Learning Federated Neural Graph Databases for Answering Complex Queries from Distributed Knowledge Graphs

Anonymous authors
Paper under double-blind review

Abstract

The increasing demand for deep learning-based foundation models has highlighted the importance of efficient data retrieval mechanisms. Neural graph databases (NGDBs) offer a compelling solution, leveraging neural spaces to store and query graph-structured data, thereby enabling LLMs to access precise, contextually relevant information. However, current NGDBs are constrained to single-graph operation, limiting their capacity to reason across multiple, distributed graphs. Furthermore, the lack of support for multi-source graph data in existing NGDBs hinders their ability to capture the complexity and diversity of real-world data. In many applications, data is distributed across multiple sources, and the ability to reason across these sources is crucial for making informed decisions. This limitation is particularly problematic when dealing with sensitive graph data, as directly sharing and aggregating such data poses significant privacy risks. As a result, many applications that rely on NGDBs are forced to choose between compromising data privacy or sacrificing the ability to reason across multiple graphs. To address these limitations, we propose to learn Federated Neural Graph DataBases (FedNGDBs), a pioneering systematic framework that empowers privacy-preserving reasoning over multi-source graph data. FedNGDB leverages federated learning to collaboratively learn graph representations across multiple sources, enriching relationships between entities and improving the overall quality of the graph data. Unlike existing methods, FedNGDBs can handle complex graph structures and relationships, making it suitable for various downstream tasks. We evaluate FedNGDBs on three real-world datasets, demonstrating its effectiveness in retrieving relevant information from multi-source graph data while keeping sensitive information secure on local devices. Our results show that FedNGDBs can efficiently retrieve answers to cross-graph queries, making it a promising approach for LLMs and other applications that rely on efficient data retrieval mechanisms.

1 Introduction

Graph Databases (GDBs) excel at efficiently storing and managing highly interconnected data, leveraging their graph structure to efficiently manage complex relationships. This capability makes them indispensable for applications such as recommendation systems (Wang et al., 2019; Cao et al., 2019) and fraud detection (Prusti et al., 2021; Sadowski & Rathle, 2014), where the graph structure is critical. GDBs offer dynamic data models that adapt to evolving structures and deliver scalable performance for intricate queries. In the era of deep learning-based foundation models such as large language models (LLMs), their importance has surged with the advent of Retrieval Augmented Generation (RAG), where agents utilize external GDBs, such as knowledge graphs (KGs) to enhance their retrieval capabilities (Gao et al., 2023; Lewis et al., 2020). This integration facilitates the creation of interactive natural language interfaces tailored to domain-specific applications, enabling more intuitive and accessible interaction with structured data and unlocking new possibilities for intelligent data-driven solutions (Matsumoto et al., 2024; Hussien et al., 2025; Jin et al., 2024).

However, traditional graph databases often suffer from two limitations: the ineffectiveness of free-text semantic search and graph incompleteness, which is a prevalent issue in real-world knowledge graphs and other

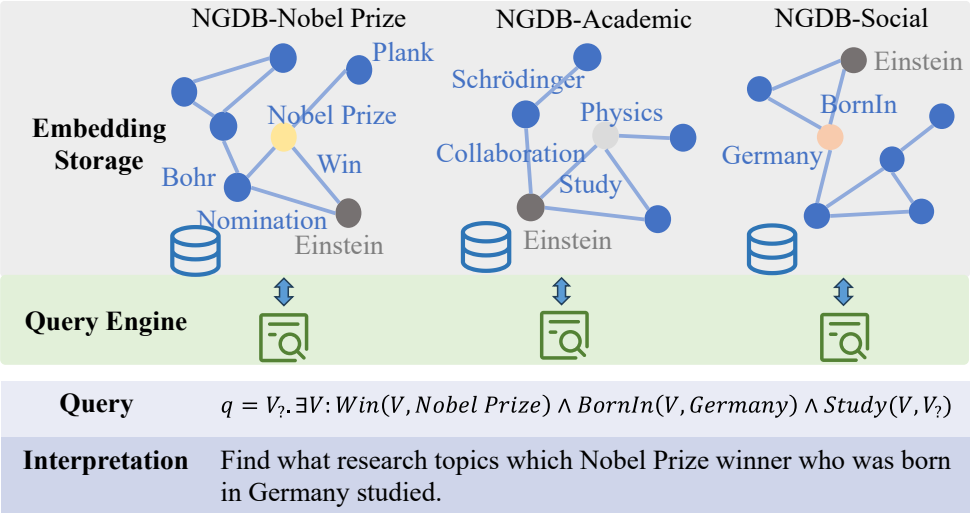


Figure 1: An example of cross graph queries on distributed neural graph databases. The relations and entities in a KG complex query can be from NGDBs which cannot be solved in a single local database.

graph-structured data. Incompleteness leads to the exclusion of relevant results, as the graph database may not capture all the necessary relationships and connections between entities by traversing (Bordes et al., 2013; Galárraga et al., 2013). These limitations hinder their ability to fully support advanced retrieval tasks. To address this, neural graph databases (NGDBs) have recently been proposed by Besta et al. (2022); Ren et al. (2023). They integrate the adaptable structure of graph data models with the powerful processing capabilities of neural networks, allowing for the effective and efficient storage, analysis of graph-structured data, and flexible representation of attributes (Zhang et al., 2024). NGDBs provide unified storage for diverse entries in an embedding space and neural query engine searching answers to input complex queries from the unified storage (Ren et al., 2023). These databases unlock stronger capabilities for intelligent data exploration, enabling users to craft complex queries and make informed inferences with the help of advanced neural network techniques (Bai et al., 2025). Among these applications, complex query answering (CQA) is an important yet challenging task in graph reasoning and can be used for supporting various downstream tasks (Bai et al.; Ren et al., 2023). CQA aims to retrieve answers that satisfy given logical expressions (Hamilton et al., 2018; Ren et al., 2020), which are often defined in predicate logic forms with relation projection operations, existential quantifiers $\exists$, logical conjunctions $\wedge$, disjunctions $\vee$, etc. As shown in Figure 1, given a logical query q, We aim to find all the research topic entities $V_?$ for which there exists Nobel Prize winners V who was born in Germany and conducted studies in that specific field.

While neural graph databases have achieved remarkable success in addressing complex query answering tasks, they are limited to utilizing a single central graph database and cannot be extended to multiple databases. As data assumes an increasingly vital role, NGDBs have experienced rapid growth in scale and scope, aggregating knowledge from diverse domains. Consequently, constructing a graph database that includes all related entities and relations has become difficult and it is impractical to access a central database with all the data needed (Peng et al., 2021; Chen et al., 2021; Zhang et al., 2022). Collaborations between various NDGB holders are essential for answering more complicated queries. For example, as shown in Figure 1, there are multiple NGDBs with different domain knowledge. A complex query may consist of entities and relations from multiple NGDBs, preventing a local query answering model on a single database from answering that cross-graph queries. However, there are various reasons hindering the data sharing between NGDB holders. For example, the growing attention on privacy, regulations such as the General Data Protection Regulation (GDPR), and commercial interest between data holders, etc. Distributed database approaches exist (Nadal et al., 2021), but they fail to leverage NGDBs' neural capabilities fully.

Federated learning offers a potential solution, enabling collaborative model training across distributed participants without sharing raw data (McMahan et al., 2017; Yang et al., 2019). In federated knowledge graph embeddings, raw graph triplets are retained on local devices, participants train local models, and only gradients are transferred to learn a global graph embedding model (Chen et al., 2021; Zhang et al., 2022; Peng et al., 2021) under the protection of privacy preservation methods, such as homomorphic encryption (HE) (Paillier, 1999), secure multi-party computation (SMPC) (Mohassel & Zhang, 2017), and differential privacy (Dwork, 2008; Geyer et al., 2017). While federated learning has been widely applied in learning graph embeddings, recent studies have indicated that learned representations can still potentially leak privacy (Hu & Song, 2023; Duddu et al., 2020), where an attacker can infer sensitive information from the learned embeddings, besides, existing works only focus on learning high-quality representations for simple downstream tasks, such as knowledge graph completion (Chen et al., 2021; Zhang et al., 2022; Huang et al., 2022; Tang et al., 2023), and lacks the ability to reason over graphs and retrieving answers to complex queries.

To overcome these gaps, we propose to learn Federated Neural Graph DataBases (FedNGDBs), a novel system that can reason over multi-source graphs avoiding sensitive raw data transmission and embeddings exposure to safeguarding privacy. FedNGDBs can be applied to different central complex query answering models. It leverages federated learning techniques to train local query answering models in local NGDBs and align graph embeddings for global queries. Different from other federated graph embedding models, the FedNGDB server not only takes the responsibility of aggregating global models but also decomposing given complex queries to sub-queries to compute the query encoding and retrieve answers from distributed NGDBs under the protection of multi-party computation to avoid global model storage. Meanwhile, to better evaluate distributed NGDB systems' performance, we create a benchmark on three widely used datasets. We evaluate our proposed FedNGDBs on the benchmark and the experiment results show the effectiveness of retrieving answers to complex queries from multi-source graphs. We summarize our major contributions as follows:

- To the best of our knowledge, we are the first to extend federated graph embedding systems to complex query answering tasks, which is critical for graph holders' collaboration.
- Based on three public datasets, we propose a benchmark for evaluating the retrieval performance of distributed NGDB systems. The benchmark systematically evaluates the retrieval performance facing cross-graph queries.
- We propose FedNGDB, a federated neural graph database system that can retrieve answers of complex queries from distributed NGDBs with privacy preserved. Extensive experiments on three datasets demonstrate its high performance when facing cross-graph queries.

The rest of the paper is organized as follows. We review the related works in Section 2. Section 3 introduces the preliminary and definition of the distributed complex query answering systems. Section 4 introduces the framework of FedNGDB in detail. Section 5 evaluates the performance of FedNGDB on the benchmark based on three real-world datasets. Finally, we make a conclusion of our work in Section 6.

2 Related Work

2.1 Neural Graph Database

Neural Graph Databases (NGDBs) neutralize traditional GDBs' storage and query planning modules, aiming for stronger intelligent data exploration capabilities. Complex query answering (CQA) is a crucial task in NGDBs, which involves training a model to process and answer complex logical queries based on graph reasoning, a process known as query encoding. These methods represent complex queries into various structures and effectively search answers among candidate knowledge graph entities. GQE (Hamilton et al., 2018), Q2B (Ren et al., 2020) and HypeE (Liu et al., 2021) encode queries to vectors, hyper-rectangle and hyperbolic embeddings, respectively. To support negation operators, various encoding methods are proposed: Q2P (Bai et al., 2022) and ConE (Zhang et al., 2021) use multiple vectors to represent complex queries. BetaE (Ren & Leskovec, 2020), GammaE (Yang et al., 2022), PERM (Choudhary et al., 2021)

propose to use various probabilistic distributions to encode complex logic graph queries. Some methods, like LMPNN (Wang et al., 2022) CQD (Arakelyan et al., 2020), Var2Vec (Wang et al., 2023a) take pre-train embeddings on simple link prediction tasks and apply logic operators to answer complex queries. Meanwhile, Neural structures are utilized to encode complex queries: BiQE (Kotnis et al., 2021) and KgTransformer (Liu et al., 2022) are proposed to use transformers, SQE (Bai et al.) applies sequential encoders, GNN-QE (Zhu et al., 2022) and StarQE (Alivanistos et al., 2021) use message-passing graph neural networks to encode queries, respectively. There are also encoding methods proposed to encode various knowledge graph types: NRN (Bai et al., 2023b) is proposed to encode numerical values, MEQE (Bai et al., 2023a) extends logical queries over events, states, and activities. P-NGDB (Hu et al., 2024) first extends the privacy preservation to neural graph databases.

While there are numerous existing complex query answering methods, these methods mainly focus on a single large graph. There are some methods proposed to reason over multi-view and temporal and varying graphs: MORA (Xi et al., 2022) ensembles multi-view knowledge graphs to scale up complex query answering, TTransE (Leblay & Chekol, 2018) and TRESCAL (Wang et al., 2015) can be applied on temporal knowledge graphs. However, these methods need raw data transmission and privacy protection is not considered. Conducting privacy-preserving complex query answering on multi-source knowledge graphs is still unexplored. With the growing attention on privacy and data protection, sensitive data cannot circulate freely among data holders, and complex query answering is forced to be conducted collaboratively on multiple knowledge graphs. Our research introduces federated learning to existing complex query answering so that we can apply reasoning over distributed NGDBs without raw data sharing.

2.2 Federated Knowledge Graph Embedding

In recent years, federated learning has emerged as a promising approach to address privacy and scalability concerns in machine learning. It allows data owners to participate in model co-construction without raw data transmission to reduce privacy leakage risks (McMahan et al., 2017; Yang et al., 2019) under the protection of privacy protection techniques such as differential privacy (DP) (Yang et al., 2019), homomorphic encryption (HE) (Zhang et al., 2020), secure multi-party computation (SMPC) (Byrd & Polychroniadou, 2020). Various studies have been conducted to explore the potential of federated learning in different domains, such as recommender system (Yang et al., 2020), finance (Long et al., 2020) etc. Federated databases are proposed to manage distributed data allowing storing and querying databases with privacy preserved (Bater et al., 2017; Berger & Schrefl, 2008; Li, 2013). Although some federated graph databases are proposed to manage graph-based data (Nadal et al., 2021), the study for the latest neural graph databases are still be ignored.

Federated knowledge graph embedding is another related topic. It tries to represent entities and their semantic relations into embedding spaces. FedE (Chen et al., 2021) learns knowledge graph embeddings locally and aggregates all local models in a global server for higher representation quality. FedR (Zhang et al., 2022) proposes to learn representation with privacy-preserving relation aggregation to avoid privacy leakage risks in entity embedding and reduce communication costs. FedCKE (Huang et al., 2022), FedMKGC (Tang et al., 2023) extend federated learning to learn global representations from different domains and multilingual knowledge graphs. FedEC (Chen et al., 2022b) applies contrastive learning to tackle data heterogeneity in knowledge graphs. MaKEr (Chen et al., 2022a) and MorseE (Chen et al., 2022c) utilize meta-learning to transfer knowledge among knowledge graphs to train graph neural networks for unseen knowledge extrapolation and inductive learning. DP-FLames (Hu et al., 2023) quantifies the privacy threats and incorporates private selection in federated knowledge embeddings. FLEST (Wang et al., 2023b) decomposes the embedding matrix and enables the sharing of latent representations to reduce the risks of privacy leakage and communication costs, FedM (Hu et al., 2022a) splits the duty of aggregating entities and relations to reduce the risks of graph reconstruction attacks. FKGE (Peng et al., 2021) applies differential privacy and avoids the need for a central server. While existing federated knowledge graph embedding methods are proposed to distributedly learn high-quality representations with privacy preservation, there are some works indicate that the learned embeddings are informative and vulnerable to various privacy attacks (Hu et al., 2022b), even in federated scenarios (Hu & Song, 2024; Hu et al., 2023). Besides, these methods all lack the ability to answer complex queries on multi-source knowledge graphs which is critical for more complicated downstream

tasks. Our research expands the simple federated knowledge graph embedding to answer complex queries on distributed knowledge graphs.

3 Preliminary and Problem Formulation

3.1 Preliminary

Following the general setting of federated knowledge graph embeddings, we denote a set of graph-structured data from various data owners as $\mathcal{G} = \{g_1, g_2, ..., g_N\}$, where N is the total number of graphs. Data owners have their own graph data and cannot access to other's databases. Let $g_k = (\mathcal{V}_k, \mathcal{R}_k, \mathcal{T}_k)$ denotes the k-th graph in $\mathcal{G}$, where $\mathcal{V}_k$ denotes the set of vertices representing entities in the graph g_k, $\mathcal{R}_k$ denotes the set of relations, $\mathcal{T}_k$ denotes the set of triplets. Specifically, $\mathcal{T}_k = \{(v_h, r, v_t)\} \subseteq \mathcal{V}_k \times \mathcal{R}_k \times \mathcal{V}_k$ denotes there is a relation between v_h and v_t, where $v_h, v_t \in \mathcal{V}_k$, $r \in \mathcal{R}_k$. We denote $\mathcal{V} = \cup_{k=1}^{N} \mathcal{V}_k$, $\mathcal{R} = \cup_{k=1}^{N} \mathcal{R}_k$, $\mathcal{T} = \cup_{k=1}^{N} \mathcal{T}_k$ as the set of vertices, relations, and triplets of all graph data, respectively.

3.2 Complex Logical Query

The complex logical query is defined in existential positive first-order logic form, consisting of various types of logic expressions like existential quantifiers $\exists$, logic conjunctions $\wedge$, and disjunctions $\vee$. In the logical expression, there is a set of anchor entities $V_a \in \mathcal{V}$ denotes given context, existential quantified variables $V_1, V_2, ..., V_k \in \mathcal{V}$, and a unique variable $V_?$ denotes our query target. The complex query intends to find the target answers $V_? \in \mathcal{V}$, such that there are $V_1, \cdots, V_k \in \mathcal{V}$ in the graph-structured data that can satisfy the given logical expressions simultaneously. Following the definition in Ren et al. (2020), the complex query expression can be converted to the disjunctive normal form (DNF) in the following:

$$\begin{aligned} q[V_?] &= V_?.V_1, ..., V_k : c_1 \vee c_2 \vee ... \vee c_n \\ c_i &= e_{i,1} \wedge e_{i,2} \wedge ... \wedge e_{i,m}, \end{aligned} \tag{1}$$

where $e_{i,j}$ is the atomic logic expression, which can be the triplet (V, r, V') denotes relation r between entities V and V', c_i is the conjunction of several atomic logic expressions $e_{i,j}$. V, V' are either anchor entities or existentially quantified variables.

3.3 Distributed Graph Set Query

Graph databases are owned by different data holders and cannot be shared directly with each other, therefore, a complex query q can involve entities and relations from different graphs. We define those queries as follows:

Definition 3.1 (Cross-graph Query) *A complex query q is a cross-graph query if there exists query answers $V_? \in \mathcal{V}$, such that there are $V_1, \cdots, V_k \in \mathcal{V}$ in the graph that can satisfy the given logical expressions and the atomic expressions in the query can not be found in a single graph.*

For example in figure 1, the entity "Physics" is the answer to the query q because there exists an existentially quantified variable "Einstein" that can satisfy the logical expression. The query q is a cross-graph query as the atomic expressions in the query are from different graph databases and can not be found in a single graph. For example, Win(Einstein, Nobel Prize) and BornIn(Einstein, Germany) are two atomic expressions from different graph databases. When the distributed graph databases face cross-graph queries, the answer can not be inferred from a single graph. Besides, We have the definition of in-graph query correspondingly:

Definition 3.2 (In-graph Query) *A complex query q is an in-graph query if for all answers $V_? \in \mathcal{V}$ to the query, such that there are $V_1, \cdots, V_k \in \mathcal{V}$ in the graph that can satisfy the given logical expressions and the atomic expressions in the query are from a single graph database.*

These in-graph queries can retrieve answers according to one of the graphs in the graph set and can be solved with existing complex query answering models. However, in a distributed graph neural graph database system, the in-graph queries may retrieve more answers as more knowledge is provided.

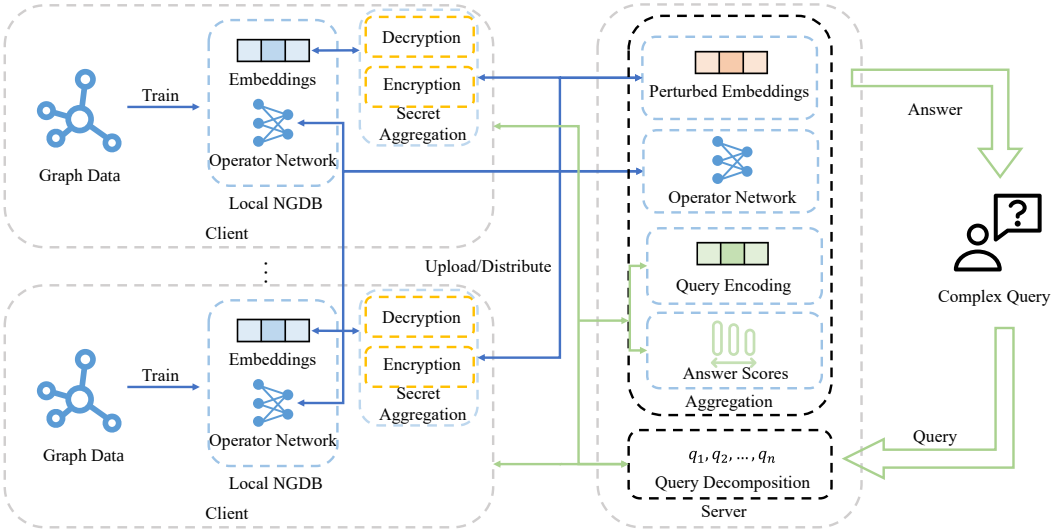


Figure 2: The training and retrieval process of FedNGDB. The blue line denotes the training process, and the green line denotes the retrieval process. In the training, clients' NGDB models are trained on respective graph-structured data. At each round, local NGDBs are aggregated at the server and updated using the global parameters. Among them, the embeddings are protected by secret aggregation so that the server can not access them. In the retrieval, each query is decomposed into sub-queries. Clients compute sub-query embeddings which the server is used to aggregate query embeddings. Answer scores are computed at clients and are aggregated at a server to retrieve answers.

3.4 Problem Formulation

Given a graph-structured data set $\mathcal{G} = \{g_1, ..., g_N\}$ with N graphs. Every graph is owned by independent data holders and can not be shared to construct a unified graph database. We assume that the triples are sensitive as they describe the informative relations between entities while the index of entities and relations can be shared, which means that the triples in each graph database will stay private on local devices. The graphs in the $\mathcal{G}$ are related and have part of entities and relation overlapped. There are complex queries involving elements from the graph set $\mathcal{G}$ and can be classified as cross-graph queries and in-graph queries. We aim to construct a distributed neural graph database system to reason over multi-source graphs and retrieve answers to complex logical queries while keeping privacy preserved, especially the triple information indicating the relations between entities. To achieve this, we assume that there is an honest but curious server managing the federated neural graph database system. Because the learned embeddings are vulnerable to various privacy attacks, the embeddings cannot be exposed to the server and should be further protected before being transferred to the server.

4 Federated Neural Graph Databases

In this section, we introduce the learning and query retrieval process of our proposed FedNGDB.

4.1 Model Learning

We first introduce the training of FedNGDB. As shown in Figure 2, FedNGDB has a central server and a set of clients. Each client has a graph with overlapping entities of others. The server takes the responsibility of aggregating parameters and organizing the training and retrieval process. The clients train a local NGDB model based on their graph-structured data. According to the sensitivity of the parameters, we divide the query encoding methods into two parts: operator function with parameter Θ and entity embeddings $\mathbf{E}$. For the operator function, the client directly sends the parameter to the server for aggregation and receives the

global function to update the local operator function, which is similar to FedAvg (McMahan et al., 2017). Therefore, in the following parts, we only introduce the entity embeddings aggregation in FedNGDB.

4.1.1 Secret Aggregation

There are various techniques, like homomorphic encryption (HE) (Paillier, 1999), secure multi-party computation (MPC) (Mohassel & Zhang, 2017), and differential privacy (Dwork, 2008; Geyer et al., 2017) to protect the uploaded parameters, however the protection of aggregated global model are often ignored. Unfortunately, the global model is informative and vulnerable to privacy attacks (Zhang et al., 2022; Hu & Song, 2024). Hence we propose a secret aggregation applied in parameter aggregation which can prevent the global server from knowing the aggregated parameters using homomorphic encryption. Assume that at each client i, a parameter denoted as θ_i is uploaded to a server for aggregating global parameter θ. The procedure of secret aggregation is described in Algorithm 1. In the beginning, each client C_i randomly generates perturbed parameters θ_i^r and shares them with other clients with encryption (Shown in Appendix A). After the sharing, each client has a set of parameters $\{\theta_1^r, \theta_2^r. \cdots, \theta_n^r\}$. In the training process, for each client, C_i uploads perturbed parameter $(\theta_i + \theta_i^r)$ to the server under the protection of homomorphic encryption. The server collects all perturbed parameters from clients, computes aggregated perturbed parameter θ^r, and sends it back to all clients. Finally, after receiving the perturbed parameters, the clients first decrypt the parameters and can compute aggregated parameter θ by removing the perturbed parameters and preventing exposing it to the server.

Besides, the parameters are also protected by the differential privacy to prevent the potential privacy leakage (Zhu et al., 2019), we use local differential privacy (LDP) to the local gradients to further improve the difficulty to recover sensitive information from transmission. As proposed in Yi et al. (2021), we clip the local gradients based on the L_∞ norm with the threshold $\mathcal{C}$ and apply zero-mean Laplacian noise with strength λ, the upper bound of the privacy budget ϵ is $\frac{2\mathcal{C}}{\lambda}$. The detailed discussion can be found in Appendix B.3.

4.1.2 Model Training

Similar to Chen et al. (2021), the server constructs a set of mapping matrices $\{\mathbf{M}^i \in \{0,1\}^{n\times n_i}\}_{i=1}^N$ and existence vectors $\{\mathbf{v}^i \in \{0,1\}^{n\times 1}\}_{i=1}^N$ to denote the entities in each client, where n is the number of all unique entities in KG set and n_i is the number of entities from client C_i. $\mathbf{M}^i_{m,n} = 1$ if the m-th entity in entity table corresponds to the n-th entity from client C_i. $\mathbf{v}^i_m = 1$ indicates that the m-th entity in entity table exists in client C_i.

FedNGDB performs secret aggregation for entity embeddings. First, the client C_i will randomly initialize the local entity embeddings $\mathbf{E}^i_0 \in \mathbb{R}^{n_i\times d}$ and perturbation embeddings $\mathbf{E}^i_r \in \mathbb{R}^{n_i\times d}$. Every client will share the permutation embeddings with all clients. At round t, the server will select part of the clients $\mathbf{C}$ participating in the training. After local training of CQA on respective local graphs, client C_i sends perturbed local entity embeddings $\mathbf{E}^i_t + \mathbf{E}^i_r$ to the server. The server will aggregate the entity embeddings using perturbed local embeddings:

$$\mathbf{E}^r_{t+1} \leftarrow \left(\mathbb{1} \oslash \sum_{i\in\mathbf{C}} \mathbf{v}^i\right) \otimes \sum_{i\in\mathbf{C}} \mathbf{M}^i(\mathbf{E}^i_{t+1} + \mathbf{E}^i_r), \tag{2}$$

where $\mathbb{1}$ denotes all-one vector, $\oslash$ denotes element-wise division for vectors and $\otimes$ denotes element-wise multiply with broadcasting. After aggregation, the server sends the aggregated entity embeddings back to all clients, and the client C_i receives:

$$\mathbf{E}^{r,i}_{t+1} \leftarrow {\mathbf{M}^i}^\top \mathbf{E}^r_{t+1}, \tag{3}$$

and the client C_i can compute and update the local entity embeddings as:

$$\mathbf{E}^i_{t+1} \leftarrow \mathbf{E}^{r,i}_{t+1} - {\mathbf{M}^i}^\top \left(\mathbb{1} \oslash \sum_{j\in\mathbf{C}} \mathbf{v}^j\right) \otimes \sum_{j\in\mathbf{C}} \mathbf{M}^j \mathbf{E}^j_r. \tag{4}$$

After secret aggregation, the entity embeddings of all clients are shared without exposing the sensitive information to the server. Besides the training of entity embeddings, the operator networks are trained and aggregated using FedAvg (McMahan et al., 2017), and the detailed descriptions of the FedNGDB are shown in Algorithm 2.

4.2 Query Retrieval

After training, the server in FedNGDB manages the process of retrieving answers to complex queries. The server first tries to arrange related clients to encode the coming queries and retrieves answers from all local graph databases based on the encoding.

4.2.1 Query Encoding

Query encoding methods commonly represent queries to embeddings and retrieve answers according to scoring functions where similarity functions are widely used. FedNGDB encodes queries and treats two types of queries differently. For in-graph queries, as these queries only involve a single graph data, FedNGDB can directly encode queries using corresponding local complex query answering models. For cross-graph queries, as the query involves entities from multiple graphs, the server will take the responsibility to plan the entire encoding process: First, the server will decompose the query to atomic expressions and send each expression to corresponding local graphs. The clients encode received atomic queries using their own local CQA models and send back the results to the server. The server collects all the encoding results and uses global operator function models to compute the representations of the queries. The query is iteratively updated by communicating between the server and clients until the original queries are encoded.

4.2.2 Answer Retrieval

Because the entity embeddings are not stored in the central server, we can only retrieve answers from all distributed local graphs after encoding the queries. Given a query encoding q, we score all the candidate entities at each local graph database, at client C_i:

$$\mathbf{S}_q^i \leftarrow f_s^i(\mathcal{V}_i, q) \in \mathbb{R}^{n_i \times 1}, \tag{5}$$

where f_s^i is a score function in the client C_i. Then the score will be uploaded to the server to aggregate a score table for all unique entities in the graph sets:

$$\mathbf{S} \leftarrow \left(\mathbb{1} \oslash \sum_{i=1}^{N} \mathbf{v}^i \right) \otimes \sum_{i=1}^{N} \mathbf{M}^i \mathbf{S}^i. \tag{6}$$

The final answers to the queries are retrieved globally from the graph database set according to the score table.

5 Experiments

In this section, we create a benchmark of distributed graph complex logical query answering problems for distributed neural graph databases and evaluate our proposed FedNGDB's performance on the benchmark.

5.1 Datasets and Experiment Setting

We introduce the detailed information of our used datasets and the setting of our experiments.

5.1.1 Datasets

In our experiment, following previous work, we use the three commonly used knowledge graphs as graph-structured data: FB15k (Bollacker et al., 2008; Bordes et al., 2013), FB15k-237 (Toutanova & Chen, 2015),

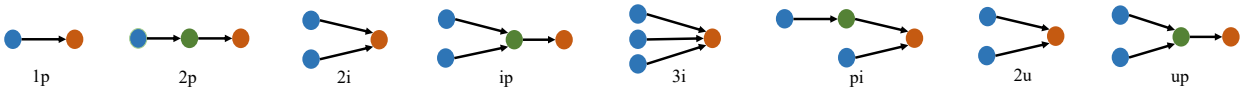


Figure 3: The query structures used for evaluation in the experiments. Naming for each query structure is provided under each subfigure, for brevity, the p, i, and u represent the projection, intersection, and union operations respectively.

Table 1: The statistics of three datasets used for experiments.

Graphs	#Clients	#Nodes	#Relations	#Edges
FB15k-237	3	13,651	79	103,359
	5	12,639	47.4	62,015
FB15k	3	14,690	448.3	197,404
	5	14,279	269	118,442
NELL995	3	40,204	66.7	47,601
	5	28,879	40	28,560

and NELL995 (Carlson et al., 2010) to construct the distributed query answering benchmark. In each dataset, there are vertices describing entities and edges describing relations. To evaluate the distributed complex query, we conduct experiments assuming having 3 and 5 clients in each federated neural graph database system, respectively. We randomly select relations into clients and split triples into clients according to selected relations as a local graph database. The triplets in each local graph are separated into training, validation, and testing with a ratio of 8:1:1 respectively. Following previous works (Ren et al., 2020), we construct training graph $\mathcal{G}_{train}$, validation graph $\mathcal{G}_{val}$, and test graph $\mathcal{G}_{test}$ in each client by training edges, training+validation edges, and training+validation+testing edges, respectively. The detailed statistics are listed in the table 1. #Clients denotes the number of clients, #Nodes, #Relations, #Edges denote the average number of nodes, relations, and edges in each client respectively.

5.1.2 Query Sampling

Following previous work (Hamilton et al., 2018; Bai et al., 2023b), we evaluate the complex logical query answering performance on the following eight general query types with abbreviations of $1p, 2p, 2i, ip, 3i, pi, 2u$, and up. As shown in figure 3, each subgraph denotes one query type, where each edge represents either a projection or a logical operator, and each node represents either a set of entities, the anchor entities and relations are to be specified to instantiate logical queries. We use the sampling method commonly used in previous works Bai et al.; Ren et al. (2020) to randomly sample complex queries from graphs. We randomly sample two sets of queries from the graph sets: in-graph queries and cross-graph queries to evaluate local and global answer retrieval performance. For local model evaluation, we first obtain training, validation, and testing queries from the formerly constructed local graph databases respectively. Then for the training queries, we conduct a graph search to find corresponding training answers on the local training graph. For the validation queries, we search for the answers on both the training graph and the validation graph and only use those queries that have different numbers of answers on the two graphs. For the testing queries, we use those queries that have different answers on the testing graph from answers on the training graph and validation graph. For global model evaluation, we construct global training, validation, and testing graphs using all local graphs, and sample testing queries with atomic expressions from different local graphs, finally we search for answers on three global graphs and only use those queries that have different answers on the testing graph from other two graphs. We collect statistics of complex queries in three datasets and the statistics are shown in Table 2. The number of in-graph queries is the average number of the client's local queries.

5.1.3 Baselines

We can use various existing query encoding methods as our local base model, to evaluate the effectiveness and generalization ability of our proposed FedNGDB, we select three commonly used complex encoding

Table 2: The statistics of queries sampled from three datasets used for experiments.

Graphs	#C	In-graph Train.	In-graph Valid.	In-graph Test.	Cross-graph Test.
FB15k-237	3	317,226	11,528	11,539	32,573
	5	180,552	6,619	6,673	31,469
FB15k	3	592,573	19,206	19,267	53,660
	5	344,418	11,409	11,437	53,154
NELL995	3	208,070	8,810	8,750	24,954
	5	117,231	5,177	5,118	24,237

Table 3: The retrieval performance of distributed neural graph databases when there are 3 clients. The average results of HR@3 and MRR of all clients are reported. The best results are underlined. The best results of distributed models are in bold.

Graph	Setting	GQE				Q2P				Tree-LSTM			
		In-graph		Cross-graph		In-graph		Cross-graph		In-graph		Cross-graph	
		HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR
FB15k-237	Local	12.64	12.03	-	-	14.55	13.63	-	-	13.32	12.73	-	-
	Central	13.13	12.39	13.03	12.28	14.93	14.66	15.02	14.81	13.28	12.61	13.36	12.91
	FedE	**13.72**	**13.23**	**12.74**	**11.63**	14.82	14.27	14.79	13.93	13.12	12.23	**12.62**	**12.08**
	FedR	12.89	11.98	-	-	14.32	14.23	-	-	**13.92**	**12.92**	-	-
	FedNGDB	13.54	12.43	12.63	11.32	**15.32**	**14.32**	**14.83**	**14.11**	12.93	12.11	12.55	11.96
FB15k	Local	22.05	18.21	-	-	24.32	22.64	-	-	22.87	20.51	-	-
	Central	29.53	25.65	30.21	25.33	38.62	34.14	38.03	34.36	38.87	35.86	37.97	36.13
	FedE	24.31	26.74	**27.95**	**25.21**	43.68	39.62	39.72	35.95	34.27	30.18	**31.19**	26.03
	FedR	20.29	18.61	-	-	25.32	22.71	-	-	23.64	20.97	-	-
	FedNGDB	**25.63**	**26.87**	24.77	25.17	**44.02**	**39.27**	**40.27**	**36.31**	**34.85**	**33.83**	31.80	**28.99**
NELL995	Local	11.85	11.03	-	-	15.86	13.02	-	-	13.85	13.85	12.94	-
	Central	12.87	11.95	13.06	12.46	16.74	14.82	16.42	15.63	15.41	14.23	16.27	15.83
	FedE	13.29	12.72	12.46	11.82	**17.23**	14.12	**16.28**	14.01	14.27	13.81	14.18	13.71
	FedR	12.01	11.23	-	-	16.04	13.26	-	-	12.48	11.67	-	-
	FedNGDB	**14.21**	**13.27**	**13.76**	**12.67**	16.62	**15.28**	16.27	**16.23**	**16.28**	**15.38**	**16.09**	**15.27**

methods GQE (Hamilton et al., 2018), Q2P (Bai et al., 2022), Tree-LSTM (Bai et al.) as our base model. GQE is a graph query encoding model that encodes a complex query into a vector in embedding space; Q2P represents complex queries using multiple vectors; Tree-LSTM recursively represents complex queries and treats all operations, entities, and relations as tokens.

To the best of our knowledge, there are no existing federated complex query answering methods but several federated knowledge graph embedding methods, therefore, we choose to compare our methods with FedE (Chen et al., 2021) and FedR (Zhang et al., 2022), two commonly used federated knowledge graph embedding methods as baselines. FedE aggregates both entity embeddings and relation embeddings in a server, while FedR only aggregates relation embeddings for privacy concerns and communication efficiency. We utilize these two methods with slight modifications to train a global complex query answering model: FedE aggregates all query encoding parameters and FedR aggregates relation embeddings and query encoding networks. Besides, we also compare our FedNGDB with local and central settings. In the local setting, there is no collaboration between clients, while in the central setting, all distributed graphs in the graph set are aggregated for a global graph for training, we sample complex queries from global training, and validation graphs for training and validation.

If there is no further statement, we use the following implementation settings in the experiments. We tune hyper-parameters on the validation local queries for the base query encoding methods and set the dimension of entities and relations as 400 for all models for fair comparison and use AdamW (Loshchilov & Hutter, 2018) as optimizer. We set the gradient clip threshold $C = 0.1$ and Laplacian noise scale $\lambda = 0.2$ to achieve 1-DP.

Table 4: The retrieval performance of distributed neural graph databases when there are 5 clients. The average results of HR@3 and MRR of all clients are reported. The best results are underlined. The best results of distributed models are in bold.

Graph	Setting	GQE				Q2P				Tree-LSTM			
		In-graph		Cross-graph		In-graph		Cross-graph		In-graph		Cross-graph	
		HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR
FB15k-237	Local	11.44	10.65	-	-	14.65	13.8	-	-	11.23	10.37	-	-
	Central	13.72	12.87	12.99	12.74	15.93	14.62	15.74	15.23	13.62	12.54	11.86	11.53
	FedNGDB	**12.42**	**11.60**	**11.20**	**10.79**	**16.13**	**15.78**	**15.28**	**14.91**	**12.48**	**11.91**	**11.49**	**11.02**
FB15k	Local	19.83	17.51	-	-	36.10	35.04	-	-	20.03	18.62	-	-
	Central	20.58	20.14	21.21	21.02	42.56	41.87	39.57	39.46	24.88	24.76	24.93	24.72
	FedNGDB	**21.40**	**20.83**	**20.71**	**19.94**	**40.81**	**37.96**	**38.56**	**35.73**	**24.59**	**23.75**	**23.85**	**22.90**
NELL995	Local	10.48	10.09	-	-	15.26	14.37	-	-	14.52	13.89	-	-
	Central	14.56	14.28	13.75	13.42	15.63	15.14	15.27	14.75	16.47	15.92	15.74	14.47
	FedNGDB	**13.79**	**13.27**	**12.74**	**12.18**	**15.44**	**15.81**	**15.28**	**14.24**	**15.68**	**14.28**	**14.57**	**12.89**

5.1.4 Evaluation Metrics

Following the previous work (Bai et al., 2023b), we evaluate the generalization capability of models by calculating the rankings of answers that cannot be directly retrieved from an observed graph. Given a testing query q, the training, validation, and public testing answers are denoted as M_{train}, M_{val}, and M_{test}, respectively. We evaluate the quality of retrieved answers using Hit ratio (HR) and Mean reciprocal rank (MRR). HR@K metric evaluates the accuracy of retrieval by measuring the percentage of correct hits among the top K retrieved items. The MRR metric evaluates the performance of a ranking model by computing the average reciprocal rank of the first relevant item in a ranked list of results. The metric can be defined as:

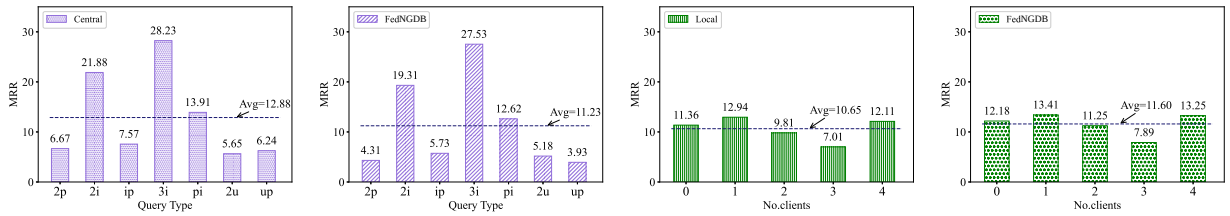
$$\text{Metric}(q) = \frac{1}{|M_{test}/M_{val}|} \sum_{v \in M_{test}/M_{val}} m(rank(v)), \tag{7}$$

$m(r) = \mathbf{1}[r \leq K]$ if the metric is HR@K and $m(r) = \frac{1}{r}$ if the metric is MRR. Higher values denote better reasoning performance. We train local models at each client by using the in-graph training queries and tune hyper-parameters using the validation queries. The evaluation is then finally conducted on the testing queries, including the evaluation of in-graph queries on local query encoding models and cross-graph queries on the global federated neural graph database system, respectively.

5.2 Performance Evaluation

We evaluate FedNGDB's complex query answering performance on three datasets and compare it to other baselines. We apply FedNGDB on three base query encoding models and evaluate the average retrieval performance on various queries. The results are summarized in Table 3 and Table 4. Table 3 reports the retrieval performance of various distributed graph complex query answering models when there are 3 clients. For each model, we evaluate performance facing in-graph queries and cross-graph queries, respectively. For in-graph queries, the average scores of all clients are reported. We report results in HR@3 and MRR which higher scores indicate better performance. The best results are underlined. The best results of distributed models are in bold. As shown in Table 3, our proposed methods can effectively retrieve complex query answers from distributed graph databases. In comparison to local settings, we can see that FedNGDB can utilize all participated local graph databases and performs better facing in-graph queries. For example, GQE model with FedNGDB can achieve 14.21 HR@3 on average while can only reach 11.85 without collaboration. Besides, in comparison to other federated knowledge graph embedding methods, our proposed FedNGDB can reach comparable performance in both in-graph queries and cross-graph queries without exposing sensitive entity embeddings to the server. For example, FedNGDB achieves the best performance in cross-graph queries in more than half of datasets and base query encoding models.

In Table 4, we present the performance of FedNGDB when there are 5 clients. We compare the performance with local training without collaboration to demonstrate the influence of client numbers. As shown in the



(a) Cross-graph Queries (b) Cross-graph Queries (c) In-graph Queries (d) In-graph Queries

Figure 4: The evaluation results of FedNGDB-GQE facing cross-graph queries on FB15k-237 (subfigures (a), (b)). The evaluation results of FedNGDB-GQE facing in-graph queries on FB15k-237 (subfigures (c), (d)).

table, FedNGDB performs well compared to complex query answering models without collaboration, there are performance improvements in all datasets after applying FedNGDB to various base query encoding models, demonstrating that FedNGDB can utilize the intrinsic information in the distributed knowledge sets. The collaboration allows FedNGDB to reason over various logical paths to improve performance.

5.3 Query Types

FedNGDB can globally reason over distributed graphs and retrieve answers to cross-graph queries. To evaluate FedNGDB's performance on various types of complex queries, we conduct experiments to evaluate the retrieval performance of FedNGDB and compare it to central learning on FB15k-237 when there are 3 clients. Figure 4(a) and Figure 4(b) show the FedNGDB-GQE's performance facing cross-graph queries. As we can see from the figure, FedNGDB performs well on most various types of queries compared to the central model. For example, on query types '2i' and 'pi', FedNGDB can reach more than 90% MRR compared to the central model.

5.4 Local Influence

Because the in-graph queries can be processed by a single local neural graph database, in this part, we evaluate the performance of FedNGDB on these queries to assess the influence of FedNGDB on local queries. We conduct experiments on FB15k-237 and the number of clients is 5. We evaluate the performance of FedNGDB based on GQE and compare the model with no collaboration. The results are summarized in the Figure 4(c) and Figure 4(d). As shown in the figure, although each client has different performance due to the triplets correlation being different in each sub-dataset, FedNGDB can improve all clients' performance compared to the model without collaboration.

6 Conclusion

In this work, we present a federated neural graph database, FedNGDB to reason over distributed knowledge sets with privacy preservation, allowing graph database holders to collaboratively build a distributed graph reasoning system without sharing raw data. We define the distributed graph complex logical query answering problem. To solve the problem, we propose secret aggregation for federated learning where the aggregated parameters can be kept secret to the server. Besides, we design a distributed query retrieval process for answering queries from distributed graph database sets to protect clients' privacy. To evaluate FedNGDB model performance, we construct a benchmark based on three commonly used knowledge graph complex query answering datasets: FB15k-237, FB15k, and NELL995. Extensive experiments on the benchmark demonstrate the effectiveness of our proposed FedNGDB. FedNGDB can retrieve answers given a query while keeping the sensitive information secret at local graph databases. In the future, we aim to propose new methods for better answering complex queries by exploiting intrinsic information in the distributed neural graph databases.

References

Dimitrios Alivanistos, Max Berrendorf, Michael Cochez, and Mikhail Galkin. Query embedding on hyper-relational knowledge graphs. In *International Conference on Learning Representations*, 2021.

Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. Complex query answering with neural link predictors. In *International Conference on Learning Representations*, 2020.

Jiaxin Bai, Tianshi Zheng, and Yangqiu Song. Sequential query encoding for complex query answering on knowledge graphs. *Transactions on Machine Learning Research*.

Jiaxin Bai, Zihao Wang, Hongming Zhang, and Yangqiu Song. Query2particles: Knowledge graph reasoning with particle embeddings. *Findings of the Association for Computational Linguistics: NAACL 2022-Findings*, 2022.

Jiaxin Bai, Xin Liu, Weiqi Wang, Chen Luo, and Yangqiu Song. Complex query answering on eventuality knowledge graph with implicit logical constraints. *Advances in Neural Information Processing Systems*, 36:30534–30553, 2023a.

Jiaxin Bai, Chen Luo, Zheng Li, Qingyu Yin, Bing Yin, and Yangqiu Song. Knowledge graph reasoning over entities and numerical values. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 57–68, 2023b.

Jiaxin Bai, Zihao Wang, Yukun Zhou, Hang Yin, Weizhi Fei, Qi Hu, Zheye Deng, Jiayang Cheng, Tianshi Zheng, Hong Ting Tsang, et al. Top ten challenges towards agentic neural graph databases. *arXiv preprint arXiv:2501.14224*, 2025.

Johes Bater, Gregory Elliott, Craig Eggen, Satyender Goel, Abel Kho, and Jennie Rogers. Smcql: secure querying for federated databases. *Proceedings of the VLDB Endowment*, 10(6):673–684, 2017.

Stefan Berger and Michael Schrefl. From federated databases to a federated data warehouse system. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*, pp. 394–394. IEEE, 2008.

Maciej Besta, Patrick Iff, Florian Scheidl, Kazuki Osawa, Nikoli Dryden, Michal Podstawski, Tiancheng Chen, and Torsten Hoefler. Neural graph databases. In *Learning on Graphs Conference*, pp. 31–1. PMLR, 2022.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 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, 2008.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

David Byrd and Antigoni Polychroniadou. Differentially private secure multi-party computation for federated learning in financial applications. In *Proceedings of the First ACM International Conference on AI in Finance*, pp. 1–9, 2020.

Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*, pp. 151–161, 2019.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam Hruschka, and Tom Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, pp. 1306–1313, 2010.

Mingyang Chen, Wen Zhang, Zonggang Yuan, Yantao Jia, and Huajun Chen. Fede: Embedding knowledge graphs in federated setting. In *Proceedings of the 10th International Joint Conference on Knowledge Graphs*, pp. 80–88, 2021.

Mingyang Chen, Wen Zhang, Zhen Yao, Xiangnan Chen, Mengxiao Ding, Fei Huang, and Huajun Chen. Meta-learning based knowledge extrapolation for knowledge graphs in the federated setting. In Lud De Raedt (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 1966–1972. International Joint Conferences on Artificial Intelligence Organization, 7 2022a. doi: 10.24963/ijcai.2022/273. URL https://doi.org/10.24963/ijcai.2022/273. Main Track.

Mingyang Chen, Wen Zhang, Zonggang Yuan, Yantao Jia, and Huajun Chen. Federated knowledge graph completion via embedding-contrastive learning. *Knowledge-Based Systems*, 252:109459, 2022b. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2022.109459. URL https://www.sciencedirect.com/science/article/pii/S0950705122007316.

Mingyang Chen, Wen Zhang, Yushan Zhu, Hongting Zhou, Zonggang Yuan, Changliang Xu, and Huajun Chen. Meta-knowledge transfer for inductive knowledge graph embedding. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, pp. 927–937, New York, NY, USA, 2022c. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531757. URL https://doi.org/10.1145/3477495.3531757.

Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan Reddy. Probabilistic entity representation model for reasoning over knowledge graphs. *Advances in Neural Information Processing Systems*, 34:23440–23451, 2021.

Whitfield Diffie and Martin E Hellman. New directions in cryptography. In *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, pp. 365–390. 2022.

Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. Quantifying privacy leakage in graph embedding. In *MobiQuitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pp. 76–85, 2020.

Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pp. 1–12. Springer, 2006.

Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pp. 1–19. Springer, 2008.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pp. 413–422, 2013.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. *Advances in neural information processing systems*, 31, 2018.

Ce Hu, Baisong Liu, Xueyuan Zhang, Zhiye Wang, Chennan Lin, and Linze Luo. A federated multi-server knowledge graph embedding framework for link prediction. In *2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 366–371. IEEE, 2022a.

Hui Hu, Lu Cheng, Jayden Parker Vap, and Mike Borowczak. Learning privacy-preserving graph convolutional network with partially observed sensitive attributes. In *Proceedings of the ACM Web Conference 2022*, pp. 3552–3561, 2022b.

Qi Hu and Yangqiu Song. Independent distribution regularization for private graph embedding. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, pp. 823–832, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701245. doi: 10.1145/3583780.3614933. URL https://doi.org/10.1145/3583780.3614933.

Qi Hu and Yangqiu Song. User consented federated recommender system against personalized attribute inference attack. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pp. 276–285, 2024.

Qi Hu, Haoran Li, Jiaxin Bai, Zihao Wang, and Yangqiu Song. Privacy-preserved neural graph databases. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1108–1118, 2024.

Yuke Hu, Wei Liang, Ruofan Wu, Kai Xiao, Weiqiang Wang, Xiaochen Li, Jinfei Liu, and Zhan Qin. Quantifying and defending against privacy threats on federated knowledge graph embedding. In *Proceedings of the ACM Web Conference 2023*, WWW '23, pp. 2306–2317, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394161. doi: 10.1145/3543507.3583450. URL https://doi.org/10.1145/3543507.3583450.

Wei Huang, Jia Liu, Tianrui Li, Shenggong Ji, Dexian Wang, and Tianqiang Huang. Fedcke: Cross-domain knowledge graph embedding in federated learning. *IEEE Transactions on Big Data*, 2022.

Mohamed Manzour Hussien, Angie Nataly Melo, Augusto Luis Ballardini, Carlota Salinas Maldonado, Rubén Izquierdo, and Miguel Angel Sotelo. Rag-based explainable prediction of road users behaviors for automated driving using knowledge graphs and large language models. *Expert Systems with Applications*, 265: 125914, 2025.

Mingyu Jin, Qinkai Yu, Dong Shu, Chong Zhang, Lizhou Fan, Wenyue Hua, Suiyuan Zhu, Yanda Meng, Zhenting Wang, Mengnan Du, et al. Health-llm: Personalized retrieval-augmented disease prediction system. *arXiv preprint arXiv:2402.00746*, 2024.

Bhushan Kotnis, Carolin Lawrence, and Mathias Niepert. Answering complex queries in knowledge graphs with bidirectional sequence encoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4968–4977, 2021.

Julien Leblay and Melisachew Wudage Chekol. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, pp. 1771–1776, 2018.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Yingjie Li. *A federated query answering system for semantic web data.* PhD thesis, USA, 2013.

Lihui Liu, Boxin Du, Heng Ji, ChengXiang Zhai, and Hanghang Tong. Neural-answering logical queries on knowledge graphs. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 1087–1097, 2021.

Xiao Liu, Shiyu Zhao, Kai Su, Yukuo Cen, Jiezhong Qiu, Mengdi Zhang, Wei Wu, Yuxiao Dong, and Jie Tang. Mask and reason: Pre-training knowledge graph transformers for complex logical queries. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1120–1130, 2022.

Guodong Long, Yue Tan, Jing Jiang, and Chengqi Zhang. Federated learning for open banking. In *Federated Learning: Privacy and Incentive*, pp. 240–254. Springer, 2020.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Nicholas Matsumoto, Jay Moran, Hyunjun Choi, Miguel E Hernandez, Mythreye Venkatesan, Paul Wang, and Jason H Moore. Kragen: a knowledge graph-enhanced rag framework for biomedical problem solving using large language models. *Bioinformatics*, 40(6), 2024.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*, pp. 19–38. IEEE, 2017.

Sergi Nadal, Alberto Abelló, Oscar Romero, Stijn Vansummeren, and Panos Vassiliadis. Graph-driven federated data management. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):509–520, 2021.

Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pp. 223–238. Springer, 1999.

Hao Peng, Haoran Li, Yangqiu Song, Vincent Zheng, and Jianxin Li. Differentially private federated knowledge graphs embedding. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 1416–1425, 2021.

Debachudamani Prusti, Daisy Das, and Santanu Kumar Rath. Credit card fraud detection technique by applying graph database model. *Arabian Journal for Science and Engineering*, 46(9):1–20, 2021.

H Ren, W Hu, and J Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *International Conference on Learning Representations (ICLR)*, 2020.

Hongyu Ren and Jure Leskovec. Beta embeddings for multi-hop logical reasoning in knowledge graphs. *Advances in Neural Information Processing Systems*, 33:19716–19726, 2020.

Hongyu Ren, Mikhail Galkin, Michael Cochez, Zhaocheng Zhu, and Jure Leskovec. Neural graph reasoning: Complex logical query answering meets graph databases. *arXiv preprint arXiv:2303.14617*, 2023.

Gorka Sadowski and Philip Rathle. Fraud detection: Discovering connections with graph databases. *White Paper-Neo Technology-Graphs are Everywhere*, 13:1–13, 2014.

Wei Tang, Zhiqian Wu, Yixin Cao, Yong Liao, and Pengyuan Zhou. Fedmkgc: Privacy-preserving federated multilingual knowledge graph completion. *arXiv preprint arXiv:2312.10645*, 2023.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pp. 57–66, 2015.

Dingmin Wang, Yeyuan Chen, and Bernardo Cuenca Grau. Efficient embeddings of logical variables for query answering over incomplete knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 4652–4659, 2023a.

Maolin Wang, Dun Zeng, Zenglin Xu, Ruocheng Guo, and Xiangyu Zhao. Federated knowledge graph completion via latent embedding sharing and tensor factorization. In *2023 IEEE International Conference on Data Mining (ICDM)*, pp. 1361–1366. IEEE, 2023b.

Quan Wang, Bin Wang, and Li Guo. Knowledge base completion using embeddings and rules. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 950–958, 2019.

Zihao Wang, Yangqiu Song, Ginny Wong, and Simon See. Logical message passing networks with one-hop inference on atomic formulas. In *The Eleventh International Conference on Learning Representations*, 2022.

Zhaohan Xi, Ren Pang, Changjiang Li, Tianyu Du, Shouling Ji, Fenglong Ma, and Ting Wang. Reasoning over multi-view knowledge graphs. *arXiv preprint arXiv:2209.13702*, 2022.

Dong Yang, Peijun Qing, Yang Li, Haonan Lu, and Xiaodong Lin. Gammae: Gamma embeddings for logical queries on knowledge graphs. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 745–760, 2022.

Liu Yang, Ben Tan, Vincent W Zheng, Kai Chen, and Qiang Yang. Federated recommendation systems. *Federated Learning: Privacy and Incentive*, pp. 225–239, 2020.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

Jingwei Yi, Fangzhao Wu, Chuhan Wu, Ruixuan Liu, Guangzhong Sun, and Xing Xie. Efficient-fedrec: Efficient federated learning framework for privacy-preserving news recommendation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 2814–2824, 2021.

Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In *2020 USENIX annual technical conference (USENIX ATC 20)*, pp. 493–506, 2020.

Delvin Ce Zhang, Menglin Yang, Rex Ying, and Hady W Lauw. Text-attributed graph representation learning: Methods, applications, and challenges. In *Companion Proceedings of the ACM Web Conference 2024*, pp. 1298–1301, 2024.

Kai Zhang, Yu Wang, Hongyi Wang, Lifu Huang, Carl Yang, Xun Chen, and Lichao Sun. Efficient federated learning on knowledge graphs via privacy-preserving relation embedding aggregation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 613–621, 2022.

Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. Cone: Cone embeddings for multi-hop reasoning over knowledge graphs. *Advances in Neural Information Processing Systems*, 34:19172–19183, 2021.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

Zhaocheng Zhu, Mikhail Galkin, Zuobai Zhang, and Jian Tang. Neural-symbolic models for logical queries on knowledge graphs. In *International Conference on Machine Learning*, pp. 27454–27478. PMLR, 2022.

A Parameter Sharing

In this section, we provide an example of sharing secrets between clients. The parameters can be shared under the protection using various encryption methods, for example, the commonly used is Diffie–Hellman key exchange (Diffie & Hellman, 2022) shown as follows, we consider the sharing process between two clients:

- Client A and Client B publicly agree to use a modulus p and base g, p is a prime.
- Client A chooses a secret integer a, then sends Client B $m_A = g^a \mod p$.
- Client B chooses a secret integer b, then sends Client A $m_B = g^b \mod p$.
- Client A computes $s = m_B^a \mod p$.
- Client B computes $s = m_A^b \mod p$.

After D-H key exchange, Client A and B share a secret $s = g^{ab} \mod p$. The secret s can be used as encryption to share sensitive information between clients.

B Alogrithm

B.1 Secret Aggregation

We present the pseudo-code of secret aggregation in Algorithm 1.

Algorithm 1: Secret Aggregation

Require: n clients $C_1, C_2, \ldots, C_n$, client C_i has parameter θ_i
Each client C_i generates random parameter θ_i^r
Transmit θ_i^r to all other clients with encryption, each
client has a set of parameters $\{\theta_1^r, \theta_2^r. \cdots, \theta_n^r\}$

Client C_i:
Upload perturbed parameter $(\theta_i + \theta_i^r)$ to server with HE Encryption.
Receive encrypted parameters $\sum_{j=1}^{n}(\theta_j + \theta_j^r)$ from server and HE decryption.
Compute encrypted parameters $\theta = [\sum_{j=1}^{n}(\theta_j + \theta_j^r) - \sum_i^n \theta_i^r]/n$

Server:
for $i = 1, \cdots, n$ **do**
  Receive encrypted parameters $(\theta_i + \theta_i^r)$ from client C_i
end for
Compute encrypted parameters $\theta^r = \sum_{j=1}^{n}(\theta_j + \theta_j^r)$
Send encrypted parameters θ^r to all clients

B.2 FedNGDB Framework

We present the pseudo-code of FedNGDB in Algorithm 2.

B.3 Differential Privacy Setting

Differential privacy represents a mathematical framework for quantifying and limiting the disclosure of private information in statistical databases (Dwork, 2008). In federated learning, differential privacy enables collaborative model training across decentralized data sources while providing formal privacy guarantees. By injecting calibrated noise into the training process, these techniques ensure that individual contributions

Algorithm 2: FedNGDB Framework

Require: The number of clients N; The faction of clients selected in each round F;

Client C_i**:**
Client C_i initialize entity embeddings $\mathbf{E}^i$ and perturbation embeddings $\mathbf{E}_r^i$.
Share $\mathbf{E}_r^i$ with other clients with encryption
Receive and decryption to get $\{\mathbf{E}_r^1, \cdots, \mathbf{E}_r^N\}$
Upload $\mathbf{E}^i$ to server for secret aggregation, receive $\mathbf{E}_0^i$

Server:
Server constructs permutation matrices $\{M_i\}_{i=1}^N$, and existence vectors $\{v_i\}_{i=1}^N$ and initialize operator networks Θ_0, distribute to all clients.
for $t = 0, 1, 2, \cdots$ **do**
  Server distributes operator networks to each client.
  $\mathcal{C}_t \leftarrow$ Randomly select client set with $N \times F$ clients.
  for $C_i \in \mathcal{C}_t$ **in parallel do**
    $(\mathbf{E}_r^i + \mathbf{E}_{t+1}^i), \Theta_{t+1}^i \leftarrow$ **ClientUpdate**$(C_i, {\mathbf{M}^i}^\top \mathbf{E}_t^r, \Theta_t)$
  end for
  $\mathbf{E}_{t+1}^r \leftarrow \left(\mathbb{1} \oslash \sum_{i \in \mathbf{C}_t} \mathbf{v}^i\right) \otimes \sum_{i \in \mathbf{C}_t} \mathbf{M}^i (\mathbf{E}_{t+1}^i + \mathbf{E}_r^i)$
  $\Theta_{t+1} \leftarrow 1/|\mathcal{C}_t| \sum_{i \in \mathcal{C}_t} \Theta_t^i$
end for

ClientUpdate$(C_i, \mathbf{E}^r, \Theta)$**:**
$\mathbf{E} \leftarrow \mathbf{E}^r - {\mathbf{M}^i}^\top \left(\mathbb{1} \oslash \sum_{j \in \mathbf{C}} \mathbf{v}^j\right) \otimes \sum_{j \in \mathbf{C}} \mathbf{M}^j \mathbf{E}_r^j$
for $e = 1, \cdots, E$ **do**
  $\mathbf{E}, \Theta \leftarrow$ **LocalUpdate**$(\mathbf{E}, \Theta)$
end for
return $\mathbf{E} + \mathbf{E}_r^i, \Theta$

cannot be reliably identified or extracted from the global model (Geyer et al., 2017). The privacy-utility tradeoff can be tuned via the privacy budget parameter ϵ, allowing practitioners to balance learning performance against disclosure risk. The formal definition of ϵ-differential privacy (ϵ-DP) is as follows (Dwork, 2006): A randomized algorithm M satisfies ϵ-differential privacy if for all neighboring datasets D_1 and D_2 that differ in at most one record, and for all possible outputs $S \subseteq Range(M)$:

$$Pr[M(D_1) \in S] \leq e^\epsilon Pr[M(D_2) \in S]$$

Where ϵ is the privacy parameter that quantifies the privacy loss.

In the former part, we introduce Laplacian noise to the local gradients to meet the guarantee of ϵ-DP for in-graph query in local neural graph databases. However, for cross-graph query, we compute the score on each graph database for each query. The Post-Processing Theorem (Dwork et al., 2014) which states the composition of a data-independent mapping with an ϵ-DP algorithm M is also ϵ-DP. Since the ranking operation of FedNGDB is a post-processing step applied to the outputs $(M_1(D), \cdots, M_n(D))$ from each local databases and the overlapped candidate answers share same scores, by the post-processing theorem of differential privacy, the final ranked list maintains the same privacy guarantee of ϵ-DP.

C Auxiliary Experiments

Here we present some auxiliary experiments to further evaluate the performance of FedNGDB.

C.1 More Clients

In the former experiments, we evaluate FedNGDB's performance when there are 3 or 5 clients in the federated system. To further evaluate models' performance when there are more clients. We split the graph-structured data into 10 subgraphs and evaluate the retrieval performance of FedNGDB using GQE as the base model. As shown in Table 5, FedNGDB can still improve the retrieval performance compared to local training when there are more clients participating in the distributed system, demonstrating the effectiveness of FedNGDB.

Table 5: The performance of GQE when #C=10.

Graph	Setting	FB15k-237				FB15k				NELL995			
		In-graph		Cross-graph		In-graph		Cross-graph		In-graph		Cross-graph	
		HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR
GQE	Local	8.96	8.47	-	-	13.46	13.04	-	-	8.36	7.83	-	-
	Central	13.15	12.76	13.24	12.89	19.76	19.43	20.27	19.87	13.76	13.28	13.53	13.27
	FedNGDB	10.67	10.17	10.23	10.04	15.39	14.98	14.86	14.45	12.93	12.11	12.55	11.96

C.2 Relation Overlap

In the experiments, we evaluate the retrieval performance when there are no overlap relations between local graph databases, however, various graph databases can have shared relations in real life, to evaluate the performance in such a scenario, we evaluate the FedNGDB with GQE's retrieval performance. The graph-structured data is randomly split into 3 subgraphs. The results are shown in Table 6, showing that FedNGDB can successfully retrieve answers from distributed graph databases.

Table 6: The MRR of GQE when relation overlapped.

Graph	Setting	FB15k-237				FB15k				NELL995			
		In-graph		Cross-graph		In-graph		Cross-graph		In-graph		Cross-graph	
		HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR	HR@3	MRR
GQE	Local	10.48	10.22	-	-	20.76	20.21	-	-	10.03	9.64	-	-
	Central	13.47	13.27	13.86	13.54	30.74	30.46	30.27	30.16	13.82	13.68	13.56	13.12
	FedNGDB	11.96	11.42	11.67	11.43	23.89	22.47	23.22	22.89	11.68	11.36	12.03	11.89

C.3 Convergence Rate

We evaluate the convergence speed of three federated frameworks. The results are presented by the average number of communication round ratios relative to FedE. As shown in Table 7, FedNGDB's convergence speed is faster than FedR while slightly slower than FedE, demonstrating that our FedNGDB can protect stronger protection while remaining competitive efficiency.

Table 7: The statistics of communication rounds .

Setting	FedE	FedR	FedNGDB
Relative Rounds to FedE	1.00	1.32	1.09