的原生混合查询，支持结构化和非结构化约束。*CoRR*，arXiv:2203.13601，2022年。[59] Weaviate：开源向量数据库。https://github.com/weaviate。访问日期：2026年1月20日。[60] C. Wei、B. Wu、S. Wang、R. Lou、C. Zhan、F. Li 和 Y. Cai。AnalyticDB-V：面向结构化和非结构化数据查询融合的混合分析引擎。《*VLDB*会议论文集》，13(12)：3152–3165，2020年8月。[61] W. Wu、J. He、Y. Qiao、G. Fu、L. Liu 和 J. Yu。HQANN：针对结构化和非结构化约束混合查询的高效稳健相似性搜索。*CoRR*，arXiv:2207.07940，2022年。[62] W. Xia、M. Yang、W. Li 和 W. Wang。过滤近似最近邻搜索的成本估算。*CoRR*，arXiv:2602.06721，2026年。[63] W. Yang、T. Li、G. Fang 和 H. Wei。PASE：PostgreSQL 超高维近似最近邻搜索扩展。载于*2020*年 *ACM SIGMOD* 国际数据管理会议论文集，SIGMOD '20，第2241–2253页，纽约，美国，2020年。计算机协会。[64] Q. Zhang、S. Xu、Q. Chen、G. Sui、J. Xie、Z. Cai、Y. Chen、Y. He、Y. Yang、F. Yang、M. Yang 和 L. Zhou。VBASE：通过松弛单调性统一在线向量相似性搜索与关系查询。载于第*17*届 *USENIX* 操作系统设计与实现研讨会（*OSDI 23*），第377–395页，波士顿，美国，2023年7月。USENIX 协会。[65] Y. Zhang、M. Li、D. Long、X. Zhang、H. Lin、B. Yang、P. Xie、A. Yang、D. Liu、J. Lin、F. Huang 和 J. Zhou。Qwen3 嵌入：借助基础模型推进文本嵌入和重排序。*CoRR*，arXiv:2506.05176，2025年。[66] Y. Zhang、S. Liu 和 J. Wang。在关系数据库中支持向量数据管理是否存在根本限制？以 PostgreSQL 为例的案例研究。载于*2024*年 *IEEE* 第*40*届国际数据工程会议（*ICDE*），第3640–3653页，2024年。

A 补充实验

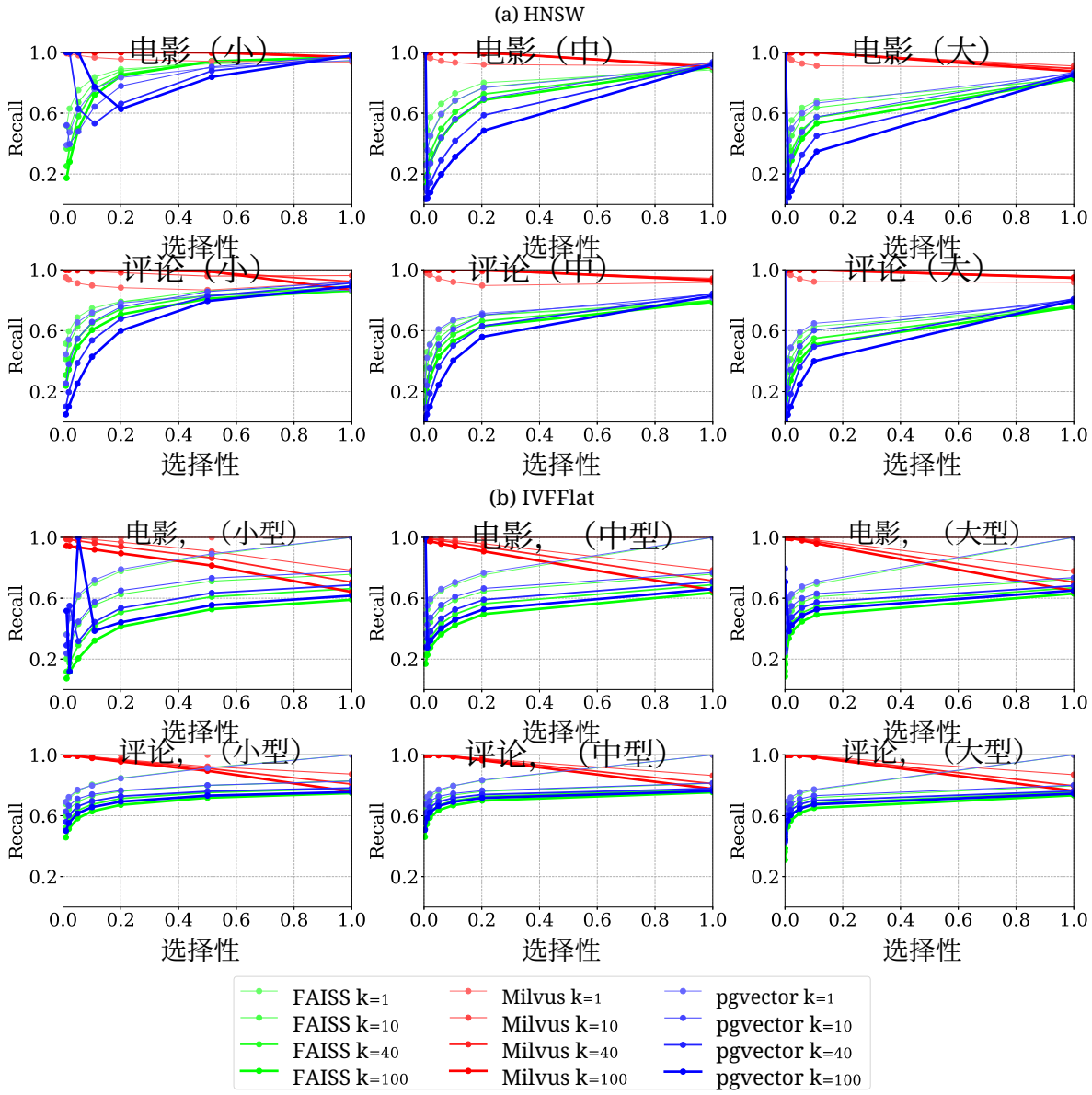


图 14：不同 k 值下的召回率–选择性曲线。

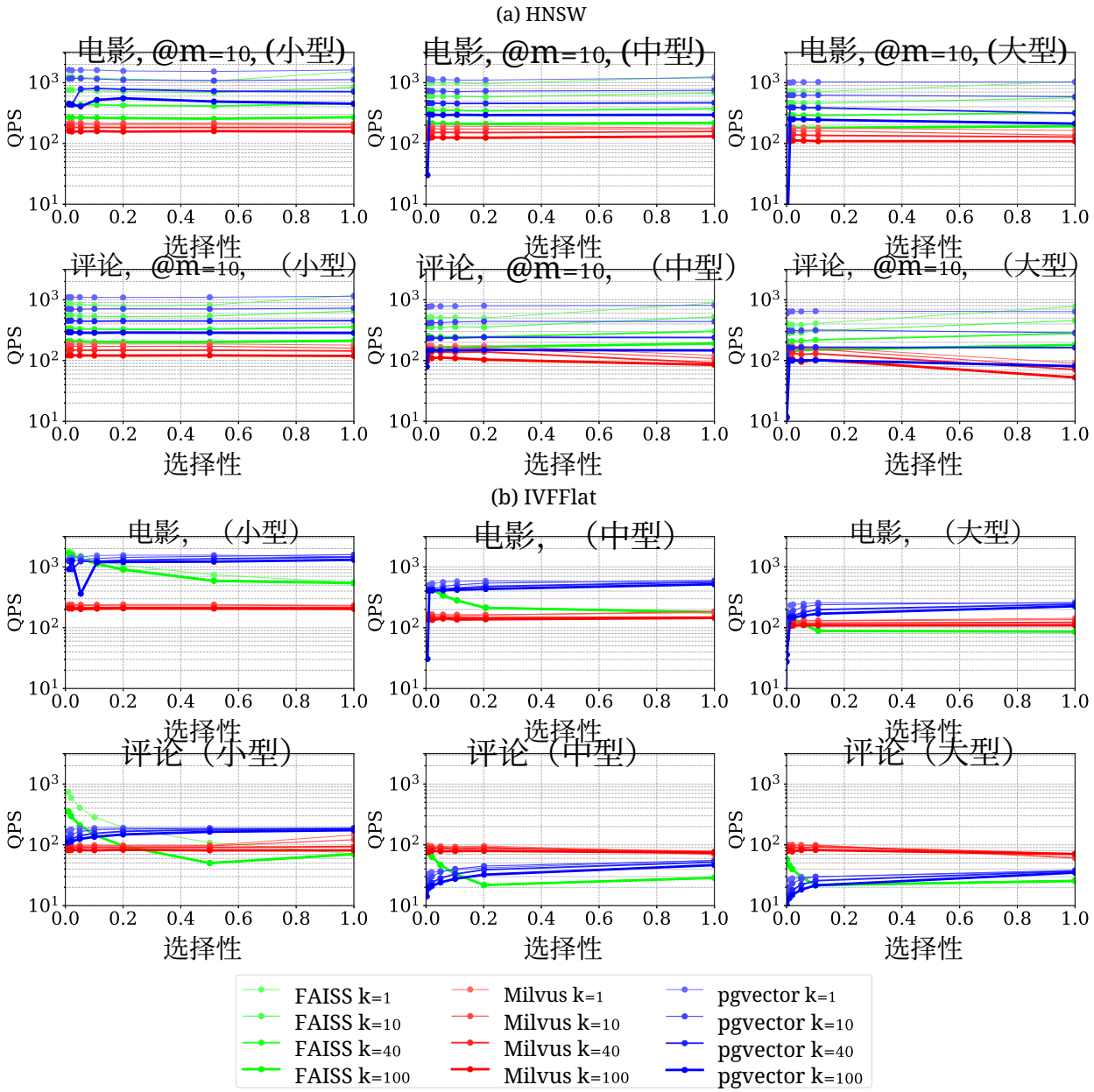


图15：不同 k 值下的 QPS–选择性曲线。

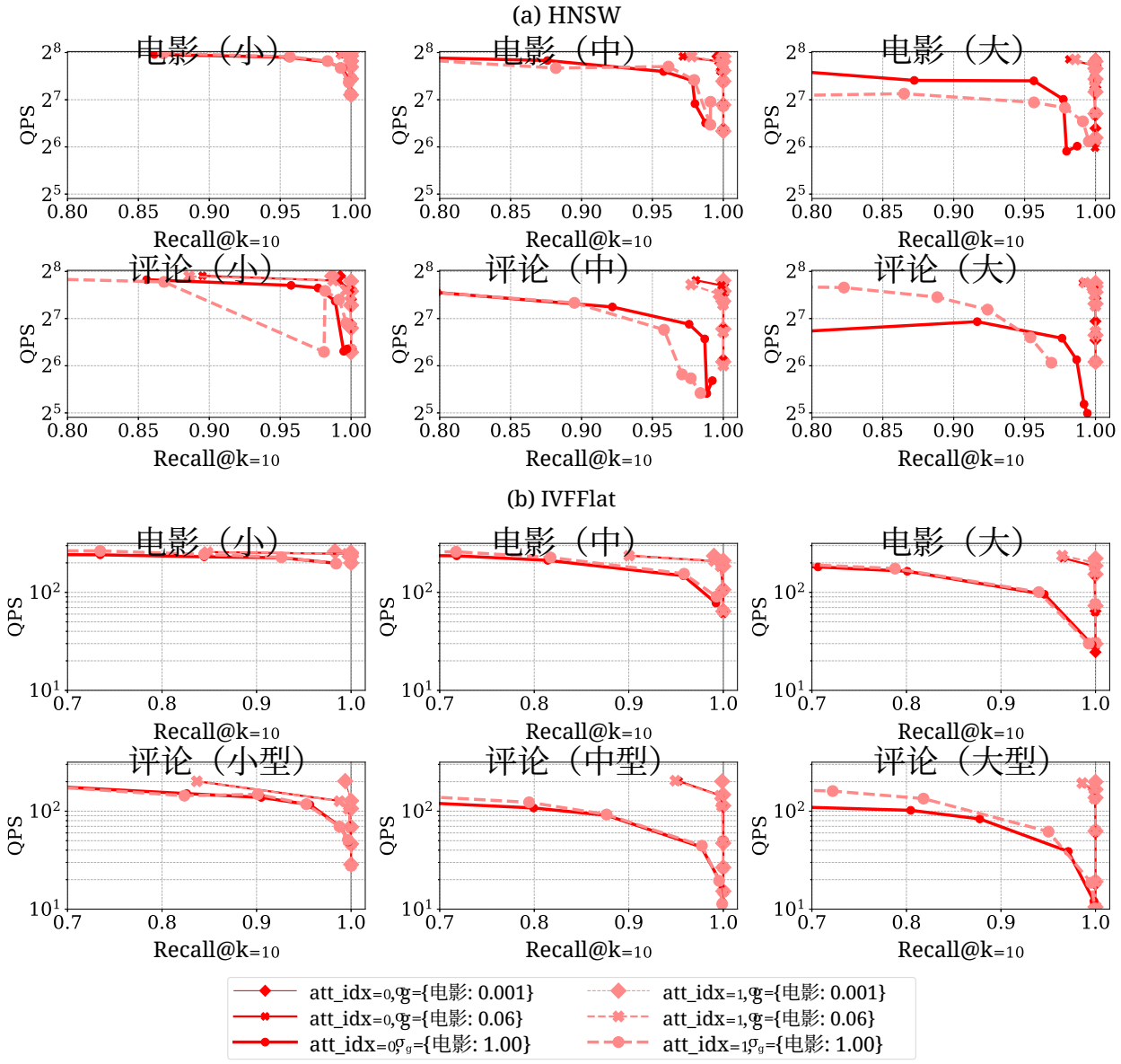


图16：*Milvus* 中 HNSW 和 IVFFlat 在筛选属性上建立索引与未建立索引时的 QPS–召回率曲线。

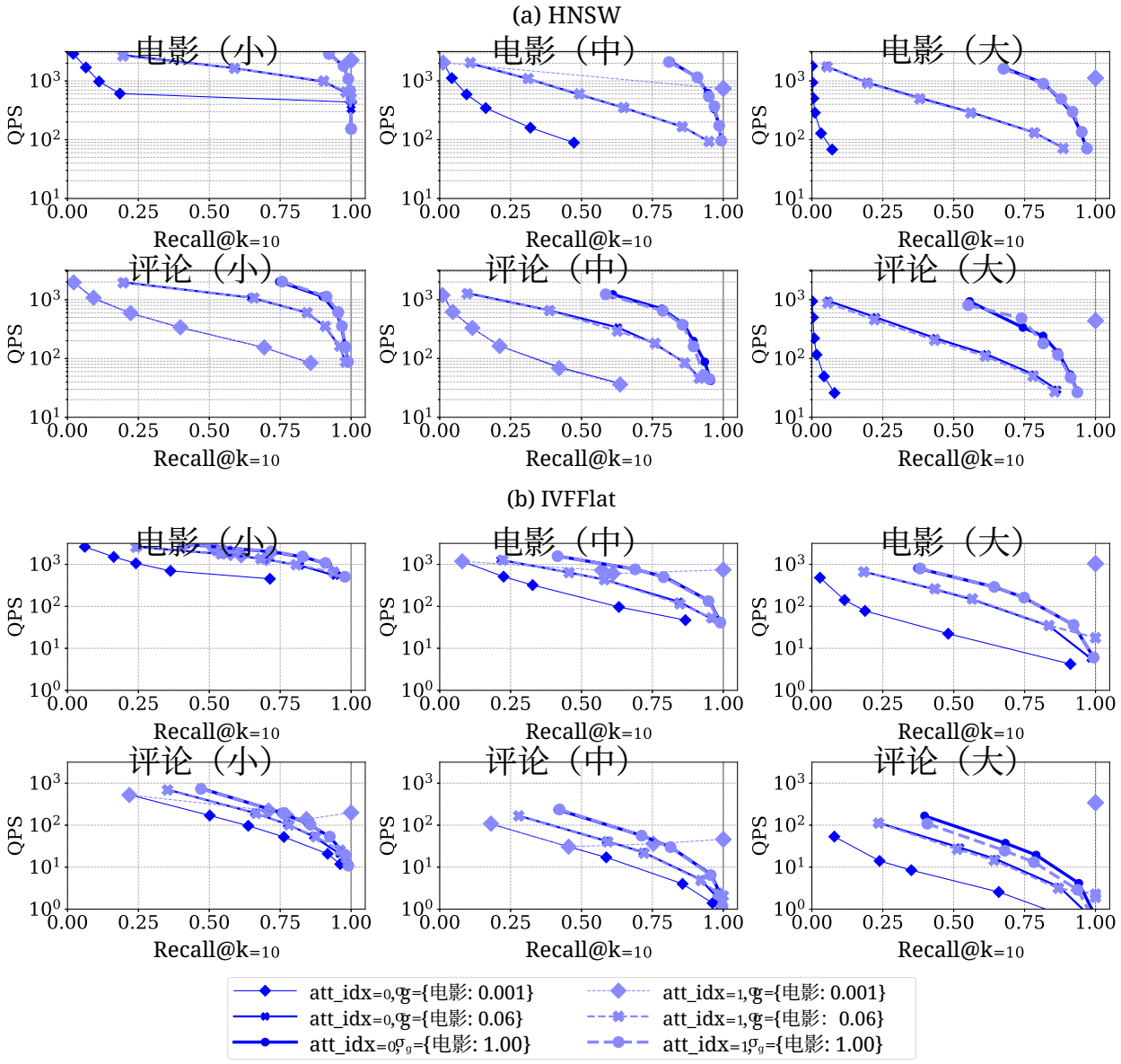


图17：*pgvector* 中，针对 HNSW 和 IVFFlat，筛选属性有无索引时的每秒查询数–召回率曲线。