Advances of Query Processing in Vector Databases

Jiadong Xie
The Chinese University of Hong Kong
Hong Kong SAR, China
jdxie@se.cuhk.edu.hk

Yingfan Liu
Xidian University
Xi'an, China
liuyingfan@xidian.edu.cn

Jeffrey Xu Yu
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China
jeffreyxuyu@hkust-gz.edu.cn

ABSTRACT

Due to the rapid advancements in AI technology, particularly in deep learning and large language models, vectors are believed to be the universal data representation of the future, connecting various modalities and domains. This underscores the necessity of vector databases to manage these vectors effectively and efficiently support different query types on vectors. This tutorial covers recent advances and challenges in query processing of vector databases. We will begin with an introduction to the background and motivation behind the emergence of vector databases. Next, we will review the techniques that focus on various query types: similarity search, multi-similarity search, filtered similarity search, and similarity join. Lastly, we will provide open challenges and future research directions, intending to foster innovation in the field.

PVLDB Reference Format:
Jiadong Xie, Yingfan Liu, and Jeffrey Xu Yu. Advances of Query Processing in Vector Databases. PVLDB, 19(12): 4893 - 4899, 2026.
doi:10.14778/3827998.3828146

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://vdb-query-processing.github.io/.

1 INTRODUCTION

With the rapid advancement of AI technology, particularly in deep learning and large language models, unstructured data is increasingly being represented by high-dimensional vectors. This is because vectors can better encapsulate semantic meaning in applications such as image recognition and beyond. Vectors are believed to be the future universal data representation, capable of connecting different modalities and domains and becoming the foundation of cross-domain innovations. Vectors currently drive numerous AI applications, such as Retrieval Augmented Generation, and OpenSearch in Amazon and Alibaba. To handle the expanding volume of large-scale vector datasets effectively, recent researches propose maintaining them in vector databases, and their focus extends to studying query processing in vector databases, including queries like similarity search, filtered similarity search, and similarity joins. This tutorial aims to systematically review these recent advances in query processing within vector databases.

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828146

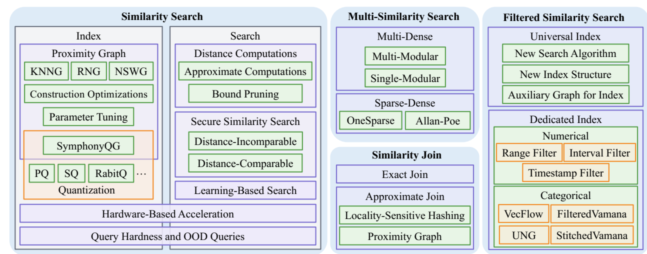

Figure 1: Overview of Recent Advances of Query Processing

Overview: As outlined in Fig. 1, this tutorial provides a comprehensive overview of recent advances and open challenges in query processing for vector databases. It is structured into six parts and designed for two 1.5-hour sessions (3 hours total): the first session covers parts (1) and (2), and the second session covers parts (3)–(6).
(1) *Background and Motivation (15 minutes).* The first part, which will last 15 minutes, introduces the background and motivations behind the vector databases. We will discuss the emerging unified data representation and the essentiality of vector databases.
(2) *Similarity Search (75 minutes).* The second part, spanning 75 minutes, introduces techniques for similarity search. We will first present the problem definition (2 minutes) and the methods based on proximity graphs (20 minutes) and quantizations (10 minutes). Next, we will introduce approaches of index maintenance (10 minutes), distance computations (8 minutes), hardware-based acceleration (8 minutes), handling out-of-distribution queries (10 minutes), and secure similarity search (7 minutes).
(3) *Multi-Similarity Search (20 minutes).* The third part, lasting 20 minutes long, will first present the problem definition (2 minutes), and then review the techniques for multi-dense (12 minutes) and sparse-dense similarity search (6 minutes).
(4) *Filtered Similarity Search (40 minutes).* The fourth part, requiring 40 minutes, will discuss the universal index (20 minutes) and dedicated index (20 minutes) for filtered similarity search, where dedicated indexes will be discussed based on specific filter types.
(5) *Similarity Join (20 minutes).* The fifth part, a 20-minute session on similarity join, will introduce problem definition and delve into both exact and approximate similarity join techniques.
(6) *Open Challenges and Discussions (10 minutes).* The final part, concluding in 10 minutes, summarizes the open challenges and future research directions in this rapidly evolving field.

Target Audience & Assumed Background: This tutorial is intended for researchers with a keen interest in vector databases. The tutorial does not require prerequisites beyond a basic understanding of databases. By the end of this tutorial, participants will have gained insights into the state-of-the-art techniques for processing different query types in vector databases, equipping them to explore the cutting-edge research and applications in this domain.

Related Tutorials: Recent tutorials have provided in-depth coverage of *filtered vector search* [13], and another surveys vector search from a *hardware-architecture* perspective, tracing the evolution from memory-resident designs to heterogeneous memory and cloud-native multi-tiered systems [74]. In contrast, these two tutorials are one component of our broader tutorial, and our emphasis centers on query-processing algorithms and their optimization trade-offs.

2 TUTORIAL OUTLINE

This is a *survey-style* tutorial, where each part follows a consistent pattern: (1) the problem definitions of the query; and (2) the representative state-of-the-art methods for processing these queries. In this section, we outline the tutorial contents and structure.

2.1 Background and Motivation

Universal Data Representation: With the rapid advancement of AI technology, especially deep learning-based embedding models, an increasing amount of unstructured data is now represented as high-dimensional vectors. This is because vectors can better capture semantic meaning across various applications, such as images and text. As a result, vectors are increasingly viewed as a universal representation that bridges diverse modalities and domains, and may serve as a foundation for cross-domain applications.

The Essentiality of Vector Databases: Vectors already underpin many modern applications, including retrieval-augmented generation (RAG) for large language models (LLMs) such as ChatGPT. Although LLMs perform well in general settings, they can fail in domain-specific or time-sensitive scenarios because they do not inherently access up-to-date external knowledge; when relevant information is missing, their outputs may deviate from facts. Vector databases address this gap by storing curated domain or real-time knowledge as embeddings and enabling efficient similarity search at query time. Given a user query, the system retrieves the most relevant items from the vector database and incorporates them into the prompt, producing a better-grounded input for the LLM.

Consequently, these emerging applications make vector databases essential for managing and retrieving vector data effectively and efficiently across a variety of query types.

2.2 Similarity Search

2.2.1 Problem Definition. Similarity search in vector databases is formulated as an approximate nearest neighbor (ANN) search. Given a dataset D of n vectors in $\mathbb{R}^d$ and a query vector $q \in \mathbb{R}^d$, nearest neighbor (NN) search aims to find the vector $p \in D$ closest to q. However, exact NN search is costly and often requires scanning nearly the entire dataset due to the curse of dimensionality [35]. Therefore, similarity search usually targets k-approximate nearest neighbor (k-ANN) search, which returns k vectors $p_1, p_2, \cdots, p_k \in D$ that are sufficiently close to q.

2.2.2 Techniques for Similarity Search. For processing the k-ANN search in memory, the state-of-the-art approaches fall into two categories: proximity graph-based and quantization-based methods.
Proximity Graph-Based Approaches: A proximity graph (PG) $G = (V, E)$ models a dataset D as a directed graph where each node corresponds to a vector and edges connect nearby vectors; despite different edge-selection rules, most PG indices share a common k-ANN query procedure, beam search, which maintains a w-size, beam width, candidate pool and iteratively expands the closest unexpanded node to refine the pool. Existing PGs are often grouped by edge-selection strategies into (i) <u>KNNG</u>, which connects each node to its k nearest neighbors and is motivated as an approximation to the Delaunay graph (DG) [45], with KGraph [15] being a strong constructor via iterative 2-hop candidate refinement; however, KNNG can be more prone to local optima and weaker connectivity, leading to inferior search performance in practice [49, 79]. (ii) <u>RNG</u> graphs prune KNNG edges to keep a small constant out-degree and typically add extra edges to improve connectivity; many such methods lack a theoretical 1-NN guarantee (e.g., NSG [19], SSG [18], DPG [49]), while others provide guarantees only under constraints, such as MRNG [19] and Vamana [39] when the query exists in D, and FANNG [32], τ-MG [66], and α-CG [46] when the 1-NN lies within a preset radius τ; these lines further yield approximation guarantees (e.g., Vamana [39] and α-CG [46]) and culminate in LMG [89] for unconstrained 1-NN/k-NN guarantees, though such guarantees typically incur high construction cost and motivate practical variants without guarantees such as DiskANN [39], τ-MNG [66], α-CNG [46], and ALMG [89]. (iii) <u>NSWG</u> indices are inspired by small-world navigability [61] and build graphs via incremental insertion (e.g., NSW [57], HNSW [58], LSH-APG [109]), with HNSW [58] widely regarded as a state-of-the-art baseline and FastHNSW/FastPG [97] improving both search and construction via layer-wise designs and per-layer RNG construction. Beyond these canonical families, several works target <u>better search performance</u> and <u>lower build cost</u>: CSPG [94] partitions the dataset with controlled overlaps to reduce exploration during search; learning-based and adaptive early-termination techniques can deliver target declarative recall [9, 47], while PGTuner [16] automates hyperparameter tuning via pre-training/transfer; delayed-synchronization traversal adapts beam search to utilize multi-thread computation resources and reduce latency [42]. On the construction side, DiskANN [39], RNN-Descent [63], LSH-APG [109], and ParlayANN [59] accelerate building, but often trading off search quality, e.g., batch insertion in ParlayANN [59], whereas FastPG [97] proposed to speed up RNG/NSWG construction without sacrificing index quality.
Quantization-Based Approaches: Quantization compresses each vector into short codes (e.g., 128/256 bits) to cut memory and accelerate approximate distance evaluation, typically with reranking: retrieve $L(> k)$ candidates by code-distance, then compute exact distances for the final top-k. PQ [40] and its variants [4, 25, 62] discretize subspaces via (rotated) K-Means-style codebooks; RaBitQ [24] offers strong accuracy with a sharp error bound and RaBitQ+ [22] extends it with longer codes, while SymphonyQG [27] integrates quantization with PG traversal to better match fast scanning primitives and reduce explicit reranking [1].
Index Maintenance: Indexes must support insertions and deletions. NSWG indices are insertion-friendly [57, 58], but deletions are often handled via bitmap-based virtual deletes and periodic rebuild/merge in systems such as ADBV [84, 103] and Milvus [75], with rebuilds accelerated using similarity join [88]. In-place maintenance aims to avoid global rebuilds: FreshDiskANN [71] repairs neighbors after deletions via candidate relinking/pruning, Wolverine++ [53] expands to filtered 2-hop candidates, and SPFresh [92] maintains cluster-based indexes via merge/split under updates.

Accelerating Distance Computations: Since distance evaluations dominate latency in ANN search, acceleration either approximates distances (e.g., quantization [31, 40], hashing [76]) or reduces exact computations via cheap lower bounds (e.g., ADSampling [23], BSA [96], TRIM [73]) that prune candidates whose bound already exceeds the current top-k threshold.

Hardware-Based Acceleration: Hardware-aware designs exploit SSD/remote memory/SIMD/GPU: DiskANN [39] and Starling [78] optimize disk-resident graph search with memory-resident codes, while second-tier memory via RDMA/CXL is proposed to address SSD index-storage limits [12] and realized via CXL disaggregation/co-design [38]. To mitigate disk compute, i.e., I/O coupling, dynamic traversal control is used [30]; SIMD quantization speeds construction [77]; GPU work adapts PG traversal for throughput (SONG [108]), accelerates RNG construction [50], or builds GPU-native graph pipelines (GGNN [28], GANNS [101], CAGRA [64]).

Query Hardness and Out-of-Distribution (OOD) Queries: Hardness measures include LID [2] and PG-specific metrics such as Steiner-hardness [82] and Escape Hardness [34], which characterize how traversal paths relate to the query's k-NN region. OOD/high-hardness robustness is improved by incorporating OOD queries during construction [10, 37] (either indexing them [37] or using them then excluding [10]) or by search-time rectification [34]. Beyond query-side OOD, embedding-model shifts can also induce distribution mismatch; recent work leverages local isometries to support cross-model top-k search without re-embedding, mitigating model-mismatch-induced hardness [93].

Secure Similarity Search: Secure k-ANN search outsources encrypted vectors plus an auxiliary index while protecting privacy [20, 21, 69, 85, 111, 112]. Distance-incomparable ciphers (AES/DES) [72] force client-side distance checks after downloading many candidates, whereas distance-comparable encryption enables cloud comparisons but varies in cost/privacy: homomorphic [29, 110] and matrix-based [111] are expensive, scalar-product-preserving schemes [85, 107, 113] can leak [55], approximate comparison trades accuracy [20], and recent DCE supports exact comparisons [55].

2.3 Multi-Similarity Search

2.3.1 Problem Definition. Given a dataset $O = \{o_1, o_2, \cdots, o_n\}$, where each object $o_i = (v_{i,1}, v_{i,2}, \cdots, v_{i,m})$ contains multiple vectors. Note that the dimensions of vectors $v_{i,1}, v_{i,2}, \cdots, v_{i,m}$ may not always be the same; they could be generated by different embedding models. A multi-similarity search is specified as $q = (q_1, q_2, \cdots, q_m)$, which also contains multiple vectors, and the distance is defined as the aggregated functions among each vector pair $(v_{i,j}, q_j)$ for $j \in \{1, 2, \cdots, m\}$, e.g., $dist(o_i, q) = \sum_{j=1}^{m} \delta_i(v_{i,j}, q_j)$, where $\delta_i(\cdot, \cdot)$ denote the distance metrics in i-th vector space.

2.3.2 Techniques for Multi-Dense Vector Similarity Search. Multi-dense similarity search aggregates distances from multiple dense embeddings, but the top-k under an aggregate metric may not appear in any single-vector top-k list. Milvus [75] addresses this by iteratively expanding per-vector candidate sets: it retrieves top-k' for each vector, aggregates over the mk' candidates, and doubles k' until the final top-k stabilizes across iterations. VBASE [106] reduces such iteration overhead via a two-phase strategy: first finds a 1-ANN and then incrementally expands candidates by indices, enabling a coordinated per-vector process that progressively returns results without repeatedly restarting with larger k'. DEG [98] builds a dedicated index and, at query time, extracts an α-specific searchable subgraph using Pareto frontier search to efficiently support the special case of two vectors ($m = 2$) with a weighted aggregate distance. RadiusSearch [87] focuses on the setting where the query's multi-vectors are generated by the same module, and proposes a two-phase search strategy to process such queries efficiently.

2.3.3 Techniques for Dense and Sparse Combined Similarity Search. Dense-sparse combined search mixes semantic embeddings (dense) with lexical signals (sparse, e.g., BM25-like term vectors), where the sparse side is high-dimensional but extremely sparse. OneSparse [11] couples an inverted index (sparse) with a quantization-based index (dense) and retrieves candidates by intersecting the sorted posting-list IDs selected for each modality, using efficient multi-list merge to find common IDs before scoring. Proximity-graph-based solutions further study distribution alignment between dense and sparse distances via pre-sampling and propose computation ordering/short-circuiting (compute dense distances first, then selectively compute sparse distances) to skip unnecessary work and reduce end-to-end cost [105]. Beyond two-vector setups, AllanPoe [51] moves toward a unified graph-based index that jointly integrates dense vectors, sparse vectors, full-text, and knowledge graph signals, leveraging GPUs for efficient construction and search.

2.4 Filtered Similarity Search

2.4.1 Problem Definition. Given a dataset $O = \{o_1, o_2, \cdots, o_n\} = \{(v_1, a_1), (v_2, a_2), \cdots, (v_n, a_n)\}$ consisting of n objects, where each o_i is with a d-dimensional vector $v_i \in \mathbb{R}^d$, and an attribute a_i. Existing works consider two types of attributes or a mix of them: categorical and numerical attributes. Let A_C be a domain of categorical values and A_N be a domain of numbers. An attribute a_i can be either a set of categorical values, $a_i = \{l_1, l_2, \cdots, l_{|a_i|}\}$, for $l_i \in A_C$, a single value $a_i \in A_N$, or a range value $a_i = [a_l, a_r] \subseteq A_N$. A filtered k-ANN query is specified as $q = (v_q, c_q)$, where v_q is a user-given query vector in $\mathbb{R}^d$ and c_q is a predicate on attributes. Let $O_{c_q} \subseteq O$ be the set of objects that satisfy the predicate c_q. The filtered k-ANN query $q = (v_q, c_q)$ is to find the k-ANN in O_{c_q} for the user-given query vector v_q.

2.4.2 Techniques for Filtered Similarity Search. Filtered k-ANN methods largely follow two design choices: *universal* indexes that support arbitrary attributes/predicates with (mostly) one PG, and *dedicated* indexes that build predicate-aware structures, typically specialized for categorical or numerical filters.

Universal Index Approaches: Using a single PG, existing work either (i) adapts the *search algorithm* over the same graph [75, 84, 106], (ii) modifies the PG into a *predicate-aware* variant (e.g., via 2-hop expansion) [65], or (iii) *enhances* the PG by adding an auxiliary attribute graph [90]. On the algorithmic side, Milvus [75] and ADBV [84, 103] adopt pre-filtering (restrict to O_{c_q} then search) and post-filtering (search $k' > k$ then filter), while VBASE [106] introduces a relaxed-monotonicity two-phase traversal that first moves toward v_q and then explores neighbors under c_q. Structurally, ACORN [65] builds a 2-hop PG (ACORN-γ/ACORN-1) and restricts traversal to predicate-satisfying nodes, and UniFilter [90] integrates an AND/OR graph for categorical and numerical predicates to guide predicate-aware edges adding and to prune traversal.

Dedicated Index Approaches: Dedicated designs build predicate-specialized indexes to speed up filtered search. For categorical predicates, FilteredVamana and StitchedVamana [26] construct label-aware graphs (incremental or per-label then stitched/pruned) and run beam search over nodes satisfying c_q; UNG [8] builds one PG per distinct categorical set and a navigating graph over these sets to locate minimal supersets for arbitrary c_q; VecFlow [86] targets GPUs by grouping IVF posting lists by label-distribution patterns and building a graph per group. For numerical predicates, range filtering is supported by multi-graph/hierarchical decompositions: SeRF [114] constructs compressed 1D-search-graph layers of HNSW variants; HSIG [52] uses sampled sorting with equi-depth histogram partitions and builds PGs for partition combinations; SuperPost-filtering [17] builds overlapping-range PGs and answers by post-filtering on the smallest enclosing range; iRangeGraph [91] uses a segment tree [14] with a PG per node, while DIGRA [41] replaces B-tree with segment tree for supporting maintenance; WoW [83] maintains a window-hierarchy to bound update cost and searches by selecting windows covering the query range; RangePQ [104] provides a quantization-based alternative with maintenance support. Beyond simple ranges, interval filtering is handled by transforming intervals to 2D points and querying rectangles (Hi-PNG [95] via QuadTree and PGs), while timestamp filtering compresses many per-timestamp PGs into one structure and extracts the relevant subgraph at query time (TS-Graph [80]). More recently, generalized range-range filtering further associates each vector with a range-valued attribute and supports arbitrary predicates between query and object ranges using multi-segment tree graphs (MSTG [54]).

2.5 Similarity Join

2.5.1 Problem Definition. There are two types of similarity join: ϵ-similarity join and k-similarity join. (1) **ϵ-Similarity Join:** Given two sets of vectors, $X = \{x_1, x_2, \cdots, x_n\}$ and $Y = \{y_1, y_2, \cdots, y_m\}$, in $\mathbb{R}^d$, for $d > 0$, the ϵ-similarity join $X \bowtie_\epsilon Y$ is defined as $X \bowtie_\epsilon Y = \{(x_k, y_l) \in X \times Y \mid \delta(x_k, y_l) \leq \epsilon\}$, where $\delta(x_k, y_l)$ is the distance metric, e.g., Euclidean distance, between $x_k \in X$ and $y_l \in Y$, and ϵ is a user-given threshold where $\epsilon > 0$. (2) **k-Similarity Join:** The k-similarity join $X \bowtie_k Y$ between two d-dimensional dataset X and Y is defined as $X \bowtie_k Y = \{(x_i, y_j) \in X \times Y \mid y_j \in TopK(x_i)\}$, where $TopK(x_i)$ is the set of top-k nearest neighbors of x_i in Y.

2.5.2 Techniques for Exact Similarity Join. Prior work on exact high-dimensional similarity join covers both ϵ-similarity join [5, 7, 44, 67, 70] and k-NN join [6, 99], though the notion of “high-dimensional” in these works (often around tens of dimensions [6, 44, 70, 99]) differs from today’s much higher-dimensional vector embeddings. State-of-the-art exact join pipelines typically follow *filtering + refinement* [36, 43, 56, 60, 67]: the filtering stage generates candidate pairs via indexing or ordering, and the refinement stage computes exact distances to obtain the final result. Because building predicate-supporting spatial indexes (e.g., R-trees [7]) can be expensive, many methods instead rely on sorting/ordering schemes that preserve join locality, such as Epsilon Grid Order [5].

2.5.3 Techniques for Approximate Similarity Join. Approximate similarity join is often reduced to running an approximate ϵ-range query per data point, supported by either LSH-based indexes [3, 33, 48, 68, 81, 100, 102] or proximity-graph-based indexes [75, 106]. LSH-based pipelines typically (i) hash/project vectors, (ii) equi-join colliding buckets to form candidates, and (iii) refine by exact distance checks [3, 33, 48, 68, 81, 100, 102]; their effectiveness hinges on the hash functions’ collision behavior, and can degrade on skewed real datasets since distribution is not explicitly modeled [79]. SimJoin [88] accelerates join processing by reusing intermediate join outcomes and optimizing the processing order, while also studying k-similarity join support.

2.6 Open Challenges and Discussions

Bridging Guarantees and Practicality: A persistent gap is that many approaches are largely heuristic, whereas approaches with theoretical guarantees often rely on strong assumptions or incur high construction costs. An open challenge is to design indexes that provide guarantees while remaining practical to build and achieving performance comparable to state-of-the-art approaches.

Toward Unified Query Processing: First, vector queries are often optimized in isolation, and vectors are rarely treated as first-class data types that interact naturally with relational operators. Developing unified query models, optimizer abstractions, and cost/statistics models for composing vector predicates with filters, joins, and aggregations remains open. Second, supporting diverse query types (similarity, multi-vector, filtered search, and join) typically requires multiple specialized indexes, sometimes duplicated across execution environments (in-memory, SSD, and GPU). A key challenge is to reduce this fragmentation by unifying index backbones or designing an all-in-one index family that supports multiple query types and hardware settings with low build time and storage overhead.

3 PRESENTERS

Jiadong Xie is a PhD candidate in the Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, advised by Prof. Jeffrey Xu Yu. His research interests are mainly vector databases and graph algorithms.

Yingfan Liu is a tenured Associate Professor in the School of Computer Science and Technology at Xidian University. Before joining Xidian University, he obtained his PhD degree from the Department of Systems Engineering and Engineering Management at the Chinese University of Hong Kong in 2019. His research interests include vector databases, and high-performance computing.

Jeffrey Xu Yu is a Professor and Acting Head of Data Science and Analytics Thrust, The Hong Kong University of Science and Technology (Guangzhou). His current main research interests include vector databases, graph algorithms and systems, graph neural networks, query processing, and optimization. He served/serves in over 300 organization committees and program committees in international conferences/workshops. He has held several prestigious roles, including Information Director and member of the ACM SIGMOD Executive Committee, Associate Editor of IEEE TKDE, and Associate Editor of the VLDB Journal. He currently serves as an Associate Editor for ACM TODS and WWW Journal.

REFERENCES

[1] Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2015. Cache locality is not enough: High-Performance Nearest Neighbor Search with Product Quantization Fast Scan. *Proc. VLDB Endow.* 9, 4 (2015), 288–299.

[2] Martin Aumüller and Matteo Ceccarello. 2021. The role of local dimensionality measures in benchmarking nearest neighbor search. *Inf. Syst.* 101 (2021), 101807.

[3] Martin Aumüller and Matteo Ceccarello. 2022. Implementing Distributed Similarity Joins using Locality Sensitive Hashing. In *EDBT 2022*. OpenProceedings.org, 1:78–1:90.

[4] Artem Babenko and Victor Lempitsky. 2014. Additive quantization for extreme vector compression. In *CVPR*. 931–938.

[5] Christian Böhm, Bernhard Braunmüller, Florian Krebs, and Hans-Peter Kriegel. 2001. Epsilon Grid Order: An Algorithm for the Similarity Join on Massive High-Dimensional Data. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*. ACM, 379–388.

[6] Christian Böhm and Florian Krebs. 2004. The *k*-Nearest Neighbour Join: Turbo Charging the KDD Process. *Knowl. Inf. Syst.* 6, 6 (2004), 728–749.

[7] Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. 1993. Efficient Processing of Spatial Joins Using R-Trees. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993*. ACM Press, 237–246.

[8] Yuzheng Cai, Jiayang Shi, Yizhuo Chen, and Weiguo Zheng. 2024. Navigating Labels and Vectors: A Unified Approach to Filtered Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 6 (2024), 246:1–246:27.

[9] Manos Chatzakis, Yannis Papakonstantinou, and Themis Palpanas. 2025. DARTH: Declarative Recall Through Early Termination for Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 3, 4 (2025), 242:1–242:26.

[10] Meng Chen, Kai Zhang, Zhenying He, Yinan Jing, and X. Sean Wang. 2024. RoarGraph: A Projected Bipartite Graph for Efficient Cross-Modal Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 17, 11 (2024), 2735–2749.

[11] Yaoqi Chen, Ruicheng Zheng, Qi Chen, Shuotao Xu, Qianxi Zhang, Xue Wu, Weihao Han, Hua Yuan, Mingqin Li, Yujing Wang, Jason Li, Fan Yang, Hao Sun, Weiwei Deng, Feng Sun, Qi Zhang, and Mao Yang. 2024. OneSparse: A Unified System for Multi-index Vector Search. In *WWW*. ACM, 393–402.

[12] Rongxin Cheng, Yifan Peng, Xingda Wei, Hongrui Xie, Rong Chen, Sijie Shen, and Haibo Chen. 2024. Characterizing the Dilemma of Performance and Index Size in Billion-Scale Vector Search and Breaking It with Second-Tier Memory. *CoRR* abs/2405.03267 (2024).

[13] Yannis Chronis, Helena Caminal, Yannis Papakonstantinou, Fatma Özcan, and Anastasia Ailamaki. 2025. Filtered vector search: State-of-the-art and research opportunities. *Proceedings of the VLDB Endowment* 18, 12 (2025), 5488–5492.

[14] Mark de Berg. 1997. *Computational geometry: algorithms and applications.* Springer.

[15] Wei Dong, Moses Charikar, and Kai Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011*. ACM, 577–586.

[16] Hao Duan, Yitong Song, Bin Yao, and Anqi Liang. 2025. PGTuner: An Efficient Framework for Automatic and Transferable Configuration Tuning of Proximity Graphs. *Proc. ACM Manag. Data* 3, 4 (2025), 261:1–261:27.

[17] Joshua Engels, Benjamin Landrum, Shangdi Yu, Laxman Dhulipala, and Julian Shun. 2024. Approximate Nearest Neighbor Search with Window Filters. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

[18] Cong Fu, Changxu Wang, and Deng Cai. 2022. High Dimensional Similarity Search With Satellite System Graph: Efficiency, Scalability, and Unindexed Query Compatibility. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 8 (2022), 4139–4150.

[19] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2019. Fast Approximate Nearest Neighbor Search With The Navigating Spreading-out Graph. *Proc. VLDB Endow.* 12, 5 (2019), 461–474.

[20] Georg Fuchsbauer, Riddhi Ghosal, Nathan Hauke, and Adam O'Neill. 2022. Approximate Distance-Comparison-Preserving Symmetric Encryption. In *SCN 2022 (Lecture Notes in Computer Science, Vol. 13409)*. Springer, 117–144.

[21] Teddy Furon, Hervé Jégou, Laurent Amsaleg, and Benjamin Mathon. 2013. Fast and secure similarity search in high dimensional space. In *WIFS 2013*. IEEE, 73–78.

[22] Jianyang Gao, Yutong Gou, Yuexuan Xu, Yongyi Yang, Cheng Long, and Raymond Chi-Wing Wong. 2025. Practical and Asymptotically Optimal Quantization of High-Dimensional Vectors in Euclidean Space for Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 3, 3 (2025), 202:1–202:26.

[23] Jianyang Gao and Cheng Long. 2023. High-Dimensional Approximate Nearest Neighbor Search: with Reliable and Efficient Distance Comparison Operations. *Proc. ACM Manag. Data* 1, 2 (2023), 137:1–137:27.

[24] Jianyang Gao and Cheng Long. 2024. RaBitQ: Quantizing High-Dimensional Vectors with a Theoretical Error Bound for Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 3 (2024), 167.

[25] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2013. Optimized product quantization. *IEEE TPAMI* 36, 4 (2013), 744–755.

[26] Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, Amit Singh, and Harsha Vardhan Simhadri. 2023. Filtered-DiskANN: Graph Algorithms for Approximate Nearest Neighbor Search with Filters. In *Proceedings of the ACM Web Conference 2023, WWW 2023*. ACM, 3406–3416.

[27] Yutong Gou, Jianyang Gao, Yuexuan Xu, and Cheng Long. 2025. SymphonyQG: Towards Symphonious Integration of Quantization and Graph for Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 3, 1 (2025), 80:1–80:26.

[28] Fabian Groh, Lukas Ruppert, Patrick Wieschollek, and Hendrik PA Lensch. 2022. GGNN: Graph-based gpu nearest neighbor search. *IEEE Transactions on Big Data* 9, 1 (2022), 267–279.

[29] Yunguo Guan, Rongxing Lu, Yandong Zheng, Jun Shao, and Guiyi Wei. 2021. Toward Oblivious Location-Based k-Nearest Neighbor Query in Smart Cities. *IEEE Internet Things J.* 8, 18 (2021), 14219–14231.

[30] Hao Guo and Youyou Lu. 2025. Achieving Low-Latency Graph-Based Vector Search via Aligning Best-First Search Algorithm with SSD. In *OSDI*. USENIX Association, 171–186.

[31] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating Large-Scale Inference with Anisotropic Vector Quantization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020 (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 3887–3896.

[32] Ben Harwood and Tom Drummond. 2016. FANNG: Fast Approximate Nearest Neighbour Graphs. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*. IEEE Computer Society, 5713–5722.

[33] Xiao Hu, Ke Yi, and Yufei Tao. 2019. Output-Optimal Massively Parallel Algorithms for Similarity Joins. *ACM Trans. Database Syst.* 44, 2 (2019), 6:1–6:36.

[34] Zhiyuan Hua, Qiji Mo, Zebin Yao, Lixiao Cui, Xiaoguang Liu, Gang Wang, Zijing Wei, Xinyu Liu, Tianxiao Tang, Shaozhi Liu, and Lin Qu. 2025. Dynamically Detect and Fix Hardness for Efficient Approximate Nearest Neighbor Search. *CoRR* abs/2510.22316 (2025).

[35] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. 604–613.

[36] Edwin H. Jacox and Hanan Samet. 2008. Metric space similarity joins. *ACM Trans. Database Syst.* 33, 2 (2008), 7:1–7:38.

[37] Shikhar Jaiswal, Ravishankar Krishnaswamy, Ankit Garg, Harsha Vardhan Simhadri, and Sheshansh Agrawal. 2022. OOD-DiskANN: Efficient and Scalable Graph ANNS for Out-of-Distribution Queries. *CoRR* abs/2211.12850 (2022).

[38] Junhyeok Jang, Hanjin Choi, Hanyeoreum Bae, Seungjun Lee, Miryeong Kwon, and Myoungsoo Jung. 2024. Bridging Software-Hardware for CXL Memory Disaggregation in Billion-Scale Nearest Neighbor Search. *ACM Trans. Storage* 20, 2 (2024), 10:1–10:30.

[39] Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. 2019. DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc.

[40] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE TPAMI* 33(1) (2011), 117–128.

[41] Mengxu Jiang, Zhi Yang, Fangyuan Zhang, Guanhao Hou, Jieming Shi, Wenchao Zhou, Feifei Li, and Sibo Wang. 2025. DIGRA: A Dynamic Graph Indexing for Approximate Nearest Neighbor Search with Range Filter. *Proc. ACM Manag. Data* 3, 3 (2025), 148:1–148:26.

[42] Wenqi Jiang, Hang Hu, Torsten Hoefler, and Gustavo Alonso. 2025. Fast Graph Vector Search via Hardware Acceleration and Delayed-Synchronization Traversal. *Proc. VLDB Endow.* 18, 11 (2025), 3797–3811.

[43] Dmitri V. Kalashnikov. 2013. Super-EGO: fast multi-dimensional similarity join. *VLDB J.* 22, 4 (2013), 561–585.

[44] Nick Koudas and Kenneth C. Sevcik. 2000. High Dimensional Similarity Joins: Algorithms and Performance Evaluation. *IEEE Trans. Knowl. Data Eng.* 12, 1 (2000), 3–18.

[45] Govinda D. Kurup. 1992. *Database Organized on the Basis of Similarities with Applications in Computer Vision.* Ph. D. Dissertation.

[46] Binhong Li, Xiao Yan, and Shangqi Lu. 2025. Fast-Convergent Proximity Graphs for Approximate Nearest Neighbor Search. *CoRR* abs/2510.05975 (2025).

[47] Conglong Li, Minjia Zhang, David G. Andersen, and Yuxiong He. 2020. Improving Approximate Nearest Neighbor Search through Learned Adaptive Early Termination. In *SIGMOD*. ACM, 2539–2554.

[48] Hangyu Li, Sarana Nutanong, Hong Xu, Chenyun Yu, and Foryu Ha. 2019. C2Net: A Network-Efficient Approach to Collision Counting LSH Similarity Join. *IEEE Trans. Knowl. Data Eng.* 31, 3 (2019), 423–436.

[49] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2020. Approximate Nearest Neighbor Search on High Dimensional Data - Experiments, Analyses, and Improvement. *IEEE Trans. Knowl. Data Eng.* 32, 8 (2020), 1475–1488.

[50] Zhonggen Li, Xiangyu Ke, Yifan Zhu, Bocheng Yu, Baihua Zheng, and Yunjun

Gao. 2025. Scalable Graph Indexing using GPUs for Approximate Nearest Neighbor Search. *CoRR* abs/2508.08744 (2025).

[51] Zhonggen Li, Yougen Li, Yifan Zhu, Zhaoqiang Chen, and Yunjun Gao. 2025. All-in-one Graph-based Indexing for Hybrid Search on GPUs. *CoRR* abs/2511.00855 (2025).

[52] Anqi Liang, Pengcheng Zhang, Bin Yao, Zhongpu Chen, Yitong Song, and Guangxu Cheng. 2024. UNIFY: Unified Index for Range Filtered Approximate Nearest Neighbors Search. *CoRR* abs/2412.02448 (2024).

[53] Dawei Liu, Bolong Zheng, Ziyang Yue, Fuhao Ruan, Xiaofang Zhou, and Christian S. Jensen. 2025. Wolverine: Highly Efficient Monotonic Search Path Repair for Graph-based ANN Index Updates. *Proc. VLDB Endow.* 18, 7 (2025), 2268–2280.

[54] Yingfan Liu, Tong Wu, Jiadong Xie, Yang Zhao, Jeffrey Xu Yu, and Jiangtao Cui. 2026. Generalized Range Filtering Approximate Nearest Neighbor Search: Containment and Overlap. In *SIGKDD*. ACM.

[55] Yingfan Liu, Yandi Zhang, Jiadong Xie, Hui Li, Jeffrey Xu Yu, and Jiangtao Cui. 2025. Privacy-preserving approximate nearest neighbor search on high-dimensional data. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE, 3017–3029.

[56] Youzhong Ma, Shijie Jia, and Yongxin Zhang. 2017. A novel approach for high-dimensional vector similarity join query. *Concurr. Comput. Pract. Exp.* 29, 5 (2017).

[57] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. 2014. Approximate nearest neighbor algorithm based on navigable small world graphs. *Inf. Syst.* 45 (2014), 61–68.

[58] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836.

[59] Magdalen Dobson Manohar, Zheqi Shen, Guy E. Blelloch, Laxman Dhulipala, Yan Gu, Harsha Vardhan Simhadri, and Yihan Sun. 2024. ParlayANN: Scalable and Deterministic Parallel Graph-Based Approximate Nearest Neighbor Search Algorithms. In *Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming, PPoPP 2024*. ACM, 270–285.

[60] Ahmed Metwally and Christos Faloutsos. 2012. V-SMART-Join: A Scalable MapReduce Framework for All-Pair Similarity Joins of Multisets and Vectors. *Proc. VLDB Endow.* 5, 8 (2012), 704–715.

[61] Stanley Milgram. 1967. The small world problem. *Psychology today* 2, 1 (1967), 60–67.

[62] Mohammad Norouzi and David J. Fleet. 2013. Cartesian k-means. In *CVPR*. IEEE, 3017–3024.

[63] Naoki Ono and Yusuke Matsui. 2023. Relative NN-Descent: A Fast Index Construction for Graph-Based Approximate Nearest Neighbor Search. In *Proceedings of the 31st ACM International Conference on Multimedia, MM 2023*. ACM, 1659–1667.

[64] Hiroyuki Ootomo, Akira Naruse, Corey Nolet, Ray Wang, Tamas Feher, and Yong Wang. 2024. CAGRA: Highly Parallel Graph Construction and Approximate Nearest Neighbor Search for GPUs. In *ICDE*. IEEE, 4236–4247.

[65] Liana Patel, Peter Kraft, Carlos Guestrin, and Matei Zaharia. 2024. ACORN: Performant and Predicate-Agnostic Search Over Vector Embeddings and Structured Data. *Proc. ACM Manag. Data* 2, 3 (2024), 120.

[66] Yun Peng, Byron Choi, Tsz Nam Chan, Jianye Yang, and Jianliang Xu. 2023. Efficient Approximate Nearest Neighbor Search in Multi-dimensional Databases. *Proc. ACM Manag. Data* 1, 1 (2023), 54:1–54:27.

[67] Martin Perdacher, Claudia Plant, and Christian Böhm. 2019. Cache-oblivious High-performance Similarity Join. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019*. ACM, 87–104.

[68] Sébastien Rivault, Mostafa Bamha, Sébastien Limet, and Sophie Robert. 2022. A Scalable Similarity Join Algorithm Based on MapReduce and LSH. *Int. J. Parallel Program.* 50, 3-4 (2022), 360–380.

[69] Sacha Servan-Schreiber, Simon Langowski, and Srinivas Devadas. 2022. Private approximate nearest neighbor search with sublinear communication. In *S&P 2022*. IEEE, 911–929.

[70] Kyuseok Shim, Ramakrishnan Srikant, and Rakesh Agrawal. 1997. High-Dimensional Similarity Joins. In *ICDE*. IEEE Computer Society, 301–311.

[71] Aditi Singh, Suhas Jayaram Subramanya, Ravishankar Krishnaswamy, and Harsha Vardhan Simhadri. 2021. FreshDiskANN: A Fast and Accurate Graph-Based ANN Index for Streaming Similarity Search. *CoRR* abs/2105.09613 (2021).

[72] Gurpreet Singh. 2013. A study of encryption algorithms (RSA, DES, 3DES and AES) for information security. *International Journal of Computer Applications* 67, 19 (2013).

[73] Yitong Song, Pengcheng Zhang, Chao Gao, Bin Yao, Kai Wang, Zongyuan Wu, and Lin Qu. 2025. TRIM: Accelerating High-Dimensional Vector Similarity Search with Enhanced Triangle-Inequality-Based Pruning. *CoRR* abs/2508.17828 (2025).

[74] Yitong Song, Xuanhe Zhou, Christian S Jensen, and Jianliang Xu. 2026. Vector Search for the Future: From Memory-Resident, Static Heterogeneous Storage, to Cloud-Native Architectures. *arXiv preprint arXiv:2601.01937* (2026).

[75] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *SIGMOD '21*. ACM, 2614–2627.

[76] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. 2018. A Survey on Learning to Hash. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 4 (2018), 769–790.

[77] Mengzhao Wang, Haotian Wu, Xiangyu Ke, Yunjun Gao, Yifan Zhu, and Wenchao Zhou. 2025. Accelerating Graph Indexing for ANNS on Modern CPUs. *Proc. ACM Manag. Data* 3, 3 (2025), 123:1–123:29.

[78] Mengzhao Wang, Weizhi Xu, Xiaomeng Yi, Songlin Wu, Zhangyang Peng, Xiangyu Ke, Yunjun Gao, Xiaoliang Xu, Rentong Guo, and Charles Xie. 2024. Starling: An I/O-Efficient Disk-Resident Graph Index Framework for High-Dimensional Vector Similarity Search on Data Segment. *Proc. ACM Manag. Data* 2, 1 (2024), V2mod014:1–V2mod014:27.

[79] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. 2021. A Comprehensive Survey and Experimental Comparison of Graph-Based Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 14, 11 (2021), 1964–1978.

[80] Yuxiang Wang, Ziyuan He, Yongxin Tong, Zimu Zhou, and Yiman Zhong. 2025. Timestamp Approximate Nearest Neighbor Search over High-Dimensional Vector Data. In *ICDE*. IEEE Computer Society, 3043–3055.

[81] Yifan Wang, Vyom Pathak, and Daisy Zhe Wang. 2024. Xling: A Learned Filter Framework for Accelerating High-Dimensional Approximate Similarity Join. *CoRR* abs/2402.13397 (2024).

[82] Zeyu Wang, Qitong Wang, Xiaoxing Cheng, Peng Wang, Themis Palpanas, and Wei Wang. 2024. Steiner-Hardness: A Query Hardness Measure for Graph-Based ANN Indexes. *Proc. VLDB Endow.* 17, 13 (2024), 4668–4682.

[83] Ziqi Wang, Jingzhe Zhang, and Wei Hu. 2025. WoW: A Window-to-Window Incremental Index for Range-Filtering Approximate Nearest Neighbor Search. *CoRR* abs/2508.18617 (2025).

[84] Chuangxian Wei, Bin Wu, Sheng Wang, Renjie Lou, Chaoqun Zhan, Feifei Li, and Yuanzhe Cai. 2020. AnalyticDB-V: A Hybrid Analytical Engine Towards Query Fusion for Structured and Unstructured Data. *Proc. VLDB Endow.* 13, 12 (2020), 3152–3165.

[85] Wai Kit Wong, David Wai-lok Cheung, Ben Kao, and Nikos Mamoulis. 2009. Secure kNN computation on encrypted databases. In *SIGMOD*. 139–152.

[86] Jingyi Xi, Chenghao Mo, Ben Karsin, Artem M. Chirkin, Mingqin Li, and Minjia Zhang. 2025. VecFlow: A High-Performance Vector Data Management System for Filtered-Search on GPUs. *Proc. ACM Manag. Data* 3, 4 (2025), 271:1–271:27.

[87] Jiadong Xie, Jeffrey Liang, Siyi Teng, Jeffrey Xu Yu, and Yingfan Liu. 2026. Breaking the Single-Reference-Vector Barrier in Approximate Nearest Neighbor Search. In *Proceedings of the ACM on Web Conference 2026, WWW*. ACM.

[88] Jiadong Xie, Jeffrey Xu Yu, and Yingfan Liu. 2025. Fast Approximate Similarity Join in Vector Databases. *Proc. ACM Manag. Data* 3, 3 (2025), 158:1–158:26.

[89] Jiadong Xie, Jeffrey Xu Yu, and Yingfan Liu. 2025. Graph Based K-Nearest Neighbor Search Revisited. *ACM TODS* (May 2025).

[90] Jiadong Xie, Jeffrey Xu Yu, Siyi Teng, and Yingfan Liu. 2025. Beyond Vector Search: Querying With and Without Predicates. *Proc. ACM Manag. Data* 3, 6 (2025), 300:1–300:26.

[91] Yuexuan Xu, Jianyang Gao, Yutong Gou, Cheng Long, and Christian S. Jensen. 2024. iRangeGraph: Improvising Range-dedicated Graphs for Range-filtering Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 6 (2024), 239:1–239:26.

[92] Yuming Xu, Hengyu Liang, Jin Li, Shuotao Xu, Qi Chen, Qianxi Zhang, Cheng Li, Ziyue Yang, Fan Yang, Yuqing Yang, Peng Cheng, and Mao Yang. 2023. SPFresh: Incremental In-Place Update for Billion-Scale Vector Search. In *SOSP*. ACM, 545–561.

[93] Beining Yang, Yang Cao, and Yang Ren. 2025. Integrating Vector Databases across Embedding Models. *Proc. ACM Manag. Data* 3, 6 (2025), 1–28.

[94] Ming Yang, Yuzheng Cai, and Weiguo Zheng. 2024. CSPG: Crossing Sparse Proximity Graphs for Approximate Nearest Neighbor Search. In *NeurIPS*.

[95] Ming Yang, Yuzheng Cai, and Weiguo Zheng. 2025. Hi-PNG: Efficient Interval-Filtering ANNS via Hierarchical Interval Partition Navigating Graph. In *SIGKDD*. 3518–3529.

[96] Mingyu Yang, Wentao Li, Jiabao Jin, Xiaoyao Zhong, Xiangyu Wang, Zhitao Shen, Wei Jia, and Wei Wang. 2025. Effective and General Distance Computation for Approximate Nearest Neighbor Search. In *ICDE*. IEEE, 1098–1110.

[97] Shuo Yang, Jiadong Xie, Yingfan Liu, Jeffrey Xu Yu, Xiyue Gao, Qianru Wang, Yanguo Peng, and Jiangtao Cui. 2025. Revisiting the Index Construction of Proximity Graph-Based Approximate Nearest Neighbor Search. *PVLDB* 18, 6 (2025), 1825–1838.

[98] Ziqi Yin, Jianyang Gao, Pasquale Balsebre, Gao Cong, and Cheng Long. 2025. DEG: Efficient Hybrid Vector Search Using the Dynamic Edge Navigation Graph. *Proc. ACM Manag. Data* 3, 1 (2025), 29:1–29:28.

[99] Cui Yu, Bin Cui, Shuguang Wang, and Jianwen Su. 2007. Efficient index-based KNN join processing for high-dimensional data. *Inf. Softw. Technol.* 49, 4 (2007), 332–344.

[100] Chenyun Yu, Sarana Nutanong, Hangyu Li, Cong Wang, and Xingliang Yuan. 2017. A Generic Method for Accelerating LSH-Based Similarity Join Processing.

IEEE Trans. Knowl. Data Eng. 29, 4 (2017), 712–726.

[101] Yuanhang Yu, Dong Wen, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2022. GPU-accelerated proximity graph approximate nearest Neighbor search and construction. In *ICDE*. IEEE, 552–564.

[102] Xingliang Yuan, Xinyu Wang, Cong Wang, Chenyun Yu, and Sarana Nutanong. 2017. Privacy-Preserving Similarity Joins Over Encrypted Data. *IEEE Trans. Inf. Forensics Secur.* 12, 11 (2017), 2763–2775.

[103] Chaoqun Zhan, Maomeng Su, Chuangxian Wei, Xiaoqiang Peng, Liang Lin, Sheng Wang, Zhe Chen, Feifei Li, Yue Pan, Fang Zheng, and Chengliang Chai. 2019. AnalyticDB: Real-time OLAP Database System at Alibaba Cloud. *Proc. VLDB Endow.* 12, 12 (2019), 2059–2070.

[104] Fangyuan Zhang, Mengxu Jiang, Guanhao Hou, Jieming Shi, Hua Fan, Wenchao Zhou, Feifei Li, and Sibo Wang. 2025. Efficient Dynamic Indexing for Range Filtered Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 3, 3 (2025), 152:1–152:26.

[105] Haoyu Zhang, Jun Liu, Zhenhua Zhu, Shulin Zeng, Maojia Sheng, Tao Yang, Guohao Dai, and Yu Wang. 2024. Efficient and Effective Retrieval of Dense-Sparse Hybrid Vectors using Graph-based Approximate Nearest Neighbor Search. *CoRR* abs/2410.20381 (2024).

[106] Qianxi Zhang, Shuotao Xu, Qi Chen, Guoxin Sui, Jiadong Xie, Zhizhen Cai, Yaoqi Chen, Yinxuan He, Yuqing Yang, Fan Yang, Mao Yang, and Lidong Zhou. 2023. VBASE: Unifying Online Vector Similarity Search and Relational Queries via Relaxed Monotonicity. In *17th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2023*. USENIX Association, 377–395.

[107] Zhilin Zhang, Ke Wang, Chen Lin, and Weipeng Lin. 2018. Secure Top-k Inner Product Retrieval. In *CIKM 2018*. ACM, 77–86.

[108] Weijie Zhao, Shulong Tan, and Ping Li. 2020. SONG: Approximate nearest neighbor search on gpu. In *ICDE*. IEEE, 1033–1044.

[109] Xi Zhao, Yao Tian, Kai Huang, Bolong Zheng, and Xiaofang Zhou. 2023. Towards Efficient Index Construction and Approximate Nearest Neighbor Search in High-Dimensional Spaces. *Proc. VLDB Endow.* 16, 8 (2023), 1979–1991.

[110] Yandong Zheng, Rongxing Lu, and Jun Shao. 2019. Achieving efficient and privacy-preserving k-NN query for outsourced ehealthcare data. *Journal of Medical Systems* 43 (2019), 1–13.

[111] Yandong Zheng, Rongxing Lu, Songnian Zhang, Jun Shao, and Hui Zhu. 2024. Achieving Practical and Privacy-Preserving kNN Query over Encrypted Data. *IEEE TDSC* (2024).

[112] Mingxun Zhou, Elaine Shi, and Giulia Fanti. 2024. Pacmann: Efficient Private Approximate Nearest Neighbor Search. *IACR Cryptol. ePrint Arch.* (2024), 1600.

[113] Youwen Zhu, Rui Xu, and Tsuyoshi Takagi. 2013. Secure k-NN computation on encrypted cloud data without sharing key with query users. In *SCC@ASIACCS*. ACM, 55–60.

[114] Chaoji Zuo, Miao Qiao, Wenchao Zhou, Feifei Li, and Dong Deng. 2024. SeRF: Segment Graph for Range-Filtering Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 1 (2024), 69:1–69:26.