Metric Algebra: Embedding-Independence in Vector Databases

TIANJIAN YANG, University of Edinburgh, UK
YANG CAO, University of Edinburgh, UK
BEINING YANG, University of Edinburgh, UK
ZIYING CHEN, University of Edinburgh, UK
TIEJUN MA, University of Edinburgh, UK

Vector databases are increasingly central to retrieval applications, yet deployments remain embedding-siloed: different databases store vectors produced by different embedding models and ANN pipelines, making cross-database search ill-defined or forcing expensive re-embedding and index rebuilds. We propose *metric algebra*, a logical foundation for embedding-independence. It models each dataset as a metric relation, *i.e.,* identifiers equipped with a semantic distance, and provides operators for similarity search, and, critically, a semantics-preserving Union operator for cross-embedding search. We implement metric algebra in MetricDB, which treats existing vector databases as physical vector realizations that implement metric relations, and executes Union via witness maps learned from overlap anchors. Building on this logical-physical bridge, MetricDB plans hard vs. soft execution for queries over Union under imperfect witnesses and avoids rebuilds via transformation-equivariant index compilation and merging. Experiments on benchmark workloads across heterogeneous embedding models (*e.g.,* Mistral, GTE-Qwen2, NV-Embed-V2, OpenAI-Ada-002) demonstrate robust cross-embedding retrieval and efficient interoperability.

CCS Concepts: • **Information systems** → **Information integration**; **Retrieval models and ranking**.

Additional Key Words and Phrases: Vector database; Data integration; Metric algebra; Embedding independence

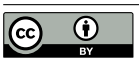
ACM Reference Format:
Tianjian Yang, Yang Cao, Beining Yang, Ziying Chen, and Tiejun Ma. 2026. Metric Algebra: Embedding-Independence in Vector Databases. *Proc. ACM Manag. Data* 4, 4 (SIGMOD), Article 279 (September 2026), 26 pages. https://doi.org/10.1145/3837117

1 Introduction

Vector databases are central to retrieval applications such as RAG, recommendation, enterprise search, and agentic workflows, and support for vector queries is now standard across DBMS and specialized vector systems [4, 11, 15, 28, 30, 36, 37]. Their core abstraction is simple: represent each object by an embedding vector and answer similarity queries by nearest-neighbor search in vector space. Despite this success, deployments remain *embedding-siloed*. Different databases store vectors produced by different embedding models, preprocessing pipelines, and ANN stacks. Consequently, even a basic operation such as searching over the union of vector collections is either ill-defined, as there is no cross-embedding distance, or forces costly re-embedding, backfilling and index rebuilds.

The root cause is a semantic-implementation entanglement. In most vector systems, the embedding model and its operational distance determine both (i) what similarity means and (ii) how it is computed. This conflation blocks a principled notion of interoperability: if two datasets are

Authors' Contact Information: Tianjian Yang, Tianjian.Yang@ed.ac.uk, University of Edinburgh, Edinburgh, UK; Yang Cao, Yang.Cao@ed.ac.uk, University of Edinburgh, Edinburgh, UK; Beining Yang, B.Yang-32@ed.ac.uk, University of Edinburgh, Edinburgh, UK; Ziying Chen, Ztuying.Chan@ed.ac.uk, University of Edinburgh, Edinburgh, UK; Tiejun Ma, tiejun.ma@ed.ac.uk, University of Edinburgh, Edinburgh, UK.



This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2026 Copyright held by the owner/author(s).
ACM 2836-6573/2026/9-ART279
https://doi.org/10.1145/3837117

embedded differently, then cross-dataset queries have no implementation-independent semantics.

At first glance, this may seem unavoidable. However, empirical work in representation learning suggests that independently trained spaces often share latent geometric structure inherent to the data manifold [13]; related behavior has been observed in cross-lingual word embeddings [1, 18], text embeddings [38] and cross-modal embeddings [34]. This suggests that a unified embedding-independent view may exist. However, it is not clear how this could lead to an embedding-independent semantics for vector databases, which we study specifically in this work.

Embedding independence. To address this, we propose to separate logic from implementation: embeddings and indices are physical artifacts and should not determine query semantics.

Metric algebra. At the logical level, we propose *metric algebra*, a coordinate-free model of vector search in which each dataset is represented as a metric relation: a finite set of identifiers placed in an abstract semantic metric schema. Similarity is expressed by range and kNN selection. The key operator is a deterministic logical Union that integrates two metric relations through overlap anchors. Union is defined when the anchors admit a unique overlap-respecting alignment, ensured by existence and rigidity conditions. Thus cross-dataset similarity search has a precise semantics before any embedding model or ANN index is chosen.

Vector realization. At the physical level, a metric relation is implemented by an embedding-specific vector realization with an operational distance and ANN index. Union is realized by a witness map learned from anchors that transports source vectors into a target space, where queries are embedded.

Once Union is separated from its physical realization, three questions arise. (1) When are two vector realizations unionable under witnesses learned in practice? (2) How does witness error affect query evaluation? (3) How do ANN indices interact with Union implemented over realizations?

MetricDB. We address these by developing MetricDB, a proof-of-concept as shown in Fig. 1 that implements metric algebra as a logical layer atop existing vector databases: it treats vector databases as realizations of metric relations and plugs in embedding alignment methods to learn witnesses. MetricDB develops plans for queries over Union and analyzes their robustness to witness error. To make Union practical, we characterize when major ANN vector index families are *transformation-equivariant* and *mergeable*, to compile and integrate existing indices through Union. These translate into the design of MetricDB: it can often reuse and consolidate existing indices for cross-embedding integration, while identifying cases where witness error can be amplified.

Evaluation on benchmarks across major embedding models *e.g.,* Mistral, GTE-Qwen2, NV-Embed-V2, OpenAI-Ada-002, demonstrates the effectiveness of metric algebra and MetricDB for cross-embedding retrieval. When vector indices are transformation-equivariant and mergeable, MetricDB synthesizes vector indices for cross-embedding queries with recall comparable or even consistent with clean indices built from scratch over the union vector realizations.

Contributions. We introduce logical-physical independence to vector databases. Specifically:

- We propose *metric algebra*, a coordinate-free embedding-independent logical layer (Section 2).
- Building upon its algebraic properties, we develop MetricDB that shows how metric algebra can be realized atop vector databases embedded by various distinct embedding models (Section 3).
- We study queries over metric Union that abstract cross-embedding vector search (Section 4).
- We formalize and characterize transformation-equivariance (Section 5) and mergeability (Section 6) of common vector indices, two critical properties that allow MetricDB to compile and merge prebuilt vector database indices for queries over metric Union.

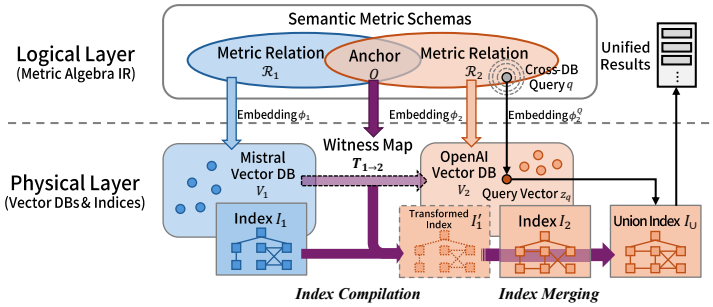


Fig. 1. Embedding-independence by MetricDB

We evaluate the benefits of metric algebra in Section 7, discuss the wider additional related research in Section 8, and conclude in Section 9 (all proofs are deferred to the full version).

2 Metric Algebra

We develop metric algebra at the logical level, independently of embeddings and ANN indices. The principle is that query semantics should be defined directly over semantic metric spaces, while embeddings and indices are implementation artifacts and thus should appear only later as the physical realizations of the logic. For convenience, we summarize frequently used notations in Table 1.

2.1 Data Model

Metric schema. A metric schema is a metric space $\mathcal{M} = (M, d_{\mathcal{M}})$, where M is the domain of semantic points and $d_{\mathcal{M}}$ is the semantic distance function[1]. As a special case, a d-dimensional Euclidean schema is $\mathcal{M}_E = (\mathbb{R}^d, \|\cdot\|_2)$, where $d_{\mathcal{M}_E}(x, y) = \|x - y\|_2$.

Metric relations (logical instances). A metric relation is a triple $\mathcal{R} = (\mathcal{E}, \mathcal{M}, \eta)$ where: (i) $\mathcal{E}$ is a finite set of entity identifiers, (ii) $\mathcal{M} = (M, d_{\mathcal{M}})$ is a metric schema, and (iii) $\eta : \mathcal{E} \to M$ is an injection that places $\mathcal{E}$ onto metric $\mathcal{M}$. The induced logical distance over entities is $d_{\mathcal{R}}(x, y) := d_{\mathcal{M}}(\eta(x), \eta(y))$, for any $x, y \in \mathcal{E}$. When $\mathcal{M}$ is a Euclidean schema, $\mathcal{R}$ is called a Euclidean metric relation.

Identifiers are *local* to each base relation: for any two distinct base relations $\mathcal{R}_i = (\mathcal{E}_i, \cdot, \cdot)$ and $\mathcal{R}_j = (\mathcal{E}_j, \cdot, \cdot)$, we assume $\mathcal{E}_i \cap \mathcal{E}_j = \emptyset$. Intuitively, the same real-world entity may appear in multiple metric relations, but it may be represented by different local identifiers in each one. Such cross-relation correspondences are captured explicitly by using independent local identifiers.

A query over metric relation $\mathcal{R} = (\mathcal{E}, \mathcal{M}, \eta)$ is a point $q \in \mathcal{M}$.

Example 1: Figure 2a shows two Euclidean metric relations $\mathcal{R}_1$ and $\mathcal{R}_2$, modeling two collections of scientific abstracts with identifiers $\{s_1, s_2, s_3, s_4, s_5\}$ and $\{t_1, t_2, t_3, t_4, t_5\}$, respectively. Three abstracts are shared across the two collections, but are indexed differently as s_1, s_2, s_3 in $\mathcal{R}_1$ and t_1, t_2, t_3 in $\mathcal{R}_2$. □

2.2 Algebra Operators

Metric algebra is a closed algebra, *i.e.,* each operator outputs a metric relation, allowing compositional semantics similar to relational algebra. It captures the essence of similarity search via three operators: *range selection, kNN selection*, and *metric* Union. Range selection and kNN selection correspond directly to range-based and kNN-based similarity search, while Union is an entirely

[1]We use the term "metric" for clarity. Extending to non-metric dissimilarities is possible for *e.g.,* kNN, but where metric properties are required we state assumptions explicitly.

Table 1. Summary of frequently used notations

Notation	Meaning
$\mathcal{R} = (\mathcal{E}, \mathcal{M}, \eta)$	logical metric relation
$V = (\mathcal{E}_V, \phi, \delta, \text{Index})$	physical vector realization
$\mathcal{O} = (O, \iota_1, \iota_2)$	anchor specification; O is the anchor-id set
$T_{1\to 2}$	witness map from source V_1 to target V_2
$\mathcal{R}_1 \sqcup_{\mathcal{O}} \mathcal{R}_2$	logical UNION between source $\mathcal{R}_1$ and target $\mathcal{R}_2$
(z, s, t)	query vector z in target/home realization V_t

new operation that is key to the algebra and gives exact semantics to queries across multiple metric relations that can possibly be implemented via different embedding models.

Selection operators. We start with the two selection operators. Consider metric relation $\mathcal{R} = (\mathcal{E}, \mathcal{M}, \eta)$ over schema $\mathcal{M} = (M, d_{\mathcal{M}})$ and any $S \subseteq \mathcal{E}$. Denote by $\mathcal{R} \restriction_S$ the induced metric relation $(S, \mathcal{M}, \eta|_S)$, where $\eta|_S$ is mapping η restricted to identifiers in S; we refer to it as the projection of $\mathcal{R}$ on S. A query q over $\mathcal{R}$ is a query point in the schema, *i.e.,* $q \in M$.

Range selection ($\boldsymbol{\rho_{q,r}(\mathcal{R})}$). Given metric relation $\mathcal{R}$, query q and a radius $r \geq 0$, define $B_{\mathcal{R}}(q, r) := \{x \in \mathcal{E} \mid d_{\mathcal{M}}(\eta(x), q) \leq r\}$. The *range selection* operator returns $\rho_{q,r}(\mathcal{R}) := \mathcal{R} \restriction_{B_{\mathcal{R}}(q,r)}$.

kNN selection ($\boldsymbol{\kappa_{q,k}(\mathcal{R})}$). Fix a global total order $\prec$ on entity identifiers for deterministic tie-breaking. Given metric relation $\mathcal{R}$, query q, and $k \in \mathbb{N}$, let $N_{\mathcal{R}}(q, k)$ be the k elements minimizing the lexicographic key $(d_{\mathcal{M}}(\eta(x), q), x)$. The *kNN selection* operator returns $\kappa_{q,k}(\mathcal{R}) := \mathcal{R} \restriction_{N_{\mathcal{R}}(q,k)}$.

Metric (logical) UNION. We next focus on metric UNION.

Geometrical preliminaries. We start with the preliminary notions.

Anchors. Given metric relations $\mathcal{R}_1 = (\mathcal{E}_1, \cdot)$ and $\mathcal{R}_2 = (\mathcal{E}_2, \cdot)$, an *anchor set* of $\mathcal{R}_1$ and $\mathcal{R}_2$ is a triple $\mathcal{O} = (O, \iota_1, \iota_2)$, where O is a finite set of entity identifiers, and $\iota_i : O \to \mathcal{E}_i$ injectively maps identifiers of $\mathcal{O}$ to those of $\mathcal{E}_i$ ($i \in \{1, 2\}$). Intuitively, $\iota_1(o)$ and $\iota_2(o)$ refer to the same underlying entity.

Isometry group. Let $\mathcal{M} = (M, d_{\mathcal{M}})$ be a metric schema. Its isometry group, denoted by $\text{Iso}(M)$, is

$$\{g : M \to M \text{ bijective} \mid d_{\mathcal{M}}(g(a), g(b)) = d_{\mathcal{M}}(a, b) \quad \forall a, b \in M\}.$$

Determining sets. For any subset $S \subseteq M$, define

$$\text{Stab}_M(S) \;:=\; \{g \in \text{Iso}(M) \mid g(s) = s \quad \forall s \in S\}.$$

Then we call S *determining* in M if $\text{Stab}_M(S) = \{\text{Id}\}$, where Id is the identity function that maps each point in M to itself. In particular, when M is a d-dimensional Euclidean schema, any set S containing $d + 1$ affinely independent points is determining in M.

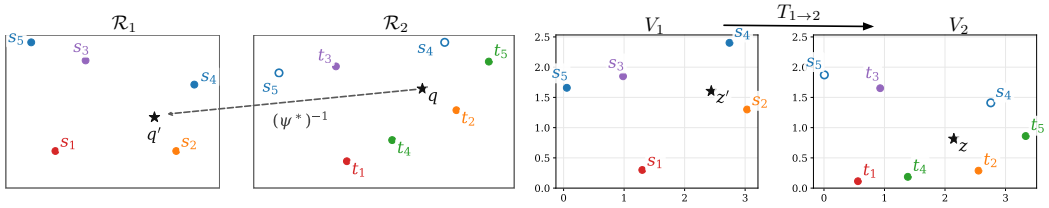
Unionability. Let $\mathcal{R}_1 = (\mathcal{E}_1, \mathcal{M}_1 = (M_1, d_1), \eta_1)$ and $\mathcal{R}_2 = (\mathcal{E}_2, \mathcal{M}_2 = (M_2, d_2), \eta_2)$ be metric relations, and let $\mathcal{O} = (O, \iota_1, \iota_2)$ be an anchor set of $\mathcal{R}_1$ and $\mathcal{R}_2$. We say $\mathcal{R}_1$ is *unionable into* $\mathcal{R}_2$ on $\mathcal{O}$ if:

(1) *Existence:* There exists an isometry $\psi : M_1 \to M_2$ such that

$$\psi(\eta_1(\iota_1(o))) \;=\; \eta_2(\iota_2(o)) \quad \forall o \in O.$$

(2) *Rigidity:* Let $S := \eta_2(\iota_2(O)) \subseteq M_2$. Then $\text{Stab}_{M_2}(S) \;=\; \{\text{Id}\}$.

Lemma 1: *If $\mathcal{R}_1$ is unionable into $\mathcal{R}_2$ on $\mathcal{O}$, then the isometry ψ satisfying the unionability constraints* w.r.t. *$\mathcal{O}$ above is unique.* □



(a) *Metric relations* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ (b) *Vector realizations (*2D *Euclidean) of* $\mathcal{R}_1$ *and* $\mathcal{R}_2$

Fig. 2. Example metric relations and their vector realizations in 2D **Euclidean space**

When $\mathcal{R}_1$ is unionable into $\mathcal{R}_2$ with this unique isometry ψ on O, we refer to ψ as the *overlap-respecting* isometry from $\mathcal{R}_1$ to $\mathcal{R}_2$.

Semantics of **Union**. If $\mathcal{R}_1 = (\mathcal{E}_1, \mathcal{M}_1, \eta_1)$ is unionable into $\mathcal{R}_2 = (\mathcal{E}_2, \mathcal{M}_2, \eta_2)$ on O, let ψ^* be the unique overlap-respecting isometry. For simplicity, we assume the only cross-dataset duplicates we identify are those in the anchor set O; we choose the $\mathcal{E}_2$ identifiers as canonical representatives and drop $\iota_1(O)$. Specifically, define the set $\mathcal{E}_\sqcup := (\mathcal{E}_1 \cup \mathcal{E}_2) \setminus \iota_1(O)$, and let $\eta_\sqcup : \mathcal{E}_\sqcup \to M_2$ be

$$\eta_\sqcup(x) \;:=\; \begin{cases} \eta_2(x), & x \in \mathcal{E}_2, \\ \psi^*(\eta_1(x)), & x \in \mathcal{E}_1 \setminus \iota_1(O). \end{cases}$$

The logical Union of $\mathcal{R}_1$ and $\mathcal{R}_2$, denoted by $\mathcal{R}_1 \sqcup_O \mathcal{R}_2$ (or $\mathcal{R}_1 \sqcup \mathcal{R}_2$ when O is clear from context), is

$$\mathcal{R}_1 \sqcup_O \mathcal{R}_2 \;:=\; (\mathcal{E}_\sqcup, \mathcal{M}_2, \eta_\sqcup).$$

Thus the target relation's schema $\mathcal{M}_2$ serves as the schema of the merged relation, target identifiers are kept as canonical anchor representatives, and every non-anchor source identifier is transported into $\mathcal{M}_2$ through ψ^*. By Lemma 1, metric Union is well-defined and deterministic. Furthermore, under anchor identifier correspondence, Union is commutative as traditional set union as $\mathcal{R}_1 \sqcup_O \mathcal{R}_2$ and $\mathcal{R}_2 \sqcup_O \mathcal{R}_1$ return the same merged semantic relation. When $\mathcal{R}_1$ and $\mathcal{R}_2$ are over the same metric schema $\mathcal{M}$, metric Union reduces to an ordinary union, denoted by $\mathcal{R}_1 \cup \mathcal{R}_2$, where we simply take union of identifiers of $\mathcal{R}_1$ and $\mathcal{R}_2$ and keep schema $\mathcal{M}$ and distance η.

Example 2: Continuing Example 1, the three shared abstracts are recorded as an anchor set $O = (\{o_1, o_2, o_3\}, \tau_1, \tau_2)$ of $\mathcal{R}_1$ and $\mathcal{R}_2$, with $\tau_1(o_i) = s_i$, $\tau_2(o_i) = t_i (i \in \{1, 2, 3\})$. Since the anchors are non-collinear in the Euclidean schema and hence determining, $\mathcal{R}_1$ and $\mathcal{R}_2$ admit a unique overlap-respecting isometry, namely ψ^*. Union transports the non-anchor source identifiers (*i.e.,* s_4, s_5) through ψ^*, which are shown as hollow blue points in $\mathcal{R}_2$ of Figure 2a. □

For convenience, if $\mathcal{S} = \mathcal{R}_1 \restriction X$ is any source-side subrelation, we write $\overline{\mathcal{S}}$ for the target-schema relation obtained by applying the same transport-and-canonicalization construction to $\mathcal{S}$ as we do for realizing $\mathcal{R}_1 \sqcup_O \mathcal{R}_2$ above: non-anchor identifiers in X are placed by the overlap-respecting isometry ψ^*, and any surviving source anchors are replaced by their target representatives.

2.3 Algebraic Properties

The value of metric algebra is not only that metric Union is well-defined, but also that its operators satisfy useful identities for compositional semantics and query rewriting.

(1) Coordinate-free semantics. Metric algebra depends on metric structure, not on coordinates.

Proposition 2: *Let* $\mathcal{R} = (\mathcal{E}, \mathcal{M}, \eta)$, $h : \mathcal{M} \to \mathcal{N}$ *be an isometry, and* $\mathcal{R}^h := (\mathcal{E}, \mathcal{N}, h \circ \eta)$ *be the re-coordinatized relation by applying* h *to* $\mathcal{R}$. *Then, for every query* $q \in \mathcal{M}$, *radius* $r \geq 0$, *and* $k \in \mathbb{N}$,

- *the answer identifiers of* $\rho_{q,r}(\mathcal{R})$ *and* $\rho_{h(q),r}(\mathcal{R}^h)$ *coincide, and*
- *the answer identifiers of* $\kappa_{q,k}(\mathcal{R})$ *and* $\kappa_{h(q),k}(\mathcal{R}^h)$ *coincide.* □

Thus metric algebra is coordinate-free: re-coordinatizing a relation by an isometry and re-coordinatizing the query point by the same isometry does not change the answer identifiers.

(2) Selection pushdown through Union. Since $\mathcal{R}_1 \sqcup_O \mathcal{R}_2$ is defined in the target schema $\mathcal{M}_2$, a target query $q \in \mathcal{M}_2$ induces the corresponding source query $q' := (\psi^*)^{-1}(q) \in \mathcal{M}_1$. Assume that the k-th and $(k+1)$-th semantic distances in $\mathcal{R}_1 \sqcup_O \mathcal{R}_2$ for q are distinct.

Theorem 3: *Let* $\mathcal{R}_1$ *be unionable into* $\mathcal{R}_2$ *on* O *and* $q \in \mathcal{M}_2$*. Then:*

kNN *pushdown:* $\kappa_{q,k}(\mathcal{R}_1 \sqcup_O \mathcal{R}_2) = \kappa_{q,k}\Big(\overline{\kappa_{q',k}(\mathcal{R}_1)} \cup \kappa_{q,k}(\mathcal{R}_2)\Big)$.

Range *pushdown:* $\rho_{q,r}(\mathcal{R}_1 \sqcup_O \mathcal{R}_2) = \overline{\rho_{q',r}(\mathcal{R}_1)} \cup \rho_{q,r}(\mathcal{R}_2)$. □

Thus range and kNN queries over Union can be decomposed into a source-side subquery and a target-side subquery, with source-side answers transported into the target schema and merged there.

Example 3: Continuing Example 2, for the target-side query point q in $\mathcal{R}_2$'s schema, the transported source query is q', determined by the unique overlap-respecting isometry ψ^* as shown in Fig. 2a. Theorem 3 says that q over $\mathcal{R}_1 \sqcup_O \mathcal{R}_2$ can be evaluated either directly in the target schema over the unioned result or by pushing it down into one subquery over $\mathcal{R}_2$ at q and one over $\mathcal{R}_1$ at q'. □

(3) Partition-wise decomposition. Let $C = \{\mathcal{E}_1^{(1)}, \ldots, \mathcal{E}_1^{(m)}\}$ be any partitioning of the source identifier set $\mathcal{E}_1$. For each partition $j \in [m]$, let $\mathcal{R}_1^{(j)} := \mathcal{R}_1 \restriction \mathcal{E}_1^{(j)}$. Since all partitions inherit the same overlap-respecting isometry ψ^*, their images $\overline{\mathcal{R}_1^{(j)}}$ are all defined in the target schema $\mathcal{M}_2$.

Proposition 4: *For any partition* C *of* $\mathcal{E}_1$, $\overline{\mathcal{R}}_1 = \overline{\mathcal{R}_1^{(1)}} \cup \cdots \cup \overline{\mathcal{R}_1^{(m)}}$*, and therefore* $\mathcal{R}_1 \sqcup_O \mathcal{R}_2 = \big(\overline{\mathcal{R}_1^{(1)}} \cup \cdots \cup \overline{\mathcal{R}_1^{(m)}}\big) \cup \mathcal{R}_2$. □

Thus metric Union admits an exact partition-wise decomposition whenever a global ψ^* exists.

3 Implementing Metric Algebra

We next connect the semantics of metric algebra to existing vector databases. The key idea is that a vector database is a physical implementation of a metric relation, and a witness map learned from anchors is a realization of the overlap-respecting isometry ψ^* that underlies Union.

3.1 From Metric Relation to Vector Realization

A metric relation captures semantic distances independently of coordinates. A deployed system, however, stores vectors and answers queries by vector search. We bridge the two by *vector realizations*.

Vector Realizations (physical instances). A vector realization is a tuple $V = (\mathcal{E}_V, \phi, \delta, \mathsf{Index})$ where (i) $\mathcal{E}_V$ is a finite set of entity identifiers, (ii) $\phi : \mathcal{E}_V \to \mathbb{R}^m$ is an embedding function (*e.g.,* OpenAI-Ada) that stores a vector of $\mathbb{R}^m$ per identifier, (iii) δ is the *operational distance* on $\mathbb{R}^m$, *e.g.,* Euclidean or cosine, and (iv) Index is a vector index for approximate vector search under δ, *e.g.,* IVF or HNSW.

When V is used to answer queries over metric relation $\mathcal{R} = (\mathcal{E}, \mathcal{M}, \eta)$, we also assume an associated query embedding function $\phi^Q : \mathcal{M} \to \mathbb{R}^m$ produced by the same embedding pipeline. A logical query $q \in \mathcal{M}$ is answered physically using query vector $\phi^Q(q)$.

A vector realization implements a metric relation exactly when its vector distances agree with the logical distances of that relation, up to an isometric change of coordinates.

Realization. Consider metric relation $\mathcal{R} = (\mathcal{E}, \mathcal{M} = (M, d_{\mathcal{M}}), \eta)$ and vector realization $V = (\mathcal{E}, \phi, \delta, \cdot)$. We say V is an *exact realization* of $\mathcal{R}$ if there exists an *isometry* (bijective distance-preserving)

$$f\colon (M, d_{\mathcal{M}}) \longrightarrow (X, \delta|_X) \quad \text{for some } X \subseteq \mathbb{R}^m,$$
$$\text{such that} \qquad \phi(x) = f(\eta(x)), \qquad \forall x \in \mathcal{E}.$$

Since f is an isometry, *i.e.,* a bijective distance-preserving mapping from M to X, for any $x, y \in \mathcal{E}$, $d_{\mathcal{M}}(\eta(x), \eta(y)) = \delta(\phi(x), \phi(y))$. Intuitively, when V is an exact realization of $\mathcal{R}$, it is an exact implementation of $\mathcal{R}$ via an ideal embedding model equal to the function $f \circ \eta$, such that pairwise semantics distances of $\mathcal{R}$ are faithfully recorded by vector distances of V. Different realizations may implement the same logical relation using different embeddings.

Example 4: Continuing Example 3, Fig. 2b depicts V_1 and V_2 as exact vector realizations of $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively. Unlike metric relations, V_1 and V_2 consist of concrete vectors in $\mathbb{R}^2$. Note that the anchors are realized as different vectors across V_1 and V_2. Further, their geometric shapes in Fig. 2b are isometric to those of Fig. 2a. □

We next connect metric Union to physical vector realizations.

Physical unionability. Consider metric relations $\mathcal{R}_1 = (\mathcal{E}_1, \mathcal{M}_1, \eta_1)$ and $\mathcal{R}_2 = (\mathcal{E}_2, \mathcal{M}_2, \eta_2)$ over schemas $\mathcal{M}_1 = (M_1, d_{\mathcal{M}_1})$ and $\mathcal{M}_2 = (M_2, d_{\mathcal{M}_2})$, respectively, with overlap anchor set $O = (O, \iota_1, \iota_2)$. Let $V_1 = (\mathcal{E}_1, \phi_1, \delta_1, \cdot)$ and $V_2 = (\mathcal{E}_2, \phi_2, \delta_2, \cdot)$ be exact realizations of $\mathcal{R}_1$ and $\mathcal{R}_2$ (recall Section 2.1), respectively, witnessed by isometries

$$f_1 : (M_1, d_{\mathcal{M}_1}) \to (X_1, \delta_1|_{X_1}), \qquad f_2 : (M_2, d_{\mathcal{M}_2}) \to (X_2, \delta_2|_{X_2}),$$

so $\phi_i = f_i \circ \eta_i : \mathcal{E}_i \to \mathbb{R}^{m_i}$ on $\mathcal{E}_i (i \in \{1, 2\})$.

We say that V_1 is *physically unionable* (or simply, unionable) into V_2 on O if there exists $T_{1\to 2} : \mathbb{R}^{m_1} \to \mathbb{R}^{m_2}$ that satisfies all below:

(1) *(Isometric)* the restriction $T_{1\to 2}|_{X_1} : X_1 \to X_2$ is an isometry (*i.e.,* bijective and distance-preserving under $\delta_1|_{X_1}$ and $\delta_2|_{X_2}$);
(2) *(Anchor alignment)* for all $o \in O$, $T_{1\to 2}(\phi_1(\iota_1(o))) = \phi_2(\iota_2(o))$;
(3) *(Rigidity)* the set of anchor vectors $A := \phi_2(\iota_2(O)) \subseteq X_2$ is determining (recall Section 2.2) in X_2, *i.e.,* $\mathrm{Stab}_{X_2}(A) = \{\mathrm{Id}\}$.

When these hold, we refer to $T_{1\to 2}$ as an exact **witness map** of the physical union of V_1 and V_2. We remark that the definition is used for soundness; as we will see later MetricDB uses anchor-based witness with certificates as sufficient conditions in practice.

Vector union. When $T_{1\to 2}$ witnesses that $V_1 = (\mathcal{E}_1, \phi_1, \delta_1, \cdot)$ is physically unionable into $V_2 = (\mathcal{E}_2, \phi_2, \delta_2, \cdot)$ on O, the result of the physical union, also referred to as the *vector union* of V_1 into V_2 and denoted by $V_1 \sqcup_O V_2$, is defined as $(\mathcal{E}_\sqcup, \phi_\sqcup, \delta_\sqcup, \cdot)$, where (i) $\mathcal{E}_\sqcup$ is the fused identifier set *w.r.t.* O, exactly the same as in metric Union; (ii)

$$\phi_\sqcup(x) := \begin{cases} \phi_2(x), & x \in \mathcal{E}_2, \\ T_{1\to 2}(\phi_1(x)), & x \in \mathcal{E}_1 \setminus \iota_1(O); \end{cases}$$

(iii) $\delta_\sqcup$ is δ_2. We also call V_2 the target space.

Logical-physical equivalence. Let $\mathcal{R}_1 = (\mathcal{E}_1, \mathcal{M}_1, \eta_1)$ and $\mathcal{R}_2 = (\mathcal{E}_2, \mathcal{M}_2, \eta_2)$ be metric relations with overlap anchors $O = (O, \iota_1, \iota_2)$. Let V_1 and V_2 be exact realizations of $\mathcal{R}_1$ and $\mathcal{R}_2$. Then we have:

Theorem 5: *(1) If V_1 is unionable into V_2 on O, then $\mathcal{R}_1$ is unionable into $\mathcal{R}_2$ on O and $V_1 \sqcup_O V_2$ is an exact realization of $\mathcal{R}_1 \sqcup_O \mathcal{R}_2$.*

(2) If both $\mathcal{R}_i$ and V_i are Euclidean ($i \in \{1,2\}$), then $\mathcal{R}_1$ is unionable into $\mathcal{R}_2$ on O if and only if *V_1 is physically unionable into V_2 on O with a witness $T_{1\to2} : \mathbb{R}^{m_1} \to \mathbb{R}^{m_2}$ (assume $m_2 \geq m_1$) of the form*

$$T_{1\to2}(z) \;=\; Qz + t,$$

where $t \in \mathbb{R}^{m_2}$ and $Q \in \mathbb{R}^{m_2 \times m_1}$ satisfies $Q^\top Q = I$. □

Theorem 5 formalizes the logical-physical separation that we seek. Theorem 5(1) is a semantic soundness guarantee: whenever a physical witness exists, vector union correctly implements logical Union. Theorem 5(2) identifies the most important practical case: for Euclidean semantics and Euclidean vector realizations, logical unionability is exactly the existence of a linear isometric witness. This is why Procrustes-type alignment [6, 26] is a natural witness choice for implementing Union.

Example 5: Continuing Example 4, suppose that V_1 and V_2 exactly realize $\mathcal{R}_1$ and $\mathcal{R}_2$ through isometries f_1 and f_2, respectively. Then the exact witness from V_1 to V_2 is $T_{1\to2} = f_2 \circ \psi^* \circ f_1^{-1}$, where ψ^* is the overlap-respecting isometry from $\mathcal{R}_1$ to $\mathcal{R}_2$. In the Euclidean setting of Fig. 2b, orthogonal Procrustes on the three anchor vector pairs recovers a witness of the form $T_{1\to2}(z) = Q_T z + c$ from Theorem 5(2). Applying $T_{1\to2}$ transports the non-anchor source vectors into V_2, shown as the hollow blue points in Fig. 2b, while query point q from Example 3 is embedded as query vectors z and z'. □

3.2 Witnesses in the Wild

Theorem 5 gives an exact semantic baseline. However, real deployments are approximate: learned embeddings distort semantic distances, learned witness maps fit anchors only approximately, and ANN indices introduce search error. So, we treat exact realization as reference semantics and use learned witnesses with anchor-based certificates to decide when cross-embedding Union is reliable.

Concretely, we uniformly treat a witness from V_1 to V_2 as a map $T_{1\to2} : \mathbb{R}^{m_1} \to \mathbb{R}^{m_2}$, regardless of how that map is represented. Given such a witness, we can evaluate source vectors in the target space by replacing $\phi_1(x)$ with $T_{1\to2}(\phi_1(x))$ and measuring distances using the target metric δ_2. Witnesses thus provide a target-space representation of the requested Union that reuses the target realization's search machinery. We consider three witness families.

(W1) Isometry witness (iso). For Euclidean realizations, the canonical family is a global isometry witness $T_{1\to2}(z) = Qz + t$ with $Q^\top Q = I$ (Theorem 5(2)). We learn (Q, t) by orthogonal Procrustes [6, 26] on anchors and certify it by two anchor-derived quantities:

(a) Anchor residual (fit quality). Given overlap anchors $O = (O, \iota_1, \iota_2)$, the residual of $T_{1\to2}$ is evaluated in the target metric as

$$\mathrm{Res}(T_{1\to2}; O) := \Big(\frac{1}{|O|}\sum_{o\in O} \delta_2(T_{1\to2}(\phi_1(\iota_1(o))),\ \phi_2(\iota_2(o)))^2\Big)^{1/2}.$$

Intuitively, Res measures how well $T_{1\to2}$ aligns the overlap anchors O from V_1 to V_2: the smaller $\mathrm{Res}(T_{1\to2}; O)$ is, the better $T_{1\to2}$ fits $V_1[O]$ to $V_2[O]$, where $V_i[O]$ denotes the anchor vectors in V_i.

(b) Rigidity proxy (stability). Small residual alone does not rule out degenerate anchor configurations (e.g., nearly collinear anchors) that admit many very different transformations with similar anchor fit. MetricDB thus computes a rigidity proxy that measures whether anchors provide a well-conditioned alignment signal by the ratio of minimum and maximum singular values of the Procrustes cross-covariance of anchors [38]; small rigidity indicates that Union may be unstable even when Res is small.

(W2) Multi-isometry witness (miso). When a global isometry is too restrictive and only local neighborhoods align as empirically discovered in [38] for many text embeddings, a *multi-isometry* witness family is preferable. Let $C = \{\mathcal{E}_1^{(1)}, \ldots, \mathcal{E}_1^{(m)}\}$ be a partition of the source identifiers, exactly as in

Proposition 4, and for each $j \in [m]$ let O_j be the anchors belonging to that partition. We learn one Euclidean witness $T_j(z) = Q_j z + t_j$ with $Q_j^\top Q_j = I$ from O_j for each partition j and transform source partition $\mathcal{E}_1^j$ with witness T_j and integrate with the target metric relation.

Proposition 4 gives the logical motivation for MISO. At the semantic level, metric UNION already admits an exact partition-wise decomposition over any partitioning of source identifiers, provided all partitions are transported by the same overlap-respecting isometry ψ^*. MISO preserves the same logical target $\mathcal{R}_1 \sqcup_O \mathcal{R}_2$, but relaxes its physical realization: instead of learning one global approximation to ψ^*, it learns several locally better fits, one per source partition.

(W3) Neural witnesses (NN). Another choice of witness maps is to learn a mapping over vector realizations of anchors via *e.g.,* MLP [12], CCA [22] or Gromov-Wasserstein learning [1]. These can achieve low anchor residual but may distort cross-realization distances away from anchors; accordingly, MetricDB treats them as pluggable witness providers and relies on held-out anchor diagnostics and model-specific generalization controls in place of rigidity proxies.

3.3 MetricDB: **Putting Metric Algebra in Action**

MetricDB implements metric algebra as a thin logical layer above existing vector databases. Users do not manipulate metric relations directly. Instead, it serves as an internal semantic intermediate representation that gives precise meaning to cross-embedding queries.

MetricDB maintains a catalog $\{(\mathcal{R}_1, V_1), \ldots, (\mathcal{R}_n, V_n)\}$, where each V_i refers to a registered vector database and $\mathcal{R}_i$ is the metric relation that identifies its logical semantics internally. For each ordered pair of realizations, MetricDB profiles overlap anchors, learns a witness family (ISO, MISO or NN) from the source into the target realization, computes associated certificates, and records whether the pair is reliable enough to support cross-embedding union queries.

Query interface. A query to MetricDB is specified by a query vector z, a source realization V_s, and a target realization V_t; we also refer to V_t as a home realization for z as they are both embedded by the same embedding function. We write this as (z, s, t), with the convention that $z = \phi_t^Q(q)$ for some query $q \in \mathcal{M}_t$. That is, the query vector always lives in the home/target space, and any additional source realization is transported into that space. Given (z, s, t), MetricDB works as follows.

(a) If $s = t$, it performs a standard vector search over V_t, corresponding to $\rho_{q,r}(\mathcal{R}_t)$ or $\kappa_{q,k}(\mathcal{R}_t)$.

(b) If $s \neq t$ and the source realization V_s is supported in the target realization V_t by a witness map, then the query has unambiguous logical meaning over UNION $\mathcal{R}_s \sqcup_O \mathcal{R}_t$, where O is the profiled anchor set. MetricDB then decides a *physical plan* to answer that logical query (more below)

(c) If $s \neq t$ but no acceptable witness is available, then the requested cross-embedding union is not semantically supported. MetricDB returns the local result over V_t, but reports it explicitly as a local-only fallback instead of an answer to the requested union.

This captures the motivating federated scenario: a query originates in one autonomous vector database, but the user wishes to search jointly over that home collection and additional foreign collections that may live in a different embedding space and use different identifiers and ANN indices.

Planning. For supported V_s and V_t vector realization pairs, MetricDB picks from two physical plans that mirror the pushdown laws of Theorem 3.

(a) Hard Union. We follow the left side of rewriting rules of Theorem 3: compute $\mathcal{R}_s \sqcup \mathcal{R}_t$ via vector union with learned witness, and then evaluate $z = \phi_t^Q(q)$ over the union result. In essence, it directly follows the query expression without leveraging rewriting.

(b) Soft Union. Alternatively, MetricDB keeps V_s and V_t separate. For a target-space query vector $z = \phi_t^Q(q)$, it queries V_t directly with z, and queries V_s with a source-side proxy of z that approximates the transported query point $q' = (\psi^*)^{-1}(q)$ from Theorem 3. The returned source candidates are then mapped into the target space and merged with the target candidates there.

Hard union favors reuse when many queries target the same unioned realization. Soft union avoids materialization but is more sensitive to witness quality as we will see in Section 4. Sections 5-6 then study how these physical plans interact with ANN indices.

4 Query Planning over Unions

Section 3 introduced hard and soft plans for queries over vector unions. We next analyze how witness error affects those plans. Throughout the section, we fix a unionable source-target pair $(\mathcal{R}_1, V_1), (\mathcal{R}_2, V_2)$, with anchor set $O = (O, \iota_1, \iota_2)$, and we keep the source/target convention of Section 3: $\mathcal{R}_1, V_1$ denote the source side, $\mathcal{R}_2, V_2$ the target side, and query points live in the target schema $\mathcal{M}_2$ of $\mathcal{R}_2$. We focus on kNN selection, as it is the dominant primitive in practice (range selection is analogous). To isolate witness error from ANN approximation, we assume exact vector search within every realization in this section. Sections 5 and 6 then study the interaction with ANN indices.

Setup. Let $\mathcal{R}_\sqcup := \mathcal{R}_1 \sqcup_O \mathcal{R}_2$ be the Union relation with anchors $O = (O, \iota_1, \iota_2)$ over schema $\mathcal{M}_2 = (M_2, d_2)$. Let $V_1 = (\mathcal{E}_1, \phi_1, \delta_1, \cdot)$ and $V_2 = (\mathcal{E}_2, \phi_2, \delta_2, \cdot)$ be the vector realizations, and let $T_{1\to2} : \mathbb{R}^{m_1} \to \mathbb{R}^{m_2}$ be a witness map from V_1 to V_2. The induced vector union realization is $V_\sqcup := V_1 \sqcup_O V_2 = (\mathcal{E}_\sqcup, \phi_\sqcup, \delta_2, \cdot)$, and a query is a schema point $q \in M_2$ with target query vector $z := \phi_2^Q(q) \in \mathbb{R}^{m_2}$. Recall metric algebra treats kNN selection as an unordered set, with deterministic tie-breaking by the global total ID order $\prec$. For any set S and any score function $s : S \to \mathbb{R}$, define $\mathrm{TopK}(S, s, k)$ as the set of k elements minimizing the lexicographic key $(s(x), x)$.

In the analysis we focus on the most important special case from Theorem 5(2): a Euclidean isometry witness $T_{1\to2}(z) = Qz + t$ with $Q^\top Q = I$. We also assume V_1 and V_2 are Euclidean vector realizations so δ_1 and δ_2 are L2 distances. We will link the accuracy of answers to q over $R_\sqcup$ with the anchor residual of witness $T_{1\to2}$ observed over O, *i.e.,* $\mathrm{Res}(T_{1\to2}; O)$ (recall Sec. 3.2). Since q is usually not from O, or worse, may not even be in $\mathcal{E}_2$, to logically imply an error bound on non-anchor points from witness error $\mathrm{Res}(T_{1\to2}; O)$ observed over O, we adopt the standard generalization assumption commonly used in model analysis: for the query point q of interest,

$$\sup_{x \in \mathcal{E}_\sqcup} \left| \widehat{D}(x \mid q) - D(x \mid q) \right| \le \max_{o \in O} \left\| T_{1\to2}(\phi_1(\iota_1(o))) - \phi_2(\iota_2(o)) \right\|_2,$$

where $D(x \mid q) := d_{\mathcal{R}_\sqcup}(x, q)$ is the ideal semantic distance between x and q and $\widehat{D}(x \mid q) := \delta_2(\phi_\sqcup(x), z)$ is the computed vector distance over the query answer vector space (*i.e., V_2*). Intuitively, the right side of the inequality is the observed maximum pairwise distance between an anchor's embedding in V_2 and the mapped image of its embedding in V_1 transformed by $T_{1\to2}$, and the left side is the gap extended to all points of concern in the unioned relation.

Stability. We next develop analytical bounds that characterize the stability of hard and soft union against witness errors. We will use a boundary margin parameter, denoted by $\Delta_k(q)$:

$$\Delta_k(q) := D(x_{(k+1)} \mid q) - D(x_{(k)} \mid q),$$

where $x_{(k)}$ and $x_{(k+1)}$ are the k-th and $(k+1)$-th nearest IDs under ideal distance function $D(\cdot \mid q)$.

Hard union stability condition. Hard union evaluating $\kappa_{q,k}$ over $\mathcal{R}_1 \sqcup_O \mathcal{R}_2$ returns set $\mathrm{TopK}(\mathcal{E}_\sqcup, \widehat{D}(x \mid q), k)$. Hard union is robust to witness errors: a query q is safe over Union with hard union implementation when the gap $\Delta_k(q)$ is sufficiently large compared to the observed anchor residual of the witness map $T_{1\to2}$. Specifically:

Lemma 6: *If* $\Delta_k(q) > 2\sqrt{|O|}\,\mathrm{Res}(T_{1\to 2}; O)$*, then hard union returns exactly the correct answer* $\kappa_{q,k}(\mathcal{R}_1 \sqcup \mathcal{R}_2)$ *with no error.* □

Soft union stability condition. Note that soft union executes in two phases: a retrieval phase where q and its reversed image q' search over $\mathcal{R}_2$ (via V_2) and $\mathcal{R}_1$ (via V_1), respectively, yielding $C_2(q) \subseteq \mathcal{E}_2$ (target-side candidates) and $C_1(q) \subseteq \mathcal{E}_1 \setminus \iota_1(O)$ (source-side candidates). Denote by $C(q) := C_1(q) \cup C_2(q)$. Then soft union that implements $\kappa_{q,k}$ over $\mathcal{R}_1 \sqcup_O \mathcal{R}_2$ returns set $\mathrm{TopK}(C(q), \widehat{D}(x \mid q), k)$. By comparing this with hard union above, we can see that the correctness of soft union is also affected by whether $C(q)$ would contain all exact answers, in addition to the margin condition that safeguards the performance of $\widehat{D}$ against D as for hard union.

Specifically, let $N_k(q) := \mathrm{TopK}(\mathcal{E}_\sqcup, D(\cdot \mid q), k)$ be the ideal logical answer to $\kappa_{q,k}$ over $\mathcal{R}_1 \sqcup_O \mathcal{R}_2$.

Corollary 7: *If* $N_k(q) \not\subseteq C(q)$ *, then soft union is not correct even if* $\mathrm{Res}(T_{1\to 2}; O) = 0$*. If* $N_k(q) \subseteq C(q)$ *and* $\Delta_k(q) > 2\sqrt{|O|}\,\mathrm{Res}(T_{1\to 2}; O)$*, then soft union returns exactly the correct answer* $N_k(q)$*.* □

Observations. (1) In terms of correctness against witness error, hard union requires only the margin condition $\Delta_k(q) > 2\sqrt{|O|}\mathrm{Res}(T_{1\to 2}; O)$, while soft union requires the same margin condition and, in addition, the coverage condition $N_k(q) \subseteq C(q)$. Hence, queries guaranteed to be correct for soft union are also guaranteed to be correct for hard union with the same witness $T_{1\to 2}$, but not vice versa.

(2) Soft union is query-sensitive and deteriorates with out-of-distribution (OOD) queries, while hard union is less query-sensitive and handles OOD queries better. To build $C_1(q)$, soft union must search V_1, whose space is $\mathbb{R}^{m_1}$. Under a Euclidean isometry witness $T_{1\to 2}(z)$, the natural "proxy" for the source query vector is the left-inverse of $T_{1\to 2}$, which implements the transported query of the rewriting rule in Theorem 3. When $T_{1\to 2}$ is imperfect, the proxy may be mislocated, which in particular often gets worse for OOD queries as they move away from anchors. In contrast, hard union is less query-sensitive as it never needs a source-side query proxy.

Implication for MetricDB[MISO]. The analysis above is stated for ISO witness family, but the same conclusion extends directly to the multi-isometry witness family (MISO) of Section 3.2. Under MetricDB[MISO], the logical target remains the same metric UNION $\mathcal{R}_1 \sqcup_O \mathcal{R}_2$; only its physical realization changes. Each source partition $\mathcal{E}_1^{(j)} \in \mathcal{C}$ is materialized through its own local witness T_j, with partition-wise certificates Res_j and Rigid_j. Thus hard union benefits directly from replacing one global approximation with several locally better fits. In contrast, soft union remains additionally exposed to query-transport and candidate-coverage failure, and in the MISO case it must also route the query to the correct partition. Hence MISO strengthens hard union most directly, while soft union remains the more fragile plan even when local witnesses are used.

5 Transformation-Equivariant Index

Section 4 showed that hard union is the more robust plan under imperfect witnesses. However, hard union has to materialize target-space union $V_\sqcup = V_1 \sqcup_O V_2$ and then build an ANN index over the materialized vectors to answer subsequent queries efficiently. Rebuilding an index from scratch can be costly. This section asks the question that makes hard union practical: *Can a prebuilt source index be compiled into the target space under a witness map, instead of being rebuilt from scratch?*

The key observation is that the witnesses used to realize metric UNION are, in the exact case, distance-preserving changes of coordinates, and in the approximate case they are designed to be close to such changes of coordinates. ANN indices are fundamentally comparison-based: their build procedures and query procedures depend on vector distances and on the orderings induced by those distances. This suggests that many source indices can be *compiled* under a witness transformation.

We formalize this property as *transformation-equivariance*, decide it for common ANN primitives, and then analyze how imperfect witnesses affect compiled indices in practice.

5.1 Transformation-Equivariance

We formalize transformation-equivariance. We identify a vector realization $V = (\mathcal{E}, \phi, \|\cdot\|_2, \mathit{Index})$ with map $\phi_V : \mathcal{E} \to \mathbb{R}^d$, where $\phi_V(e) = x$ means entity identifier $e \in \mathcal{E}$ is stored as vector $x \in \mathbb{R}^d$ in V. We consider Euclidean realizations V_1 (m_1-dimensional) and V_2 (m_2-dimensional), and the witness family $\mathcal{T}_{m_1 \to m_2}$ from Theorem 5(2) comprising all maps $T : \mathbb{R}^{m_1} \to \mathbb{R}^{m_2}$ of form $T(x) = Qx + t$, where $Q \in \mathbb{R}^{m_2 \times m_1}$, $t \in \mathbb{R}^{m_2}$, and $Q^\top Q = I_{m_1}$. For any $T \in \mathcal{T}_{m_1 \to m_2}$ and any vector realization $V : \mathcal{E} \to \mathbb{R}^{m_1}$, we write $T(V)$ for the transformed vector realization, *i.e.,* map $T \circ \phi_V : \mathcal{E} \to \mathbb{R}^{m_2}$.

Index primitives. We view an ANN index primitive f as a deterministic map from input index states to output index states. An index state consists of (a) *vector state*, *e.g.,* vector-valued fields of an ambient dimension d, such as centroids, codebook, stored vectors, and (b) *discrete state*, *e.g.,* IDs, adjacency lists, code indices, often index dependent. We treat a vector realization as a trivial vector state whose vector-valued fields are simply the vectors.

Let $\mathrm{State}_f(d)$ denote the set of all valid output states produced by running f on d-dimensional inputs (*i.e.,* vector states are d-dimensional), with f's hyperparameters fixed. For any index state S with vector-valued fields in $\mathbb{R}^{m_1}$, and any witness $T : \mathbb{R}^{m_1} \to \mathbb{R}^{m_2}$, we write $T[S]$ for the state obtained by applying T to *every vector* in S, while leaving discrete state unchanged.

Transformation-equivariance. An index primitive f is $\mathcal{T}$*-equivariant* if for every $T \in \mathcal{T}_{m_1 \to m_2}$ there exists a state transformer $\Gamma_f(T) : \mathrm{State}_f(m_1) \to \mathrm{State}_f(m_2)$ such that for *every* valid m_1-dimensional input state S of f, $f\big(T[S]\big) = \Gamma_f(T)\big(f(S)\big)$. We also refer to $\Gamma_f(T)$ as a $\mathcal{T}$-equivariant *compilation* of index f. Note that $\Gamma_f(T)$ is not arbitrary but instead tightly coupled to f and T, because it must satisfy the commutation identity for *all* valid S. In particular, if there exist S_1 and S_2 such that $f(S_1) = f(S_2)$ but $f(T[S_1]) \neq f(T[S_2])$, then no such $\Gamma_f(T)$ can exist.

Composite indices. Many practical indices are composites, built as a pipeline $\mathsf{A} = f_s \circ \cdots \circ f_1$ of stages (preprocessing, coarse structure, compression). We show that $\mathcal{T}$-equivariance composes.

Lemma 8: *If each stage f_i of* A *is $\mathcal{T}$-equivariant with state transformer Γ_{f_i}, then* A *is $\mathcal{T}$-equivariant with state transformer $\Gamma_{\mathsf{A}}(\cdot) := \Gamma_{f_s}(\cdot) \circ \Gamma_{f_{s-1}}(\cdot) \circ \cdots \circ \Gamma_{f_1}(\cdot)$. That is, for every witness $T \in \mathcal{T}$ and every valid input vector realization V, we have $\mathsf{A}(T(V)) = \Gamma_{\mathsf{A}}(T)(\mathsf{A}(V))$.* □

Lemma 8 suggests a modular approach: it reduces compilation of a composite ANN index to verifying $\mathcal{T}$-equivariance of primitives and composing the corresponding state transformers.

5.2 Deciding Transformation-Equivariance

We now decide $\mathcal{T}$-equivariance for common ANN primitives. By Lemma 8, these primitive-level decisions lift directly to composite indices built as pipelines. We cover (i) vector transforms (L_2-normalization, PCA), (ii) coarse structures (IVF, HNSW), and (iii) fine quantizers (PQ, SRQ), all built in the popular FAISS library that is widely used in production vector search systems [15].

Vector transforms. We start with vector transform primitives.

L_2-normalization. L_2-normalization normalizes vectors and converts non-Euclidean vector realizations into Euclidean ones. It can be viewed as a function $\nu : \mathbb{R}^m \setminus \{0\} \to \mathbb{R}^m$ by $\nu(x) = x/\|x\|_2$.

PCA. PCA (Principal Component Analysis) is a popular dimension reduction method. Fix $K \le m$. For a vector realization $V : \mathcal{E} \to \mathbb{R}^m$, let $\mu_V := \frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}} \phi_V(e)$ and $\Sigma_V := \frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}} (\phi_V(e)-\mu_V)(\phi_V(e)-\mu_V)^\top \in \mathbb{R}^{m\times m}$. Let $U_K \in \mathbb{R}^{m\times K}$ have orthonormal columns spanning a top-K eigenspace of Σ_V.

PCA outputs the projected realization $\pi_K^V : \mathcal{E} \to \mathbb{R}^K$ with $\pi_K^V(e) := U_K^\top(\phi_V(e) - \mu_V)$.

Lemma 9: *(1) L_2-normalization is $\mathcal{T}$-equivariant if $t = 0$. (2) PCA is $\mathcal{T}$-equivariant.* □

Coarse structures. We next examine IVF and HNSW.

IVF. An IVF state is a centroid codebook $C = \{c_1, \ldots, c_m\} \subseteq \mathbb{R}^d$ (vector state) and inverted lists $(L_j)_{j=1}^m$ (discrete state), where each $L_j \subseteq \mathcal{E}$ stores IDs assigned to centroid c_j. IVF is normally built with KMeans; we assume deterministic KMeans initialization and tie-breaking.

HNSW. An HNSW state is a multi-layer graph over $\mathcal{E}$ (adjacency lists per layer) plus the stored vectors $\phi_V(e)$ at vertices. We assume deterministic level assignments and tie-breaking.

Lemma 10: *Both IVF and HNSW are $\mathcal{T}$-equivariant.* □

Fine quantizers. We next consider vector quantization methods. We start with a brief overview two candidate primitives, Product Quantization (PQ) and Stacked Residual Quantization (SRQ).

Product Quantization (PQ). PQ fixes a partition of coordinates into $B \geq 1$ blocks and trains a KMeans codebook per block (vector index state); each ID stores a block-wise code tuple (discrete index state).

Stacked residual quantization (SRQ). SRQ [25] compresses a vector x as a sum of L codewords. At each layer ℓ, it stores a codebook $C_\ell = \{c_{\ell,1}, \ldots, c_{\ell,M_\ell}\} \subseteq \mathbb{R}^d$ (vector state), and each ID $e \in \mathcal{E}$ stores a code index $a_\ell(e) \in [M_\ell]$ (discrete state). SRQ represents the vector $\phi(e)$ of e by its decoded reconstruction $\phi_f(e) := \sum_{\ell=1}^L c_{\ell,a_\ell(e)} \in \mathbb{R}^d$.

Lemma 11: *(1) For any PQ configuration with at least two blocks ($B \geq 2$), PQ is not $\mathcal{T}$-equivariant. (2) SRQ is $\mathcal{T}$-equivariant.* □

Lemmas 9-11 provide a decision procedure for transformation-equivariance. By Lemma 8, composite indices built from $\mathcal{T}$-equivariant primitives (*e.g.*, IVF with PCA, IVF with L_2-normalization, or HNSW with residual quantization) are transformation-equivariant. Indices with PQ are not compilable and thus require rebuilding to function over the integrated database.

5.3 Robustness Analysis for Witness in the Wild

Section 5.2 decides exact transformation-equivariance: if $T \in \mathcal{T}_{m_1 \to m_2}$ is an exact witness that captures the isometry in the definition of logical Union, and an index primitive is $\mathcal{T}$-equivariant, then compilation $\Gamma_f(T)$ produces *exactly* the same index as rebuilding on $T(V)$. In practice, witnesses are learned from anchors and are imperfect. We next quantify how such witness errors, modeled as noise distributions, are propagated through the index compilation.

Witness noise model. Fix Euclidean realizations and let $V_1 : \mathcal{E} \to \mathbb{R}^{m_1}$ be a source realization materialized into target space $\mathbb{R}^{m_2}$. Let $T \in \mathcal{T}_{m_1 \to m_2}$ be the (unknown) ideal witness from Theorem 5(2), and $\widehat{T}$ be the learned witness. We model the discrepancy as additive noise: for each $e \in \mathcal{E}$, $\widehat{T}(V_1(e)) = T(V_1(e)) + \varepsilon_e$, where $\varepsilon_e \in \mathbb{R}^{m_2}$ are independent across all $e \in \mathcal{E}$, and follow either: (i) Gaussian noise $\varepsilon_e \sim \mathcal{N}(0, \sigma^2 I_{m_2})$, or (ii) coordinate-wise Laplace noise with $(\varepsilon_e)_i \overset{\text{i.i.d.}}{\sim} \text{Laplace}(0, b)$, so $\mathbb{E}[\varepsilon_e] = 0$ and $\text{Cov}(\varepsilon_e) = 2b^2 I_{m_2}$. The anchor residual $\text{Res}(\widehat{T}; O)$ (Section 3.2) estimates the typical scale of $\|\varepsilon_e\|_2$ on anchors. These two error models are widely used and also yield closed-form error bounds that explain empirical findings. We use them as analytically tractable first-order witness-error models; extending the analysis to other witness errors is left to future work.

Ranking stability. All ANN indices ultimately fail under witness error through ranking inversions. Fix a query vector $z \in \mathbb{R}^{m_2}$ and two (ideal) candidates $x, y \in \mathbb{R}^{m_2}$ with $\|z - x\|_2 < \|z - y\|_2$. Let

noisy candidates be $x' := x + \varepsilon_x$ and $y' := y + \varepsilon_y$. While x is closer to z compared to y, noise may flip the order. Let the *clean (squared-distance) gap* between x and y *w.r.t.* query z be $\Delta := \|z-y\|_2^2 - \|z-x\|_2^2 > 0$; let the *noisy gap* be $\Delta' := \|z-y'\|_2^2 - \|z-x'\|_2^2$. We write $\mathrm{swap}(x,y)$ for the event that noise flips the ordering: $\mathrm{swap}(x,y) := \{\Delta' \le 0\}$. Let Φ be the CDF of $\mathcal{N}(0,1)$.

Lemma 12: *Under first-order approximation, we have* $\Delta' = \Delta + Z$, *where* $Z := 2\langle \varepsilon_x, z-x\rangle - 2\langle \varepsilon_y, z-y\rangle$.

(1) For Gaussian noise, $\Pr[\mathrm{swap}(x,y)] = \Pr[\Delta' < 0] = \Phi(-\frac{\Delta}{\sqrt{\mathrm{Var}(Z)}})$.

(2) For Laplacian noise, $\Pr[\mathrm{swap}(x,y)] = \Pr[\Delta' < 0] \le \frac{\mathrm{Var}(Z)}{\mathrm{Var}(Z)+\Delta^2}$. □

Lemma 12 gives us the basic analytic unit. The remainder of the analysis studies how preprocessing and quantization change Δ and $\mathrm{Var}(Z)$, and how IVF/HNSW amplify or damp the effect of swaps.

L_2-normalization. Consider L_2-normalization $\nu(x) := x/\|x\|_2$ for $x \neq 0$. After normalization, we rank points $x \in \mathbb{R}^{m_2}$ by $D(x) := \|u - \nu(x)\|_2^2$. Hence, the distance perturbation for x *w.r.t.* query vector u in the presence of noise ε is $\delta D(x) := \|u - \nu(x+\varepsilon)\|_2^2 - \|u - \nu(x)\|_2^2$.

Lemma 13: *Under the Gaussian witness noise model and ignoring* $O(\|\varepsilon\|_2^2)$ *terms,* $\delta D(x)$ *is Gaussian with mean* 0 *and variance*

$$\mathrm{Var}(\delta D(x)) = 4\,\frac{\sigma^2}{\|x\|_2^2}\Big(1 - \langle u, \nu(x)\rangle^2\Big) \;\le\; 4\,\frac{\sigma^2}{\|x\|_2^2}.$$

For Laplacian noise, the same bound holds up to replacing σ^2 *by* $2b^2$. □

Lemma 13 implies L_2 normalization suppresses witness noise by damping perturbations by $1/\|x\|_2$. After normalization, the swap noise term of Lemma 12 becomes $Z_\nu := \delta D(y) - \delta D(x)$. By Lemma 13, $\mathrm{Var}(Z_\nu) = \mathrm{Var}(\delta D(y)) + \mathrm{Var}(\delta D(x)) \le 4\sigma^2(\|x\|_2^{-2} + \|y\|_2^{-2})$ (Gaussian noise), so swaps are suppressed by a factor roughly $1/\|x\|_2^2$.

PCA. Let $U \in \mathbb{R}^{m_2 \times K}$ have orthonormal columns learned from the ideal transformed dataset $T(V_1)$ by PCA. For vector x in $T(V_1)$, its PCA projected vector is $\tilde{x} := U^\top x$ and projected noise is $\tilde{\varepsilon} := U^\top \varepsilon$.

Lemma 14: *(1) Under Gaussian witness noise model,* $\tilde{\varepsilon} \sim \mathcal{N}(0, \sigma^2 I_K)$.

(2) Under Laplacian witness noise, $\mathbb{E}[\tilde{\varepsilon}] = 0$ *and* $\mathrm{Cov}(\tilde{\varepsilon}) = 2b^2 I_K$. □

PCA search compares $\tilde{z}$ to $\tilde{x}, \tilde{y}$ in $\mathbb{R}^K$. Lemma 14 shows that PCA preserves noise scale per retained coordinate, so Lemma 12 applies in K dimensions by replacing $(z-x, z-y)$ with $(\tilde{z}-\tilde{x}, \tilde{z}-\tilde{y})$. PCA helps when it preserves margins while discarding noisy directions; it hurts when the discriminative signal is in the discarded subspace.

IVF. IVF builds a "codebook" $C = \{c_1, \ldots, c_M\} \subseteq \mathbb{R}^{m_2}$ and inverted lists $(L_j)_{j=1}^M$, where each c_i is the centroid of vectors in L_i. It probes a small set of centroids and searches only within $\mathrm{Cand}_{\mathrm{IVF}}(z) := \cup_{j \in P(z)} L_j$. Thus IVF has an extra failure mode beyond swaps: if a true neighbor is not in $\mathrm{Cand}_{\mathrm{IVF}}(z)$, it is unrecoverable.

The key instability is boundary crossing: for a point x assigned to centroid c_a and a competitor centroid c_j, define $\Delta_{x\to j} := \|x-c_j\|_2^2 - \|x-c_a\|_2^2 > 0$ and the noisy gap $\Delta'_{x\to j} := \|x+\varepsilon-c_j\|_2^2 - \|x+\varepsilon-c_a\|_2^2$.

Lemma 15: *Under first-order approximation,* $\Delta'_{x\to j} = \Delta_{x\to j} + Z_{x\to j}$, *where*

$$Z_{x\to j} = 2\langle \varepsilon,\, c_a - c_j\rangle, \quad \mathrm{Var}(Z_{x\to j}) = \begin{cases} 4\sigma^2\|c_a - c_j\|_2^2, & \textit{Gaussian,} \\ 8b^2\|c_a - c_j\|_2^2, & \textit{Laplacian.} \end{cases}$$

□

Interpretation. This has the same margin-vs-noise form as Lemma 12: boundary crossings become

likely when $\Delta_{x\to j}$ is comparable to $\sqrt{\mathrm{Var}(Z_{x\to j})}$. Since list membership is discrete, such crossings can cause bucket misses even when distance swaps are rare.

HNSW and SRQ. Witness errors impact HNSW and SRQ differently. HNSW query execution is an adaptive sequence of distance comparisons along a graph. A noisy result can differ from clean result only if one or more of those comparisons is flipped. SRQ is exactly compilable under an exact witness (Lemma 11), but under witness noise each layer performs a nearest-codeword decision on a noisy residual, so small errors can trigger code changes and discontinuous jumps in the reconstruction.

Interpretation. These analyses tell us the following.

(1) Several important ANN index primitives are exactly compilable under the witness family underlying metric Union, so hard union does not necessarily require rebuilding indices from scratch.

(2) Under imperfect witnesses, different primitives amplify witness noise in very different ways: L_2-normalization is stabilizing, IVF is sensitive to cell boundaries, HNSW degrades more gracefully, and PQ is not compilable at all. With source-side compilation in place, the next question is whether the compiled source index and the target index can themselves be *merged* into a single union index for hard-union execution. Section 6 studies exactly this question.

6 Mergeability of Vector Indices

Section 5 showed when a source index built on a realization V_1 can be compiled into the target space under a witness $T_{1\to 2}$, avoiding a rebuild on $T_{1\to 2}(V_1)$. After compilation, we hold two indices in the same space: one over the transformed source vectors and one over the target vectors. A straightforward physical plan is to query both indices and merge the two result lists. This plan is always feasible but doubles query-time work and complicates memory management; moreover, it is essentially a physical analogue of soft union.

Instead, we ask when it is possible to *merge* two prebuilt indices into a *single* index over union, without rebuilding from scratch. This enables a single-index for hard union. Mergeability is also of independent interest: vector databases routinely build indices over disjoint shards for parallel construction, incremental ingestion, and rebalancing, and efficient mergeability determines whether such indices can be consolidated without rescanning all vectors.

6.1 Mergeability: Cost Model and Error Model

As in Section 5.1, a Euclidean vector realization is a map $\phi_V : \mathcal{E} \to \mathbb{R}^d$. Let $V_\alpha : \mathcal{E}_\alpha \to \mathbb{R}^d$ and $V_\beta : \mathcal{E}_\beta \to \mathbb{R}^d$ be two realizations with $\mathcal{E}_\alpha \cap \mathcal{E}_\beta = \emptyset$. Their union $V_\alpha \cup V_\beta$ is defined by the pointwise union of maps. Recall that, for an ANN index primitive f, its index state $\mathrm{State}_f(d)$ is the set of valid output states produced by running f on d-dimensional inputs with f's hyperparameters fixed.

Mergeability. Index f is *mergeable* if there exists a deterministic function $\mathrm{Merge}_f : \mathrm{State}_f(d) \times \mathrm{State}_f(d) \to \mathrm{State}_f(d)$ such that:

(1) Merge_f is *vector-sublinear* time: it accesses $o(n)$ vectors over $f(V_\alpha)$ and $f(V_\beta)$, where $n = |\mathcal{E}_\alpha| + |\mathcal{E}_\beta|$, *i.e.,* total number of vectors.

(2) $\mathrm{Merge}_f(\cdot,\cdot)$ is *error-bounded*, *i.e.,* for all V_α and V_β, $\mathrm{Merge}_f(f(V_\alpha), f(V_\beta))$ approximates $f(V_\alpha \cup V_\beta)$ with bounded error.

To complete the notion, we still need to define when Merge_f is error-bounded. In particular, we need to quantify the approximation error of the merged index independently of queries, as index merging and construction are preprocesses before queries are issued.

Representational error. For many index families, an index state induces a representative vector

for each ID, which allows us to quantify the error of Merge_f. Formally, assume $f(V)$ defines a representative map $\phi_f : \mathcal{E} \to \mathbb{R}^d$, where $\phi_f(e)$ is the vector representation of $\phi_V(e)$. Define per-ID error $\texttt{err}(e) := \|\phi_V(e) - \phi_f(e)\|_2$ and representational error $\texttt{err}_f(V) := \sum_{e\in\mathcal{E}} \texttt{err}(e)^2$.

This aligns with standard index build objectives of many common ANN indices. For instance, for IVF, $\phi_f(e)$ is the centroid of e's inverted list and $\texttt{err}_f(V)$ is exactly the KMeans within-cluster SSE; for quantizers, $\phi_f(e)$ is the decoded reconstruction and $\texttt{err}_f(V)$ is quantization SSE [25]; for PCA, $\phi_f(e)$ is the rank-K reconstruction and $\texttt{err}_f(V)$ is PCA reconstruction loss [2].

Error-bounded merging & optimality. Merge_f is *error-bounded* if there exists a certificate Δ_{Merge_f} computable from the *vector states* of $f(V_\alpha)$ and $f(V_\beta)$ only such that, for all disjoint V_α, V_β,

$$\texttt{err}_{\text{Merge}_f}(V_\alpha \cup V_\beta) \le \texttt{err}_f(V_\alpha) + \texttt{err}_f(V_\beta) + \Delta_{\text{Merge}_f}.$$

We say Merge_f is *c-optimal* ($c \ge 1$) if among all vector-sublinear, error-bounded merge operators $\Delta_{\text{Merge}'_f}$ for f, we have $\Delta_{\text{Merge}_f} \le c \cdot \Delta_{\text{Merge}'_f}$.

6.2 Mergeability of Vector Indices

We study the mergeability of main ANN index primitives.

Representational indices. We determine the mergeability of *representational* ANN indices, where the objectives of the corresponding index construction algorithms align with their representation errors. Note that L_2-normalization is per-vector by definition so it is exactly mergeable. Below we focus on other representational cases in Section 5, including PCA, IVF, SRQ quantization.

Theorem 16: *(1) Both IVF and SRQ are mergeable, and moreover, c-optimal for some constant c.*

(2) PCA is not mergeable. □

Below we give a constructive proof for the mergeable cases (IVF and SRQ) of Theorem 16.

Merging IVF. An IVF state consists of a centroid codebook $C = \{c_1, \ldots, c_m\} \subseteq \mathbb{R}^d$ (vector state) and inverted lists $(L_j)_{j=1}^m$ (discrete state). Given IVF states over V_α and V_β and a target number of merged centroids K, we develop $\text{Merge}_{\text{IVF}}$ working as follows:

(1) It carries out a weighted K-means clustering [17] over the union of centroids of IVF states over V_α and V_β. Specifically, index all input inverted lists from both states as $r \in [m]$, where m is the total number of centroids. For list r define its weight $w_r := |L_r|$ and centroid $\mu_r \in \mathbb{R}^d$. For any assignment $\pi : [m] \to [K]$, define the induced center for group t as $\mu_t(\pi) := \frac{\sum_{r:\pi(r)=t} w_r \mu_r}{\sum_{r:\pi(r)=t} w_r}$. Then the induced weighted k-means objective is $\arg\min_{\pi:[m]\to[K]} \sum_{t=1}^K \sum_{r:\pi(r)=t} w_r \, \|\mu_r - \mu_t(\pi)\|_2^2$.

(2) For π returned in step (1), for all i such that $\pi(i) = t$, merge L_i to L_t, update c_t in the merged index state with $\mu_t(\pi)$.

Proposition 17: $\text{Merge}_{\text{IVF}}$ *is (a) vector-sublinear, (b) error-bounded and (c) 10-optimal.* □

Merging SRQ. Recall SRQ index state from Section 5.2. Given SRQ states over realizations V_α and V_β, and target codebook sizes $(K_\ell)_{\ell=1}^L$, we present procedure $\text{Merge}_{\text{SRQ}}$ that merges SRQ layer-by-layer:

(1) For each layer $\ell \in [L]$, it carries out a weighted K_ℓ-means clustering [17] over the union of layer-ℓ codewords from the two SRQ states, similar to the reduction to weighted KMeans for IVF: each layer-ℓ codeword r corresponds to a centroid of IVF and its weight is the usage count, *i.e.,* the number of IDs whose layer-ℓ code equals r.

(2) For π_ℓ returned in (1), we update the layer-ℓ codebook similarly to step (2) of $\text{Merge}_{\text{IVF}}$ above.

Proposition 18: $\text{Merge}_{\text{SRQ}}$ *is (a) vector-sublinear, (b) error-bounded, and (c) 10-optimal.* □

HNSW. Unlike IVF/SRQ, HNSW stores the original vectors verbatim (so representational error is always 0) and its approximation comes from graph navigation. Thus, the induced error of the merging HNSW must be discussed in terms of query-dependent top-k recall, not in terms of query-agnostic representational distortion.

For any query vector $q \in \mathbb{R}^d$, let K_q be the ground-truth k-NN set in $V_\alpha \cup V_\beta$ (ties broken deterministically). Assume q has a set A_q of k IDs as answers; we measure its k-NN recall as $\text{recall}(q) := \frac{|A_q \cap K_q|}{k}$. For a realization $V \subset \mathbb{R}^d$ and any point $p \in V$, define $N_V(p) := \{q \in \mathbb{R}^d | \forall y \in V \setminus \{p\}, \|q-p\|_2 < \|q-y\|_2\}$. Fix HNSW hyperparameters with a deterministic query procedure. Let $\text{Merge}_{\text{HNSW}}$ be any deterministic merging procedure for HNSW. For any $k \geq 2$, we have:

Theorem 19: *If* $\text{Merge}_{\text{HNSW}}$ *is vector-sublinear, then there exist realizations* $V_\alpha, V_\beta \subset \mathbb{R}^d$ *and a query set* $Q \subseteq \cup_{p \in V_\alpha} N_{V_\alpha}(p)$ *such that, for every* $q \in Q$, $\text{recall}(q) = \frac{1}{k}$ *when using index of* $\text{Merge}_{\text{HNSW}}$. □

Theorem 19 shows that to avoid catastrophic recall loss, a merged HNSW must create cross-dataset links that allow queries of V_1 to reach the relevant points in V_2. Establishing such links requires substantial cross-dataset distance evaluation beyond vector-sublinear in the worst case.

Incremental merging. Although sublinear merging is infeasible in the worst case for HNSW, a practical heuristic for HNSW is to merge by insertion: keep index I_1 of V_1 and insert all HNSW vertices from V_2 into I_1 using the standard HNSW insertion procedure. This creates cross edges by construction and often yields good empirical recall. It also saves computation by fully reusing index of V_1.

7 Experimental Study

Using benchmarks and deep embedding models, we evaluated the feasibility of embedding-independence via metric algebra (Section 7.2), and the effectiveness of transformation-equivariant index compilation (Section 7.3) and merging (Section 7.4) for querying Union.

7.1 Experimental Setup

We start with the experimental settings.

Benchmarks. We used five public benchmarks spanning different sizes and domains: SciFact [35] from 5,183 scientific articles, ArguAna [32] from 8,674 debate arguments, FiQA [23] from 57,638 financial articles, Quora [32] for duplicate question detection from 0.5 million question corpus, Fever [33] for fact verification from 5.4 million Wiki facts. Each benchmark provides built-in query-answer pairs for retrieval quality evaluation.

Embedding models. We used 4 major embedding models to implement vector realizations of benchmarks: `Mistral` [14], `OpenAI-Ada` (Ada-002 [10]), `GTE-Qwen2` [21], and `NV-Embed-V2` [19].

Metric databases. We created a metric database from each benchmark and evaluated it under multiple ordered embedding-model pairings. Specifically, given a benchmark dataset D, we randomly partitioned it into three disjoint groups, namely, D_1, D_O and D_2. We then let metric relation $\mathcal{R}_1$ be $D_1 \cup D_O$ and $\mathcal{R}_2$ be $D_2 \cup D_O$, where D_O serves as the set of anchor data points. We then generated the vector realizations $V_{\texttt{emb}_1}$ and $V_{\texttt{emb}_2}$ of $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively, by using two different embedding models $\texttt{emb}_1$ and $\texttt{emb}_2$. This gives us 12 different implementations for each benchmark.

Queries. For each benchmark and each built-in (query q, answer ans) pair, we generated a kNN selection query $\kappa_{q,k}$ and evaluated it over vector realizations of Union $\mathcal{R}_1 \sqcup \mathcal{R}_2$, using both hard and soft union implemented via (a) exact vector search, (b) search with index built from scratch, and (c) search with index obtained via transformation-equivariant index compilation and merging.

Table 2. Recall@100 (%) on `Mistral` ⊔ `OpenAI-Ada`

Method		ArguAna			SciFact		
		recall@100	reL@100	reR@100	**recall**@100	reL@100	reR@100
DirectUnion		49.75	0.00	99.57	50.00	0.00	100.00
MetricDB	[NN]	66.34	49.18	83.50	52.08	29.43	74.73
	[ISO]	47.97	2.14	93.86	39.33	4.00	74.67
	[MISO]	**84.45**	85.21	83.69	**77.41**	80.92	74.50

Table 3. Varying model pairings (each cell reports recall@100(%) of DirectUnion/MetricDB[MISO])

`M`: `Mistral`, `O`: `OpenAI-Ada`, `G`: `GTE-Qwen2`, `N`: `NV-Embed-V2`

	ArguAna				SciFact			
	`M`	`G`	`N`	`O`	`M`	`G`	`N`	`O`
`M`⊔	-	50.0/88.5	50.0/86.3	49.8/84.5	-	50.0/89.0	50.0/82.3	50.0/77.4
`G`⊔	49.8/85.7	-	50.0/89.9	49.8/86.2	50.0/90.1	-	50.0/80.7	50.0/75.2
`N`⊔	49.8/85.1	50.0/90.8	-	49.8/86.0	50.0/89.7	50.0/88.1	-	50.0/76.7
`O`⊔	49.8/83.5	50.0/88.9	50.0/86.7	-	50.0/87.7	50.0/84.0	50.0/79.2	-

Rows: vector realization of $\mathcal{R}_1$; Columns: realization of $\mathcal{R}_2$ (for querying $\mathcal{R}_1 \sqcup \mathcal{R}_2$).

Table 4. Varying benchmarks (each cell reports the recall@100 of DirectUnion/MetricDB[MISO])

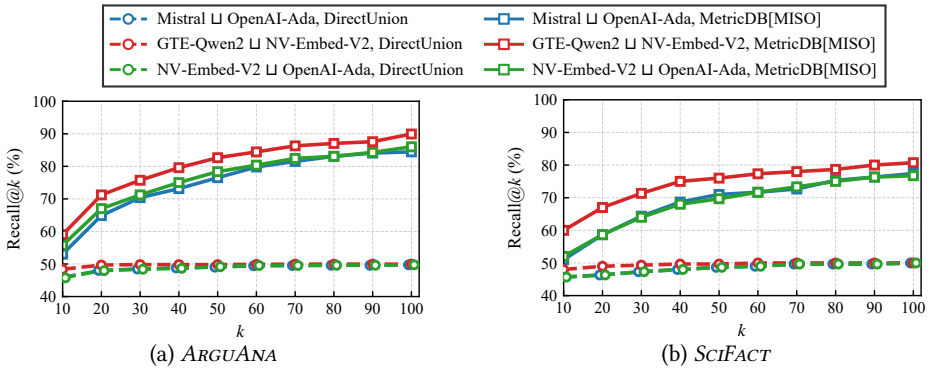
	ArguAna	SciFact	Quora	Fever	FiQA
`Mistral` ⊔ `OpenAI`	49.75/84.45	50.00/77.41	49.95/80.48	49.34/74.48	49.05/70.59
`QWen2` ⊔ `NV-Embed`	49.96/89.94	50.00/80.73	50.00/82.61	49.67/76.21	49.25/72.97
`NV-Embed` ⊔ `OpenAI`	49.75/86.01	50.00/76.74	49.95/80.67	49.59/75.39	49.17/71.02

We balance benchmark queries by the locations of their ground-truth answers. We first randomly partition non-answer corpus items into D_1, D_O, and D_2, and then randomly assign answer-bearing corpus items evenly to D_1 and D_2. Thus, queries are not treated as randomly partitioned database records; instead, their relevant answers are balanced across the two sides of the union.

We evaluated MetricDB for answering all queries over different implementations, by searching $V_{\texttt{emb}_1} \sqcup V_{\texttt{emb}_2}$ using query vectors $\texttt{emb}_2(q)$ produced by $\texttt{emb}_2$ for all query items q. We measured the quality of answers by standard recall@k. For each query q with its relevant items $\text{ans}(q)$ in benchmark ground truth, let $\pi_k(q)$ be the returned top-k list by a method over a vector realization. Then $\text{recall@}k := \frac{1}{|Q|}\sum_{q\in Q}\mathbb{I}(\pi_k(q) \cap \text{ans}(q) \neq \emptyset)$, where $\mathbb{I}(x) = 1$ if x is *True*. We also reported recall@k for answers in $\mathcal{R}_1$ and in $\mathcal{R}_2$ separately, denoted by reL@k and reR@k, respectively. Here, reL@k and reR@k condition on the location of the ground-truth answer: reL@k is the recall@k computed over queries whose answers reside in $\mathcal{R}_1$, and reR@k over queries whose answers reside in $\mathcal{R}_2$. In all tests over $\mathcal{R}_1 \sqcup \mathcal{R}_2$, query vectors are embedded by the same embedding model that realizes $\mathcal{R}_2$.

Implementation. Following Section 3.3, we implemented MetricDB.

Witness maps. We implemented all three families of witness maps. For isometry witness (ISO), we used Procrustes Analysis [29, 31] to learn the map $T_{1\to 2}$ from vector realizations of $\mathcal{R}_1$ to those of $\mathcal{R}_2$. To learn multi-isometry witness (MISO), following [38] we group the anchors into clusters via KMeans, where the number of clusters consistent with the number of inverted lists of IVF index over the anchors, and then learn an isometry per each cluster; a query vector is then processed by the nearest isometry, *i.e.,* isometry with the closest cluster centriod. We also implemented neural witness via Gromov-Wasserstein alignment learning [1]. We write MetricDB[ISO] for MetricDB equipped with ISO isometry; similarly for other variants *e.g.,* MetricDB[MISO] and MetricDB[NN].



(a) ArguAna (b) SciFact

Fig. 3. Recall@k under varying retrieval depths

Table 5. HardUnion vs. SoftUnion: each cell reports recall@100(**%**)/ per-query latency(**ms**)

MetricDB	Mistral ⊔ OpenAI		GTE-Qwen2 ⊔ NV-Embed		NV-Embed ⊔ OpenAI	
	ArguAna	SciFact	ArguAna	SciFact	ArguAna	SciFact
HardUnion	**84.45**/0.15	77.41/0.07	**89.94**/0.35	**80.73**/0.57	**86.09**/0.35	**76.74**/0.96
SoftUnion	75.68/6.00	**78.70**/11.19	87.09/6.70	63.46/25.81	78.82/6.50	57.81/29.22

ANN indices. In MetricDB, when an index is both transformation-equivariant and mergeable according to Sections 5-6, we use the corresponding index compilation and merging algorithms to generate vector index of the vector union realization of $\mathcal{R}_1 \sqcup \mathcal{R}_2$ for query evaluation; specifically, HNSW is regarded as mergeable via the incremental merging procedure of Section 6.2. We also compared with baselines where indices were reconstructed from scratch. For those baselines, we use default vector index configuration in FAISS [15], *e.g.*, $nlist = 4\sqrt{N}$ (database size N) and $nprobe = 8$ for IVF.

All tests were run on an Intel i7-12700H CPU (32GB RAM) and NVIDIA GeForce RTX 3060 GPU.

7.2 Embedding-Independence and Unions

We first demonstrated that embedding independence is achievable via MetricDB. We then compared soft and hard union for querying Union across vector realizations of distinct embedding models.

Embedding-independence. We evaluated retrieval quality of all benchmark queries over vector realizations across embedding models and benchmarks. We compared MetricDB with baseline DirectUnion that directly takes the union of vector realizations and then does vector search. To avoid noise from ANN indices, we used exact vector search and hard union plan for MetricDB.

(1) Embedding-independent retrieval. MetricDB with miso witness answers queries over metric unions with significantly higher recall than DirectUnion. As shown in Table 2, MetricDB[miso] achieves recall@100 over 84% on ArguAna while DirectUnion is 50%; case for SciFact is similar.

The real gap is even more revealing. Recall that for $\mathcal{R}_1 \sqcup \mathcal{R}_2$, query vectors are from the same embedding space of $\mathcal{R}_2$'s realization; thus reL@k reflects the performance of cross-embedding retrieval while reR@k indicates how well same-embedding space retrieval is retained. MetricDB has well-balanced and consistently high recall, reL and reR, which demonstrates its effectiveness for embedding-independent querying. In stark contrast, as DirectUnion almost never retrieves query answers from $\mathcal{R}_1$, and as answer-bearing queries are balanced across $\mathcal{R}_1$ and $\mathcal{R}_2$, DirectUnion has near-zero reL@100, near-perfect reR@100, and about 50% overall recall. This justifies that practical witnesses can be effective in supporting embedding-independence and cross-embedding queries.

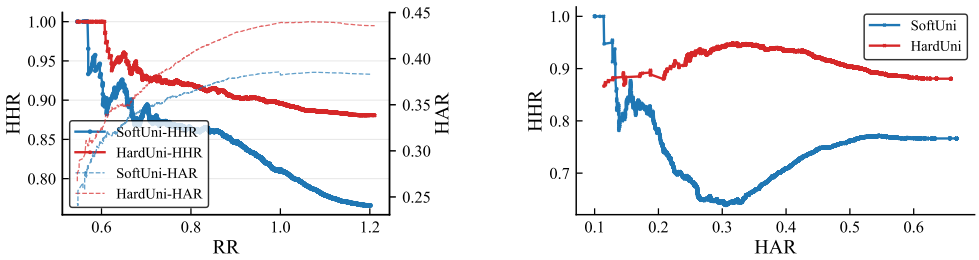


Fig. 4. Relationship between radius ratio, head hit rate, and head average residual

Table 6. Index quality (recall@100, %) and build time (ms)

M: `Mistral`, O: `OpenAI-Ada`, G: `GTE-Qwen2`, N: `NV-Embed-V2`

Index	Method	SciFact				ArguAna			
		M ⊔ O	G ⊔ N	N ⊔ O	Build T.	M ⊔ O	G ⊔ N	N ⊔ O	Build T.
IVF	Rebuild	72.67	67.91	72.67	4335.3	81.31	88.87	85.02	7545.5
	MetricDB	70.33	67.90	72.51	**212.7**	80.74	86.31	83.74	**355.9**
HNSW	Rebuild	73.33	71.67	77.67	9644.7	92.58	99.04	94.29	17593.5
	MetricDB	72.81	69.15	74.69	**300.3**	91.82	98.41	92.35	**532.7**

We also observe that MISO has the best performance among all witness families. That is, MISO has the expressiveness of higher-order NN maps, while its isometry-based structure allows it to enjoy all the nice theoretical properties (*e.g.,* index compilation, robustness) of isometries. This also justifies the practical relevance of the isometry-based theoretical analysis in across Sections 4-6.

(2) Varying embedding model-pairing. To rule out model-pair bias, Table 3 reports recall@100 for *all* pairings among the four embedding models. The same holds across all pairings: DirectUnion stays close to the balanced-side baseline, while MetricDB[MISO] consistently achieves high recall@100.

(3) Varying scale and domains. We further examined recall@100 of MetricDB across benchmarks that span five different application domains with scales up to 5.4M documents/vectors (Fever benchmark). As shown in Table 4, MetricDB[MISO] consistently improves over DirectUnion by a significant gap, *e.g.,* 76.74–80.73% on SciFact, 84.45–89.94% on ArguAna, 80.48–82.61% on Quora, 74.48–76.21% on Fever, and 70.59–72.97% on FiQA, while DirectUnion remains around 49–50%. This indicates MetricDB remains effective across scales and domains of benchmarks.

(4) Varying retrieval depth k. To examine whether the effectiveness of MetricDB depends on a large retrieval budget, we vary the retrieval depth k from 10 to 100 and report recall@k in Fig. 3. Across all model pairings, MetricDB[MISO] consistently outperforms DirectUnion, with substantial gains already at the smallest tested budget $k = 10$. Further, MetricDB improves steadily as k increases, while DirectUnion cannot capitalize on this. This shows that metric algebra and its implementation in MetricDB are effective not only at recall@100, but also under smaller retrieval depths.

Soft vs. Hard unions. To validate our analysis in Section 4, we further compared the quality, efficiency and stability of hard union and soft union using MetricDB[MISO].

(1) Quality. Table 5 shows that hard union is consistently better in answer quality (recall@100) than soft union. Soft union is markedly less stable across vector realizations of the same database.

(2) Efficiency. As also shown in Table 5, soft union consistently incurs higher per-query latency due to extra query transportation and answer merging costs. These overheads are data dependent and

Table 7. Recall@100 (%) under varying witness noise

	Gaussian Noise				Laplacian Noise			
	Cluster		*Uniform*		*Cluster*		*Uniform*	
σ/b =	0.05	0.1	0.05	0.1	0.05	0.1	0.05	0.1
Rebuild (Flat)	92.07	82.54	90.73	83.35	88.57	74.80	88.27	73.19
Flat ∘ L2Norm	98.07	96.56	97.81	95.54	97.49	94.47	97.25	94.04
IVF	89.80	81.69	31.72	31.25	80.60	69.33	33.11	29.20
IVF ∘ L2Norm	98.10	96.13	57.20	55.68	97.27	94.42	56.88	55.67
HNSW	89.75	82.14	90.50	80.51	86.77	73.14	86.80	71.19
HNSW ∘ L2Norm	92.53	85.95	91.79	82.37	89.24	77.00	89.30	73.33

Table 8. Recall@100 (%) and build time (ms) of merged indices

Method		SciFact				ArguAna			
		`OpenAI`	`NV.`	`Mist.`	Build T.	`OpenAI`	`NV.`	`Mist.`	Build T.
IVF	Rebuild	82.74	87.49	88.00	4335.3	95.29	98.29	96.35	7545.5
	MetricDB	87.04	91.05	91.56	**177.8**	96.92	98.82	97.94	**203.3**
IVF∘IVF	Rebuild	70.80	74.54	73.89	3876.2	73.24	77.46	72.36	7368.7
	MetricDB	66.62	70.67	68.77	**788.2**	69.01	73.87	68.93	**1498.4**
HNSW	Rebuild	90.57	92.61	94.27	9644.7	96.24	99.14	97.90	17593.5
	MetricDB	90.55	93.63	94.83	**5782.7**	96.64	98.92	98.01	**11321.0**
HNSW∘IVF	Rebuild	77.94	83.68	85.73	4541.6	86.67	87.88	84.13	7973.8
	MetricDB	74.11	80.32	82.24	**2918.2**	83.19	85.64	81.22	**5418.8**

increase as dataset size grows, and can dominate end-to-end latency at query time. All these make soft union less attractive for high-throughput settings, despite being more lightweight.

(3) Stability. We further validated the stability analysis of soft and hard unions in Section 4, by examining the impact of query distribution *w.r.t.* anchors. Recall that MISO is a collection of isometries, each built from a cluster of anchors. Accordingly, we use the radius ratio (RR) to measure how far a query point q_i lies from the selected isometry cluster center, normalized by the cluster radius: $\mathrm{RR}(q_i) = \frac{\|q_i - \mu_i\|_2}{r_i}$, where $r_i = \max_{x \in C_i} \|x - \mu_i\|_2$ and C_i (centroid μ_i) is the cluster/isometry selected to answer q_i. We defined the *head hit rate* (HHR) over t, as the probability that a benchmark answer, located within t RR distance of the isometry centroid μ_i, is returned by the union method. We also monitor the alignment quality of the witness with respect to t, referred to as the *head average residual* (HAR). This is computed by prefix-averaging the anchor residual Res (Section 3.2) over anchor points within t RR distance to the centroid.

Figure 4 shows the average HHR (left y-axis) and HAR (right y-axis) across all cases with varying RR distance. We see two consistent patterns: (i) the average HHR of soft union drops faster as RR increases, indicating higher sensitivity to OOD queries, and (ii) the drop correlates with increasing HAR, suggesting that proxy-query transport of soft union plans amplifies witness errors when the selected isometry map is inaccurate. Together, these results empirically validate the Section 4 prediction that hard union is more robust under witness errors and query shift.

7.3 Index Compilation and Robustness

We examine the effectiveness and robustness of index compilation.

Effectiveness & scalability. Table 6 compares the quality (recall@100) and the index build time of Rebuild, which constructs clean ANN indices from scratch, with MetricDB, which uses index compilation to transform prebuilt indices. The results show that the compiled indices maintain

Table 9. Recall@100 (%) with index compilation + merging

M: Mistral, O: OpenAI-Ada, G: GTE-Qwen2, N: NV-Embed-V2

Method	SciFact			ArguAna		
	M⊔O	G⊔N	N⊔O	M⊔O	G⊔N	N⊔O
MetricDB [miso, IVF, Rebuild]	76.67	78.33	75.33	83.51	87.08	82.80
MetricDB [miso, IVF, HardUnion]	76.41	78.07	75.08	83.02	88.02	83.74
MetricDB [miso, HNSW∘IVF, Rebuild]	75.33	78.67	75.67	83.44	86.94	82.66
MetricDB [miso, HNSW∘IVF, HardUnion]	71.43	76.08	71.76	79.74	84.16	80.67

quality (recall) almost the same as that of indices built from scratch, while offering significant savings in index building time. For instance, over SciFact and ArguAna, for both IVF and HNSW, queries evaluated on the compiled indices have recall@100 values that are at most ~2% below those of indices reconstructed from scratch, while saving on average 96.1% of index-building time.

We further evaluated the quality and efficiency of index compilation over the largest benchmark Fever (5.4M vectors). As shown in Table 10 (the *compilation* row), while both Rebuild and MetricDB took more time, MetricDB still saved *e.g.,* ~95% of IVF construction time of Rebuild while maintaining the same retrieval recall of a clean IVF over Fever; similarly for HNSW∘IVF, a composite index where we build an inner HNSW for each cluster of the outer IVF.

Robustness against witness noise. We next evaluated how inexact witnesses affect index behavior, following the noise model in Section 5.3. We compared $\texttt{INDEX}(F'(D))$ against $\texttt{INDEX}(F(D))$, where F is an isometry and $F'(x) = F(x) + \varepsilon$ adds isotropic error. We evaluated recall@100 on two synthetic datasets in $\mathbb{R}^{128}$, each with $N = 10{,}000$ points:

(i) *Uniform*, where points are sampled i.i.d. from $U([-1, 1]^{128})$; and

(ii) *Clustered*, where we first sample a set of cluster centroids uniformly from $[-1, 1]^{128}$ and then generate points by adding i.i.d. Gaussian noise around the assigned centroid.

We used 1,500 random queries. For the Gaussian witness noise $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ Laplacian noise model $\varepsilon \sim \mathrm{Lap}(0, b)$, we varied $\sigma/b \in \{0.05, 0.1\}$. The results are shown in Table 7.

(1) Preprocessing such as L_2-normalization remains highly stable under all different noise configurations. It further positively improves robustness of other indices when used as a composite index.

(2) IVF works well against localized, clustered errors, *i.e.,* over *Cluster* dataset, but can degrade sharply on uniform data due to boundary-crossing errors in discrete cell assignment.

(3) In contrast, HNSW degrades more gracefully over both *Uniform* and *Cluster*. These findings are consistent with the theoretical analyses in Section 5.3, and further justifies the effectiveness of index compilation under different witness errors.

7.4 Merging Vector Indices for Querying Union

Finally, we evaluated the performance of index mergeability of Section 6, and its effectiveness for supporting hard-union execution in MetricDB.

Effectiveness of merging. Following Section 6, we first used two disjoint vector realizations V_α and V_β of the same embedding model and evaluated queries on their union $V_\cup = V_\alpha \cup V_\beta$. For each index family, we compared Rebuild, which builds index from scratch on $V_\cup$, with MetricDB, which merges prebuilt indices on V_α and V_β. We report recall@100 and build time in Table 8 and Table 10 (*Merging* row); we use IVF∘IVF (resp. HNSW∘IVF) to denote a composite index where we build an inner IVF (resp. HNSW) for each cluster of the outer IVF.

Table 10. Recall, index building time, and query latency on Fever (5.4M)

		IVF		HNSW∘IVF	
Compilation		Rebuild	MetricDB	Rebuild	MetricDB
`Mistral` ⊔ `OpenAI-Ada`	recall@100	67.54	67.01	69.84	68.25
	Build Time (s)	1010.6	15.5	2298.4	235.8
Merging		Rebuild	MetricDB	Rebuild	MetricDB
`Mistral`	recall@100	84.47	86.19	79.91	75.63
`OpenAI-Ada`	recall@100	81.64	81.9	85.69	82.35
Build Time (s)		1009.3	28.7	2363.4	1211.3
Compilation + Merging		Rebuild	MetricDB	Rebuild	MetricDB
`Mistral` ⊔ `OpenAI-Ada`	recall@100	72.81	71.59	70.33	69.68
	Build Time (s)	1017.3	35.9	2388.4	1584.8
	Query Latency (ms)	1.53	1.76	1.06	1.2

(1) The results confirm that index merging consistently preserves recall@100 of queries over union of vectors when compared to a completely rebuilt, clean index. Over `OpenAI-Ada` on SciFact and ArguAna, the merged HNSW by MetricDB achieves 90.55% of recall@100, comparable to the 90.57% achieved by a clean HNSW built from scratch. This consistency extends to other HNSW scenarios. Surprisingly, a merged IVF can provide even higher recall than a reconstructed IVF, *e.g.,* 91.56% vs. 88.00% over `Mistral` on SciFact. This is because the merged IVF can moderately enlarge some inverted lists, which enhances candidate coverage while maintaining search scope.

(2) The quality of the merged index extends to composite indices. For example, both merged IVF∘IVF and HNSW∘IVF consistently achieve recall@100 within ~4% and ~3%, respectively, of their reconstructed composite counterparts, across all embedding models.

(3) While maintaining index quality, index merging offers significant savings on index building time, *e.g.,* 24x faster for IVF compared to Rebuild over SciFact despite SciFact's modest size; the gap will grow with dataset scale due to its vector-sublinear time complexity. The savings on HNSW are smaller compared to IVF, which is consistent with the analysis in Section 6.

(4) The quality and the time saving benefit of the merged index remain the same over larger benchmark, *e.g.,* on Fever as shown in Table 10. For instance, on Fever, merging IVF even slightly improves the benchmark query recall for both `Mistral` and `OpenAI-Ada` compared to reconstructing IVF from scratch, while saving 97.2% of index build time; similarly for HNSW∘IVF composite indices.

Putting all together. By integrating index compilation with merging, we evaluated the performance of MetricDB in querying the union of vector realizations from different embedding models using compiled and merged indices. As reported in Table 9 and Table 10 (the *Compilation + Merging* row), MetricDB's index pipeline effectively implements a hard-union plan by compiling and merging prebuilt indices. This significantly reduces index building time without compromising index quality or increasing query latency. For instance, when querying `Mistral`⊔`OpenAI-Ada`, the compiled and merged IVF index over the union'ed vector realization achieves the same recall@100 on SciFact (~76%), ArguAna (~83%) and Fever (~72%); results for composite index (HNSW∘IVF) are similar.

8 Related Work

We position our work within the broader research landscape.

Vector databases and ANN search. A large body of work studies ANN search and vector database systems. ANN search centers on vector indices, *e.g.,* IVF and HNSW [24], and product/residual quantization, with mature libraries such as FAISS [7, 15]. They have been implemented either

as standalone vector DBMSs, *e.g.,* Milvus, Manu, and SingleStore-V [4, 11, 28, 36], or as RDBMS extensions, *e.g.,* PostgreSQL pgvector [30] and Oracle AI Vector Search [27].

These systems focus on efficient retrieval within a fixed embedding space under a fixed operational distance. Their query semantics are thus tied to a single physical embedding space. Our work is complementary: MetricDB does not propose a new ANN index or another vector DBMS. Instead, it adds a logical layer above vector realizations so that independently embedded databases can be queried together. On the system side, our transformation-equivariant index compilation and index mergeability results show when prebuilt ANN indices can be reused after a cross-embedding witness transformation, avoiding full re-embedding and index reconstruction.

Embedding alignment and cross-space retrieval. Another closely related line of work studies mappings between independently learned representation spaces. Cross-lingual and cross-domain alignment methods use orthogonal maps, CCA/manifold alignment, or optimal-transport and Gromov-Wasserstein objectives to relate spaces [1, 16, 18, 26], while more recent work studies broader geometric consistency across embedding models [13, 38]. In particular, local convergence [38] has been validated as an alignment for text embeddings optimized cross-embedding retrieval. We use such methods as witness providers, but focus on different questions: alignment work asks whether a useful map can be learned; we ask what cross-view queries mean once such maps are only approximate, when their answers remain robust under witness error, and when existing ANN index states can be reused after alignment. In other words, our novelty is not a stronger embedding alignment method, but the database layer around alignment: deterministic logical union, witness-aware planning and robustness analysis, and transformation-equivariant compilation and mergeability results that make embedding-independence practical over existing vector databases.

Logical independence and data integration. On the database side, our work is closest in spirit to the classical separation of logical query semantics from physical storage/access paths [3, 5], and to unified querying over heterogeneous sources in data-integration and polystore settings [8, 9, 20]. The resemblance, however, is architectural rather than semantic. Metric algebra is not intended as a new user-facing replacement for relational algebra; it is an internal logical intermediate representation for similarity search over vector collections. More importantly, relational operators are defined over tuples and set membership, and do not by themselves assign meaning to kNN or range search across independently embedded collections. Our work is precisely to define that missing semantics: anchors and rigidity make a logical union well-defined at the metric level, after which similarity operators have embedding-independent meaning independent of any particular vector representation.

9 Conclusion

We presented metric algebra, a logical foundation for embedding-independence in vector databases. Metric algebra lifts vector datasets to metric relations and defines similarity operators with implementation-independent semantics, including a deterministic logical Union operator for cross-embedding integration and queries. We implemented the ideas in MetricDB, which treats existing vector databases as vector realizations and executes Union using anchor-derived transformation witnesses. Building on this, we studied query planning for searching over Union and showed how to reduce integration cost via transformation-equivariant index compilation and index mergeability.

MetricDB assumes overlap anchors are available. A key next step is to acquire anchors automatically, *e.g.,* via entity linking or weak supervision. Another is to extend Union to the multiway setting and analyze how errors propagate under more diverse witness error models.

Acknowledgments

This work is supported by RAEng RF\201920\19\319 and the Huawei-Edinburgh Joint Lab.

References

[1] David Alvarez-Melis and Tommi S. Jaakkola. 2018. Gromov-Wasserstein Alignment of Word Embedding Spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, 1881–1890. doi:10.18653/V1/D18-1214

[2] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer New York. https://dl.acm.org/doi/10.5555/1162264

[3] Michael L. Brodie and Joachim W. Schmidt. 1982. Final Report of the ANSI/X3/SPARC DBS-SG Relational Database Task Group. *SIGMOD Record* 12, 4 (1982), i–62. doi:10.1145/984555.1108830

[4] Cheng Chen, Chenzhe Jin, Yunan Zhang, Sasha Podolsky, Chun Wu, Szu-Po Wang, Eric Hanson, Zhou Sun, Robert Walzer, and Jianguo Wang. 2024. SingleStore-V: An Integrated Vector Database System in SingleStore. *Proceedings of the VLDB Endowment* 17, 12 (Aug. 2024), 3772–3785. doi:10.14778/3685800.3685805

[5] Edgar F. Codd. 1970. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM* 13, 6 (1970), 377–387. doi:10.1145/362384.362685

[6] Zhen Cui, Hong Chang, Shiguang Shan, and Xilin Chen. 2014. Generalized Unsupervised Manifold Alignment. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger (Eds.), Vol. 27. Curran Associates, Inc., 2429–2437. https://proceedings.neurips.cc/paper_files/paper/2014/file/f306a2961c3dcdd21904a759d51f4e22-Paper.pdf

[7] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2026. The Faiss Library. *IEEE Trans. Big Data* 12, 2 (2026), 346–361. doi:10.1109/TBDATA.2025.3618474

[8] Jennie Duggan, Aaron J. Elmore, Michael Stonebraker, Magda Balazinska, Bill Howe, Jeremy Kepner, Sam Madden, David Maier, Tim Mattson, and Stan Zdonik. 2015. The BigDAWG Polystore System. *SIGMOD Record* 44, 2 (2015), 11–16. doi:10.1145/2814710.2814713

[9] Hector Garcia-Molina, Yannis Papakonstantinou, Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey D. Ullman, Vasilis Vassalos, and Jennifer Widom. 1997. The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems* 8, 2 (1997), 117–132. doi:10.1023/A:1008683107812

[10] Ryan Greene, Ted Sanders, Lilian Weng, and Arvind Neelakantan. 2022. New and improved embedding model. https://openai.com/index/new-and-improved-embedding-model/. Accessed: 2025-07-08.

[11] Rentong Guo, Xiaofan Luan, Long Xiang, Xiao Yan, Xiaomeng Yi, Jigao Luo, Qianya Cheng, Weizhi Xu, Jiarui Luo, Frank Liu, Zhenshan Cao, Yanliang Qiao, Ting Wang, Bo Tang, and Charles Xie. 2022. Manu: a cloud native vector database management system. *Proceedings of the VLDB Endowment* 15, 12 (Aug. 2022), 3548–3561. doi:10.14778/3554821.3554843

[12] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer. https://hastie.su.domains/ElemStatLearn

[13] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. 2024. Position: The Platonic Representation Hypothesis. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*, Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (Eds.). PMLR, 20617–20642. https://proceedings.mlr.press/v235/huh24a.html

[14] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL] https://arxiv.org/abs/2310.06825

[15] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2021), 535–547. doi:10.1109/TBDATA.2019.2921572

[16] Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, Brussels, Belgium, 2979–2984. doi:10.18653/v1/D18-1330

[17] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. 2004. A local search approximation algorithm for k-means clustering. *Computational Geometry* 28, 2 (2004), 89–112. Special Issue on the 18th Annual Symposium on Computational Geometry - SoCG2002. doi:10.1016/j.comgeo.2004.03.003

[18] Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=H196sainb

[19] Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2025. NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models. arXiv:2405.17428 [cs.CL] https://arxiv.org/abs/2405.17428

[20] Maurizio Lenzerini. 2002. Data Integration: A Theoretical Perspective. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 233–246. doi:10.1145/543613.543644

[21] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards General Text Embeddings with Multi-stage Contrastive Learning. arXiv:2308.03281 [cs.CL] https://arxiv.org/abs/2308.03281

[22] Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep Multilingual Correlation for Improved Word Embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 250–256. doi:10.3115/v1/n15-1028

[23] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. WWW'18 Open Challenge: Financial Opinion Mining and Question Answering *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1941–1942. doi:10.1145/3184558.3192301

[24] Yu A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (April 2020), 824–836. doi:10.1109/tpami.2018.2889473

[25] Julieta Martinez, Holger H. Hoos, and James J. Little. 2014. Stacked Quantizers for Compositional Vector Compression. arXiv:1411.2173 [cs.CV] https://arxiv.org/abs/1411.2173

[26] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting Similarities among Languages for Machine Translation. arXiv:1309.4168 [cs.CL] https://arxiv.org/abs/1309.4168

[27] Oracle. 2026. *Oracle AI Vector Search*. https://www.oracle.com/uk/database/ai-vector-search/

[28] James Jie Pan, Jianguo Wang, and Guoliang Li. 2024. Survey of vector database management systems. *The VLDB Journal* 33, 5 (July 2024), 1591–1615. doi:10.1007/s00778-024-00864-x

[29] Xavier Perez-Sala, Fernando De la Torre, Laura Igual, Sergio Escalera, and Cecilio Angulo. 2016. Subspace Procrustes Analysis. *International Journal of Computer Vision* 121, 3 (Aug. 2016), 327–343. doi:10.1007/s11263-016-0938-x

[30] pgvector contributors. 2026. pgvector: Open-source vector similarity search for Postgres. https://github.com/pgvector/pgvector.

[31] Hüseyin Temiz, Berk Gökberk, and Lale Akarun. 2019. Multi-view Reconstruction of 3D Human Pose with Procrustes Analysis. In *2019 Ninth International Conference on Image Processing Theory, Tools and Applications (IPTA)*. 1–5. doi:10.1109/IPTA.2019.8936071

[32] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. arXiv:2104.08663 [cs.IR] https://arxiv.org/abs/2104.08663

[33] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Marilyn Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, New Orleans, Louisiana, 809–819. doi:10.18653/v1/N18-1074

[34] Shengbang Tong, David Fan, Jiachen Zhu, Yunyang Xiong, Xinlei Chen, Koustuv Sinha, Michael Rabbat, Yann LeCun, Saining Xie, and Zhuang Liu. 2025. MetaMorph: Multimodal Understanding and Generation via Instruction Tuning. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 17001–17012. doi:10.1109/iccv51701.2025.01579

[35] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or Fiction: Verifying Scientific Claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 7534–7550. doi:10.18653/v1/2020.emnlp-main.609

[36] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD/PODS '21)*. ACM, 2614–2627. doi:10.1145/3448016.3457550

[37] Chuangxian Wei, Bin Wu, Sheng Wang, Renjie Lou, Chaoqun Zhan, Feifei Li, and Yuanzhe Cai. 2020. AnalyticDB-V: A Hybrid Analytical Engine Towards Query Fusion for Structured and Unstructured Data. *Proc. VLDB Endow.* 13, 12 (2020). doi:10.14778/3415478.3415541

[38] Beining Yang, Yang Cao, and Yang Ren. 2025. Integrating Vector Databases across Embedding Models. *Proceedings of the ACM on Management of Data* 3, 6 (Dec. 2025), 1–28. doi:10.1145/3769803

Received 17 January 2026; revised 19 May 2026; accepted 12 June 2026