FedBridge: A Federated Query Engine over Embedding-Heterogeneous Vector Databases

Yuxiang Wang
Beihang University
Beijing, China
yuxiangwang@buaa.edu.cn

Ruixi Hu
Beihang University
Beijing, China
ruixihu@buaa.edu.cn

Ziyuan He
Beihang University
Beijing, China
hzy_he@buaa.edu.cn

Zhilin Liang
Beihang University
Beijing, China
zlliang@buaa.edu.cn

Xinyu Zhao
Beihang University
Beijing, China
xinyuzhao2321@buaa.edu.cn

Yongxin Tong
Beihang University
Beijing, China
yxtong@buaa.edu.cn

ABSTRACT

Vector databases have emerged as core infrastructure for managing unstructured data by enabling efficient semantic search over high-dimensional embeddings. While existing systems largely assume a centralized setting, real-world deployments are often federated, with data distributed across autonomous silos. A fundamental challenge in such environments is embedding heterogeneity: different silos and query users may adopt distinct embedding models, making their vector representations incomparable and hindering effective retrieval. In this paper, we present FedBridge, a federated query engine designed for embedding-heterogeneous vector databases. FedBridge introduces a lightweight alignment technique to bridge heterogeneous embedding spaces, along with two novel sampling strategies that effectively probe data silos and improve query efficiency. Besides, it provides a unified query interface for embedding-heterogeneous vectors, supports diverse embedding models and data formats, and offers adapters for popular vector database systems. We demonstrate the deployment and usage of FedBridge via a representative scientific question answering scenario.

PVLDB Reference Format:
Yuxiang Wang, Ruixi Hu, Ziyuan He, Zhilin Liang, Xinyu Zhao, and Yongxin Tong. FedBridge: A Federated Query Engine over Embedding-Heterogeneous Vector Databases. PVLDB, 19(12): XXX-XXX, 2026. doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Roy-buaa/FedBridge.

1 INTRODUCTION

Vector databases have become essential for managing unstructured data such as images, text, and audio [8]. They store high-dimensional embeddings of raw objects, enabling efficient semantic search. Given a query, the system embeds it using an embedding model and retrieves its nearest neighbors, supporting applications like image search and retrieval-augmented generation (RAG).

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

While most systems assume a centralized database, real-world deployments are often federated [4, 9, 12, 14], where data is distributed across autonomous silos such as scientific data centers, hospitals, or law firms. In such settings, queries are dispatched across data silos and the returned results are aggregated. However, this paradigm introduces a key challenge in practice: embedding heterogeneity, which is rarely addressed in prior systems [3, 7]. Specifically, each data owner may adopt a distinct embedding model tailored to its local data characteristics or resource constraints [1, 5], resulting in embeddings that are not directly comparable across silos.

The presence of embedding heterogeneity necessitates executing federated queries over heterogeneous vector databases. Given a query object q and the associated query embedding model M_q, the query user aims to retrieve the top-k nearest objects to q from n silos, where each silo stores vectors generated by its own embedding model. The goal is to achieve high retrieval accuracy and low query latency despite embedding heterogeneity, while preserving the privacy of raw data during query processing.

In this paper, we propose FedBridge, a federated query engine over embedding-heterogeneous vector databases with high efficiency and usability. At the algorithm level, FedBridge employs a lightweight alignment technique to bridge heterogeneous embedding spaces. It further introduces two novel sampling strategies to balance the retrieval effort across parties, together with a feedback mechanism to improve efficiency [13]. Compared with prior federated approaches, FedBridge achieves up to a 6.2× speedup. At the system level, FedBridge provides a unified query interface for embedding-heterogeneous vector retrieval. In addition, it supports the integration of diverse embedding models and raw data formats [13], incorporates configurable privacy protection mechanisms for raw data [6], and offers adapters for different vector database systems such as Milvus [7] and Qdrant [11].

We demonstrate the design and usage of FedBridge through key workflows, including data silo management, data visualization, and federated query processing. A scientific question answering scenario showcases its interactive visualizations and user interfaces.

2 SYSTEM DESIGN

In this section, we first present an overview of FedBridge, then introduce the user interface followed by the complete query execution workflow, and finally describe the implementation details.

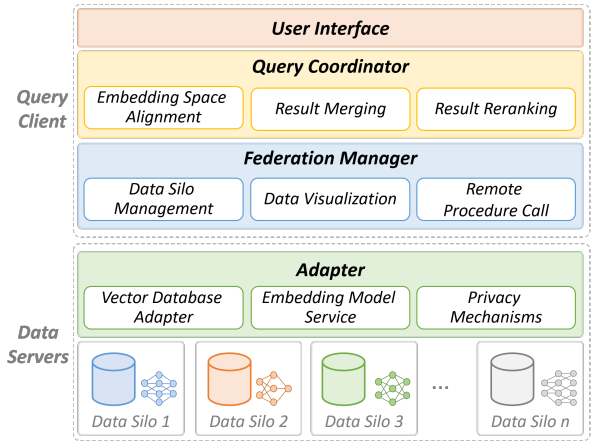


Figure 1: System architecture of FedBridge.

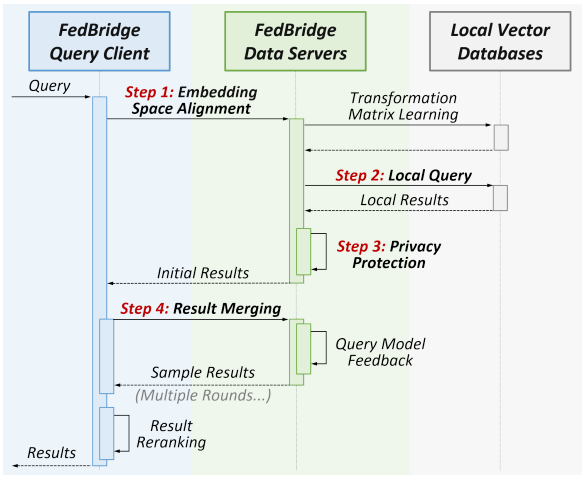


Figure 2: The query execution workflow of FedBridge.

2.1 FedBridge Overview

As shown in Figure 1, FedBridge consists of both the query client side and the data server side.

Query Client Side. The query client consists of three components: the user interface, query coordinator, and federation manager. The *user interface* provides an easy-to-use entry point for query clients to issue federated queries, including selecting target silos, choosing the query embedding model, and configuring retrieval parameters. The issued query is then passed to the *query coordinator*, which generates the execution plan and orchestrates the overall query workflow. The *federation manager* operates under the query coordinator to manage and coordinate data silos. It maintains model and data information for each silo, manages connections to active silos, and provides communication support during query execution.

Data Server Side. The data server side of FedBridge interfaces with local vector databases that are autonomously operated by individual data silos. It provides adapters for popular vector database systems and integrates embedding model services for seamless query processing. In addition, it incorporates privacy mechanisms for text and image data.

2.2 User Interface

FedBridge provides a set of user interfaces for managing data silos with embedding-heterogeneous vector databases and issuing federated queries.

Data Silo Management. FedBridge enables users to register new data silos through a unified interface, abstracting away differences in underlying vector database systems. A silo can be registered by specifying its host address and corresponding embedding model.

```
FedBridge.register(silo="S1", model="ResNet", addr=...)
FedBridge.register(silo="S2", model="SWIN", addr=...)
FedBridge.register(silo="S3", model="CLIP", addr=...)
```

In this example, FedBridge registers three image data silos, each using a distinct embedding model to generate vectors.

Federated Query over Embedding-Heterogeneous Vectors. FedBridge provides a federated query interface similar to standard vector search, with the key distinction that users can specify the query embedding model to suit their downstream applications.

```
FedBridge.search(
  query=img, k=10, silos=["S1", "S2", "S3"]
  query_model="ViT", distance="L2", params=...
)
```

In the above example, the user issues a query to retrieve images from the three data silos that are similar to `img` under a user-specified embedding model (`ViT`). The `params` argument specifies additional options, including privacy mechanisms and result merging strategies, which are described in Section 2.3.

2.3 Execution Workflow

The execution workflow of FedBridge is illustrated in Figure 2. A query is executed over a set of data silos, each of which stores its raw data along with the corresponding embedding vectors and may maintain local indexes to accelerate vector search. A federated query over these embedding-heterogeneous silos aims to retrieve relevant objects under a user-specified model. FedBridge executes the query through a four-step workflow:

Step 1: Embedding Space Alignment. Each data silo may employ a distinct embedding model, making direct similarity comparisons across silos infeasible, as their vectors reside in different embedding spaces. To address this challenge, FedBridge aligns the embedding spaces of different silos, enabling vectors generated by heterogeneous models to be mapped into the user-specified embedding space for similarity search. Existing approaches typically rely on complex embedding transformation methods, which incur substantial preprocessing overhead during query processing [17, 18]. In contrast, FedBridge adopts a lightweight embedding space alignment mechanism whose preprocessing can be completed within seconds and supports the seamless integration of new embedding models. Specifically, for each silo S_i, we learn a transformation matrix $\mathcal{T}_i \in \mathbb{R}^{d_i \times d_q}$ that maps vectors from the local embedding space of dimension d_i to the query embedding space of dimension d_q. The transformation matrix is trained via gradient descent on a

small public dataset D_{Pub}, using the following loss function:

$$\mathcal{L} = \sum_{o \in D_{pub}} \left\| M_i(o)\mathcal{T}_i - M_q(o) \right\|_2$$

where $M_i(o)$ and $M_q(o)$ denote the embedding vectors of object o under M_i (local embedding model of silo S_i) and M_q (query embedding model), respectively. With the learned transformation, the distance between a query q and an object o can be approximated as $dis(M_q(q), M_i(o)\mathcal{T}_i)$, enabling efficient similarity comparison across heterogeneous embedding spaces.

Step 2: Local Query. Local query processing aims to retrieve objects from each silo that are similar to q under the embedding space of M_q. First, the system searches the local vector database using vector $M_i(q)$ to retrieve candidate objects based on the similarity in M_i's embedding space, leveraging existing vector indexes like IVF or HNSW for efficient retrieval. The retrieved candidates are then transformed using the learned matrix $\mathcal{T}_i$ (from Step 1) and re-ranked in M_q's embedding space, producing the final local results.

Moreover, for the inner product distance metric, we can directly use the transformed query vector $M_q(q)\mathcal{T}_i^\top$ to search the local vector database. In this case, the re-ranking step can be avoided due to the following identity:

$$\begin{aligned} &dis_{IP}(M_q(q), (M_i(o)\mathcal{T}_i)) = M_q(q) \cdot (M_i(o)\mathcal{T}_i)^\top \\ &= M_i(o)\mathcal{T}_i \cdot (M_q(q))^\top = M_i(o) \cdot (M_q(q)\mathcal{T}_i^\top)^\top \\ &= dis_{IP}(M_q(q)\mathcal{T}_i^\top, M_i(o)) \end{aligned}$$

Step 3: Privacy Protection. Privacy protection is applied to local results before raw objects are sent to the query client. FedBridge supports mainstream privacy-preserving techniques for raw data: (1) *Data anonymization*: applies lightweight anonymization techniques to remove, replace or mask sensitive entities in text and image data [6]. (2) *Differential privacy*: injects calibrated noise into raw data to provide formal privacy guarantees while preserving utility for similarity search [15, 16].

Step 4: Result Merging. This step collects local results from each silo and merges them into the final result set. Due to approximation errors introduced by embedding space alignment and noise introduced by privacy mechanisms, the distances of local results may not accurately reflect their true distances under the query embedding model. To obtain accurate rankings, raw objects may need to be transferred and re-embedded under M_q, and re-ranked accordingly, operations that are time-consuming. Therefore, the merging process aims to minimize transmission and re-embedding operations, while maintaining high accuracy of retrieval.

To this end, FedBridge leverages the non-IID nature of federated vector data and incorporates two novel sampling algorithms for efficient and accurate result merging [13]:

- Competition-based sampling: iteratively retrieves candidates from one of the current best-performing silos.
- Contribution-based sampling: maintains the cumulative contributions of silos, and adaptively sampling silo according to their contributions.

Moreover, the data server side of FedBridge further improves efficiency of result merging by leveraging feedback from the query model to enhance the quality of subsequent candidates [13]. Specifically, candidates that are closer to objects with smaller true distances to q under the embedding space of M_q are assigned higher priority for re-embedding.

2.4 Implementation

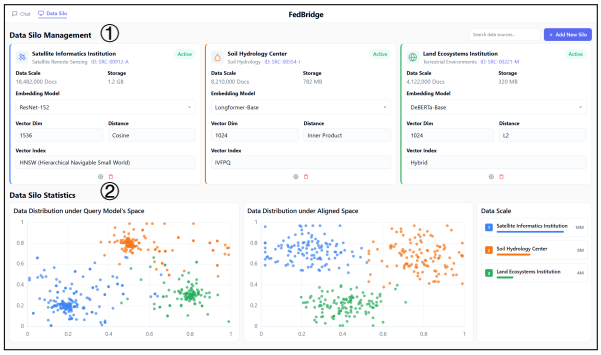
We then describe the implementation of FedBridge on both the query client side and the data server side.

Query Client Side. The query client is implemented as a lightweight module for fast deployment and efficient query execution. (1) *Federation Manager* is built on gRPC and Protocol Buffers to support efficient communication between the query client and data servers. It adopts a modular design with basic RPC primitives and message schemas, while supporting customizable message formats for different result merging algorithms. It can also optionally collect communication statistics (*e.g.*, latency and transferred bytes) for performance monitoring.
(2) *Query Interface and Coordinator* is built on top of the federation manager and incorporates lightweight functionalities from faiss [2] and PyTorch [10] to support efficient distance computation, and on-the-fly embedding under the query model. Moreover, the coordinator enables non-blocking execution while waiting for silo responses, improving concurrency and overall latency.

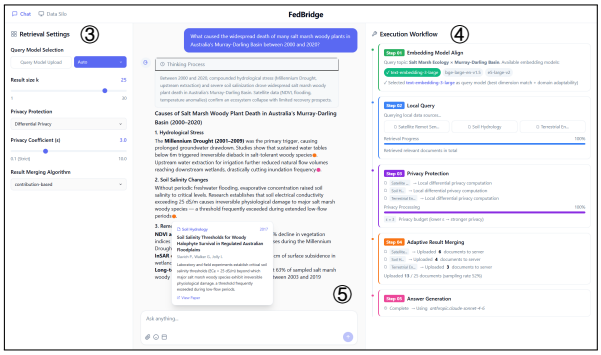
Data Server Side. Each data silo joins the federation by deploying a FedBridge adapter configured with its local vector database backend, embedding model service, and dataset reader.
(1) *Vector Database Adapter* supports popular vector database systems such as Milvus [7] and Qdrant [11] by specifying the service address and credentials, or by implementing a custom connector that provides core vector search interfaces.
(2) *Embedding Model Service* provides a unified interface to local embedding services. A silo can connect to an existing model service or implement a custom adapter exposing the embedding interface.
(3) *Dataset Reader* supports raw data ingestion from sources with different formats, simplifying the integration of raw data into the local query pipeline.

3 DEMONSTRATION SCENARIO: SCIENTIFIC QUESTION ANSWERING

In this part, we center on the scenario of question answering with multi-source scientific documents and demonstrate the usage of FedBridge. Comprehensive scientific question answering often requires synthesizing knowledge across multiple institutions [4]. However, in practice, these institutions operate as autonomous data silos, each independently managing its documents with heterogeneous embedding models. Accordingly, a cross-institutional setting is leveraged to illustrate how FedBridge harmonizes these disparate embedding spaces to enable federated query and verifiable answer generation. We demonstrate the usage of FedBridge in two folds: *(i) Federation Management* (Figure 3a), dedicated to configuring and visualizing heterogeneous data silos; and *(ii) Federated Query and Question Answering* (Figure 3b), designed for executing the federated retrieval workflow and verifiable answer generation.



(a) Federation management.



(b) Federated query and question answering.

Figure 3: Demonstration of FedBridge.

3.1 Federation Management

Data Silos. Panel ① illustrates three scientific data silos, each characterized by metadata such as its embedding model and data volume. In addition, new silos can be added through the user console.

Visualization of Data Distribution. FedBridge provides a visualization interface to monitor data distributions across all silos, as shown in panel ②. It samples data objects from each silo and projects both the embeddings under the query model and those in the aligned space into a two-dimensional representation using Principal Component Analysis (PCA). This visualization highlights the inherent heterogeneity of the initial data distributions while demonstrating the effectiveness of the embedding alignment process: the aligned space approximately preserves the relative spatial relationships of the query model's embedding space.

3.2 Federated Query and Question Answering

Query Setup. Within panel ③, users configure query parameters such as the result size, privacy mechanisms, result merging algorithms, and the query embedding model. In particular, FedBridge supports two modes of embedding model selection: *(i) Manual Selection:* users choose an embedding model from repositories (*e.g.*, HuggingFace) or upload locally fine-tuned models for specific tasks; and *(ii) Automated Selection:* a LLM automatically determines the optimal embedding model based on the question. The selected configurations are then used to execute the federated query.

Federated Query. Upon receiving a scientific query, FedBridge triggers the federated query execution workflow, as illustrated in panel ④. First, it automatically selects an appropriate embedding model for the query domain and performs embedding space alignment. Participating silos then conduct local query in the aligned space and inject differential privacy noise into the results. Subsequently, FedBridge performs contribution-based sampling for result merging and transmits the relevant documents to the query client. The workflow then proceeds to the answer generation phase.

Answer Generation. As shown in panel ⑤, the large language model integrates the retrieved information and generate answers with fine-grained citations that link each statement to its corresponding data source and context, ensuring that answers to complex scientific queries remain transparent and verifiable.

ACKNOWLEDGEMENT

This work was partially supported by National Science Foundation of China (NSFC) (Grant Nos. 62425202, 62336003), the Beijing Natural Science Foundation (Z230001), the Fundamental Research Funds for the Central Universities No. JKF-2026007844235, and the State Key Laboratory of Complex & Critical Software Environment (SKLCCSE). Yongxin Tong is the corresponding author.

REFERENCES

[1] Databracks. 2025. Improving Retrieval and RAG with Embedding Model Finetuning. https://www.databricks.com/blog/improving-retrieval-and-rag-embedding-model-finetuning
[2] Faiss. 2026. A library for efficient similarity search and clustering of dense vectors. https://github.com/facebookresearch/faiss
[3] Zeheng Fan, Yuxiang Zeng, Zhuanglin Zheng, and Yongxin Tong. 2025. FedVSE: A Privacy-Preserving and Efficient Vector Search Engine for Federated Databases. *PVLDB* (2025).
[4] Anastasios Kementsietsidis, Frank Neven, Dieter Van de Craen, and Stijn Vansummeren. 2008. Scalable multi-query optimization for exploratory queries over federated scientific databases. *PVLDB* (2008).
[5] LlamaIndex. 2023. Fine-Tuning Embeddings for RAG with Synthetic Data. https://www.llamaindex.ai/blog/fine-tuning-embeddings-for-rag-with-synthetic-data-e534409a3971
[6] Microsoft. 2026. Presidio: Data Protection and De-identification SDK. https://microsoft.github.io/presidio/
[7] Milvus. 2026. The High-Performance Vector Database Built for Scale. https://github.com/milvus-io/milvus
[8] James Jie Pan, Jianguo Wang, and Guoliang Li. 2024. Survey of vector database management systems. *The VLDB Journal* (2024).
[9] Bjarne Pfitzner, Nico Steckhan, and Bert Arnrich. 2021. Federated learning in a medical context: a systematic literature review. *ACM TOIT* (2021).
[10] PyTorch. 2026. PyTorch: Tensors and Dynamic neural networks. https://github.com/pytorch/pytorch
[11] Qdrant. 2026. High-Performance Vector Search at Scale. https://github.com/qdrant/qdrant
[12] Yuxiang Wang, Shuyuan Li, Yongxin Tong, Shuyue Wei, and Zimu Zhou. 2026. Efficient shapley-based data valuation for federated trajectories. *FCS* (2026).
[13] Yuxiang Wang, Yongxin Tong, Zimu Zhou, Ziyuan He, Ruixi Hu, and Ke Xu. 2026. Federated Retrieval over Embedding-Heterogeneous Vector Databases. *ICDE 2026*.
[14] Yuxiang Wang, Yuxiang Zeng, Shuyuan Li, Yuanyuan Zhang, Zimu Zhou, and Yongxin Tong. 2024. Efficient and private federated trajectory matching. *IEEE TKDE* (2024).
[15] Zekun Xu, Abhinav Aggarwal, Oluwaseyi Feyisetan, and Nathanael Teissier. 2020. A Differentially Private Text Perturbation Method Using Regularized Mahalanobis Metric. *PrivateNLP* (2020).
[16] Haonan Yan, Xiaoguang Li, Wenjing Zhang, Qian Chen, Bin Wang, Hui Li, and Xiaodong Lin. 2024. Coder: Protecting privacy in image retrieval with differential privacy. *IEEE TDSC* (2024).
[17] Beining Yang, Yang Cao, and Yang Ren. 2025. Integrating Vector Databases across Embedding Models. *SIGMOD* (2025).
[18] Jinsung Yoon and Sercan O Arik. 2025. Embedding-Converter: A Unified Framework for Cross-Model Embedding Transformation. *ACL* (2025).