Towards Reliable Vector Database Management Systems: A Software Testing Roadmap for 2030

SHENAO WANG, Huazhong University of Science and Technology, China YANJIE ZHAO, Huazhong University of Science and Technology, China YINGLIN XIE, Huazhong University of Science and Technology, China ZHAO LIU, 360 AI Security Lab, China XINYI HOU, Huazhong University of Science and Technology, China QUANCHEN ZOU, 360 AI Security Lab, China HAOYU WANG, Huazhong University of Science and Technology, China

The rapid growth of Large Language Models (LLMs) and AI-driven applications has propelled Vector Database Management Systems (VDBMSs) into the spotlight as a critical infrastructure component. VDBMS specializes in storing, indexing, and querying dense vector embeddings, enabling advanced LLM capabilities such as retrieval-augmented generation, long-term memory, and caching mechanisms. However, the explosive adoption of VDBMS has outpaced the development of rigorous software testing methodologies tailored for these emerging systems. Unlike traditional databases optimized for structured data, VDBMS face unique testing challenges stemming from the high-dimensional nature of vector data, the fuzzy semantics in vector search, and the need to support dynamic data scaling and hybrid query processing. In this paper, we begin by conducting an empirical study of VDBMS defects and identify key challenges in test input generation, oracle definition, and test evaluation. Drawing from these insights, we propose the first comprehensive research roadmap for developing effective testing methodologies tailored to VDBMS. By addressing these challenges, the software testing community can contribute to the development of more reliable and trustworthy VDBMS, enabling the full potential of LLMs and data-intensive AI applications.

1 Introduction

The advent of Large Language Models (LLMs) and the AI revolution has catapulted Vector Database Management Systems (VDBMSs) [43], such as Weaviate [42], Pinecone [29], and Qdrant [31], into the forefront as a foundational infrastructure for the data-intensive era [17, 27, 35]. Unlike traditional databases optimized for tabular or document-based data, VDBMS specializes in storing, indexing, and querying vector embeddings—dense mathematical representations of unstructured data that power the core functionality of LLMs, recommendation engines, semantic search, and multimodal applications. VDBMS plays a pivotal role in enabling advanced LLM capabilities [11, 17, 39], such as Retrieval-Augmented Generation (RAG) systems [38], Long-Term Memory (LTM) applications [44], and LLM caching mechanisms [2]. As LLM systems become increasingly sophisticated and data-hungry, VDBMS emerges as a critical technology for harnessing the full potential of these models and applications.

While embedding-based retrieval algorithms have been studied for over a decade in the academic community [6, 9, 41], the transition from theoretical frameworks to production-grade VDBMS marks a watershed moment for software testing. Over the past few years, this transition has spurred the rapid commercialization of VDBMS, with more than 40 specialized platforms now vying to optimize vector storage, real-time similarity search, and integration with LLM pipelines. **However, this explosive growth has not been matched by commensurate progress in rigorous software**

Authors' Contact Information: Shenao Wang, shenaowang@hust.edu.cn, Huazhong University of Science and Technology, Wuhan, China; Yanjie Zhao, yanjie_zhao@hust.edu.cn, Huazhong University of Science and Technology, Wuhan, China; Yinglin Xie, xieyinglin@hust.edu.cn, Huazhong University of Science and Technology, Wuhan, China; Zhao Liu, r3pwnx@gmail.com, 360 AI Security Lab, Beijing, China; Xinyi Hou, xinyihou@hust.edu.cn, Huazhong University of Science and Technology, Wuhan, China; Quanchen Zou, zouquanchen@gmail.com, 360 AI Security Lab, Beijing, China; Haoyu Wang, haoyuwang@hust.edu.cn, Huazhong University of Science and Technology, Wuhan, China.

testing methodologies tailored for these emerging systems. Unlike traditional databases optimized for structured data, VDBMS is purpose-built for storing, indexing, and querying vector embeddings. As such, they face unique testing challenges stemming from the high-dimensional nature of vector data [27, 35, 43], the approximate nature of vector search [12, 14], and the need to support dynamic data scaling and hybrid query processing [10, 27, 28].

Current testing efforts for VDBMS are still in their infancy, primarily focusing on isolated performance benchmarks [1, 24, 30, 45] or simplified metamorphic testing cases [38] that address specific aspects like false vector matches in LLM-augmented generation systems. While these studies shed light on the critical impact of VDBMS defects on downstream applications, they represent initial steps, and **comprehensive testing methodology that holistically addresses the unique characteristics and challenges of VDBMS remains an open research gap.** Lessons learned from the traditional **Database Management System (DBMS)** domain caution us that even after extensive research and rigorous testing, modern DBMS still exhibit systemic defects [5, 7, 19, 37]. Similarly, the deep learning ecosystem, which shares similarities with VDBMS in terms of data-intensive workloads and complex computational pipelines, has encountered numerous reliability issues [3, 16, 20], potentially leading to crashes, hangs, build failures, and silent data corruption [36, 40]. Given the pivotal role of VDBMS in enabling advanced LLM capabilities and the potentially far-reaching impact of failures, envisioning comprehensive testing tailored to VDBMS is imperative for the 2030 software engineering landscape.

However, testing VDBMS poses unique challenges that must be carefully considered: (1) The high-dimensional nature of vector data introduces complexities in test data generation, resource overhead, and performance trade-offs. High-dimensional vectors spanning across multiple storage pages cause memory explosion for fuzz testing tools [18, 27, 28]. Moreover, the O(d) time complexity of vector similarity computation, where d denotes the vector dimension, drastically decreases testing speed compared to traditional O(1) attribute predicates [27, 28]. (2) The fuzzy semantics in vector search make the oracle definition challenging. ANN algorithms have non-deterministic ϵ -error bounds [12], rendering traditional boolean assertions ineffective. (3) Dynamic data scaling and hybrid query processing across data modalities demand testing strategies for evolving workloads and heterogeneous data types. Data distribution shifts due to dynamic updates can cause indexing strategies to become imbalanced over time, often requiring full index rebuilds [27]. Furthermore, hybrid queries involving predicate filtering before, during, or after vector similarity search across multimodal data necessitate diverse testing strategies. (4) The integration with complex LLM pipelines necessitates end-to-end testing to capture error propagation, as small embedding perturbations can cascade into semantic errors after LLM decoding [38], with error accumulation effects amplifying retrieval-stage issues [13].

In light of these challenges, this paper proposes the first roadmap for future research on VDBMS testing. We begin by defining the common components and architecture of VDBMS, which typically consist of a vector storage engine, indexing subsystem, query processing pipeline, and client-side SDK. We then systematically analyze the bugs, vulnerabilities, and defects that have been discovered in existing VDBMS implementations, categorizing them into different components and symptoms. Drawing from these insights, we structure our investigation into three key aspects of VDBMS testing: test input generation, oracle definition, and test evaluation. For each of these aspects, we identify the unique challenges stemming from the intrinsic characteristics of VDBMS and propose a future research roadmap to address these challenges.

In summary, our contributions are detailed as follows:

• Empirical Study of VDBMS Defects. We conduct an empirical study of bugs across four major open-source VDBMS projects. Our analysis reveals a high prevalence of crash/hang bugs

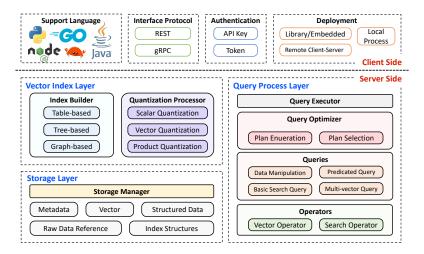


Fig. 1. Typical Architecture of VDBMSs.

(23.1%), with storage components severely impacted. Incorrect behavior bugs are most common (43.0%), severely affecting query processing components. These findings highlight the need for tailored testing approaches to address critical defects in VDBMS.

- Identification of Key Testing Challenges. Drawing from the empirical study of defects in open-source VDBMS projects and the unique characteristics of VDBMS, we identify and articulate the key challenges in three fundamental aspects of VDBMS testing: test input generation, test oracle design, and test evaluation. These challenges are particularly critical in the VDBMS context due to the high-dimensional nature of vector data, the impact of dynamic data scaling on indexing strategies, and the complexity of vector operations and queries.
- Proposal of the Research Roadmap. Guided by the identified challenges, we propose the first comprehensive research roadmap that outlines future directions for addressing the critical aspects of VDBMS testing. Following this roadmap, the software testing community can contribute to the development of more reliable and trustworthy VDBMS, enabling the full potential of LLMs and data-intensive AI applications.

2 VDBMS

2.1 Architecture

VDBMSs are specialized systems designed to efficiently store, index, and query high-dimensional vector embeddings. While the specific architectures can vary across different VDBMS implementations, they generally consist of several interconnected components, as shown in Figure 1.

Storage. This layer handles the persistent storage of vector embeddings, associated metadata, raw data references, index structures, and structured data. The storage manager component oversees the storage and retrieval of these different data elements, often leveraging techniques like compression and partitioning for optimized space utilization.

Vector Index. The vector index layer enables efficient similarity search over vast vector collections through specialized indexing structures and quantization techniques tailored for high-dimensional data. The index builder constructs and maintains vector index structures based on table-based methods (e.g., LSH [15]), tree-based methods (e.g., ANNOY [34]), and graph-based techniques (e.g., HNSW [21]). The quantization processor employs vector compression techniques, including scalar quantization, vector quantization, and product quantization.

System	Type	Sub-type	Language	Max Dim	Index Methods	Pre. Query
Pinecone [29]	Native	Vector	Rust	20,000	FreshDiskANN	V
Chroma [4]	Native	Vector	Python	Unlimited	HNSW	×
Weaviate [42]	Native	Hybrid	Go	65,535	FLAT/FLAT-BQ/HNSW	✓
Milvus [22]	Native	Hybrid	Go/C++	Unlimited	FLAT/HNSW/ANNOY	~
Qdrant [31]	Native	Hybrid	Rust	65,536	FLAT/HNSW/HNSW-PQ	~
Redis [32]	Extended	NoSQL	C	Unlimited	FLAT/HNSW	✓
MongoDB [23]	Extended	NoSQL	C++/Java	4,096	FLAT/HNSW	✓
Neo4j [26]	Extended	NoSQL	Java/Scala	2048	HNSW	~
SingleStore [33]	Extended	Relational	C++/Go	Unlimited	FLAT/IVF/HNSW	~
MyScale [25]	Extended	Relational	C++	Unlimited	FLAT/MSTG/IVF	~

Table 1. Feature Comparison of 10 Representative VDBMSs.

Query Process. This layer is responsible for parsing, optimizing, and executing vector queries. It furnishes a set of operators tailored for data manipulation on vector tables. Search operators play a pivotal role in supporting similarity queries, enabling nearest neighbor retrieval or range searches. Advanced query types cater to more intricate use cases. Predicated queries amalgamate vector conditions with structured data predicates. Multi-vector queries address scenarios where a single real-world entity is represented by multiple vectors, aggregating scores across these vectors. The query optimizer focuses primarily on predicated queries and systematically explores alternative execution strategies through plan enumeration. The query executor then implements the chosen plan through the coordinated use of distributed architectures and hardware acceleration.

Client Side. The client-side components of a VDBMS provide critical interfaces between end-users/applications and the underlying vector processing infrastructure. VDBMS clients usually provide multi-language SDKs (Python, Go, Node.js, Java) with dual interface protocols REST APIs for metadata operations and gRPC [8] for high-throughput vector transfers. Security combines API key authentication with token-based authorization (JWT/OAuth2), while deployment flexibility spans embedded libraries, local standalone processes, and remote client-server architectures.

2.2 Comparison of Existing VDBMS

Modern VDBMSs exhibit significant architectural and functional diversity, as summarized in Table 1. Our comparison focuses on key systems across critical dimensions that define modern vector data management capabilities, including system type, sub-type, implementation language, maximum supported vector dimension, indexing methods, and support for predicated queries. These diverse features and characteristics pose unique challenges for software testing and validation in the context of VDBMSs.

Various systems are designed as native vector databases (e.g., Pinecone [29], Chroma [4], Weaviate [42], Milvus [22], Qdrant [31]) or extend existing engines (e.g., Redis [32], MongoDB [23], SingleStoreDB [33], Neo4j [26], MyScale [25]) to incorporate vector capabilities. In terms of their data models, some systems are purely vector-oriented, while others adopt a hybrid approach combining vector and scalar data, necessitating comprehensive testing for handling different data types and their interactions. Turning to the implementation aspect, the languages used vary, requiring testing for language-specific issues and compatibility concerns across different technology stacks. Another key factor influencing testing requirements is the maximum supported vector dimension, which ranges from specific limits to arbitrary high dimensions across different systems, demanding testing across diverse dimensionalities including boundary cases and extreme values. Additionally,

VDBMS	PRs	Bug PRs	Ma	nual Ana	Vulnerabilities		
			Storage	Index	Query	Client	
Milvus	9,576	414	26	5	42	35	13
Qdrant	1,035	35	10	0	3	13	8
Chroma	972	126	14	3	28	49	2
Weaviate	1,499	109	29	4	39	21	3
Total	13,082	684	79	12	112	118	26

Table 2. Analysis of Bug-Related Issues and Vulnerabilities in 4 Open Source VDBMSs.

Table 3. Categorization of Bugs by Symptom and Component in Milvus/Qdrant/Chroma/Weaviate.

Symptom	Storage	Index	Query	Client	Total
Crash/Hang	3/7/4/15	1/0/1/2	5/0/3/9	11/4/7/2	74 (23.1%)
Build Failure	0/0/0/0	0/0/0/0	0/0/0/0	0/5/20/6	31 (9.7%)
Incorrect Behavior	3/1/10/10	0/0/2/2	17/2/24/28	16/2/13/8	138 (43.0%)
Performance Issue	6/2/0/4	1/0/0/0	5/0/1/2	7/1/1/0	30 (9.3%)
Others	14/0/0/0	3/0/0/0	15/1/0/0	1/1/8/5	48 (15.0%)

the choice of indexing technique plays a crucial role in balancing search accuracy, performance, and memory efficiency. Indexing techniques like Hierarchical Navigable Small World (HNSW) [21], Inverted File Index (IVF), Product Quantization (PQ), and Approximate Nearest Neighbors Oh Yeah (ANNOY) [34] are employed, with HNSW being particularly popular for large-scale similarity searches. Finally, support for predicated queries, enabling filtered retrieval based on conditions, is an important feature in many vector database applications. Most of the listed systems support predicated queries, requiring rigorous testing of various predicate combinations and complexities to ensure accurate and efficient query processing.

3 Current State: Empirical Study of VDBMS Defects

To understand the current state of software quality in VDBMSs, we conducted an initial empirical study on four prominent open-source projects in Table 1: Milvus, Qdrant, Chroma, and Weaviate. To conduct a comprehensive analysis of bug-related issues and vulnerabilities in VDBMSs, we followed a rigorous process. As shown in Table 2, we first collected data on **Pull Requests (PRs)** from the repositories of four prominent open-source VDBMS projects. We then filtered these PRs using keywords related to bugs to obtain the subset of "Bug PRs". From this subset, we further filtered for closed and merged PRs, which resulted in a set of confirmed bug fixes. We manually analyzed and categorized based on the affected VDBMS components: storage, indexing, query processing, and client-side components. To ensure the accuracy and consistency of our analysis, we developed a set of criteria and guidelines for categorizing bugs based on their manifestation and impact on different VDBMS components. Two researchers independently analyzed and categorized a subset of the reported bugs. Any discrepancies or disagreements in the categorization were discussed and resolved through consensus.

Table 3 presents the categorization of the manually analyzed bugs based on their symptoms and the affected VDBMS components. The data is ordered as Milvus/Qdrant/Chroma/Weaviate across each row. The table reveals several noteworthy patterns and highlights the prevalence of various types of bugs across different VDBMS components:

- Widespread Crash Bugs (23.1%). All four VDBMS systems exhibit a significant number of crash/hang bugs affecting storage, indexing, query processing, and client components. Such severe bugs can render the system completely inoperable, severely impacting its reliability and availability. Notably, the Weaviate system has the highest number of crash/hang bugs (15) related to its storage component. While the oracle design for detecting crashes is relatively straightforward (e.g., monitoring for process termination or unresponsiveness), generating effective test cases that trigger crash/hang bugs in VDBMSs poses unique challenges.
- Prevalence of Incorrect Behavior (43.0%). Bugs that cause the system to exhibit incorrect or unexpected behavior, such as functional errors or incorrect output, are the most prevalent across all VDBMS components. Query processing components are particularly affected, with Milvus (17), Chroma (24), and Weaviate (28) exhibiting a high number of such bugs in query scenarios. Effectively detecting incorrect behavior in VDBMSs poses significant challenges for test oracles. In addition to validating the correctness of query results, test oracles must account for various aspects, including the accuracy of similarity scores, ranking order, and overall result quality. Furthermore, oracles for testing query components must handle diverse data distributions, high-dimensional vectors, and complex similarity functions.
- Performance Degradation (9.3%). Performance-related bugs, which can lead to unacceptable system responsiveness or resource utilization, are consistently present across all VDBMSs. Detecting performance bugs in VDBMSs poses unique challenges. First, it requires generating complex test cases and workloads to construct and query large vector indexes, simulating real-world scenarios. Additionally, defining precise oracles for performance testing is more demanding, as it involves monitoring system metrics, resource consumption, and establishing acceptable performance thresholds.
- Build and Integration Issues (9.7%). While build failures are relatively rare, client-side components in Chroma (20) and Weaviate (6) exhibit a significant number of such issues. These bugs can hinder the integration and deployment of these systems, potentially impacting their adoption and usability. VDBMS testing should include build and integration testing to ensure seamless deployment and usability, especially for client-side components that interface with various applications and environments.

4 Challenges and Future Research Roadmap

The empirical study of bug patterns and defects in prominent open-source VDBMSs reveals several critical challenges that must be addressed to improve the reliability of these emerging systems. As illustrated in Figure 2, we propose the first roadmap for future research on VDBMS testing. Specifically, we will discuss the key challenges and future research opportunities centered around three fundamental aspects: input generation, oracle definition, and test evaluation.

Test Input Generation. Test input generation for vector databases poses significant challenges due to the unique characteristics of vector data and the complexity of operations involved. The challenges and potential research opportunities can be categorized into the following aspects:

• Vector Data Generation. Generating representative input vector data is a critical challenge. High-dimensional vectors may exhibit characteristics like clustering, sparsity, and long-tailed distributions, making it difficult to generate test data that covers all potential scenarios. Research opportunities include developing techniques for generating high-dimensional, diverse, and representative vector data distributions. Additionally, the optimization techniques employed during vector storage and indexing processes may introduce precision errors or optimization traps, necessitating the generation of inputs that expose boundary conditions in floating-point operations, such as values close to zero, overflow, or underflow. Furthermore, VDBMSs support

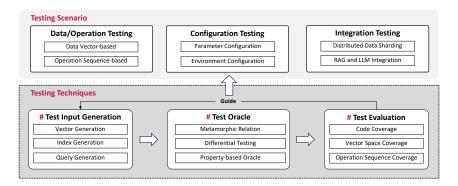


Fig. 2. Challenges and Future Research Roadmap for VDBMS Testing.

varying vector dimensions, requiring the generation of extreme value vectors or large-scale test inputs (e.g., 10 billion vectors) under resource constraints to test boundary conditions.

- Index Generation. Testing index construction in VDBMSs requires considering different index types (e.g., HNSW, IVF, LSH) and their parameter combinations (e.g., M and efConstruction for HNSW), which can lead to an exponential growth in the input space. Exhaustively testing all parameter combinations is impractical, necessitating the selection of representative test inputs. A potential research opportunity lies in exploring strategies for selecting representative test inputs for index construction, considering different index types, parameter combinations, and vector data characteristics. Additionally, scenarios like incremental indexing should be considered when generating targeted test cases.
- Query Generation. Test input generation should cover various query types, including not only basic data operations and search queries but also focusing on complex scenarios such as predicate queries and multi-vector queries. Predicate queries involve combinations of vector search and structured filtering conditions (e.g., WHERE category = A AND vector ≈ [...]), leading to an exponential input space due to the possible combinations of structured conditions (numeric ranges, string matching, boolean logic) and vector search conditions. A potential research opportunity is developing techniques for generating test inputs that cover such complex query scenarios. Multi-vector joint searches (e.g., vector1 ≈ [...] AND vector2 ≈ [...]) require generating semantically related vector pairs (e.g., different modality features describing the same object), with diverse supported operations (e.g., set operations on vector collections, batch similarity computations). Leveraging domain knowledge to guide the generation of semantically meaningful and relevant test inputs for vector databases is another potential research direction.

Test Oracle. Establishing effective test oracles for vector databases presents significant challenges due to the unique characteristics of vector data and the complexity of operations involved. While traditional test oracles for detecting crashes and hangs are still applicable, they account for a relatively small proportion of bugs (23.1%) observed in VDBMSs. The majority of bugs (43.0%) are related to incorrect behavior, necessitating the adaptation of test oracles to the specific features and nuances of vector databases.

Metamophic Relations (MRs) One challenge lies in the design of MRs, which capture the
expected relationships between inputs and outputs of a system under test. In the context of
VDBMSs, the complexity of compound operations, such as transactional operations involving
both insertions and queries, makes it difficult to formalize MRs that accurately capture the expected state consistency. Additionally, the inherent uncertainty introduced by high-dimensional

data perturbations, where minor vector modifications (e.g., adding noise) may alter the similarity ordering, can render MRs ineffective. Future research should focus on developing systematic approaches for designing effective MRs that capture the intricate relationships between inputs, outputs, and system states in vector databases, while accounting for the complexities of compound operations and high-dimensional data perturbations.

- Differential Tesing. Differential testing, which compares the outputs of multiple systems or versions for the same inputs, may face challenges in interpreting result discrepancies. For instance, when performing approximate similarity searches, the difference between the top-k results returned by two systems may be ambiguous in terms of whether it falls within a reasonable error margin or indicates a legitimate bug. This ambiguity arises from factors like varying indexing parameters across systems, rather than inherent defects. Establishing principled techniques for interpreting and resolving ambiguities in result discrepancies, potentially leveraging domain knowledge or statistical methods to distinguish between legitimate bugs and expected approximations, is a crucial research direction.
- Property-based Oracle. Property-based testing, which involves checking if a system exhibits certain desirable properties, can be hindered by the difficulty in defining a comprehensive set of properties for high-dimensional vector data due to its complex mathematical nature. Furthermore, dynamic system states, such as index rebuilding, may lead to time-varying properties, necessitating the consideration of temporal dependencies in property definitions. Future research should explore techniques for eliciting and formalizing properties of vector databases, drawing from domain knowledge, data characteristics, and mathematical properties of high-dimensional spaces, while considering the dynamic nature of system states.

Test Evaluation. Evaluating the effectiveness of testing VDBMSs poses unique challenges. While traditional code coverage metrics (e.g., line coverage, branch coverage) can partially reflect test effectiveness, they struggle to differentiate critical code segments and branches in the context of VDBMSs. API test coverage, which measures the coverage of specific important APIs, may not adequately capture the complexity of API parameter combinations and cross-API state dependencies. Future research should investigate techniques for measuring the coverage of API parameter combinations and cross-API state dependencies, potentially leveraging approaches such as operation sequence coverage or adapted variants. Operation sequence coverage, which measures the coverage of sequences of operations, can be a more informative metric. However, it faces the challenge of prioritization bias, where randomly generated sequences may overly cover simple operations (e.g., single queries) while neglecting high-risk compound operations (e.g., "insert \rightarrow query \rightarrow delete \rightarrow index rebuild"). Additionally, validating long operation sequences can be computationally expensive. Due to the high-dimensional nature of vector data, evaluating vector space coverage may also be necessary. However, effective design and measurement of vector space coverage require careful consideration to ensure adequate coverage of sparse regions or cluster boundaries.

5 Conclusion

VDBMS have emerged as critical enablers for advanced LLM capabilities and data-intensive AI applications. However, the unique characteristics of VDBMS, including high-dimensional vector data, approximate vector search semantics, dynamic data scaling, and integration with complex LLM pipelines, pose significant challenges for software testing. In this paper, we have systematically analyzed these challenges and proposed the first comprehensive research roadmap for future work on VDBMS testing. By addressing the key aspects of test input generation, oracle definition, and test evaluation tailored to VDBMS, the software testing community can contribute to the development of more reliable and trustworthy VDBMS implementations.

References

- Martin Aumüller, Erik Bernhardsson, and Alexander John Faithfull. 2020. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. Inf. Syst. 87 (2020). doi:10.1016/J.IS.2019.02.006
- [2] Fu Bang. 2023. GPTCache: An Open-Source Semantic Cache for LLM Applications Enabling Faster Answers and Cost Savings. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, Liling Tan, Dmitrijs Milajevs, Geeticka Chauhan, Jeremy Gwinnup, and Elijah Rippeth (Eds.). Association for Computational Linguistics, Singapore, 212–218. doi:10.18653/v1/2023.nlposs-1.24
- [3] Junjie Chen, Yihua Liang, Qingchao Shen, Jiajun Jiang, and Shuochuan Li. 2023. Toward Understanding Deep Learning Framework Bugs. ACM Trans. Softw. Eng. Methodol. 32, 6 (2023), 135:1–135:31. doi:10.1145/3587155
- [4] chroma-core. 2025. The AI-native open-source embedding database. https://github.com/chroma-core/chroma. Accessed: 2025-02-21.
- [5] Ziyu Cui, Wensheng Dou, Yu Gao, Dong Wang, Jiansen Song, Yingying Zheng, Tao Wang, Rui Yang, Kang Xu, Yixin Hu, Jun Wei, and Tao Huang. 2024. Understanding Transaction Bugs in Database Systems. In Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14-20, 2024. ACM, 163:1–163:13. doi:10.1145/3597503.3639207
- [6] Shiv Ram Dubey. 2022. A Decade Survey of Content Based Image Retrieval Using Deep Learning. IEEE Trans. Circuits Syst. Video Technol. 32, 5 (2022), 2687–2704. doi:10.1109/TCSVT.2021.3080920
- [7] Xiyue Gao, Zhuang Liu, Jiangtao Cui, Hui Li, Hui Zhang, Kewei Wei, and Kankan Zhao. 2023. A Comprehensive Survey on Database Management System Fuzzing: Techniques, Taxonomy and Experimental Comparison. CoRR abs/2311.06728 (2023). doi:10.48550/ARXIV.2311.06728 arXiv:2311.06728
- [8] grpc. 2025. grpc. https://github.com/grpc/grpc. Accessed: 2025-02-21.
- [9] Jiafeng Guo, Yinqiong Cai, Yixing Fan, Fei Sun, Ruqing Zhang, and Xueqi Cheng. 2022. Semantic Models for the First-Stage Retrieval: A Comprehensive Review. ACM Trans. Inf. Syst. 40, 4 (2022), 66:1–66:42. doi:10.1145/3486250
- [10] Rentong Guo, Xiaofan Luan, Long Xiang, Xiao Yan, Xiaomeng Yi, Jigao Luo, Qianya Cheng, Weizhi Xu, Jiarui Luo, Frank Liu, Zhenshan Cao, Yanliang Qiao, Ting Wang, Bo Tang, and Charles Xie. 2022. Manu: A Cloud Native Vector Database Management System. Proc. VLDB Endow. 15, 12 (2022), 3548–3561. doi:10.14778/3554821.3554843
- [11] Yikun Han, Chunjiang Liu, and Pengfei Wang. 2023. A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge. CoRR abs/2310.11703 (2023). doi:10.48550/ARXIV.2310.11703 arXiv:2310.11703
- [12] Rubing Huang, Chenhui Cui, Junlong Lian, Dave Towey, Weifeng Sun, and Haibo Chen. 2024. Toward Cost-Effective Adaptive Random Testing: An Approximate Nearest Neighbor Approach. *IEEE Trans. Software Eng.* 50, 5 (2024), 1182–1214. doi:10.1109/TSE.2024.3379592
- [13] Ikaros-521. 2024. LanceDBError in Anything-LLM. https://github.com/Mintplex-Labs/anything-llm/issues/1131. Accessed: 2025-02-21.
- [14] Piotr Indyk and Rajeev Motwani. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998, Jeffrey Scott Vitter (Ed.). ACM, 604-613. doi:10.1145/276698.276876
- [15] Piotr Indyk and Rajeev Motwani. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998, Jeffrey Scott Vitter (Ed.). ACM, 604-613. doi:10.1145/276698.276876
- [16] Md Johirul Islam, Giang Nguyen, Rangeet Pan, and Hridesh Rajan. 2019. A comprehensive study on deep learning bug characteristics. In Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019, Marlon Dumas, Dietmar Pfahl, Sven Apel, and Alessandra Russo (Eds.). ACM, 510–520. doi:10.1145/3338906.3338955
- [17] Zhi Jing, Yongye Su, Yikun Han, Bo Yuan, Haiyun Xu, Chunjiang Liu, Kehai Chen, and Min Zhang. 2024. When Large Language Models Meet Vector Databases: A Survey. CoRR abs/2402.01763 (2024). doi:10.48550/ARXIV.2402.01763 arXiv:2402.01763
- [18] Timo Kersten, Viktor Leis, Alfons Kemper, Thomas Neumann, Andrew Pavlo, and Peter A. Boncz. 2018. Everything You Always Wanted to Know About Compiled and Vectorized Queries But Were Afraid to Ask. Proc. VLDB Endow. 11, 13 (2018), 2209–2222. doi:10.14778/3275366.3275370
- [19] Jie Liang, Zhiyong Wu, Jingzhou Fu, Mingzhe Wang, Chengnian Sun, and Yu Jiang. 2024. Mozi: Discovering DBMS Bugs via Configuration-Based Equivalent Transformation. In Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14-20, 2024. ACM, 135:1–135:12. doi:10.1145/3597503.3639112
- [20] Tarek Makkouk, Dong Jae Kim, and Tse-Hsun Peter Chen. 2022. An Empirical Study on Performance Bugs in Deep Learning Frameworks. In IEEE International Conference on Software Maintenance and Evolution, ICSME 2022, Limassol, Cyprus, October 3-7, 2022. IEEE, 35–46. doi:10.1109/ICSME55016.2022.00012
- [21] 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. doi:10.1109/

TPAMI 2018 2889473

- [22] milvus-io. 2025. High-Performance Vector Database Built for Scale. https://github.com/milvus-io/milvus. Accessed: 2025-02-21.
- [23] MongoDB. 2025. Atlas Vector Search. https://www.mongodb.com/products/platform/atlas-vector-search. Accessed: 2025-02-21.
- [24] MyScale. 2023. MyScale Vector Database Benchmark. https://myscale.github.io/benchmark/. Accessed: 2025-02-21.
- [25] MyScale. 2025. Run Vector Search with SQL. https://myscale.com/. Accessed: 2025-02-21.
- [26] Neo4j. 2025. Neo4j Vector Index and Search. https://neo4j.com/labs/genai-ecosystem/vector-search/. Accessed: 2025-02-21.
- [27] James Jie Pan, Jianguo Wang, and Guoliang Li. 2024. Survey of vector database management systems. VLDB J. 33, 5 (2024), 1591–1615. doi:10.1007/S00778-024-00864-X
- [28] James Jie Pan, Jianguo Wang, and Guoliang Li. 2024. Vector Database Management Techniques and Systems. In Companion of the 2024 International Conference on Management of Data, SIGMOD/PODS 2024, Santiago AA, Chile, June 9-15, 2024, Pablo Barceló, Nayat Sánchez-Pi, Alexandra Meliou, and S. Sudarshan (Eds.). ACM, 597–604. doi:10.1145/ 3626246.3654691
- [29] Pinecone. 2025. Pinecone: The vector database to build knowledgeable AI. https://www.pinecone.io/. Accessed: 2025-02-21.
- [30] Qdrant. 2024. Vector Database Benchmarks. https://qdrant.tech/benchmarks/. Accessed: 2025-02-21.
- [31] Qdrant. 2025. Qdrant: High-performance, massive-scale Vector Database and Vector Search Engine for the next generation of AI. https://github.com/qdrant/qdrant. Accessed: 2025-02-21.
- [32] RediSearch. 2025. RediSearch. https://github.com/RediSearch/RediSearch. Accessed: 2025-02-21.
- [33] SingleStore. 2025. The Real-Time Data Platform for Intelligent Applications. https://www.singlestore.com/. Accessed: 2025-02-21.
- [34] spotify. 2024. annoy. https://github.com/spotify/annoy. Accessed: 2025-02-21.
- [35] Toni Taipalus. 2024. Vector database management systems: Fundamental concepts, use-cases, and current challenges. Cogn. Syst. Res. 85 (2024), 101216. doi:10.1016/J.COGSYS.2024.101216
- [36] Florian Tambon, Amin Nikanjam, Le An, Foutse Khomh, and Giuliano Antoniol. 2024. Silent bugs in deep learning frameworks: an empirical study of Keras and TensorFlow. Empir. Softw. Eng. 29, 1 (2024), 10. doi:10.1007/S10664-023-10389-6
- [37] Matthew Siu-Hin Tang, T. H. Tse, and Zhi Quan Zhou. 2023. Detecting Hidden Failures of DBMS: A Comprehensive Metamorphic Relation Output Patterns Approach. In 47th IEEE Annual Computers, Software, and Applications Conference, COMPSAC 2023, Torino, Italy, June 26-30, 2023, Hossain Shahriar, Yuuichi Teranishi, Alfredo Cuzzocrea, Moushumi Sharmin, Dave Towey, A. K. M. Jahangir Alam Majumder, Hiroki Kashiwazaki, Ji-Jiang Yang, Michiharu Takemoto, Nazmus Sakib, Ryohei Banno, and Sheikh Iqbal Ahamed (Eds.). IEEE, 1768–1773. doi:10.1109/COMPSAC57700.2023. 00273
- [38] Guanyu Wang, Yuekang Li, Yi Liu, Gelei Deng, Tianlin Li, Guosheng Xu, Yang Liu, Haoyu Wang, and Kailong Wang. 2024. MeTMaP: Metamorphic Testing for Detecting False Vector Matching Problems in LLM Augmented Generation. In Proceedings of the 2024 IEEE/ACM First International Conference on AI Foundation Models and Software Engineering, FORGE 2024, Lisbon, Portugal, 14 April 2024, David Lo, Xin Xia, Massimiliano Di Penta, and Xing Hu (Eds.). ACM, 12–23. doi:10.1145/3650105.3652297
- [39] Shenao Wang, Yanjie Zhao, Xinyi Hou, and Haoyu Wang. 2024. Large Language Model Supply Chain: A Research Agenda. CoRR abs/2404.12736 (2024). doi:10.48550/ARXIV.2404.12736 arXiv:2404.12736
- [40] Shenao Wang, Yanjie Zhao, Zhao Liu, Quanchen Zou, and Haoyu Wang. 2025. SoK: Understanding Vulnerabilities in the Large Language Model Supply Chain. arXiv:2502.12497 [cs.CR] https://arxiv.org/abs/2502.12497
- [41] Tianshi Wang, Fengling Li, Lei Zhu, Jingjing Li, Zheng Zhang, and Heng Tao Shen. 2024. Cross-Modal Retrieval: A Systematic Review of Methods and Future Directions. Proc. IEEE 112, 11 (2024), 1716–1754. doi:10.1109/JPROC.2024. 3525147
- [42] Weaviate. 2025. Weaviate: Open-source Vector Database. https://github.com/weaviate/weaviate. Accessed: 2025-02-21.
- [43] Xingrui Xie, Han Liu, Wenzhe Hou, and Hongbin Huang. 2023. A Brief Survey of Vector Databases. In 2023 9th International Conference on Big Data and Information Analytics (BigDIA). 364–371. doi:10.1109/BigDIA60676.2023. 10429609
- [44] Yanjie Zhao, Xinyi Hou, Shenao Wang, and Haoyu Wang. 2024. LLM App Store Analysis: A Vision and Roadmap. CoRR abs/2404.12737 (2024). doi:10.48550/ARXIV.2404.12737 arXiv:2404.12737
- [45] zilliztech. 2025. VectorDBBench: A Vector Database Benchmark Tool. https://zilliz.com/benchmark. Accessed: 2025-02-21.