Research Article

Top Ten Challenges Towards Agentic Neural Graph Databases

Jiaxin Bai[1], Zihao Wang[1], Yukun Zhou[1], Hang Yin[2], Weizhi Fei[2], Qi Hu[1], Zheye Deng[1], Jiayang Cheng[1], Tianshi Zheng[1], Hong Ting Tsang[1], Yisen Gao[3], Zhongwei Xie[4], Yufei Li[5], Lixin Fan[6], Binhang Yuan[1], Wei Wang[1], Lei Chen[1], Xiaofang Zhou[1], Yangqiu Song[1]

1. Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong; 2. Department of Mathematical Sciences, Tsinghua University, China; 3. Institute of Artificial Intelligence, Beihang University, Beijing, China; 4. Wuhan University, China; 5. Sichuan University, China; 6. AI Group, WeBank, China

Graph databases (GDBs) like Neo4j and TigerGraph excel at handling interconnected data but lack advanced inference capabilities. Neural Graph Databases (NGDBs) address this by integrating Graph Neural Networks (GNNs) for predictive analysis and reasoning over incomplete or noisy data. However, NGDBs rely on predefined queries and lack autonomy and adaptability. This paper introduces Agentic Neural Graph Databases (Agentic NGDBs), which extend NGDBs with three core functionalities: autonomous query construction, neural query execution, and continuous learning. We identify ten key challenges in realizing Agentic NGDBs: semantic unit representation, abductive reasoning, scalable query execution, and integration with foundation models like large language models (LLMs). By addressing these challenges, Agentic NGDBs can enable intelligent, self-improving systems for modern data-driven applications, paving the way for adaptable and autonomous data management solutions.

Corresponding authors: Jiaxin Bai, jbai@cse.ust.hk; Zihao Wang, zwanggc@cse.ust.hk

1. Introduction

Graph databases like Neo4j[1], TigerGraph[2], and Azure Cosmos DB are useful tools for representing and querying interconnected data using nodes and edges. These databases are adept at handling the complex relationships inherent in graph-structured data, providing efficient mechanisms for storage and retrieval.

 doi.org/10.32388/V8QVHA

A Neural Graph Database (NGDB), as introduced in[3], represents a system architecture that merges the predictive capabilities of Graph Neural Networks (GNNs) with the rich data representation features of graph databases (GDBs). NGDBs enhance graph databases by leveraging GNNs for advanced machine-learning tasks while preserving and utilizing the information embedded within the graph data model.

However, methodologies for conducting inferences within this latent neural space are yet to be thoroughly explored. To address this gap, the integration of neural execution engines on top of neural graph storage has been proposed[4]. By utilizing neural embeddings and neural networks, NGDBs enhance their ability to perform complex reasoning and more effectively infer hidden relationships, which are the capabilities that traditional graph databases lack. This fusion of symbolic graph representations with neural computation paves the way for more intelligent and adaptable data management systems to address contemporary applications' diverse demands. The process of "neuralization" is particularly beneficial for inferring missing information within the underlying graph data model, enriching the database with additional knowledge.

From a broader perspective, the principles of data management systems revolve around efficiently storing, retrieving, and managing data while providing a layer of abstraction to users. These systems aim to handle large volumes of data and complex operations, concealing the underlying complexities from end-users. Motivated by this principle, we propose the concept of **Agentic Neural Graph Databases (Agentic NGDBs)**, extending neuralization to further automate data and data management processes. Here, we summarize the challenges regarding the Agentic NGDB from the following three perspectives interface, learning, and system:

- **Interface:** The Agentic NGDB should automatically construct appropriate queries that generate useful answers for a given task in a specific context.
- **Learning and Inference:** Agentic NGDB should leverage neural networks to execute queries and derive meaningful answers as neural network predictions, even when the underlying data model is incomplete.
- **System:** The Agentic NGDB should remain compatible with existing graph databases, supporting most standard GDB operators. Additionally, it should function as an adaptor for foundation models, enhancing knowledge and reasoning capabilities. Furthermore, it must actively learn by constructing and executing appropriate `CREATE`, `UPDATE`, or `DELETE` queries in a given context.

There are significant challenges to achieving each of these aspects, as illustrated in Figure 1. We identify the most critical challenges for realizing these functionalities based on recent progress in the research community on logical query answering and logical hypothesis generation for relational graphs.

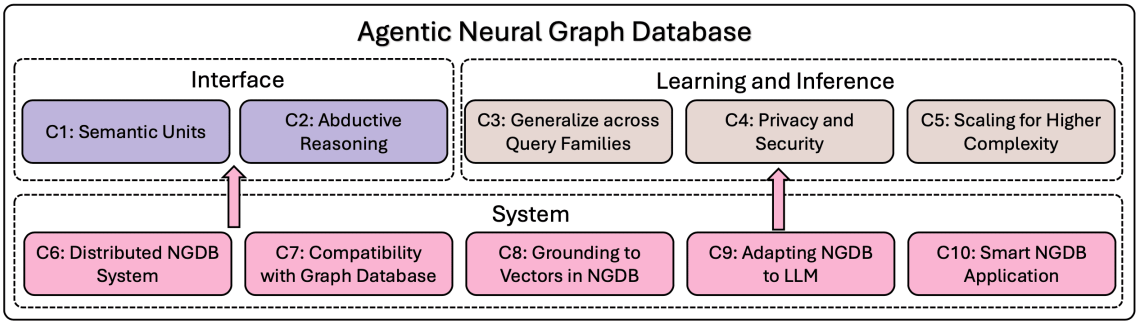

Figure 1. The top ten challenges in achieving Agentic NGDB. Its three perspectives include interface, learning, and system.

Interface

The first significant challenge in the Interface component is addressing *fundamental semantic units* (**Challenge 1**) within the neural graph database's query and data model. Semantic units refer to the data types associated with nodes and edges, such as atomic IDs, text strings (e.g., entities and events), numbers, and dates. Constructing queries that effectively handle these diverse semantic units presents a significant obstacle. Beyond managing individual semantic units, another critical challenge lies in connecting these units to construct more complex queries. The Interface component also requires advanced *abductive reasoning capabilities* (**Challenge 2**). In Agentic NGDBs, abductive reasoning refers to identifying the optimal NGDB query that best explains or supports a specific task in a given context. This capability ensures that the database can adaptively generate meaningful, task-relevant queries. The generated queries can then be executed symbolically by a graph database or through neural execution within the system.

Learning and Inference

Neural query execution is the core functionality of traditional NGDBs, referring to the ability to perform tasks or actions according to a predefined plan or strategy within the neural space[5] by learning and inferences. However, several practical challenges remain in this area. One major challenge is enhancing

inference capabilities for better generalization (**Challenge 3**) across query families. This involves ensuring that NGDB systems can effectively handle and generalize across diverse query structures and types, even when presented with novel or complex combinations of queries. Another critical hurdle involves maintaining *data privacy and security* (**Challenge 4**). Due to their inherent vulnerability to extracting latent knowledge, NGDBs must safeguard sensitive information against advanced inference attacks, particularly in neural models. Robust privacy mechanisms are essential for building trust and ensuring security in NGDB applications. The *scaling laws of neural query execution* (**Challenge 5**) must be explored. Scaling laws in NGDBs describe how the system's performance improves as key factors, such as the number of model parameters, the size of the training dataset, and training costs, are increased. This concept is rooted in neural scaling laws observed in deep learning, where larger models generally lead to better performance, albeit at higher computational costs.

System

The System component focuses on building a system on top of the learning and inference algorithms that can ensure continuous learning and adaptation within Agentic NGDBs. Efficiently processing and managing large-scale data while maintaining high performance becomes especially critical when dealing with massive datasets. This challenge is further amplified in distributed NGDB architectures, where optimizing query performance under read-intensive workloads and dynamically fluctuating demands is necessary. Ensuring elastic scalability and developing NGDBs that operate effectively *as distributed systems* (**Challenge 6**) are key to achieving these goals. These systems must be capable of improving themselves by writing and executing CREATE, UPDATE, and DELETE clauses or performing model editing directly within the neural latent space. The first aspect of this functionality involves ensuring *compatibility with graph database models* (**Challenge 7**). The fundamental CRUD (CREATE, READ, UPDATE, DELETE) actions are essential for managing and modifying persistent data elements in traditional graph databases. Seamlessly integrating these actions into NGDBs is necessary for enabling effective self-improvement. The second aspect involves *grounding vectors within NGDBs* (**Challenge 8**). For effective learning and adaptation, the system must accurately identify the locations of relevant knowledge and understand how reasoning is conducted within the latent neural space. Proper grounding ensures that modifications and updates align with the underlying knowledge representation. Moreover, the Agentic NGDB must be capable of *integrating with foundation models*, such as large language models (LLMs), to enhance its reasoning and knowledge capabilities (**Challenge 9**). The NGDB can provide more reliable and contextually accurate results for various tasks by leveraging foundation models' advanced

natural language understanding and reasoning capabilities. Finally, we discuss the challenge of developing Smart Neural Graph Databases (NGDB) applications (**Challenge 10**), particularly Agentic NGDB. The challenges lie in creating systems that leverage their advanced functionalities across diverse applications.

Agentic NGDBs extend the capabilities of traditional NGDBs by incorporating autonomy, active learning, and adaptability. While NGDBs enhance graph databases by integrating Graph Neural Networks (GNNs) to perform advanced inference and reasoning and handle incomplete or noisy data, they rely heavily on predefined tasks and human-defined queries. In contrast, Agentic NGDBs introduce three core functionalities: automatic query construction tailored to specific tasks and contexts, neural query execution for predictive analysis, and continuous learning and adaptation through active updates to the knowledge base. In the following sections, we will individually discuss each identified challenge.

2. Challenge 1: Semantic Units

The NGDB primarily relies on relational graphs, where nodes and relations are the basic semantic units. Incorporating diverse semantic units, such as numbers and events, introduces complexity due to their intrinsic relationships. For example, numbers involve algebraic operations (e.g., addition, subtraction), while events involve temporal and causal relations. Addressing these complexities requires reasoning engines that can learn and process such relationships effectively.

Number literals (e.g., age, height) are critical for filtering and querying within NGDBs. Prior work includes methods like KBLRN[6], KR-EAR[7], and LitCQD[8], which improve reasoning by integrating numeric constraints into queries. Despite these advancements, challenges remain. These include developing advanced numerical operations and integrating neural-symbolic systems into NGDBs while ensuring compatibility with symbolic solvers for faithful reasoning. Existing approaches focus on entity-centric knowledge graphs, but event-centric knowledge graphs (EVKGs) like ATOMIC[9] and ASER[10] emphasize relationships between events (e.g., temporal and causal relations). Reasoning on EVKGs involves determining event occurrences and their sequences, which introduces unique challenges compared to entity-centric KGs. Recent work extends traditional reasoning by integrating temporal and occurrence constraints[11].

Moreover, beliefs, desires, and intentions (BDI) represent higher-level, abstract semantic units extending beyond simple entity-attribute relationships and eventualities. These elements are crucial for modeling

human-like reasoning, decision-making, and behavior prediction. Beliefs refer to what an agent (human or system) assumes or holds to be true about the world. These can include factual statements like *It is raining outside* and subjective perspectives like *This movie is great*. In KGs, beliefs are often represented as knowledge nodes or statements that may vary across agents or contexts, allowing for personalization or multi-agent reasoning. Intentions represent the goals or purposes behind an agent's actions or decisions and as a bridge between beliefs and actions. Intentions are often implicit and must be inferred from user behavior or contextual information. KGs are typically modeled as motivational nodes or goals that guide reasoning about why an agent performs specific actions. For instance, *PersonX intends [to buy a gift for a friend]*, which could explain why *PersonX searches for [gift shops nearby]*. On the other hand, desires represent an agent's wants, preferences, or needs, which may not always lead to concrete actions unless accompanied by intention. In knowledge graphs, desires are commonly expressed as preferences or motivational entities that influence behavior, such as *PersonX desires [to eat ice cream]*. These three elements allow knowledge graphs to capture human motivations more comprehensively. These concepts are closely connected to the Theory of Mind (ToM), which refers to the ability to understand that other agents (humans, machines, etc.) possess their own beliefs, desires, and intentions that may differ from one's own. In the context of knowledge graphs, the Theory of Mind enhances reasoning about multi-agent knowledge by enabling the understanding of diverse perspectives. Theory of Mind also enables the inference of motivations by reasoning about the interplay between beliefs, desires, and intentions.

Integrating BDI and ToM in Agentic NGDB has practical applications across various domains. In e-commerce, systems like FolkScope[12], COSMO[13], and RIG[14] are the knowledge graphs that leverage BDI to model user behavior, enabling personalized recommendations by linking user actions (e.g., purchases) with inferred desires and intentions. In commonsense reasoning, resources like ATOMIC use BDI to represent cause-effect relationships, allowing systems to reason about potential outcomes of actions. Multi-agent systems benefit from BDI-enhanced KGs by enabling cooperative and competitive interactions that account for multiple agents' goals, beliefs, and desires. Additionally, in natural language understanding, BDI helps interpret user intent in queries, conversations, and social media posts by associating semantic meanings with inferred motivations. We still need systematic storing and inference with these intention knowledge graphs.

3. Challenge 2: Abductive Reasoning with NGDB

Abductive reasoning, the process of inferring the most plausible explanations for observations, is a fundamental aspect of human cognition and artificial intelligence. In the context of knowledge graphs (KGs), abductive reasoning generates hypotheses to explain observations (entity sets) by leveraging structured relationships and entities. Complex Logical Query Answering (CLQA) has further advanced abductive reasoning by enabling multi-hop logical inferences over large, incomplete graphs. Neural Graph Databases (NGDBs) build on these advancements, offering a more flexible and robust framework for abductive reasoning.

Early methods for abductive reasoning in KGs relied on supervised learning and search-based techniques. Generative models, such as transformer-based architectures, were used to produce logical hypotheses. For example,[15] proposed a supervised generative model trained on datasets like FB15k-237 and WN18RR, which excelled in structural fidelity but struggled to generalize to unseen observations due to the limitations of supervised objectives. To address these limitations, reinforcement learning (RL) techniques were introduced. Reinforcement Learning from Knowledge Graph feedback (RLF-KG) employed proximal policy optimization (PPO) to generate hypotheses aligned with observed evidence. This approach improved explanatory power and generalizability, achieving significant gains in metrics like Jaccard similarity and Smatch scores across multiple datasets. NGDBs extend these methods by embedding knowledge graph data in a latent space, enabling flexible query processing and hypothesis generation. By leveraging latent embeddings, NGDBs can infer missing information and generate hypotheses for complex logical queries, even on incomplete graphs, outperforming traditional graph databases. NGDBs represent a significant step forward in abductive reasoning, synthesizing the strengths of CLQA and advanced generative models. However, several challenges must be addressed:

- **More Generalized Observation** In the current definition of abductive reasoning, the definition of the observations is a set of entities. However, observation can be further generalized to a context, for example, a conversation history in the conversational recommendation task setting, or a structured shopping session.
- **More Complex Structured Hypotheses** Existing abductive reasoning models on KGs primarily focus on conjunctive tree-formed queries. NGDBs, with their increased query expressiveness, require hypothesis generation models capable of handling more complex structured observations. For

instance, hypotheses should accommodate EFO_k(existential first-order logic) and cyclic queries, expanding beyond the limitations of earlier models.

- **Graph-Based Hypothesis Generation Models** Traditional sequence-based models struggle to capture the structural complexity of logical hypotheses, which are fundamentally query graphs. These graphs exhibit features like permutation invariance of logical operators, requiring models explicitly designed to generate graph-structured hypotheses.
- **NGDB as a Reward Model for Reinforcement Learning** Previous RL-based methods, such as[15], relied on symbolic execution results from knowledge graphs to provide reward signals during hypothesis generation. However, these reward signals suffer from the incompleteness inherent to the open-world assumption. NGDBs can address this issue by serving as a more robust reward model, leveraging their latent embeddings and flexible query capabilities to improve hypothesis generation.

4. Challenge 3: Generalization across Query Families

Introducing neural modules in graph databases enables the generalization to the *knowledge* in databases. However, the development of neural modules is always entangled with their targeted query families, thus naturally biased toward them due to their inductive biases, emphasizing the challenge of generalizing towards different *query types*. Compared to classic database algorithms that support an entire query family as long as it is formally defined, neural modules still suffer a loss in performance for generalization even when the query family is fixed[16]. Readers are also referred to related surveys[17][18].

4.1. Different Query Families and Their Neural Modules

Tree-formed Queries and Compositional Generalizability. The tree-formed query is a collective term that describes the whole query family that can be recursively defined in a tree structure, in which logical connectives and variables are carefully organized so that set operations can formally derive the answers[16], the set operations include set projection[19], intersection, union[20], complement[20] and set difference[21]. To tackle such kinds of queries, a line of research is known as query embeddings, where sets are modeled as embeddings, and set operations mentioned above are modeled directly by neural modules[22][21][23]. The set operations composition allows the models to generalize the entire tree-form query family. This connection between model design and query family is termed the compositional generalizability[16][24], and the performance drop with the increasing of compositional levels is still universally observed and remains a challenge to address.

EFO-1 Queries and Query Graph. It is shown that tree-formed query family is constrained by certain assumptions and fails to represent the whole family of Existential First Order queries with one free variable (EFO-1 query) such as cyclic query[25]. To handle new graph-theoretic features which cannot be represented in tree-formed queries. One commonly adopted technique for EFO-1 queries is the DNF normal form or the UCQ query-solving strategy[20][26], which solves the conjunctive query first and then takes the union of the answer set of each conjunctive query. A query graph[26] can naturally describe each conjunctive query. This formulation motivates graph-related search methods[25] or graph neural networks[26].

More Advanced Query Types. More advanced query families still exist, though the development of corresponding neural models on these topics is insufficient at the current stage. Thus, we discuss some of the challenges we might face in pursuit of more advanced queries in NGDB from the following aspects (i) *Multi-arity predicates:* The first challenge we may encounter is when the knowledge databases are constructed by $(n+1)$-ary tuples, the relation corresponds to n-ary predicate and a graph becomes a hypergraph[27]. (ii) *Support of functions* The corresponding research gap is the support for functions in the query – a function can output nodes, numbers, semantic units, or data of more advanced modality – for example, the AVG and COUNT functions in SQL but not in current CQA models. We have noted one preliminary research trying to fill this gap[8].

4.2. Minimal Assumption for Broad Generalization

Previous case studies showcase the close entanglement of the neural modules and the query types they support syntactically. In other words, the key to generalization is minimizing the query families' assumptions and the inductive biases of the neural part of NGDB. We present two types of methods with minimal assumptions.

Neuro-symbolic Methods. NGDB implies that the underlying database is a graph, meaning neural modules solely modeling the graph itself impose no assumptions on the query family it might support. Such neural modules include link predictors or knowledge graph embeddings that map a triple (s, p, o) of subject, predicate, and object into a score[28]. Therefore, the critical design task of NGDB with such modules is revising the algorithms into the neuro-symbolic forms with the scores produced by link predictors[29][25]. An apparent and more decomposed approach is to derive an instance of a classic graph database using the link predictor, and all previous research in graph databases applies directly. Notably,

the neuro-symbolic approach achieves the same level of generalizability in queries as the classic database research.

Sequence Models. Language models or sequence models are general-purpose models and thus further disentangle the inductive bias of neural modules and the specific task (queries in NGDB). Such models support the sequence inputs, which cover inputs from all possible kinds of query types. However, the performance on specific query types is transferred from designing neural architectures to curating the training datasets. The cost is transferred from the complex inference algorithms to the training phase[30].

4.3. Learning Aspects of Generalization

From the machine learning perspective, one new issue is uniformly improving the performance of all queries of a particular query family under the analogy of query types as tasks. The approach towards this goal also varies for different methods. For neuro-symbolic approaches, the generalization will be improved coherently as link prediction performance improves. For neural methods, the challenge of generalization is the same as multi-task learning. Query embeddings, as a particular case of neural methods, recent works propose adopting set operators with meta-learning, yielding the solution of meta operators[24].

5. Challenge 4: Privacy and Security

5.1. Database Privacy

Privacy in data storage refers to protecting sensitive information from unauthorized access and misuse[31]. Traditional databases are facing several privacy risks, which can be categorized into: (1) Unauthorized Access[32]: Unauthorized access to databases can result in large-scale data leakage, exposing sensitive personal information. (2) Insider Threats[33]: Employees with legitimate access may misuse their privileges, either intentionally or unintentionally compromising data privacy. (3) Data Inference Attacks[34]: Attackers can employ various techniques to deduce sensitive information from seemingly innocuous data.

To mitigate privacy risks, several protection methods have been developed: (1) Data Anonymization[35]: Techniques such as k-anonymity[36] and l-diversity[37] help mask individual identities within datasets,

making it harder to trace data back to specific individuals. (2) Encryption[38]: Data encryption ensures that unauthorized parties cannot access sensitive information even if they breach a database. (3) Access Control[32]: Access control restricts data access to authorized users only, reducing the risk of insider threats. (4) Differential Privacy[39]: This approach adds noise to data outputs, ensuring that the presence or absence of an individual in a dataset does not significantly affect the results of queries.

5.2. New Privacy Challenges in NGDBs

Graph databases, while offering advantages in managing complex relationships, introduce specific privacy risks: (1) Link Prediction Attacks[40]: Adversaries can use machine learning models to predict hidden relationships within the graph, potentially uncovering private connections. (2) Structural Attacks[41]: Even when the data content is anonymized, the graph's structure itself can reveal sensitive insights. The unique structure of graph data amplifies these risks, as the relationships between entities can reveal information that is not immediately apparent from isolated data points. Neural Graph Databases (NGDBs) represent a significant advancement in data management, combining the strengths of traditional graph databases with the capabilities of neural networks. The exploration of privacy issues in NGDBs remains largely underdeveloped, with significant gaps in research addressing potential vulnerabilities and mitigation strategies.

Potential Attacks. One of the primary strengths of NGDBs is their ability to generalize from incomplete data by inferring hidden relationships. While this capability can enhance data retrieval and knowledge discovery, it also poses significant privacy risks[42]: (1) Model Inversion Attacks[43]: Neural models can be susceptible to inversion attacks, where an adversary uses access to the model to recover the graph data used for NGDB training. (2) Membership Inference Attacks[44]: Attackers may infer whether a particular data point (node or edge) was part of the training data, revealing sensitive information in NGDBs. (3) Embedding Leakage[45]: The embeddings generated by NGDBs to represent nodes and relationships can leak sensitive information, as these embeddings often capture detailed structural and content-based features of the graph stored.

Promising Defenses. (1) Differential Privacy in NGDBs: Extending differential privacy techniques to protect neural graph databases is a key research direction. Adding noise to the model parameters or gradients during training can help mitigate membership inference and model inversion attacks[46][47]. (2) Embedding Obfuscation: Techniques to obfuscate embeddings without losing their utility for answering

complex queries need to be developed to prevent leakage of sensitive information[48]. (3) Private Distribute Training: Privacy problems in distributed NGDBs need further development[49]. Federated learning, including Secure Multi-Party Computation (SMPC)[50] and Homomorphic Encryption (HE) [51] techniques, can be adapted to NGDBs to ensure that data is processed without being revealed.

Evaluation Benchmarks. Another significant challenge in NGDBs is the evaluation of privacy protection efficacy. Assessing the effectiveness of privacy-preserving mechanisms requires robust benchmarks that can accurately measure both privacy protection and the quality of retrieved data. However, such benchmarks are currently lacking in the field. To address this challenge, standardized evaluation metrics and datasets should be developed that can facilitate comprehensive testing of privacy-preserving techniques in NGDBs. Establishing reliable benchmarks will provide insights into the strengths and weaknesses of different approaches, ultimately guiding future developments in privacy protection.

6. Challenge 5: Scaling for Higher Complexity

In deep learning, neural scaling law is an empirical law that describes the performance of neural models improves with the number of parameters, training dataset size, and training cost[52][53]. During the development of the NGDB model, scaling is also a major thread, primarily encompassing the scaling of parameter number, query data size, and training costs. The query embedding methods and sequence models often scale the training costs in the training stage, including the model parameters and queries. In contrast, the neuro-symbolic methods often scale the computation cost over the test stage to improve the performance. We mainly discuss how to scale these models further, particularly when the query structure becomes increasingly complex[25][54] and the magnitude of the knowledge databases becomes very large[55]. Specifically, we introduce the complexity of these models in the training and inference stages and discuss their efficiency and scalability challenges.

Data Scaling in the Training Stage. Both query embedding and sequence models are trained from scratch, requiring many sampled queries as training data. The quality and size of these training queries are crucial, and they typically encompass various query types. The NGDB models generally use the same dataset, with the basic 1p query type enumerating the entire knowledge graph[20]. To incorporate new features such as negation[22], cyclic queries[25], and multivariable queries[54], it is essential to sample query types that include these features. Materializing training queries becomes infeasible as the knowledge graph grows, and sampling logical queries is incompatible with traditional single-hop

frameworks based on graph partitioning. To address this challenge, SMORE[55] proposes a scalable framework that efficiently samples training data on the fly with high throughput. In contrast, neuro-symbolic methods primarily rely on pre-training for the knowledge graph completion task and depend on search algorithms to address general logical queries.

Test Time Scaling in Inference Stage. We first introduce the notion of query complexity and data complexity[56]. Data complexity captures the relation between the time complexity and the database size $|E|$ (number of the edges) when the query is fixed. In contrast, query complexity is assessed based on the size of the query $|Q|$ (number of the predicates) when assuming the database is fixed. When discussing the complexity, the query is restricted to tree-formed queries and EFO-1 queries that we have discussed before. The complexity of neural symbolic search is well studied. The complexity for tree-formed queries is $O(|Q||E|)$. Such approaches[25][29] require $O(|Q|)$ search steps, while each step requires a search over the database, which is $O(|E|)$. For the general EFO-1 query, the cyclic query makes the general complexity particularly hard and results in $O(|E|^{|Q|})$ time, which is polynomial in data but exponential in query. One distinct feature of query embeddings and sequence models compared to neuro-symbolic methods is the disentanglement of the $|E|$ term and $|Q|$ term. Notably, the neural network encoder[26] or sequence model[30] work on the query directly, which is usually $O(|Q|)$ to encode query information and $O(|E|)$ to decode the answer by embedding comparison. However, this great advantage in inference time complexity of query embeddings and neural symbolic models comes from the additional and usually resource-consuming training procedures.

7. Challenge 6: Distributed NGDB System

7.1. Scenario Features and System Requirements

Scenario Features. NGDB is targeted at a scenario where users can simultaneously conduct graph data management and graph inference. We identify four features of such a scenario that significantly affect the system design. (1) Hybrid symbolic and neural operation[4]. Users can input queries requiring algebraic, neural, or hybrid computation; (2) Massive graph data and embeddings. Not only do the graph data of different domain knowledge exhibit tremendous scale[57], but also various types of embeddings[58] of these graph data further enlarge the volume; (3) Read intensive workload. During the serving stage, most of the graph data and embeddings are queried more frequently rather than updated[59]; (4) Dynamic workload fluctuation. Different parts of the graph data and embeddings are

accessed in different time slots and the number of online users and frequency and data volume of one query fluctuate[60].

System Requirements. The neural graph database system should fulfill the following requirements to handle these features effectively and efficiently. (1) Co-located graph and embedding management. The NGDB system should support symbolic graph data and neural embedding management. (2) High query performance. The latency of a single query and system throughput for numerous tenants serving massive data should be optimized. (3) Scalability.. The hardware resource management should be scalable to handle workload fluctuation, especially computational resources, cost-efficiently. Challenges are introduced to the system design of neural graph databases to implement these system features.

7.2. Challenges of System Design

User Interface Design. Existing vector databases provide SQL-like interfaces and parameterized API[61], while most of the interfaces mainly focus on relational data. Graph databases provide numerous interfaces[58], but there is little experience in combining neural operations into symbolic graph operations. It is essential to design highly expressive declarative user interfaces as well as programming interfaces.

Query-Oriented Distributed Storage. Due to the massive volume of graph data and corresponding embeddings, which is out of the capacity of standalone storage, distributed storage is an indispensable mechanism of NGDB. Under read-intensive workload, partitioning (or sharding), acting as a distributed index, tailored for most frequent and costly types of query could remarkably reduce the intermediate data transfer, consequently enhancing the overall latency and throughput[61]. Practices in graph database community[62][63][64][65] and vector database community concludes valuable principles and strategies on distributed storage and indexing of graph data and embedding separately. However, the hybrid storage of both data types is not explored, especially in circumstances where hybrid queries, requiring both symbolic and neural processes, are of evident importance. A typical example question is about whether embeddings and raw graph data shall be co-located. Although some open source graph database[66][67] and vector databases[68][60][69][70] could be utilized as standalone storage engine in NGDB, partitioning should be carefully designed under specific query workload.

Distributed Graph Computing and Inference. There are abundant works on distributed graph query and analysis with various algorithms in the graph database community[71][72]. However, distributed system

support for knowledge graph inference has not been adequately explored. Atom[73] points out a key observation that query embedding is the performance bottleneck, which shall be one of the considerations in NGDB query execution. On the base of these two kinds of computation optimization, when encountered with hybrid queries requiring both computation, query planning, and scheduling for maximized parallelism and minimized network communication overhead, still remain an unexplored direction. There are some preliminary practice cases in relational databases[74][75][76], which consider the optimization with neural operators but are still far from mature.

Elastic Scalability. To deal with dynamic workload fluctuation, fine-grained elasticity is of great importance to distributed NGDB systems, in which case on-demand resource provision helps reduce the cost[77] of NGDB service. Besides, not all the massive data are simultaneously accessed. There are evident biases and data heat shifts in database serving scenarios. Therefore, we argue that being cloud-native with elastic scalability is a crucial requirement for the NGDB system. Manu[60] detects such workload fluctuation in industrial applications, thus fully embracing the mechanisms of elastic scalability via dedicated abstraction of hardware resource management, including GPU, CPU, and disk. Besides, storage-computation-separation is essential for cloud-native databases[78]. It is essential to explore the combination of these separate practices. Additionally, the trade-off between latency and elasticity is a critical concern since practices in vector databases reveal that embedding management requires a large memory occupation.

8. Challenge 7: Compatibility of NGDB with Traditional Graph Database

Like graph databases, Neural Graph Databases (NGDB) are another way of the data model that derives the properties from the existing graphs, including nodes and edges, to represent entities and their relationships[4]. This structural consistency makes migrating and interoperating data between the two databases relatively easy. In terms of interfaces, NGDB can maintain support for standard graph query languages[66][79] while providing vectorized query capabilities, allowing users to query and operate in familiar languages. In terms of operations, traditional CRUD operations remain fully functional, with the reasoning function of neural networks serving as enhanced features. For instance, conventional graph databases provide foundational support in query processing through mature storage and indexing technologies, while NGDB handles queries requiring missing link inference. Such compatibility design

will enable a seamless system transition, where users can migrate to get NGDB capabilities without completely reconstructing existing applications. However, NGDB faces several challenges with traditional graph databases:

Novel Query Interface. Incorporating deep learning and graph neural networks extends beyond conventional graph database functionalities, requiring novel interfaces for deep learning-based queries and inference. This creates compatibility issues when attempting to reuse existing query languages, highlighting the need to develop new query languages or extend current ones[80][79].

Performance-Consistency Trade-off. While traditional graph databases are optimized for storage and querying[81], they may struggle to meet performance requirements when handling large-scale graph-based deep-learning tasks. NGDB emphasizes representation learning on nodes and edges[4], requiring consideration of high-performance computing and distributed training paradigms. For instance, during conventional CRUD operations, NGDB may need to update node and relation embeddings, introducing additional computational overhead. Moreover, integrating neural components introduces temporal consistency challenges, where model updates may lead to temporary discrepancies between the base graph data and learned representations. Finding an optimal balance between consistency guarantees and computational efficiency remains a considerable challenge for NGDB systems.

9. Challenge 8: Grounding to Vectors with NGDB

Grounding natural language to knowledge bases has been extensively studied in conventional graph databases. Traditional approaches typically handle different grounding scenarios: hypothesis or query grounding (with free variables)[82][20], and entity[83] or event[11][84]). With the emergence of NGDBs, where structural information and semantic content are encoded as vectors, the grounding process faces new challenges and opportunities. Recent work[4] introduces a neural graph engine that learns query planning and execution strategies through interactions with Neural Graph Storage. However, grounding to general NGDBs still presents several unique challenges.

Semantic Granularity and Disambiguation. Semantic granularity and disambiguation pose fundamental difficulties. The grounding process must accurately translate natural language queries into appropriate vector representations while determining suitable levels of semantic granularity, such events, propositions, etc.[85][84]. This challenge is compounded by the need to handle abstraction and polysemy when mapping linguistic elements to vector spaces, as meanings can vary significantly based on context.

Compositional Semantics and Reasoning. Second, compositional semantics and reasoning path selection present significant challenges. NGDBs must effectively represent complex multi-hop relations while maintaining transitivity and logical consistency in vector operations. The system needs to identify relevant paths in the vector space for query resolution, which becomes particularly challenging when dealing with multiple possible reasoning paths. In addition, determining appropriate termination criteria for path exploration is crucial for both efficiency and accuracy.

Interpretation and Groundedness Evaluation. The third challenge is around interpretation and groundedness evaluation. The system is expected to reliably convert vector-based results back to natural language while providing clear explanations for its reasoning process. Additionally, it needs to report the level of groundedness for each grounding operation, ensuring semantic fidelity is maintained throughout the process. This is particularly important for applications requiring high precision and explainability.

10. Challenge 9: Adapting NGDB to LLM

This section explores the integration of Neural Graph Databases (NGDBs) with Large Language Models (LLMs) to enable joint reasoning and Retrieval-Augmented Generation (RAG). NGDBs can serve as retrieval modules for LLMs, leveraging structured data and reasoning capabilities to enhance generated outputs' accuracy, scalability, and contextual relevance. Joint learning of LLMs and NGDBs involves training these systems within a unified framework to combine natural language understanding with advanced logical reasoning.

NGDB-RAG: Definition and Components. NGDB-RAG (Neural Graph Database - Retrieval-Augmented Generation) is a system that integrates NGDBs with LLMs to enhance both retrieval and generation tasks. The NGDB-RAG system is composed of three main components. The first is the neural graph storage, which stores embeddings of nodes and edges in the graph. These embeddings capture both local and global structural relationships within the graph, providing a rich representation of the data. The second component is the neural query engine, which tries formulating and processing logical queries in the embedding space. This engine enables flexible modeling and supports logical operations such as conjunction, disjunction, and negation, allowing for robust retrieval even in incomplete or noisy graphs. The third component is integrating with LLMs, where NGDB reasoning results are incorporated into the language model. This integration can be achieved through text-based methods, by converting structured

data into natural language, or through vector-based methods, by embedding structured data as vectors for direct input into the LLM.

Functionality of NGDB-RAG. NGDB-RAG enhances retrieval by utilizing the structured relationships in NGDBs to perform advanced reasoning tasks. Unlike traditional RAG systems that rely on document similarity, NGDB-RAG leverages the intricate dependencies within knowledge graphs to retrieve more accurate and contextually relevant information. In the generation process, NGDB-RAG integrates structured knowledge and reasoning capabilities from NGDBs to improve the generated text's factual accuracy and logical consistency while reducing hallucinations. Furthermore, NGDB-RAG is designed to handle large-scale graphs and supports various query types, including temporal, spatial, and numerical reasoning, ensuring scalability and expressiveness in practical applications.

Joint Learning Framework. The joint learning framework of NGDBs and LLMs employs a co-training approach where both systems share parameters or representation spaces to enable collaborative learning. Improvements in one component positively influence the other, creating a feedback loop that enhances the overall system. The combined training objective is expressed as: $L_{\text{total}} = L_{\text{LLM}} + \lambda L_{\text{NGDB}}$. In this equation, L_{LLM} represents the loss associated with the language model, typically the cross-entropy loss for next-token prediction. L_{NGDB} denotes the loss related to NGDB reasoning tasks, such as the error between predicted and true query answers. The hyperparameter λ controls the balance between the two loss components. The objective of this joint training is to improve the reasoning capabilities of the NGDB while enhancing the LLM performance.

Future work aims to develop the co-training framework further to enable simultaneous training of NGDB reasoning engines and LLMs, ensuring parameter sharing and collaborative learning. Efforts are also being made to refine the combined loss function to balance language modeling and reasoning tasks better, enhancing both components' performance. Integration modules are being developed to incorporate NGDB reasoning results into LLMs through text-based and vector-based methods. These advancements are expected to create a unified system capable of performing advanced reasoning and generating high-quality, contextually accurate text.

11. Challenge 10: Smart Neural Graph Databases

Benefited from its rich functionalities, Agentic NGDB offers a wide range of applications across domains:

- **Autonomous Data Management:** Agentic NGDB can autonomously manage complex datasets, optimize query execution, and organize storage structures without human intervention. This is particularly useful in large-scale systems where manual optimization is impractical.
- **Personalized Recommendations:** Through continuous learning, Agentic NGDB can provide real-time personalized recommendations by analyzing user preferences and graph-based relationships. This is crucial in e-commerce and social networks, where tailored experiences drive user engagement[86].
- **Complex Event Processing:** Agentic NGDB is well-suited for handling complex event processing[11], where multiple events and data streams need to be analyzed in real-time. By leveraging their semantic understanding and neural inference, Agentic NGDB can identify correlations and patterns across seemingly unrelated events, making them valuable in cybersecurity, fraud detection, and IoT systems.

12. Conclusion

Agentic Neural Graph Databases (Agentic NGDBs) represent an advancement in data management, building on traditional graph databases and Neural Graph Databases (NGDBs) by introducing autonomy, continuous learning, and advanced reasoning.

This paper identifies ten key challenges to realizing Agentic NGDBs, including semantic representation, abductive reasoning, generalization across query types, scalability, privacy, and integration with foundation models like large language models (LLMs). Ensuring compatibility with traditional databases, grounding knowledge in vectors, and developing distributed systems are essential for achieving robust and scalable solutions.

By overcoming these challenges, Agentic NGDBs can transform modern data-driven applications. Their ability to autonomously generate and execute queries, support continuous learning, and integrate symbolic and neural reasoning offers new possibilities in autonomous data management, personalized recommendations, and complex event processing. These advancements promise to redefine how we manage, query, and reason over interconnected data for the future.

References

1. ^*Miller JJ (2013). "Graph database applications and concepts with Neo4j". In: Proceedings of the southern as sociation for information systems conference, Atlanta, GA, USA. **2324**: 141–147.*

2. ^Deutsch A, Xu Y, Wu M, Lee VE (2019). "TigerGraph: A Native MPP Graph Database". ArXiv. **abs/1901.08248**. S2CID 59222855.
3. ^Besta M, Iff P, Scheidl F, Osawa K, Dryden N, Podstawski M, Chen T, Hoefler T. "Neural Graph Databases." In: Rieck B, Pascanu R, editors. Learning on Graphs Conference, LoG 2022, 9-12 December 2022, Virtual Event. Proceedings of Machine Learning Research. 2022; **198**:31. Available from: https://proceedings.mlr.press/v198/besta22a.html. [cited 2023 Feb 17].
4. a, b, c, d, eRen H, Galkin M, Cochez M, Zhu Z, Leskovec J (2023). "Neural graph reasoning: Complex logical query answering meets graph databases". arXiv preprint arXiv:2303.14617.
5. ^Russell SJ, Norvig P. Artificial intelligence: a modern approach. Pearson; 2016.
6. ^García-Durán A, Niepert M. KBlrn: End-to-End Learning of Knowledge Base Representations with Latent, Relational, and Numerical Features. In: Globerson A, Silva R, editors. Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018. AUAI Press; 2018. p. 372-381. Available from: http://auai.org/uai2018/proceedings/papers/149.pdf. [cited 2020 Dec 15]. Available from: https://dblp.org/rec/conf/uai/Garcia-DuranN18.bib.
7. ^Lin Y, Liu Z, Sun M. Knowledge representation learning with entities, attributes and relations. In: Kambhampati S, editor. Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016. IJCAI/AAAI Press; 2016. p. 2866-2872. Available from: http://www.ijcai.org/Abstract/16/407. [cited 2019 Aug 20].
8. a, bDemir C, Wiebesiek M, Lu R, Ngonga Ngomo A-C, Heindorf S. LitCQD: Multi-hop Reasoning in Incomplete Knowledge Graphs with Numeric Literals. In: Koutra D, Plant C, Gomez Rodriguez M, Baralis E, Bonchi F, editors. Machine Learning and Knowledge Discovery in Databases: Research Track - European Conference, ECML PKDD 2023, Turin, Italy, September 18-22, 2023, Proceedings, Part III. Lecture Notes in Computer Science. Springer; 2023. p. 617-633. doi:10.1007/978-3-031-43418-1_37. Available from: https://dblp.org/rec/conf/pkdd/DemirWLNH23.bib.
9. ^Sap M, Le Bras R, Allaway E, Bhagavatula C, Lourie N, Rashkin H, Roof B, Smith NA, Choi Y (2019). "ATOMIC: An Atlas of Machine Commonsense for If-Then Reasoning." In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019. AAAI Press. pp. 3027–3035. doi:10.1609/aaai.v33i01.3301302 7. Source.

10. ^Zhang H, Liu X, Pan H, Song Y, Leung CW. "ASER: A Large-scale Eventuality Knowledge Graph." In: Huang Y, King I, Liu T, van Steen M, editors. WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020. ACM / IW3C2; 2020. p. 201-211. doi:10.1145/3366423.3380107. Available from: https://dblp.org/rec/conf/www/ZhangLPSL20.bib.

11. a, b, cBai J, Liu X, Wang W, Luo C, Song Y. "Complex Query Answering on Eventuality Knowledge Graph with Implicit Logical Constraints." In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S, editors. Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023. 2023. Available from: http://papers.nips.cc/paper_files/paper/2023/hash/6174c67b136621f3f2e4a6b1d3286f6b-Abstract-Conference.html. Source: dblp computer science bibliography.

12. ^Yu C, Wang W, Liu X, Bai J, Song Y, Li Z, Gao Y, Cao T, Yin B. "FolkScope: Intention Knowledge Graph Construction for E-commerce Commonsense Discovery". In: Findings of the Association for Computational Linguistics: ACL 2023. Toronto, Canada: Association for Computational Linguistics; 2023. p. 1173–1191. Available from: https://aclanthology.org/2023.findings-acl.76.

13. ^Yu C, Liu X, Maia J, Cao T, Li L(Y), Gao Y, Song Y, Goutam R, Zhang H, Yin B, Li Z (2024). "COSMO: A large-scale e-commerce common sense knowledge generation and serving system at Amazon". https://www.amazon.science/publications/cosmo-a-large-scale-e-commerce-common-sense-knowledge-generation-and-serving-system-at-amazon.

14. ^Bai J, Wang Z, Cheng J, Yu D, Huang Z, Wang W, Liu X, Luo C, He Q, Zhu Y, et al. Intention knowledge graph construction for user intention relation modeling. arXiv preprint arXiv:2412.11500. 2024.

15. a, bBai J, Wang Y, Zheng T, Guo Y, Liu X, Song Y. "Advancing abductive reasoning in knowledge graphs through complex logical hypothesis generation." In Proceedings of the 62st Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics. 2024.

16. a, b, cWang Z, Yin H, Song Y (2021). "Benchmarking the Combinatorial Generalizability of Complex Query Answering on Knowledge Graphs". Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks. **1**. Available from: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/7eabe3a1649ffa2b3ff8c02ebfd5659f-Abstract-round2.html.

17. ^Wang Z, Yin H, Song Y (2022). "Logical Queries on Knowledge Graphs: Emerging Interface of Incomplete Relational Data." IEEE Data Eng. Bull..

18. ^Liu L, Wang Z, Tong H (2024). "Neural-Symbolic Reasoning over Knowledge Graphs: A Survey from a Query Perspective". arXiv preprint arXiv:2412.10390.

19. ^Hamilton W, Bajaj P, Zitnik M, Jurafsky D, Leskovec J (2018). "Embedding logical queries on knowledge graphs". Advances in neural information processing systems. **31**.
20. a, b, c, d, eRen H, Hu W, Leskovec J. "Query2box: Reasoning Over Knowledge Graphs In Vector Space Using Box Embeddings". In: International Conference on Learning Representations (ICLR); 2020.
21. a, bLiu L, Du B, Ji H, Zhai C, Tong H (2021). "Neural-Answering Logical Queries on Knowledge Graphs". In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 1087–1097.
22. a, bRen H, Leskovec J (2020). "Beta embeddings for multi-hop logical reasoning in knowledge graphs". Advances in Neural Information Processing Systems. **33**: 19716–19726.
23. ^Wang Z, Fei W, Yin H, Song Y, Wong G, See S (2023). "Wasserstein-Fisher-Rao Embedding: Logical Query Embeddings with Local Comparison and Global Transport". In: Findings of the Association for Computational Linguistics: ACL 2023. pp. 13679–13696.
24. a, bYin H, Wang Z, Song Y (2024). "Meta Operator for Complex Query Answering on Knowledge Graphs". arXiv preprint arXiv:2403.10110.
25. a, b, c, d, e, fYin H, Wang Z, Song Y. "Rethinking Existential First Order Queries and their Inference on Knowledge Graphs." In: The Twelfth International Conference on Learning Representations, 2024.
26. a, b, c, dWang Z, Song Y, Wong GY, See S (2023). "Logical message passing networks with one-hop inference on atomic formulas". arXiv preprint arXiv:2301.08859. Available from: arXiv:2301.08859.
27. ^Luo H, Yang Y, Zhou G, Guo Y, Yao T, Tang Z, Lin X, Wan K, et al. NQE: N-ary Query Embedding for Complex Query Answering over Hyper-relational Knowledge Graphs. arXiv preprint arXiv:2211.13469. 2022.
28. ^Wang Q, Mao Z, Wang B, Guo L (2017). "Knowledge graph embedding: A survey of approaches and applications". IEEE Transactions on Knowledge and Data Engineering. **29** (12): 2724–2743.
29. a, bBai Y, Lv X, Li J, Hou L. "Answering Complex Logical Queries on Knowledge Graphs via Query Computation Tree Optimization." In: Proceedings of the 40th International Conference on Machine Learning. PMLR; 2023. p. 1472-1491. Available from: https://proceedings.mlr.press/v202/bai23b.html. ISSN: 2640-3498.
30. a, bBai J, Zheng T, Song Y (2023). "Sequential Query Encoding for Complex Query Answering on Knowledge Graphs". Transactions on Machine Learning Research. Available from: https://openreview.net/forum?id=ERqGqZzSu5. ISSN 2835-8856.
31. ^Olivier MS (2002). "Database privacy: balancing confidentiality, integrity and availability". ACM SIGKDD Explorations Newsletter. **4** (2): 20–27.
32. a, bBertino E, Ghinita G, Kamra A, et al. Access control for databases: Concepts and systems. Foundations and Trends® in Databases. **2011**; **3**(1–2):1–148.

33. ^Senator TE, Goldberg HG, Memory A, Young WT, Rees B, Pierce R, Huang D, Reardon M, Bader DA, Chow E, et al. Detecting insider threats in a real corporate database of computer usage activity. In: Proceedings of th e 19th ACM SIGKDD international conference on Knowledge discovery and data mining. 2013. p. 1393–1401.

34. ^Naveed M, Kamara S, Wright CV (2015). "Inference attacks on property-preserving encrypted databases". I n: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 644–6 55.

35. ^Murthy S, Abu Bakar A, Abdul Rahim F, Ramli R. A comparative study of data anonymization techniques. In: 2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS). IEEE; 2019. p. 306-309.

36. ^Sweeney L (2002). "k-anonymity: A model for protecting privacy". International journal of uncertainty, fu zziness and knowledge-based systems. **10** (05): 557–570.

37. ^Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M (2007). "l-diversity: Privacy beyond k-ano nymity". ACM Transactions on Knowledge Discovery from Data (TKDD). **1** (1): 3–es.

38. ^Basharat I, Azam F, Muzaffar AW (2012). "Database security and encryption: A survey study". Internationa l Journal of Computer Applications. **47** (12).

39. ^Dwork C. "Differential privacy." In: International colloquium on automata, languages, and programming. Springer; 2006. p. 1–12.

40. ^Lin W, Ji S, Li B (2020). "Adversarial attacks on link prediction algorithms based on graph neural network s". In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security. pp. 370–38 0.

41. ^Zhao K, Zhou H, Zhu Y, Zhan X, Zhou K, Li J, Yu L, Yuan W, Luo X (2021). "Structural attack against graph b ased android malware detection". Proceedings of the 2021 ACM SIGSAC conference on computer and comm unications security. pp. 3218–3235.

42. ^Hu Q, Li H, Bai J, Wang Z, Song Y (2024). "Privacy-Preserved Neural Graph Databases". In: Proceedings of t he 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 1108–1118.

43. ^Fang H, Qiu Y, Yu H, Yu W, Kong J, Chong B, Chen B, Wang X, Xia ST, Xu K (2024). "Privacy leakage on dnns: A survey of model inversion attacks and defenses". arXiv preprint arXiv:2402.04013. Available from: arXiv:2 402.04013.

44. ^Hu H, Salcic Z, Sun L, Dobbie G, Yu PS, Zhang X (2022). "Membership inference attacks on machine learnin g: A survey". ACM Computing Surveys (CSUR). **54** (11s): 1–37.

45. ^Song C, Raghunathan A. "Information leakage in embedding models." In: Proceedings of the 2020 ACM SIGSAC conference on computer and communications security. 2020. p. 377-390.

46. ^Truex S, Liu L, Gursoy ME, Wei W, Yu L. "Effects of differential privacy and data skewness on membership inference vulnerability." In: 2019 First IEEE international conference on trust, privacy and security in intelligent systems and applications (TPS-ISA). IEEE; 2019. p. 82–91.

47. ^Wang Y, Si C, Wu X (2015). "Regression model fitting under differential privacy and model inversion attack". Twenty-fourth international joint conference on artificial intelligence.

48. ^Hu Q, Song Y (2023). "Independent Distribution Regularization for Private Graph Embedding". In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management. pp. 823–832.

49. ^Hu Q, Jiang W, Li H, Wang Z, Bai J, Mao Q, Song Y, Fan L, Li J (2024). "Fedcqa: Answering complex queries on multi-source knowledge graphs via federated learning". CoRR.

50. ^Goldreich O (1998). "Secure multi-party computation". Manuscript. Preliminary version. **78** (110): 1–108. Citeseer.

51. ^Acar A, Aksu H, Uluagac AS, Conti M (2018). "A survey on homomorphic encryption schemes: Theory and implementation". ACM Computing Surveys (Csur). **51** (4): 1–35.

52. ^Hestness J, Narang S, Ardalani N, Diamos G, Jun H, Kianinejad H, Patwary MM, Yang Y, Zhou Y (2017). "Deep learning scaling is predictable, empirically". arXiv preprint arXiv:1712.00409. arXiv:1712.00409.

53. ^Kaplan J, McCandlish S, Henighan T, Brown TB, Chess B, Child R, Gray S, Radford A, Wu J, Amodei D (2020). "Scaling laws for neural language models". arXiv preprint arXiv:2001.08361. arXiv:2001.08361.

54. a, b Yin H, Wang Z, Fei W, Song Y (2023). "EFO$_k$-CQA: Towards Knowledge Graph Complex Query Answering beyond Set Operation". arXiv. arXiv:2307.13701 [cs].

55. a, b Ren H, Dai H, Dai B, Chen X, Zhou D, Leskovec J, Schuurmans D. Smore: Knowledge graph completion and multi-hop reasoning in massive knowledge graphs. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2022. p. 1472-1482. Available from: http://arxiv.org/abs/2110.14890.

56. ^Abiteboul S, Hull R, Vianu V. Foundations of databases. 8th ed. Addison-Wesley Reading; 1995.

57. ^Bollacker K, Evans C, Paritosh P, Sturge T, Taylor J (2008). "Freebase: a collaboratively created graph database for structuring human knowledge". In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. pp. 1247–1250.

58. a, b Khan A (2023). "Knowledge graphs querying". ACM SIGMOD Record. **52** (2): 18–29.

59. ^Tian Y. "The world of graph databases from an industry perspective". ACM SIGMOD Record. **51** (4): 60–67, 2023.

60. a, b, cGuo R, Luan X, Xiang L, Yan X, Yi X, Luo J, Cheng Q, Xu W, Luo J, Liu F, et al. (2022). "Manu: a cloud nativ e vector database management system". arXiv preprint arXiv:2206.13843. arXiv:2206.13843.

61. a, bPan JJ, Wang J, Li G (2024). "Survey of vector database management systems". The VLDB Journal. **33** (5): 1 591–1615.

62. ^Ammar AB. "Graph database partitioning: A study." In: 2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA). IEEE; 2016. p. 1–9.

63. ^Fu Z, Wu Z, Li H, Li Y, Wu M, Chen X, Ye X, Yu B, Hu X (2019). "Geabase: A high-performance distributed gra ph database for industry-scale applications". International Journal of High Performance Computing and Ne tworking. **15** (1-2): 12–21.

64. ^Li C, Chen H, Zhang S, Hu Y, Chen C, Zhang Z, Li M, Li X, Han D, Chen X, et al. ByteGraph: a high-performan ce distributed graph database in ByteDance. Proceedings of the VLDB Endowment. **15**(12):3306–3318, 2022.

65. ^Sun R, Chen J (2023). "Design of Highly Scalable Graph Database Systems without Exponential Performan ce Degradation". In: Proceedings of the International Workshop on Big Data in Emergent Distributed Enviro nments. pp. 1–6.

66. a, bNeo4j, Inc. Neo4j. 2023. Available from: https://neo4j.com. Accessed: 2024-10-28.

67. ^**TigerGraph, Inc.** TigerGraph. 2023. Available from: https://www.tigergraph.com. Accessed: 2024-10-28.

68. ^Wang J, Yi X, Guo R, Jin H, Xu P, Li S, Wang X, Guo X, Li C, Xu X, et al. Milvus: A purpose-built vector data ma nagement system. Proceedings of the 2021 International Conference on Management of Data. 2021:2614–2 627.

69. ^Qdrant Team. Qdrant. 2023. Available from: https://qdrant.tech. Accessed: 2024-10-28.

70. ^SeMI Technologies. Weaviate. https://weaviate.io, 2023. Accessed: 2024-10-28.

71. ^Bouhenni S, Yahiaoui S, Nouali-Taboudjemat N, Kheddouci H (2021). "A survey on distributed graph patter n matching in massive graphs". ACM Computing Surveys (CSUR). **54** (2): 1–35.

72. ^Meng L, Shao Y, Yuan L, Lai L, Cheng P, Li X, Yu W, Zhang W, Lin X, Zhou J (2024). "A survey of distributed g raph algorithms on massive graphs". ACM Computing Surveys. **57** (2): 1–39.

73. ^Zhou Q, Yin P, Yan X, Li C, Jiang G, Cheng J (2024). "Atom: An Efficient Query Serving System for Embeddin g-based Knowledge Graph Reasoning with Operator-level Batching". Proceedings of the ACM on Managem ent of Data. **2** (4): 1–29.

74. ^Yuan Y, Tang B, Zhou T, Zhang Z, Qin J (2024). "nsDB: Architecting the Next Generation Database by Integrating Neural and Symbolic Systems". Proceedings of the VLDB Endowment. **17** (11): 3283–3289.

75. ^Zhao Z, Cai S, Gao H, Pan H, Xiang S, Xing N, Chen G, Ooi BC, Shen Y, Wu Y, et al. NeurDB: On the Design and Implementation of an AI-powered Autonomous Database. arXiv preprint arXiv:2408.03013. 2024.

76. ^Liu S, Biswal A, Cheng A, Mo X, Cao S, Gonzalez JE, Stoica I, Zaharia M (2024). "Optimizing llm queries in relational workloads". arXiv preprint arXiv:2403.05821.

77. ^Zhang H, Liu Y, Yan J (2023). "Cost-Intelligent Data Analytics in the Cloud". arXiv preprint arXiv:2308.09569. arXiv:2308.09569.

78. ^Li F (2019). "Cloud-native database systems at Alibaba: Opportunities and challenges". Proceedings of the VLDB Endowment. **12** (12): 2263–2272.

79. a, bFrancis N, Green A, Guagliardo P, Libkin L, Lindaaker T, Marsault V, Plantikow S, Rydberg M, Selmer P, Taylor A. Cypher: An evolving query language for property graphs. In: Proceedings of the 2018 International Conference on Management of Data, SIGMOD '18. New York, NY, USA: Association for Computing Machinery; 2018. p. 1433–1445. doi:10.1145/3183713.3190657.

80. ^Wang M, Yu L, Zheng D, Gan Q, Gai Y, Ye Z, Li M, Zhou J, Huang Q, Ma C, Huang Z, Guo Q, Zhang H, Lin H, Zhao J, Li J, Smola AJ, Zhang Z (2019). "Deep Graph Library: Towards Efficient and Scalable Deep Learning on Graphs". CoRR. **abs/1909.01315**. Available from: http://arxiv.org/abs/1909.01315. [cited 2024 Jun 21].

81. ^Angles R, Arenas M, Barcelo P, Boncz P, Fletcher G, Gutierrez C, Lindaaker T, Paradies M, Plantikow S, Sequeda J, van Rest O, Voigt H. G-CORE: A Core for Future Graph Query Languages. In: Proceedings of the 2018 International Conference on Management of Data, SIGMOD '18. New York, NY, USA: Association for Computing Machinery; 2018. p. 1421–1432. doi:10.1145/3183713.3190654.

82. ^Zhong W, Xu J, Tang D, Xu Z, Duan N, Zhou M, Wang J, Yin J (2019). "Reasoning over semantic-level graph for fact checking". arXiv preprint arXiv:1909.03745.

83. ^Shen W, Wang J, Han J (2014). "Entity linking with a knowledge base: Issues, techniques, and solutions". IEEE Transactions on Knowledge and Data Engineering. **27** (2): 443–460.

84. a, bJiayang C, Qiu L, Chan C, Liu X, Song Y, Zhang Z (2024). "EventGround: Narrative Reasoning by Grounding to Eventuality-centric Knowledge Graphs". In: Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024). 2024: 6622–6642.

85. ^Chen T, Wang H, Chen S, Yu W, Ma K, Zhao X, Zhang H, Yu D (2023). "Dense x retrieval: What retrieval granularity should we use?" arXiv preprint arXiv:2312.06648.

86. ^Bai J, Luo C, Li Z, Yin Q, Song Y (2024). "Understanding Inter-Session Intentions via Complex Logical Reasoning". arXiv. arXiv:2312.13866 [cs.AI].

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.