Toward Relational-Vector Databases with Schema-Structured Embeddings

Abolfazl Asudeh
University of Illinois Chicago
asudeh@uic.edu

Mohsen Dehghankar
University of Illinois Chicago
mdehgh2@uic.edu

Mahdi Erfanian
University of Illinois Chicago
merfan2@uic.edu

ABSTRACT

Vector databases use embeddings for similarity search while relying on traditional attributes for relational query processing. This separation limits their applicability to domains where the semantic attributes required for database operations are not explicitly available. In this paper, we introduce Relational-Vector Databases (rVecDBs), a database model in which the relational structure is encoded within the embeddings. The enabling representation is the Schema-Structured Embedding (SSE), where different coordinate blocks are geometrically aligned with different semantic facets, and nested prefixes within a block represent different levels of semantic abstraction. Unlike materialized categorical attributes, SSEs represent semantic facets as continuous geometric spaces, allowing semantic comparisons beyond a predefined set of labels. We formulate a range query model over SSEs and use it to define relational database operations, including semantic filtering, similarity ranking, and joins. We further formulate grouping and aggregation over the learned semantic spaces. We present two complementary approaches for learning SSEs and address schema compatibility across relations through sequential alignment of shared semantic join keys. Our experiments on four real-world datasets demonstrate the effectiveness of SSEs in capturing facet-specific and hierarchical semantics while preserving general similarity search quality. Compared to materialized semantic labels, SSEs significantly improves filtering accuracy, achieving a query accuracy comparable to separate facet embeddings while reducing vector storage by a large factor. SSE's block structure further enables efficient query processing, achieving a major speedup over masked embeddings for a composite query on one million tuples.

This paper takes the first step toward the development of rVecDBs, opening up a range of research directions in indexing, query optimization, and physical database design. The authors hope this work lays the foundation for a new generation of vector databases where relational query processing and similarity search operate directly over learned representations.

PVLDB Reference Format:
Abolfazl Asudeh, Mohsen Dehghankar, and Mahdi Erfanian. Toward Relational-Vector Databases with Schema-Structured Embeddings. PVLDB, (): xx – yy, 2027.
doi:xx.xxxxx/xxxx

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. , No. ISSN 2150-8097.
doi:xx.xxxxx/xxxx

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

1 INTRODUCTION

Vector databases have become a core infrastructure for modern AI applications. They store high-dimensional embeddings and support approximate nearest neighbor (ANN) search over images, text, multimodal data, and other complex objects. Despite this rapid adoption, the underlying data model remains fundamentally split. Conventional relational attributes, such as time and location, support filtering and structured query processing, while the vector representations (aka embeddings) support similarity search. A constrained ANN query therefore typically has the form `SELECT * FROM T WHERE filter(A,`τ`) ORDER BY sim(E,q) LIMIT k.` The predicates are evaluated over traditional attributes A, while the similarity search (ranking) is evaluated over an embedding E [19, 21, 26].

Current vector databases can retrieve objects similar to a query, but they rely on predefined, explicitly available traditional attributes to support relational database operations such as filtering, joins, grouping, aggregation, and roll-up/drill-down [9, 28, 45, 51]. This separation is increasingly inadequate because, across many domains, such as ecological or medical data, where the metadata required for filtering and other database operations is often incomplete, expensive to curate, or entirely absent [6, 12, 32]. Some other approaches use separate embedding spaces to represent different aspects of an entity, rather than a single general-purpose embedding that supports relational queries over these aspects [16, 24, 29, 37].

In all these cases, embeddings remain opaque: although they may encode rich semantic information, they lack explicit structure, making it difficult to isolate specific concepts or attributes. In addition, embeddings are difficult to interpret. Although a dense embedding may capture rich information about real-world objects, its individual dimensions generally lack clear, human-understandable meanings. This limits their usefulness in applications where interpretability is important [2, 33, 38, 46].

In this paper, we propose *Relational-Vector Databases* (rVecDBs), a new database model in which *embeddings expose a semantic schema structure* that can be used directly in relational query processing. The foundation of rVecDBs is the *Schema-Structured Embedding* (SSE). An SSE partitions the embedding into coordinate blocks, each assigned to a semantic facet. For facets with hierarchical semantics, nested prefixes of a block represent different levels of abstraction. The schema specifies the block structures, while their geometric mapping is learned from data. Unlike conventional vector databases, where relational attributes are stored separately from embeddings, an rVecDB makes the learned semantic structure within the embedding available to database operators. Let us introduce Example 1.1

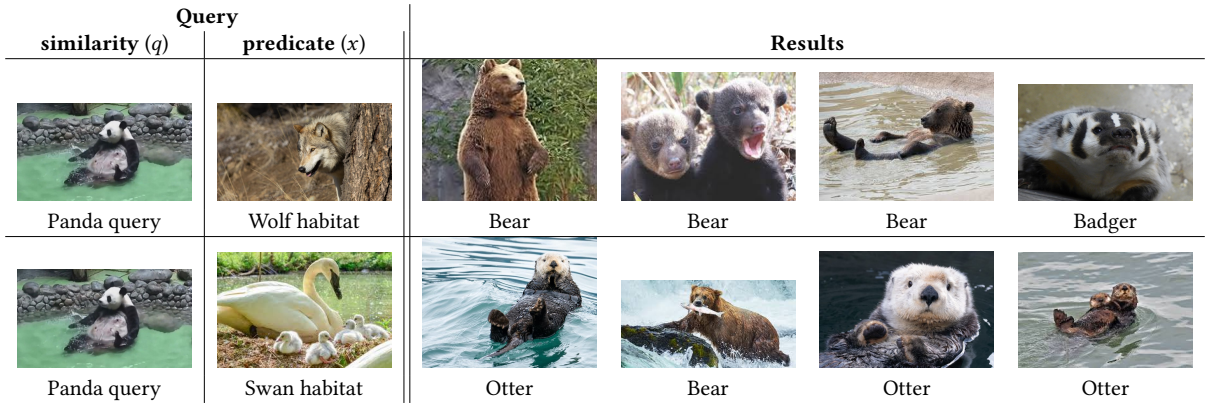


Figure 1: Retrieved results for Example 1.1. The query asks for the animal images similar to the query q (panda) at the `family` level, among the ones in the `habitat` of the predicate x.

on a biodiversity dataset containing animal images[1] as the running example across the paper.

Example 1.1. On a dataset of biodiversity images, a user may wish to `[find` k `animals with habitats similar to that of reference image (e.g., a swan), ranked by their family-level taxonomic similarity to a panda]`. Using SSE (see Section 3), which organizes embeddings into separate blocks for semantic facets such as taxonomy and habitat, and uses nested prefixes within the taxonomy block, this query can be processed by filtering based on similarity in the habitat block and ranking by similarity in the family-level prefix. Notably, neither habitat nor taxonomic information is explicitly available as structured attributes in the dataset or the query. As we shall explain in Section 4, rVecDB enables formulating the user query as follows, where x is the habitat reference image and q is the ranking reference image:

```
SELECT * FROM Animals AS t
WHERE E_habitat(t) ≈_τh E_habitat(x)
ORDER BY sim(E^(2)_tax(t), E^(2)_tax(q)) DESC LIMIT 4
```

Figure 1 shows our experiment results of this query over a dataset of animal images represented by SSEs (see Section 6 for more details). Changing the predicate tuple while keeping the ranking reference fixed changes the feasible semantic region: a wolf-habitat predicate returns predominantly bear-like animals, while a swan-habitat predicate shifts the results toward otters. Interestingly, the returned images often share visual characteristics with the reference panda image.

Indeed, an alternative approach is to train classifiers that predict semantic attributes and materialize their outputs in traditional attributes. Such an approach is useful when the required attributes have a known, fixed vocabulary. However, materialized labels do not preserve the continuous relationships among facet values. SSEs instead represent each facet as *a continuous geometric space*, allowing predicates to be specified using concept prototypes or example tuples. For instance, a habitat predicate can be defined using the habitat representation of an arbitrary image rather than selecting from a predefined list of habitat categories. This also allows semantic similarity and compatibility to be evaluated at different levels of granularity. Of course SSEs do not eliminate the need for supervision, but that they preserve learned semantic relationships as a queryable soft schema in the structure of the vectors rather than discrete traditional labels.

The rVecDB model raises several research questions. First, how can embeddings be learned so that designated blocks reliably capture their intended facets, hierarchical prefixes represent the corresponding abstraction levels, and the full embedding preserves retrieval quality? Second, how should vector indices support constrained ANN when both the filtering predicate and the ranking objective are geometric, potentially over different blocks or prefixes? Third, how can relational operators such as joins, grouping, and aggregation be defined and efficiently evaluated over continuous, learned attributes? These questions connect representation learning with relational database concepts such as database modeling, access methods, and query processing.

This paper takes *the first step* by introducing SSE and the rVecDB data and query models. Specifically, after reviewing the related work in Section 2, we first formulate schema-structured embeddings (SSEs) in Section 3. Next, in Section 4, we define our relational-vector database (rVecDB) model and formally describe the range query model over SSEs, which forms the basis for defining database operators. Our data and query models are agnostic to the techniques used for training SSEs. Still, in Section 5, we formulate our learning objective and propose two complementary learning approaches based on the training data at hand. Joining multiple relations based on semantic blocks requires schema compatibility across those relations. We address this issue in Section 5.2 through sequential alignment of shared semantic join keys. Next, in Section 6, we provide experimental evidence to validate our approach and evaluate its performance in various settings. We then provide a set of illustrative queries on our rVecDB model in Section 7 to demonstrate its expressiveness and the composition of relational operators over different semantic facets and abstraction levels. Indeed, the full-stack development of an rVecDB requires addressing a wide range of technical and systems problems, including the design of access methods that exploit the structure of SSEs, selectivity and cardinality estimation over learned semantic spaces, and query

[1]The dataset may optionally contain limited traditional attributes such as id, time, and the information of the photographer.

planning and optimization. We briefly discuss some of these research directions in our concluding remarks in Section 8. We hope this paper paves the way for a new generation of vector databases in which learned representations are integral to relational data modeling and query processing.

2 RELATED WORK

In this section, we review existing methods for answering queries that combine relational query processing with vector similarity predicates, as illustrated in Example 1.1. We categorize these approaches into two groups based on how they represent and query semantic properties: (1) **explicit-attribute approaches**, which follow the traditional database paradigm and evaluate predicates over structured attributes or labels, and (2) **continuous-representation approaches**, a more modern line of work that evaluates predicates through similarity in embedding spaces.

2.1 Using Explicit Attributes

As discussed in the introduction, existing filtered ANN search methods combine vector similarity search with predicates over explicitly stored attributes, such as category, location, or time [3, 9, 28, 44, 45, 47, 51]. These methods assume that the attributes required for filtering are explicitly available both in the dataset, as stored fields, and in the query, as specified attribute values.

Attributes unavailable in the dataset. When the required attributes are not stored in the dataset, a common alternative is to train classifiers or concept prediction models that extract semantic attributes from the underlying objects and materialize the predictions as structured fields [13, 15, 22, 27]. This enables traditional database operations over the predicted labels but has several drawbacks: (i) prediction errors in the extracted attributes propagate to query results; (ii) classifiers are restricted to a predefined set of attributes and labels, so they cannot support queries over previously unspecified semantic properties; (iii) extracting and materializing the labels adds computational and storage costs; and, most importantly, (iv) discrete labels discard the continuous relationships among attribute values. For example, a habitat classifier may label an image as *wetland* or *forest*, but these labels do not capture how similar two habitats are.

Attributes unavailable in the query. A user may not know the relevant attribute value but may have an example of it, such as another object or a concept prototype; in Example 1.1, the filter is defined by another image without any explicit label. Supporting such queries with a classifier would require mapping each query example to a predefined label at query time, which inherits drawbacks (i)–(iv) above and additionally incurs model inference latency on every query.

In contrast to these methods, SSEs represent semantic facets as continuous geometric spaces, so a predicate can be specified directly by a concept prototype or an example object, such as the habitat representation of an arbitrary image. Filtering then becomes similarity search within the corresponding facet, with no labels to predict or materialize.

2.2 Using Continuous Representations

Another line of work represents different semantic aspects of an object with aspect-specific embeddings. Multi-Aspect Dense Retrieval [16] and MURAL [37] learn aspect-specific representations for properties such as product category, brand, and color, and vector databases such as Milvus and PostgreSQL with pgvector support multiple vector fields per object [24, 29, 44]. Conditional Similarity Networks (CSN) [41] instead learn a single embedding together with one non-negative mask per notion of similarity, trained from triplet comparisons; similarity under a notion is measured on the masked embedding, and the masks are either learned or fixed to disjoint subsets of dimensions. CSN shows that several notions of similarity can share one embedding, but its masks are flat: it does not represent levels of granularity within an aspect, and it does not treat the masked subspaces as attributes for database operations.

These approaches have three limitations. First, apart from CSN, each aspect has its own embedding model and space, which is redundant when aspects are related, for example nested as in animal taxonomy (order, family, genus): each related attribute is learned and stored independently, multiplying storage and computation while ignoring their shared structure. Matryoshka Representation Learning (MRL) [17] nests representations within one embedding: prefixes of increasing length are each usable on their own, and all prefixes are trained with the same objective and labels. A shorter MRL prefix is therefore a cheaper, lower-fidelity version of the same semantics, used to trade accuracy for cost. SSEs also use nested prefixes, but each prefix is supervised at its own level of a hierarchy, so a shorter prefix encodes coarser semantics (e.g., order instead of genus) rather than a coarser approximation. Other hierarchical embeddings [7, 25, 42] capture a hierarchy within one space, but neither they nor MRL organize multiple semantic facets into a unified representation that supports facet-specific predicates.

Second, each embedding is still a high-dimensional dense space whose coordinates have no identifiable meaning. Interpretability matters [11, 35] when users must understand why an object satisfied a predicate or when results must be audited, and this opacity propagates to downstream systems built on embeddings, from retrieval pipelines to larger models [18].

Third, the relevant aspects must be identified in advance, which is restrictive when attributes or their values are hard to predefine. For example, consider a dataset of books in which attributes such as author, publication year, and publisher are well-defined, but genre, subgenre, themes, narrative style, mood, and literary tropes are open-ended, often hierarchical, overlapping, and evolving as new styles emerge. A book may be science fiction, more specifically dystopian fiction or cyberpunk, while exploring identity, isolation, and artificial intelligence; a complete set of such attributes and values for all books is hard to define.

In contrast, SSEs learn a single, general-purpose embedding that organizes semantic facets into separate blocks, with nested prefixes within a block representing levels of granularity. Related attributes share one space, facets and their hierarchical levels can be queried without separate embeddings, and the dimensions gain an interpretable organization with less redundancy.

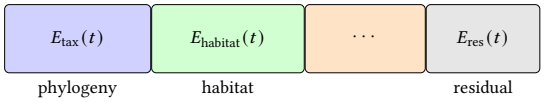


Figure 2: An SSE of an animal t, containing facet-specific blocks, each aligned with different semantic aspects, and the residual block capturing information not assigned to a predefined facet.

2.3 Query Planning

Query planning for filtered vector search has been studied extensively: existing methods choose among filter-first, vector-search-first, and interleaved strategies based on filter selectivity and index structure [5, 28, 44, 47]. These techniques are orthogonal to our work and can be applied to rVecDBs. Moreover, the structure of SSEs opens opportunities for new, more efficient query plans, which we discuss in the following sections.

3 SCHEMA-STRUCTURED EMBEDDINGS

Let t denote an object and let $\mathcal{F} = \{\phi_1, \ldots, \phi_m\}$ be a set of semantic facets specified for its domain. A Schema-Structured Embedding (SSE) maps t to a vector $E(t) \in \mathbb{R}^d$ of the form

$$E(t) = [E_1(t), \ldots, E_m(t), E_{\text{res}}(t)],$$

where each block $E_i(t) \in \mathbb{R}^{d_i}$ is associated with a semantic facet ϕ_i, and $E_{\text{res}}(t)$ is an optional residual block. The embedding schema specifies the facets and their corresponding coordinate blocks, including their dimensionalities. These assignments are shared across objects represented under the same schema. The geometric structure within each block is learned from data.

For example, an SSE for an animal image may contain separate blocks for taxonomy, habitat, diet, geography, and morphology, as illustrated in Figure 2. Each block is trained to capture similarity with respect to its designated facet. Two animals with similar habitats should have nearby representations in the habitat block, even if they differ in taxonomy or visual appearance. The taxonomy block, in contrast, should reflect taxonomic relationships. Different blocks may therefore induce different similarity orderings over the same collection of objects. Note that the facets are not statistically independent, even though each block is trained to represent the relationships relevant to its facet.

The schema specifies which semantic facets are represented, and each block acts as a *soft attribute*, whose value is a vector of real numbers rather than a categorical label with a predefined cardinality. Indeed, training the embeddings may follow predefined labels, but the resulting vector representations are not restricted to those labels. That is, two entities with the same labels in the training set, may have similar, but not identical, vector representations on their corresponding blocks.

The residual block provides capacity for retrieval-relevant information not explicitly assigned to the semantic facets. It is optimized jointly with the facet blocks to preserve general similarity-search quality over the full embedding.

3.1 Hierarchical Facets

Some semantic facets have hierarchical structure, with different levels corresponding to different degrees of semantic specificity.

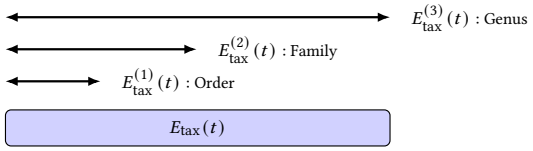


Figure 3: A hierarchical facet block for animal taxonomy.

For a hierarchical facet ϕ_i, an SSE assigns *nested prefixes* of its block E_i to successive levels of the hierarchy. Let L_i denote the number of levels, with prefix lengths $d_{i,1} < \cdots < d_{i,L_i} = d_i$. The representation of an object t at level ℓ is defined as

$$E_i^{(\ell)}(t) = E_i(t)[1 : d_{i,\ell}].$$

Shorter prefixes represent coarser semantic distinctions, while longer prefixes capture increasingly fine distinctions. The assignment of prefixes to abstraction levels is part of the embedding schema. Prefix length specifies the semantic abstraction at which objects are compared. Hence, the same facet block can support comparisons at several abstraction levels without storing a separate categorical attribute for each level. For example, as illustrated in Figure 3, the taxonomy block of an animal SSE may use dimension prefixes [1..8], [1..32], and [1..64] to represent Order, Family, and Genus, respectively. At the Order level, animals belonging to the same order should have similar representations, even when they belong to different families. Longer prefixes introduce finer distinctions, allowing similarity to be evaluated at the Family or Genus level.

3.2 Semantic Comparisons

Continuous facet representations allow semantic comparisons to be specified through reference vectors rather than only through predefined labels. A reference may be a learned prototype for a named concept, a representation constructed from several examples, or the representation of an example object. In Example 1.1, a habitat comparison may use the habitat block of the SSE of a swan image, i.e., $E_{\text{habitat}}(t_{\text{swan}})$, as its reference. The comparison then concerns similarity to the habitat represented by that particular example, without requiring the user to assign it a categorical habitat label.

A similarity threshold controls how restrictive a comparison is. Let $E_B(t)$ denote the representation of t in a selected block or hierarchical prefix, and let c be a reference vector in the same space. The condition $\text{sim}(E_B(t), c) \geq \tau$ defines a neighborhood around c according to the chosen similarity function. For a fixed space and reference, increasing τ restricts the matching values, while decreasing it admits a broader range. The same stored representations can therefore support strict or relaxed semantic matching without changing the schema or assigning new labels to the objects.

Deferring the learning of SSEs to Section 5, we next introduce our model of the relational-vector databases.

4 RELATIONAL-VECTOR DATABASE MODEL

We define a Relational-Vector Database (rVecDB) as a set of relations, where each relation $T = \{t_i\}_{i=1}^n$ is a collection of tuples. Each tuple $t \in T$ is represented using a schema-structured embedding (SSE) $E(t) \in \mathbb{R}^d$, where $E(t)$ is structured into $m < d$ blocks as

$E(t) = [E_1(t), \cdots, E_m(t), E_{\text{res}}(t)]$. Each block $E_\phi(t) \in \mathbb{R}^{d_\phi}$ is geometrically aligned with a domain concept or a semantic attribute[2]. A facet may be non-hierarchical, such as habitat or conservation status, or hierarchical, such as taxonomy, geography, topic, or organizational structure.

4.1 Query Model

Range queries are central to rVecDB's query model, enabling the database operations on top of SSEs. Formally, let $\text{sim} : \mathbb{R}^{d^*} \to \mathbb{R}^+$ be a similarity function that takes two vector representations in $\mathbb{R}^{d^*}$ as the input and measures their similarity, with larger values indicating higher similarities. Without loss of generality, we assume the similarity values are normalized in range $[0, 1]$, such that $\text{sim}(x, x) = 1,\ \forall x \in \mathbb{R}^{d^*}$. When $d^* = d$, the similarity is measured over the entire SSE, while $d^* = d_B < d$ measures the similarity over the projected dimensions of a block (or a hierarchical prefix) B.

For a reference vector $c \in \mathbb{R}^{d_B}$ in the selected space and thresholds $0 \le \perp \le \top \le 1$, define the *semantic range*

$$\mathcal{R}(T; E_B, c, \perp, \top) = \{t \in T : \perp \le \text{sim}(E_B(t), c) \le \top\}.$$

For the common case where $\top = 1$, we simplify the notation to $\mathcal{R}(T; E_B, c, \perp) = \{t \in T : \perp \le \text{sim}(E_B(t), c)\}$.

The reference c may be the representation of an example tuple, averaged over several examples, or a learned concept.

Having formulated the range queries, we next show how it enables the database operations on SSEs.

Semantic predicates. A semantic predicate is specified as a range query

$$\phi(E_\phi, c_\phi, \perp, \top) \equiv \mathcal{R}(T; E_\phi, c_\phi, \perp, \top).$$

That is, the tuples that fall inside the range $\mathcal{R}$ satisfy the predicate ϕ. For instance, let us consider Example 1 with the swan image (2nd row of Figure 1) as the reference image. The predicate $E_{\texttt{Habitat}} \approx_\tau E_{\texttt{Habitat}}(x)$ is modeled as a range query over the embedding block $E_\phi = E_{\text{habitat}}$, around the habitat block of the swan image, i.e., $c_\phi = E_{\text{Habitat}}(t_{\text{swan}})$, and a similarity threshold τ. Formally

$$E_{\text{Habitat}} \approx_\tau E_{\text{Habitat}}(x) \equiv \mathcal{R}(T; E_{\text{Habitat}}, E_{\text{Habitat}}(t_{\text{swan}}), \tau).$$

Similarly, a predicate over a hierarchical block selects a prefix $E_\phi^{(\ell)}$ and a reference vector in that prefix space.

A conjunction $\Phi = \phi_1 \wedge \cdots \wedge \phi_s$ defines the feasible set

$$T_\Phi = \{t \in T : \phi_j(t) = \text{true},\ \forall j \in [s]\} = \bigcap_{j=1}^{s} \mathcal{R}\left(T; E_{\phi_j}, c_{\phi_j}, \perp_j, \top_j\right).$$

Each predicate may use a different block, prefix, reference, and threshold interval. Disjunction corresponds to union of ranges, while negation is specified using a range query with an upper bound. For example, the predicate $E_{\texttt{Habitat}} \not\approx E_{\texttt{Habitat}}(x)$ for the swan image in Example 1.1 can be formulated using a similarity upper bound τ as

$$E_{\text{Habitat}} \not\approx E_{\text{Habitat}}(x) \equiv \mathcal{R}(T; E_{\text{Habitat}}, E_{\text{Habitat}}(t_{\text{swan}}), 0, \tau).$$

Conventional predicates can be combined with these semantic conditions.

[2]A relation may also contain traditional attributes, such as time, location, or user-provided metadata.

Similarity ranking. A constrained similarity query ranks the feasible tuples in a selected space E_ρ relative to a query vector q. Let τ_k be the kth largest ranking score among feasible tuples. Then the constrained similarity can be formulated as a range query as:

$$\texttt{ORDER BY SIM q ON E}_\rho\ \texttt{LIMIT k} = \text{Top}_k\left(T_\Phi;\ \text{sim}(E_\rho(t), q)\right) \equiv \mathcal{R}(T_\phi; E_\rho, q, \tau_k).$$

The ranking space E_ρ may coincide with a predicate space, overlap it, or use different blocks. In Example 1.1, the ranking constraint $\texttt{ORDER BY SIM } E^{(2)}_{\texttt{tax}}(q)\ \texttt{ON } E^{(2)}_{\texttt{tax}}\ \texttt{LIMIT 4}$ returns the top-4 tuples based on their similarity with the panda image (q) on the family level (level 2) of the taxonomy block, while the filtering predicate is on the habitat block.

Before moving to the next section, we would like to underscore that these definitions specify the logical result, not an execution order. An implementation may evaluate predicates first, explore the ranking order first, or process the corresponding ranges jointly.

4.2 From Query Model to Database Operations

Semantic Joins. A semantic VecJOIN compares tuples through compatible facet spaces. Let R and S be relations with compatible representations for a selected block or prefix E_B. A semantic join is defined as

$$R \bowtie_{B,\perp} S = \{(r, s) \in R \times S : \text{sim}(E_B(r), E_B(s)) \ge \perp\}.$$

Hence, a join can be formulated using a set of range queries over S, where the reference vectors are obtained from the relation R. That is,

$$R \bowtie_{B,\perp} S = \bigcup_{r \in R} \Big(\{r\} \times \mathcal{R}(S; E_B, E_B(r), \perp)\Big).$$

For example, animals and habitat records can be joined through their habitat blocks. Increasing the threshold $\perp = \tau_j$ requires closer habitat similarity, while decreasing it admits more matches.

Indeed, a semantic VecJOIN requires schema compatibility across the joined relations. In Section 5.2, we address this issue by introducing shared semantic-join keys, which are sequentially aligned during the training process.

Grouping. Unlike grouping by categorical labels, the groups over SSEs are determined by continuous facet values. A semantic GROUP BY partitions a relation T into k groups based on the representations of its tuples in a selected block or hierarchical prefix E_B. Each group G_i is represented by a center $c_i \in \mathbb{R}^{d_B}$, which need not correspond to an existing tuple. The grouping objective is

$$\max_{\{G_i, c_i\}_{i=1}^k} \sum_{i=1}^{k} \sum_{t \in G_i} \text{sim}(E_B(t), c_i),$$

where the G_i form a partition of T into k nonempty groups. Each tuple is assigned to the center with the highest similarity, with ties resolved deterministically. That is,

$$\texttt{GROUP BY}_k\ E_B \equiv \Big\{\{t \in T : \text{sim}(E_B(t), c_i) \ge \text{sim}(E_B(t), c_j),\ \forall j \ne i\}\Big\}_{i=1}^{k}.$$

with ties resolved deterministically. Thus, GROUP BY induces a partition of the projected embedding space into similarity-based regions, analogous to Voronoi cells.

A group may be represented by its center or by a set of representative tuples that expose its contents to the user. For example, consider animals grouped by habitat. To represent a group G_i, one can cluster its members using their full embeddings, obtaining h_i nonempty subgroups $H_{i,1}, \ldots, H_{i,h_i}$. We present each subgroup $H_{i,j}$ as the $\langle t^{\mu}_{H_{i,j}}, |H_{i,j}| \rangle$ pair, where $t^{\mu}_{H_{i,j}}$ is the medoid tuple of $H_{i,j}$.

Aggregation. Aggregation can operate on conventional numeric attributes or on similarity scores computed from SSEs. Let U denote the input relation or a group, E_B a selected block or prefix, and q a reference vector in the corresponding space. For an aggregate function $F \in \{\text{SUM}, \text{MIN}, \text{MAX}, \text{AVG}\}$, we define

$$\text{Agg}_F(U; E_B, q) = F\Big(\{\text{sim}(E_B(t), q) : t \in U\}\Big),$$

where the argument is a multiset of similarity scores, with one contribution per input tuple occurrence. Please note that the aggregation space may differ from the grouping space. In Example 1.1, for instance, the grouping can be done using the habitat block, while the aggregate is computed over the family level of the taxonomy block.

In Section 7, we provide a set of query examples to illustrate the composition of the database operations using `rVecDB` model.

5 LEARNING SSE

5.1 Learning Objective

The embedding schema of an SSE specifies the semantic facets, coordinate blocks, and hierarchical levels, but their geometric alignment must be learned from data. An SSE can be obtained by training an encoder directly or by transforming an existing embedding. For example, given a pretrained vector representation $z(t)$, a learnable projection f_θ produces $E(t) = f_\theta(z(t))$.

In order to train an SSE, one can minimize a loss function that jointly optimizes retrieval preservation, facet alignment, and hierarchical alignment:

$$\mathcal{L} = \mathcal{L}_{\text{ret}} + \sum_{i=1}^{m} \lambda_i \mathcal{L}_{\text{facet},i} + \sum_{i \in \mathcal{H}} \sum_{\ell=1}^{L_i} \mu_{i,\ell} \mathcal{L}_{\text{prefix},i,\ell}\,, \tag{1}$$

where $\mathcal{H}$ denotes the set of hierarchical facets, and the weights λ_i and $\mu_{i,\ell}$ are the tunable coefficients that control the contributions of various objectives. The retrieval loss $\mathcal{L}_{\text{ret}}$ preserves the similarity structure of a pretrained representation or optimizes general retrieval quality. Each facet loss $\mathcal{L}_{\text{facet},i}$ aligns its block with the designated semantic facet, while for hierarchical facets, $\mathcal{L}_{\text{prefix},i,\ell}$ encourages alignment at level ℓ using supervision from the corresponding hierarchy level.

Although not limited to any specific training approach, we discuss two complementary approaches for learning the geometric structure of the designated blocks and prefixes below.

Pairwise similarity supervision. Rather than requiring categorical labels, SSEs can be trained using pairwise semantic similarities. Let $\mathcal{P}_B$ be a set of supervised tuple pairs for a block or prefix E_B, and let $S_B(t,u)$ denote the target similarity between tuples t and u with respect to the corresponding semantic facet. We minimize the discrepancy between the learned and target similarities:

$$\mathcal{L}_{\text{sim}}(E_B, S_B) = \frac{1}{|\mathcal{P}_B|} \sum_{(t,u) \in \mathcal{P}_B} \Big(\text{sim}(E_B(t), E_B(u)) - S_B(t,u)\Big)^2.$$

The targets may be derived from graded annotations, expert-provided similarities, ontologies, or proxy features. For example, the animal images taken in nearby locations share similar habitats. Similarly, given numerical habitat annotations $a_{\text{habitat}}(t)$ generated by an off-the-shelf machine learning model, the target can be defined as the cosine similarity between the annotation vectors.

Using pairwise similarities, the objective function in Equation 1 is instantiated as $\mathcal{L}_{\text{facet},i} = \mathcal{L}_{\text{sim}}(E_i, S_i)$, and $\mathcal{L}_{\text{prefix},i,\ell} = \mathcal{L}_{\text{sim}}(E_i^{(\ell)}, S_{i,\ell})$. The retrieval-preservation loss can be instantiated similarly, using the original embedding similarities as targets:

$$\mathcal{L}_{\text{ret}} = \mathcal{L}_{\text{sim}}(E, S_0), \qquad S_0(t,u) = \text{sim}(z(t), z(u)).$$

This approach follows the principle of similarity-preserving distillation [39], which ensures the learned representation preserves pairwise relationships without reproducing the original embedding coordinates.

Contrastive supervision. When supervision identifies semantically related and unrelated examples, we can instead use block-specific or prefix-specific contrastive learning. For a hierarchical facet i at level ℓ, let $\mathcal{M}$ be a training minibatch and $P_{i,\ell}(t)$ the examples in $\mathcal{M} \setminus \{t\}$ sharing the same label as t at that level. Let $\mathcal{A}_{i,\ell}$ denote the set of tuples t (known as anchors) for which at least one positive example exists in the minibatch at hierarchy level ℓ. That is $\mathcal{A}_{i,\ell} = \{t \in \mathcal{M} \mid |P_{i,\ell}(t)| > 0\}$. Following supervised contrastive learning [14], we define

$$\mathcal{L}^{\text{con}}_{i,\ell} = -\frac{1}{|\mathcal{A}_{i,\ell}|} \sum_{t \in \mathcal{A}_{i,\ell}} \frac{1}{|P_{i,\ell}(t)|} \sum_{p \in P_{i,\ell}(t)} \log \frac{\exp\Big(\text{sim}(E_i^{(\ell)}(t), E_i^{(\ell)}(p))/\eta\Big)}{\sum_{u \in \mathcal{M} \setminus \{t\}} \exp\Big(\text{sim}(E_i^{(\ell)}(t), E_i^{(\ell)}(u))/\eta\Big)},$$

where $\eta > 0$ is a temperature parameter. The loss brings examples sharing the designated semantic property closer while separating examples with different properties. The positive and negative sets depend on the selected facet and hierarchy level. For instance, two animals belonging to different families within the same order are positives at the Order level but negatives at the Family level.

Note that both of these approaches support incomplete supervision, since the similarity loss uses only pairs for which target similarities are available, and contrastive learning uses available positive and negative relationships. As a result, they do not need to materialize a complete pairwise similarity matrix.

5.2 Schema Compatibility Across Relations

A semantic `VecJOIN` requires the participating blocks to represent the same facet in a common embedding space. However, this compatibility is not guaranteed when SSEs for different relations are learned independently. Even when two blocks represent the same semantic facet and have identical dimensions, their coordinate systems may differ. For example, under cosine similarity, an orthogonal rotation of either block preserves all within-relation similarities, but generally changes the similarities between tuples from different

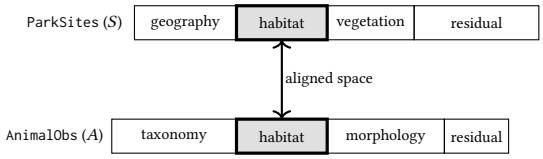


Figure 4: Illustrating two relations with different SSE schemas and a shared habitat join key, where the habitat block of ParkSites defines a fixed reference space.

relations. Consequently, independently learned blocks may capture their respective facets accurately while remaining unsuitable for cross-relation similarity comparisons.

We address this problem through *shared semantic join keys*, learned by sequentially aligning the corresponding blocks. For example, consider two relations, ParkSites (S) and AnimalObservations (A), as illustrated in Figure 4. The first contains records describing ecological sites in national parks, with facets such as geography, habitat, and vegetation. The second relation contains animal observations, with facets such as taxonomy, habitat, and morphology. Although the relations have different SSE schemas, their habitat blocks must occupy the same semantic space to support a habitat-based join. We first learn the SSE of S and use the value of its habitat block $E^S_{\text{habitat}}(s)$ as the "true label" when training the SSE of A, to align its habitat block with the fixed reference space.

Consider two relations T and S, where T refers to S using block B as the join key. Let $\mathcal{M} \subseteq T \times S$ denote the known join associations available during training. For each $(a, s) \in \mathcal{M}$, the block representation $E^S_B(s)$ serves as the supervision target for $E^A_B(a)$. Following the existing literature for learning aligned representations from paired observations [8, 50], we define the alignment loss as

$$\mathcal{L}_{\text{align}} = \frac{1}{|\mathcal{M}|} \sum_{(a,s)\in\mathcal{M}} \left[1 - \text{sim}\left(E^T_B(a), E^S_B(s)\right)\right].$$

This loss is added to the SSE learning objective, while the reference block remains fixed. Unlike pairwise similarity supervision, which preserves relative geometric relationships but is invariant to rotations, the alignment loss anchors the learned block to the reference coordinate system. The remaining blocks are learned according to their respective semantic objectives.

As a result, the shared key blocks of both relations belong to the same space, allowing the semantic join to be formulated using a set of range queries over S, as discussed in Section 4.2.

5.3 Block Size Allocation

The dimensionality of each facet block determines its representational capacity and the cost of indexing and querying it. Ideally, we would like to allocate the smallest number of dimensions that preserves the quality of semantic comparisons. Specifically, let $Q_i(d)$ denote the validation performance of facet i with d dimensions, measured on its intended semantic queries. Given a set of candidate dimensions $\mathcal{D}_i$ and a tolerance ε_i, the block size can be selected as

$$d_i^* = \min\left\{d \in \mathcal{D}_i : Q_i(d) \geq \max_{d'\in\mathcal{D}_i} Q_i(d') - \varepsilon_i\right\}.$$

d_i^* selects the smallest dimension whose validation performance is within ε_i of the best observed performance. For hierarchical facets, the allocation must also account for the performance of the designated prefixes.

Block allocation also supports schema evolution. We consider an append-only model in which the representations of existing facet blocks remain fixed, while new facets are learned using residual capacity or additional dimensions. This preserves existing facet-level predicates and indices without requiring retraining of the original blocks. Changes to the full embedding, however, may affect general similarity search and require updating its index.

6 EXPERIMENTAL EVALUATION

Having introduced the rVecDB data model and the SSE learning framework, our goal in this section is to evaluate whether a single schema-structured embedding is expressive, accurate, and efficient enough to support relational query processing over unlabeled data. Specifically, we design our evaluation to answer five research questions:

- **Q1: Representation and hierarchy (Section 6.2).** Can fixed coordinate blocks and nested prefixes faithfully capture distinct semantic facets and abstraction levels without hurting ordinary full-vector similarity search?
- **Q2: Predicates, ranking, and text queries (Section 6.3).** How accurately do SSEs answer composite range filters, rankings, similarity bands, and natural-language text predicates, especially on unseen classes and graded similarity where explicit labels fail?
- **Q3: Grouping and aggregation (Section 6.4).** Can SSEs support relational GROUP BY$_k$, representative group summarization (REP), hierarchical roll-up/drill-down, and cross-facet aggregation when no categorical columns exist?
- **Q4: Cross-relation joins (Section 6.5).** Can VecJOIN link related tuples across tables at a chosen granularity, including tables embedded by *different* encoders and aligned via shared join keys?
- **Q5: System efficiency and robustness (Section 6.6).** How compact and fast is an SSE in storage, indexing, and query execution compared to alternative architectures, how many dimensions does a block need, and do our findings hold across encoders?

Summary of observations. Our experiments answer all five questions in the affirmative and highlight why SSEs strike a sweet spot that existing approaches miss. First, compared to pipelines based on materialized classifier labels, zero-shot prompts, plain foundation embeddings, or runtime VLM scoring, SSEs are substantially more accurate across every relational operator, particularly on unseen classes, graded compatibility, and zero-shot text references, while running in milliseconds rather than minutes. Second, when multi-vector schemas and masked embeddings (CSN) are upgraded to use our own multi-level SSE losses, SSEs match their accuracy with no loss in expressive power while storing up to 7.1× **less data** than separate per-facet vectors and executing composite queries 9× **faster** with 8.8× **smaller facet indices** than masked embeddings.

We first present our experimental setup in Section 6.1, and then address Q1–Q5 in Sections 6.2–6.6.

Table 1: Benchmark datasets across modalities and domains.

Dataset	Tuples	Train / Test	Semantic Facets	Hierarchy Card.
AwA2 (images)	37,322	24.3k / 13.1k	taxonomy, habitat, diet	10 / 27 / 40
iNat (images)	100,000	56.0k / 44.0k	taxonomy	273 / 1,103 / 4,884
OpenAlex (text)	99,873	56.4k / 43.5k	scientific topic (+ Authors)	4 / 26 / 250
MovieLens (text)	13,227	9.9k / 3.3k	genre, tone, theme, style	–

6.1 Experimental Setup

Datasets and Workloads. To test whether SSEs generalize across modalities, facet structures, and hierarchy depths, we evaluate on four real-world benchmarks spanning images and text (Table 1). Every benchmark provides *graded* ground-truth facet similarity and holds out unseen leaf classes during training. **AwA2** [48] contains 37,322 animal images across 50 mammal species with continuous human ratings (0–100) over 85 semantic properties; we model 14 habitat properties and 5 diet properties as continuous facets alongside a three-level taxonomy (Order / Family / Genus), holding out AwA2's standard 10 test species (e.g., giant panda, seal, leopard) as *unseen* classes. **iNaturalist (iNat)** [40] (100,000 images across 10,000 species of animals, plants, and fungi) stresses hierarchical scaling on a deep tree of 273 orders, 1,103 families, and 4,884 genera, with 20% of species (2,000) held out as unseen. **OpenAlex** [30] provides a multi-relation scientific benchmark of 99,873 papers (titles and abstracts across all 26 fields of science) linked to a second table of 5,120 authors across 300 institutions; each paper follows a three-level topic hierarchy (Domain / Field / Subfield, e.g., *Physical Sciences* > *Chemical Engineering* > *Catalysis*), with 20% of leaf topics held out as unseen. **MovieLens** [10] tests non-hierarchical text generality over 13,227 movie plot summaries paired with continuous tag-genome relevance scores [43] grouped into four facets (genre, tone, theme, style).

SSE **Architecture and Training.** Each object is encoded once by a frozen foundation model: CLIP ViT-L/14 [31] for images (plus EVA02-CLIP-L/14 [36] in Q5), SPECTER2 [34] for scientific papers, and `bge-base` [49] for movie plots and author profiles. A lightweight two-layer MLP projector then maps this vector to a d=768 SSE (compressed to d=128 in Q5). On AwA2, the schema allocates 64 dimensions to taxonomy (nested prefixes 8/32/64 for Order, Family, and Genus), 32 to habitat, 16 to diet, and 656 to the residual block. On the larger hierarchies of iNat and OpenAlex, the taxonomy/topic block uses 128 dimensions with prefixes 16/64/128 and a 640-dim residual; MovieLens uses four 32-dim facet blocks and a 640-dim residual. We optimize Eq. 1: $\mathcal{L}_{\text{ret}}$ distills the frozen encoder's pairwise similarities; $\mathcal{L}_{\text{facet}}$ aligns each continuous block to ground-truth facet similarities; and $\mathcal{L}_{\text{prefix}}$ combines graded tree-level targets (e.g., at Family: 1.0 for same family, 0.5 for same order only, 0 otherwise) with supervised contrastive loss [14] at each prefix's abstraction level. Projector training takes seconds to a few minutes on a single RTX 5090 GPU. Hyperparameters are selected on a validation split of the training classes, and all results report the mean across three random seeds (standard deviation ≤ 0.01 unless noted).

Competing Paradigms. To ensure a controlled comparison, all baselines share the exact same frozen encoder and training supervision (Table 2), mirroring the two paradigms from Section 2:

Table 2: Competing paradigms. All methods use the same frozen encoder; trained methods use identical supervision.

Method	Paradigm	Stored / Tuple	Facet Evaluation
SSE **(ours)**	Schema-structured	1 vec (768 or 128d)	Cosine on fixed block / prefix
B1 Labels	Explicit attributes	1 vec + label cols	Predicted label overlap / match
B2 Prompts	Zero-shot profile	1 vec + score vec	Cosine of prompt-score profiles
B3 VLM	LLM/VLM operator	Raw object only	Per-pair VLM inference (0–10)
B4 Encoder	Unstructured vec	1 vec (768d)	Cosine on full embedding
B5 Separate	Multi-vector DB	6 vecs (920d)	Cosine on dedicated facet vector
B6 CSN	Learned masks	1 vec (768d) + masks	Masked 768d cosine per facet
MRL-style	Flat nested prefix	1 vec (768d)	All prefixes trained on leaf class
PCA Prefix	Unsupervised prefix	1 vec (768d)	Top-k principal components

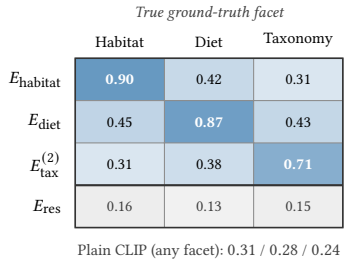
(1) *Explicit-attribute and unstructured baselines:* **B1 Labels** represents the standard filtered-ANN database pipeline: classifiers trained on the frozen embedding predict discrete labels for every facet and hierarchy level (e.g., thresholding AwA2 property scores at 50/100), materializing them as relational columns; facet similarity is label-set overlap and taxonomy similarity is tree-level match $(1, 0.5, 0)$. **B2 Prompts** requires no training, scoring each tuple against natural-language text prompts per attribute value (e.g., *"an animal that lives in water"*) and comparing prompt-score profiles. **B3 VLM** evaluates zero-shot multimodal semantic operators [27] by prompting Qwen2.5-VL-7B [1] on each reference–candidate image pair (0–10 rating). **B4 Encoder** uses the unmodified foundation embedding directly for all facets.

(2) *Continuous multi-view architectural baselines:* **B5 Separate Vectors** trains a dedicated MLP and stores a separate embedding per facet and per hierarchy level (6 vectors totaling 920 floats/tuple on AwA2, including the base vector for general search), reflecting multi-vector columns in Milvus [24] or `pgvector` [29]. **B6 CSN** [41] learns one 768-dim embedding plus one non-negative 768-dim mask per facet and level. Crucially, rather than CSN's weaker original triplet loss, we upgrade *both* B5 and B6 with our exact graded and hierarchical SSE losses. Thus, B5 and B6 serve as continuous-learning upper controls that isolate the *architectural and systems cost* of multi-vector fragmentation (B5) and dense overlapping masks (B6) versus a single fixed-block SSE.

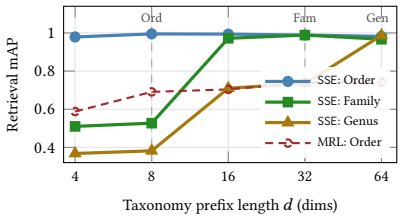
For hierarchy analysis (Q1), we also compare two nested-prefix ablations: **MRL-style** [17], which uses the identical SSE architecture but trains all prefixes $(8, 32, 64)$ on the finest label (Genus/Subfield), and **PCA Prefix**, which orders encoder coordinates by variance without supervision. In tables, boldface highlights the best result among single-representation database methods (SSE vs. B1, B2, B4), while the multi-view controls (B5, B6) are separated to highlight the accuracy–efficiency trade-off.

6.2 Representation Quality and Hierarchy (Q1)

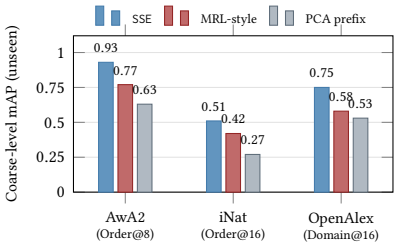
Every `rVecDB` operator treats coordinate blocks and prefixes as queryable semantic attributes. Before evaluating database queries, we therefore verify three foundational requirements: 1) each block must track its designated facet without leaking other facets, 2) nested prefixes must capture successive levels of a hierarchy, 3) structuring the vector must not hurt ordinary full-vector search.



(a) Block–facet correlation (Spearman ρ, AwA2).



(b) Prefix zoom staircase (AwA2).



(c) Coarse prefix on unseen classes.

Figure 5: Representation quality of SSEs. (a) Each block strongly tracks its designated facet along the diagonal while remaining disentangled from other facets and leaving non-facet signal in E_{res}. (b) Successive prefixes (8/32/64) unlock Order, Family, and Genus step by step while preserving coarser levels; MRL fails to represent Order because all prefixes optimize Genus. (c) Across image and text hierarchies, level-specific training generalizes coarse prefixes to unseen classes far better than MRL or PCA truncation.

Facet Alignment and Disentanglement. Over 200k random test pairs on AwA2, we measure how closely each method's facet similarity correlates with ground-truth graded similarity (Table 3). SSE blocks achieve 0.90 and 0.87 Spearman correlation on Habitat and Diet, which is nearly 3× **higher than the unstructured CLIP encoder** (0.31 / 0.28), 3.5× **higher than zero-shot prompts** (0.26 / 0.17), and +28–29 points above materialized classifier labels (0.61 / 0.59). Why do the baselines fail so sharply? Unstructured encoders (B4) entangle all visual features into a single metric: because an otter visually resembles a terrestrial weasel far more than a swan, full-vector similarity groups morphological relatives rather than aquatic animals. Zero-shot text prompts (B2) suffer from CLIP's weak attribute binding on abstract ecological concepts, while binary labels (B1) quantize continuous habitat spectra into coarse pass/fail buckets, discarding degrees of compatibility.

Equally important for relational query composition, Figure 5a shows that SSE coordinate blocks *disentangle* facets cleanly. Each block peaks sharply along the diagonal (0.90 Habitat, 0.87 Diet, 0.71 Taxonomy) while off-diagonal correlations drop to the natural ecological background between traits (0.31–0.45; e.g., cetaceans share both taxonomy and marine habitat), and the residual block E_{res} sheds facet structure (0.13–0.16) to retain general visual appearance. On MovieLens text, where plot summaries provide a noisier signal and true movie facets naturally overlap, SSE still doubles facet alignment over the plain `bge-base` embedding across all four facets (0.25–0.35 vs. 0.10–0.19), whereas thresholded binary tag classifiers collapse almost completely (0.01–0.05). Meanwhile, B5 and B6 match SSE within 0.01–0.02: confining facets to fixed disjoint coordinate slices incurs *zero accuracy penalty* compared to separate networks or full-width masks.

Hierarchical Zoom Levels via Nested Prefixes. Table 4 and Figures 5b–5c test whether nested prefixes act as semantic zoom levels across all three hierarchical benchmarks, evaluated on *unseen* species and topics never observed during training. Figure 5b reveals a striking semantic staircase as more prefix coordinates are read on AwA2: the 8-dim prefix already saturates Order retrieval (0.995 seen / 0.932 unseen) while leaving different families within an order unseparated (0.53); extending to 32 dimensions unlocks Family

Table 3: Facet alignment: Spearman ρ with ground-truth graded similarity over 200k test pairs (AwA2).

Method	Habitat	Diet	Taxonomy
SSE **(ours)**	**0.90**	**0.87**	0.71
B1 Labels (materialized)	0.61	0.59	**0.73**
B2 Prompts (zero-shot)	0.26	0.17	0.24
B4 Encoder (unstructured)	0.31	0.28	0.24
B5 Separate vectors (6 vecs)	0.91	0.87	0.72
B6 CSN (768d masks)	0.90	0.87	0.73

(0.989) while keeping genera merged (0.74); and reading all 64 dimensions resolves Genus (0.987) while *preserving* coarse Order and Family structure inside the longer prefix (pairwise Order AUC 0.95).

Comparing against MRL-style and PCA prefixes across AwA2, the 4,884-genus iNat tree, and OpenAlex (Table 4, Figure 5c) exposes a critical representation-learning insight: *coordinate nesting alone does not induce semantic hierarchy*. Because MRL trains every prefix length on leaf-level labels, its 8-dim prefix is merely a lossy compression of Genus rather than a representation of Order, dropping unseen coarse-level mAP by 16 points on AwA2 (0.77 vs. 0.93), 9 points on iNat (0.42 vs. 0.51), and 17 points on OpenAlex (0.58 vs. 0.75; 0.57 vs. 0.76 on seen topics). Unsupervised PCA truncation fares even worse (0.63 / 0.27 / 0.53) because leading variance directions capture background lighting and style rather than domain ontology. Finally, ablating the prefix objective in Eq. 1 shows why combining graded tree similarity with supervised contrastive loss is essential: on deep hierarchies like iNat (∼5k genera), random pairs rarely share fine taxa, causing pure similarity regression to under-separate fine levels (Genus mAP 0.23 vs. 0.33), whereas pure contrastive loss pushes sibling families apart and erodes coarse Order geometry inside longer prefixes. Combining both losses achieves the best or tied-best accuracy across every dataset and level.

Preservation of Full-Vector Retrieval. Imposing block and prefix structure does not compromise ordinary similarity search over the full 768-dim embedding: same-species / same-topic retrieval mAP

Table 4: Hierarchical retrieval mAP on *unseen* species/topics at each level's designated prefix (AwA2: 8/32/64; iNat & OpenAlex: 16/64/128 for Coarse / Mid / Fine).

	AwA2 (O / F / G)			iNat (O / F / G)			OpenAlex (D / F / S)		
Method	@8	@32	@64	@16	@64	@128	@16	@64	@128
SSE **(ours)**	**0.93**	**0.98**	**0.98**	**0.51**	**0.36**	**0.33**	**0.75**	**0.46**	**0.40**
MRL-style	0.77	0.95	0.97	0.42	0.35	0.32	0.58	0.44	0.39
PCA Prefix	0.63	0.85	0.93	0.27	0.20	0.17	0.53	0.36	0.38

matches or slightly exceeds the frozen encoder across AwA2 (0.82 vs. 0.82), iNat (0.12 vs. 0.11), and OpenAlex (0.43 vs. 0.41), retaining 78–86% of the encoder's exact top-10 neighbors (74% on MovieLens). Consequently, an rVecDB stores *a single* SSE *per tuple* for both structured facet operations and unstructured kNN search.

6.3 Predicates, Ranking, and Text Queries (Q2)

Having verified that individual blocks and prefixes capture their intended semantics, we now ask whether composing them in end-to-end queries, such as filtering on one facet while ranking on another, selecting similarity bands, or specifying predicates in natural language, delivers accurate answers, especially in regimes where explicit labels fail.

Composite Predicates, Ranking, and Range Bands. We run 1,000 composite queries modeled on Example 1.1 (and Q1–Q2 in Section 7): select tuples whose habitat block is close to a reference animal x, then rank the feasible set by Family-level similarity to an independent query animal q (Table 5; similar gains hold on MovieLens text, where composite F1 reaches 0.35 vs. 0.18 for labels). SSE outperforms materialized labels (B1) by +56% **relative in Filter F1** (0.75 vs. 0.48) and +47% **in Ranking NDCG@10** (0.56 vs. 0.38), while cutting cross-facet mixing error more than in half (0.15 vs. 0.35–0.40): when a swan and an otter serve as habitat references, SSE returns consistent aquatic sets despite their distant taxonomies, whereas when a polar bear and a giant panda serve as references, SSE separates arctic from bamboo-forest habitats despite their shared bear family (0.12 error vs. 0.54 for B1). We also evaluate the VLM semantic operator (**B3**, Qwen2.5-VL-7B) on a 50-query subset: pairwise VLM scoring is *both substantially less accurate* than SSE (Filter F1 0.53 vs. 0.75) and *computationally prohibitive*, processing only 16.3 image pairs/s on an RTX 5090 and requiring ∼25 minutes per query over the AwA2 test split compared to milliseconds for SSE.

Table 5 and Figure 6a pinpoint three structural reasons why explicit-attribute pipelines fail on semantic queries:

(1) *Open-world unseen classes:* When the reference x belongs to a species never labeled during training (e.g., seal or giant panda), classifiers must force it into seen label buckets (0.47 F1), whereas SSE projects visual features directly into continuous habitat geometry, retaining nearly all of its seen-species accuracy (0.79 → 0.72).

(2) *Graded similarity and band predicates:* Among pairs whose binary habitat labels *disagree*, SSE still orders true ecological compatibility with ρ=0.89 (vs. 0.59 for B1 and 0.23 for B4). This continuous geometry directly enables interval band predicates (*"moderately similar habitat"*, $\perp_h \le \text{sim} \le \top_h$), nearly doubling F1 over both labels and the plain encoder (0.60 vs. 0.35 / 0.34).

(3) *Calibrated threshold control:* As shown in Figure 6a, tightening τ_h across deciles monotonically raises true habitat similarity from 0.09 to 0.92 (ρ=1.00). By contrast, discrete label overlap oscillates non-monotonically across tie-buckets and caps at 0.76, while the plain encoder stays flat (≈ 0.42) across the bottom five deciles.

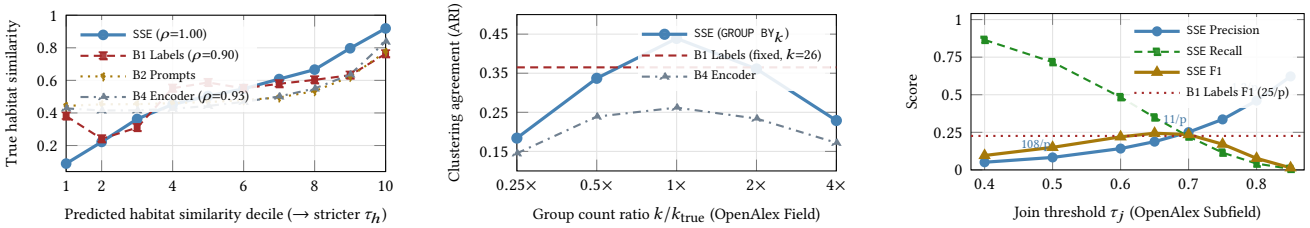
Zero-Shot Cross-Modal Text Predicates. Because CLIP aligns images and text in a shared backbone, an rVecDB user need not supply a reference image at all: a predicate can be specified in natural language. Without training on any text, we encode a species prompt (*"a photo of a polar bear"*), apply a standard mean-offset shift to bridge CLIP's modality gap [20], pass it through the image-trained SSE projector, and query E_{habitat}. As shown in Table 6, zero-shot text references recover nearly the full accuracy of image references (0.71 vs. 0.79 F1 across all 50 species) and outperform raw CLIP text-to-image search by +22 **F1 points** (0.71 vs. 0.49), including on unseen species (0.60 vs. 0.53). Raw CLIP text search retrieves animals that *look like* a polar bear; projecting the text vector into E_{habitat} isolates *where a polar bear lives*, returning whales, seals, and walruses.

6.4 Semantic Grouping and Aggregation (Q3)

Beyond selecting and ranking individual tuples, a relational query engine must partition unlabeled collections into meaningful groups and compute group-level aggregates. We next test whether GROUP BY$_k$, representative summarization (REP), hierarchical drill-down, and cross-facet aggregation work reliably over continuous SSE blocks.

Semantic Grouping and Roll-Up/Drill-Down. We execute GROUP BY$_k$ via spherical k-means on the chosen block or prefix (Section 4.2), and implement hierarchical drill-down by recursively partitioning Order groups into Family and Genus subgroups across AwA2, iNat, and OpenAlex (Table 7). On continuous facets such as AwA2 habitat, grouping by predicted binary label combinations shatters animals into 27 brittle bit-patterns (NMI 0.41, ARI 0.17), whereas GROUP BY$_4$ on E_{habitat} discovers coherent ecological biomes at NMI 0.68 (ARI 0.72, vs. 0.51 / 0.43 for B4). Figure 7 visualizes the output of SELECT REP(g,5), COUNT(*) (Q4 in Section 7): without any stored labels, SSE partitions the 6,000 test images into plains and ground, jungle and bush, marine and arctic, and forest habitats, summarizing each group via five full-embedding medoids and their represented tuple counts.

Across taxonomic and scientific hierarchies, SSE prefixes consistently beat the plain encoder at every granularity (e.g., iNat bird Orders 0.50 vs. 0.41; OpenAlex Fields 0.57 vs. 0.45). Materialized classifiers (B1) are competitive *only* when every test group is a closed-world class seen during training (AwA2 seen Orders: 0.89 vs. 0.78). Whenever data contain *unseen* species or topics, which is the realistic regime for exploratory grouping, classifiers cannot invent new group IDs and collapse (predicting only 49 of 224 unseen bird genera on iNat), whereas SSE forms accurate groups directly from geometry (0.75 **vs.** 0.47 **NMI** on iNat unseen genera; 0.69 **vs.** 0.51 on OpenAlex unseen subfields). Moreover, Figure 6b shows that varying k provides a smooth granularity knob: agreement peaks at k_{true} (OpenAlex Field ARI 0.44 vs. 0.37 for fixed labels and 0.26 for



(a) Threshold τ_h as a continuous dial (AwA2). **(b) Impact of k in `GROUP BY`$_k$ (k_{true}=26).** **(c) `VecJOIN` selectivity vs. precision.**

Figure 6: Continuous query behavior across `rVecDB` operators. (a) Sweeping τ_h across similarity deciles monotonically increases true habitat compatibility from 0.09 to 0.92, whereas predicted labels oscillate and plateau at 0.76 and the plain encoder remains flat across lower deciles. (b) `GROUP BY`$_k$ peaks at the true cardinality and smoothly rolls up ($k < k_{\text{true}}$) or drills down ($k > k_{\text{true}}$), surpassing fixed label buckets. (c) Varying τ_j in `VecJOIN` smoothly trades recall for precision and output size (matches per paper annotated), whereas a label equality join offers only one rigid operating point.

Table 5: Relational-vector query and aggregation benchmark on AwA2 (1,000 composite queries, 300 aggregation queries). Filter (All / Seen / Unseen): F1 of habitat range predicate $E_{\text{habitat}}(t) \approx_{\tau_h} E_{\text{habitat}}(x)$ split by whether reference x belongs to a seen or unseen species; Rank: NDCG@10 of Family-prefix ranking relative to q; Mixing: feasible-set Jaccard error when references share habitat but differ in taxonomy (↓ lower is better); Graded: Spearman ρ among pairs whose binary habitat labels differ; Band: F1 of interval predicate $\perp_h \le \text{sim} \le \top_h$; Agg (AVG / MAX): Spearman ρ of per-Order habitat aggregates (Section 6.4).

	Habitat Filter F1 ↑			Family	Facet	Graded	Band	Order Agg. (ρ) ↑	
Method	All	Seen x	Unseen x	Rank ↑	Mixing ↓	Sim. ↑	Pred. ↑	AVG	MAX
SSE **(ours)**	**0.75**	**0.79**	**0.72**	**0.56**	**0.15**	**0.89**	**0.60**	**0.77**	**0.85**
B1 Labels (materialized)	0.48	0.50	0.47	0.38	0.35	0.59	0.35	0.55	0.66
B2 Prompts (zero-shot)	0.47	0.43	0.49	0.41	0.40	0.21	–	–	–
B4 Encoder (unstructured)	0.55	0.48	0.61	0.39	0.35	0.23	0.34	0.53	0.70
B5 Separate vectors (6 vecs)	0.76	0.79	0.73	0.57	0.15	0.89	0.61	0.77	0.85
B6 CSN (768d masks)	0.75	0.79	0.73	0.57	0.15	0.89	0.60	0.77	0.87

Table 6: Zero-shot cross-modal habitat filtering on AwA2: natural-language text reference vs. image reference.

Query Modality & Space	All (50)	Seen (40)	Unseen (10)
SSE, image reference	**0.79**	**0.81**	**0.73**
SSE, text reference (zero-shot)	**0.71**	**0.73**	**0.60**
Raw CLIP, image reference	0.49	0.47	0.58
Raw CLIP, text reference	0.49	0.48	0.53

Table 7: Semantic `GROUP BY`$_k$ agreement (NMI; k=k_{true}) across graded facets, seen hierarchies, and unseen leaf classes.

	AwA2		iNat (Birds)		OpenAlex	
Method	Habitat	Order	Order	Genus†	Field	Subfield†
SSE **(ours)**	**0.68**	0.78	**0.50**	**0.75**	**0.57**	0.69
B1 Labels (fixed groups)	0.41	**0.89**	0.30	0.47	0.52	0.51
B4 Encoder	0.51	0.53	0.41	0.66	0.45	**0.70**
B5 Separate vectors	0.64	0.80	0.50	0.75	0.57	0.69
B6 CSN	0.65	0.81	0.53	0.75	0.57	0.69

†Evaluated exclusively on species/topics held out from training.

B4), while smaller k rolls up related fields and larger k drills down into high-purity subtopics (AwA2 habitat NMI: 0.48, 0.71, 0.68, 0.63, and 0.60 for k=2, 3, 4, 8, 16).

Cross-Facet Aggregation. The rightmost columns of Table 5 evaluate cross-facet aggregation (Q5 in Section 7): computing per-Order `AVG` and `MAX` habitat similarity to 300 reference animals, where Order membership itself is determined by the 8-dim Order prefix. Remarkably, even though B1 assigns Order membership as accurately as `SSE` (0.94 vs. 0.93), `SSE` ranks Order-level aggregates far more accurately (`AVG` ρ=0.77 vs. 0.55; `MAX` 0.85 vs. 0.66) because continuous habitat scores preserve subtle within-group variation that binary labels erase.

6.5 Cross-Relation Semantic Joins (Q4)

A relational database must also connect tuples across distinct tables. Here we ask whether `VecJOIN` can match related entities across relations at a chosen abstraction level, both when the two relations share a common encoder and in the harder, realistic setting where each relation is embedded by a different model and trained independently.

Multi-Granularity Joins and Semantic Near-Misses. On OpenAlex, we join `Papers` with `Authors` across Domain (4), Field (26), and Subfield (250) prefixes (16/64/128 dims) to answer: *"for each paper, which* 10 *authors work closest to its research topic?"* (ground

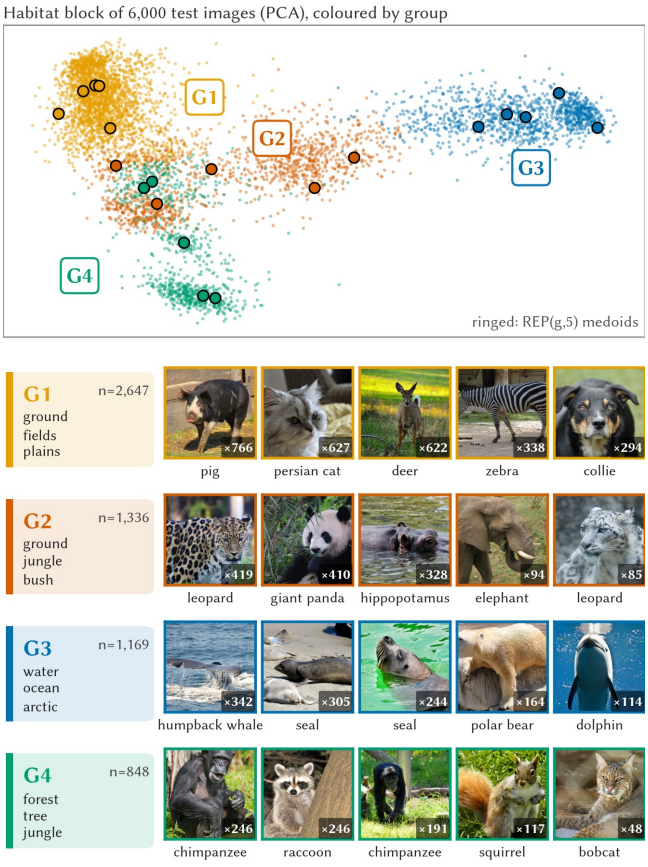


Figure 7: Execution of `SELECT REP(g,5), COUNT(*) FROM Animals GROUP BY`$_4$ E_{habitat} on 6,000 AwA2 test images. Top: 2D PCA projection of E_{habitat} colored by discovered group, with `REP(g,5)` medoids ringed. Bottom: each group (annotated on the left with size and members' top true properties for reference) summarized by five full-embedding medoids and subgroup counts.

truth: authors whose primary publication area matches the paper's area in OpenAlex's taxonomy). We first test homogeneous schema alignment, where an author's SSE is the mean SSE of their other papers (excluding the query paper, with no author supervision). As shown in Table 8 (left), SSE outperforms predicted-label joins and raw SPECTER2 across every granularity: at the 250-class Subfield level, SSE reaches P@10 of 0.36 (40× **random** at 0.009 and 1.8× **predicted labels** at 0.20) and maintains the same lead on unseen topics (0.37 vs. 0.21).

Strict exact-label scoring penalizes scientifically close matches (e.g., matching an AI paper with a Computer Vision researcher counts the same as matching Manual Therapy). We therefore evaluate partial semantic credit at the Subfield level using two independent domain-relatedness keys: *Text* (cosine similarity of `bge-large` embeddings of subfield descriptions, calibrated so identical subfields score 1 and pairs at or below the median score 0, e.g., AI↔Vision 0.32, Oncology↔Cancer Research 0.40) and *Co-activity* (empirical rate at which authors publish in both subfields). Under both

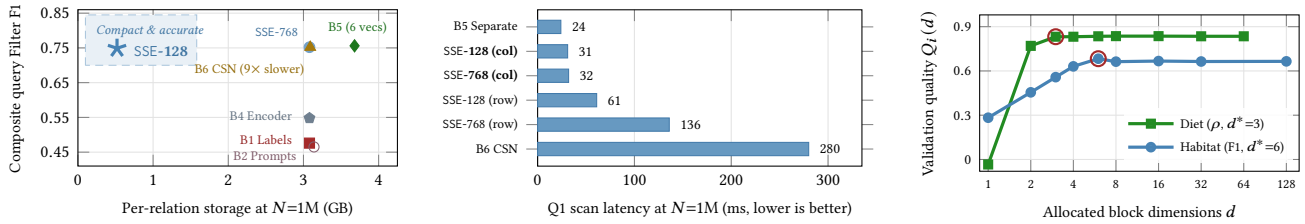
Table 8: `Papers` ⋈ `Authors` `VecJOIN` (P@10) on OpenAlex under (left) shared-encoder aggregation and (right) separately trained SSEs across *different* encoders (SPECTER2 ⋈ `bge-base`) on held-out authors. *Sem.*: Subfield P@10 with text semantic credit.

	Homogeneous Encoder				Cross-Encoder ($\mathcal{L}_{\text{align}}$)			
Method	Dom.	Fld.	Sub.	Sem.	Dom.	Fld.	Sub.	Sem.
SSE **(aligned)**	**0.81**	**0.53**	**0.36**	**0.41**	**0.81**	**0.47**	**0.24**	**0.35**
SSE (no $\mathcal{L}_{\text{align}}$)	–	–	–	–	0.20	0.04	0.01	0.08
B1 Labels (equality)	0.75	0.48	0.20	0.32	0.70	0.41	0.18	0.26
B4 Raw encoders	0.78	0.45	0.30	0.36	0.30	0.04	0.01	0.08
Random	0.27	0.04	0.01	0.07	0.28	0.04	0.01	0.07

keys, SSE preserves a clear margin (0.41 vs. 0.32 for labels on `Papers`⋈`Authors`; 0.62 vs. 0.52 on `Papers`⋈`Papers`), showing that its non-exact matches land overwhelmingly in adjacent research areas.

Heterogeneous Encoders and Shared Join Keys. The decisive test of Section 5.2 occurs when two relations use *different* encoders and train their SSEs separately. Keeping the `Papers` SSE (SPECTER2) fixed as the reference relation S, we embed `Authors` (T) from raw publication-title profiles using a distinct model, `bge-base`, and train the `Authors` SSE on 70% of authors using authorship links as join associations $\mathcal{M}$ in $\mathcal{L}_{\text{align}}$, evaluating on the held-out 30% unseen authors (Table 8, right). Two findings stand out. First, *without explicit coordinate alignment, cross-relation joins fail completely*: an `Authors` SSE trained with the exact same hierarchical losses on its own labels, but omitting $\mathcal{L}_{\text{align}}$, organizes authors well internally, yet its cross-relation join collapses to random chance (0.04 Field, 0.01 Subfield), because pairwise similarity losses are invariant to orthogonal rotations between independently learned spaces. Raw encoders (SPECTER2 vs. `bge-base`) fail identically (0.04). Second, adding $\mathcal{L}_{\text{align}}$ anchors the two schemas into a common geometric frame, boosting held-out join precision by 12× **at Field** (0.47) and 24× **at Subfield** (0.24), surpassing predicted-label equality joins at every level (0.81/0.47/0.24 vs. 0.70/0.41/0.18, and 0.26 vs. 0.18 on unseen topics).

Selectivity–Precision Control via τ_j. Unlike an equality join on predicted labels, which emits a single rigid output (25 authors/paper at Subfield, Precision 0.17, F1 0.23), the threshold τ_j of `VecJOIN` exposes a continuous selectivity–precision trade-off (Figure 6c). Sweeping τ_j from 0.50 to 0.80 smoothly contracts output cardinality from 108 to 1.2 authors/paper while lifting precision from 0.08 to 0.46; at τ_j=0.65–0.70, SSE matches or exceeds the label join's F1 (0.24 vs. 0.23) while materializing **fewer than half as many join pairs** (11 vs. 25 per paper). At coarser levels (Field), while top-k `VecJOIN` clearly beats labels (0.47 vs. 0.41), a single uncalibrated global τ_j achieves lower peak F1 (0.33 vs. 0.41) because similarity neighborhoods vary in density across broad disciplines, pointing to per-query threshold calibration as an open query-optimization problem for `rVecDBs`.



(a) Accuracy vs. storage (N=1M). **(b) Physical layout & scan latency (N=1M).** **(c) Block-size diminishing returns (AwA2).**

Figure 8: System efficiency and schema sizing. (a) SSE-128 uniquely occupies the compact, high-accuracy corner (7.1× smaller than B5 Separate Vectors; +27 F1 points above B1 Labels; B3 VLM requires ~25 min/query at F1 0.53). (b) Fixed schema offsets allow block-columnar storage, cutting scan latency by 2×–4.2× (31–32 ms regardless of residual width) and beating B6 CSN by 9×. (c) Facet quality saturates at d^*=3–6 dimensions (red circles), confirming low intrinsic facet dimensionality.

6.6 System Efficiency and Robustness (Q5)

In Q1–Q4, continuous multi-view baselines (B5 Separate Vectors and B6 CSN) matched SSE in accuracy once upgraded with our training losses. This raises a practical systems question: what does rVecDB gain from organizing everything into a single fixed-block SSE? To answer this, we evaluate physical storage, scan latency, index footprint, block-size selection, and encoder independence.

Physical Layout, Storage, and Query Latency at Scale. First, keeping the 112 facet dimensions fixed on AwA2 and compressing only the residual block from 656 to 144 (SSE-256) or 16 dimensions (SSE-128) leaves facet and composite query accuracy *unchanged* (Filter F1 0.752 → 0.749, Rank NDCG 0.560 → 0.562, Graded ρ 0.889 → 0.889, unseen Order@8 mAP 0.932 → 0.934) while retaining strong full-vector retrieval (mAP 0.80 vs. 0.82).

Table 9 and Figures 8a–8b benchmark physical storage, exact-scan latency of composite query Q1 (habitat filter at 25% selectivity + Family top-10 ranking), and `faiss` HNSW [23] (M=32) facet indexing over N=1,000,000 tuples:

- **Advantage over Separate Vectors (B5):** B5 fragments every tuple across 6 separate vectors (920 floats = 3.68 GB), requiring separate columns and model heads for every facet and hierarchy level. SSE-128 consolidates the entire schema and residual into **one compact vector (**0.52 **GB,** 7.1× **smaller)**. Moreover, because SSE block offsets are fixed by the schema, storing blocks in a *block-columnar* physical layout (zero space overhead) lets Q1 scan only the 32 habitat floats and 32 Family-prefix floats per passing tuple. This cuts scan latency by 2×–4.2× over row-major storage (61 → 31 ms for SSE-128; 136 → 32 ms for SSE-768, making query speed independent of residual width) and matches B5 within 7 ms while building identical 32-dim HNSW facet indices (0.38 GB, 0.02 ms top-10 lookup).
- **Advantage over Masked Embeddings (B6 CSN):** Inspecting CSN's learned masks reveals why soft masking is ill-suited for databases: rather than isolating sparse coordinates, each CSN mask activates 365–435 of the 768 dimensions with ~50% overlap between any two facets. Because coordinates cannot be sliced, every CSN facet evaluation requires a full 768-dim masked inner product, making Q1 9× **slower** (280 vs. 31 ms), and indexing a single facet requires a 768-dim HNSW graph (8.8× **larger**: 3.33 vs. 0.38 GB; 4.7× slower build; 3× slower search). Even pre-materializing CSN's active dimensions per facet balloons relation storage to 11.02 **GB (**21× **larger than** SSE-128**)** and slows scans to 387 ms.

Table 9: System cost at N=1M tuples: relation footprint, Q1 scan latency (median of 50 queries), and habitat HNSW index.

Physical Representation	Vecs /tup	Data (GB)	Q1 Scan (ms)	Index d / GB	Build (s)	Top-10 (ms)
SSE-**128 (block-col.)**	**1**	**0.52**	**31**	**32 / 0.38**	3.8	**0.02**
SSE-768 (block-col.)	1	3.08	32	32 / 0.38	**3.4**	**0.02**
SSE-128 / 768 (row)	1	0.52 / 3.08	61 / 136	32 / 0.38	3.8	0.02
B1 Labels (low acc.)	1	3.08	19	–	–	–
B5 Separate vectors	6	3.68	24	32 / 0.38	3.8	0.02
B6 CSN (masked 768d)	1	3.09	280	768 / 3.33	17.7	0.06
B6 CSN (active mat.)	6	11.02	387	365 / 1.72	10.0	0.04

Intrinsic Facet Dimensionality and Block Allocation. Applying the validation-driven sizing rule of Section 5 (ε=0.01, 5 training species held out for validation) across $d \in \{1, \ldots, 128\}$ reveals steep diminishing returns (Figure 8c): Diet correlation jumps from −0.03 (d=1) to 0.83 at d^*=3 and plateaus completely through d=64, while Habitat Filter F1 rises from 0.28 (d=1) to 0.68 at d^*=6 and remains flat through d=128. Doubling-based exponential search requires nearly half as many training runs as dense grid search (7–8 vs. 11–14) and selects neighboring sizes (d^*=4 and 8) within 0.02 of peak quality. This confirms that individual semantic facets possess low intrinsic dimensionality and leaves ample vector budget for schema evolution.

Robustness Across Foundation Encoders. Finally, replacing CLIP ViT-L/14 with EVA02-CLIP-L/14 [36] on AwA2, with zero changes to schema layout or hyperparameters, reproduces every conclusion: SSE blocks strongly capture Habitat, Diet, and Taxonomy (ρ = 0.93, 0.87, 0.66, vs. 0.29, 0.28, 0.23 for plain EVA and 0.50, 0.57, 0.72 for B1 Labels), composite query accuracy reaches 0.76 Filter F1 and 0.58 NDCG@10 (vs. 0.44 / 0.36 for labels and 0.53 / 0.38 for plain EVA), graded in-between similarity reaches 0.92 (vs. 0.47 for labels), and unseen prefix retrieval reaches 0.95, 0.99, and 0.99 (vs. 0.79 Order for MRL). SSEs thus provide a robust, encoder-agnostic abstraction for relational-vector query processing.

7 ILLUSTRATIVE QUERIES

The following queries illustrate the composition of these operators. We use animal and habitat records with the relevant SSE blocks and assume that blocks compared across relations are aligned. The taxonomy prefixes $E^{(1)}_{\text{tax}}$, $E^{(2)}_{\text{tax}}$, and $E^{(3)}_{\text{tax}}$ represent Order, Family, and Genus, respectively. The SQL-like notation $u \approx_\tau v$ abbreviates $\text{sim}(u, v) \geq \tau$. For grouping, `GROUP BY`$_k$ specifies similarity-based clustering into k groups, and `REP(g,h)` represents group g using up to h representative tuples and their corresponding subgroup sizes, as described above.

Q1: Independent filtering and ranking references. Retrieve animals with habitats similar to that of a swan, ranked by family-level similarity to a panda: `SELECT * FROM Animals AS t WHERE` $E_{\text{habitat}}(t) \approx_{\tau_h} E_{\text{habitat}}(t_{\text{swan}})$ `ORDER BY` $\text{sim}(E^{(2)}_{\text{tax}}(t), E^{(2)}_{\text{tax}}(t_{\text{panda}}))$ `LIMIT 10.` The predicate and ranking use different blocks and different reference objects. Varying τ_h changes the feasible habitat neighborhood while preserving the ranking criterion.

Q2: Composing ranges across facets. Combine an Order-level predicate, a habitat similarity interval, and a conventional attribute condition, then rank by morphology: `SELECT * FROM Animals AS t WHERE` $E^{(1)}_{\text{tax}}(t) \approx_{\tau_t} c_{\text{Carnivora}}$ `AND` $\bot_h \leq \text{sim}(E_{\text{habitat}}(t), E_{\text{habitat}}(t_{\text{swan}})) \leq \top_h$ `AND t.ObservationYear >= 2020 ORDER BY` $\text{sim}(E_{\text{morph}}(t), q_{\text{morph}})$ `LIMIT 10.` The habitat interval can select a band of similarity rather than all values above a lower bound. Each semantic condition uses its own reference and thresholds, independently of the morphology-based ranking.

Q3: Joining through a shared semantic join key. Find animal–park-site pairs whose habitat representations satisfy a selected similarity threshold: `SELECT A.id, P.id FROM Animals AS A VecJOIN ParkSites AS P ON` $E_{\text{habitat}}(A) \approx_{\tau_j} E_{\text{habitat}}(P)$. Each animal supplies the reference for a range over the ParkSites relation. Changing τ_j changes the required degree of similarity.

Q4: Grouping without predefined categories.. Group animals by habitat and represent each group using representative animal images: `SELECT REP(g,5), COUNT(*) FROM Animals AS t GROUP BY`$_k$ $E_{\text{habitat}}(t)$ `AS g.` Each group is summarized by up to five representative images selected using the full embeddings, along with the number of tuples each image represents.

Q5: Aggregation over semantic facets. Measure the average and maximum habitat similarity of animals in the order Carnivora to the habitat of a swan, where $q_h = E_{\text{habitat}}(t_{\text{swan}})$: `SELECT AVG(`$\text{sim}(E_{\text{habitat}}(t), q_h)$`), MAX(`$\text{sim}(E_{\text{habitat}}(t), q_h)$`) FROM Animals AS t WHERE` $E^{(1)}_{\text{tax}}(t) \approx_{\tau_t} c_{\text{Carnivora}}$. Both the predicate and the aggregates are evaluated directly over SSE blocks, without requiring traditional attributes.

Q6: Composing joins, predicates, grouping, and aggregation. Identify animals with habitats compatible with sites in Sub-Saharan Africa, restrict them to the order Carnivora, group them at the Family level, and rank the groups by their average morphological similarity to a query object:

```
SELECT REP(g,5), COUNT(*), AVG(sim(E_morph(t), q_morph)) AS score
FROM ( SELECT DISTINCT A.* FROM Animals AS A VecJOIN ParkSites AS H
ON E_habitat(A) ≈_τj E_habitat(H) WHERE E_geo(H) ≈_τg c_SubSahAfrica AND
E^(1)_tax(A) ≈_τt c_Carnivora ) AS t
GROUP BY_k E^(2)_tax(t) AS g
ORDER BY score DESC LIMIT 10.
```

The habitat block determines join membership, geography and an Order-level prefix determine feasibility, a Family-level prefix determines grouping, and morphology determines the aggregate used for ranking the groups. The result represents each group through representative animal images and their corresponding counts.

8 FUTURE WORK AND FINAL REMARKS

In this paper, we introduced Schema-Structured Embeddings (SSEs) and the Relational-Vector Database (rVecDB) data and query models, together with learning methods and experimental evidence for their use. Developing a full rVecDB system raises several further research problems. We briefly mention some of these problems below.

Index Construction. Range queries and approximate nearest neighbor (ANN) search are the core operations in rVecDBs. Various indices have been developed for these problems, including space-partitioning methods, tree-based data structures, and graph-based indices [26]. Efficiently adapting these indices to constrained similarity search is already a challenging problem in conventional vector databases [19]. In rVecDBs, the problem becomes more challenging since various database operations are modeled as similarity queries across different possibly independent subspaces. Moreover, the query point in the ranking space may correspond to an object that does not satisfy any of the filtering predicates. Designing indices that exploit the block and hierarchical structure of SSEs to support efficient search under such geometric constraints is an important algorithmic research direction.

Semantic Join Processing. Vector similarity joins have been studied in prior work, including recent efforts such as DiskJoin [4] that enable large-scale similarity joins on disk-resident vectors. Extending these techniques to rVecDBs is an interesting research direction. In particular, join keys in rVecDBs may be specified over different blocks or hierarchical prefixes, and a join may involve several semantic predicates with different similarity thresholds. Unlike ANN search, a semantic join may require reporting all matching pairs, as missing pairs may affect subsequent database operations such as grouping and aggregation. Developing scalable join algorithms with explicit guarantees on the completeness of the join results is therefore of particular interest.

Parameter Tuning and Query Optimization. Queries in rVecDBs introduce several parameters, including similarity thresholds for predicates and joins, and the number of groups for semantic grouping. Specifying appropriate values for these parameters may not be straightforward for users. For example, the same similarity threshold may yield substantially different join cardinalities across facets or reference tuples. Automatically tuning these parameters based on user-specified requirements, such as result cardinality or semantic accuracy, is an interesting research problem. Moreover, once the query parameters are fixed, estimating the selectivity of geometric predicates and the cost of alternative execution plans is central to query optimization in rVecDBs.

Query Accuracy. Unlike traditional attributes, the attribute values in SSEs are learned and may not perfectly reflect the intended semantics. Moreover, approximate query processing introduces another source of error. These errors may propagate through a

sequence of database operations. Quantifying and controlling the accuracy of complex queries over SSEs is an interesting direction for future work.

In this paper, we provided a common basis for addressing research challenges that enable relational-vector databases at scale. We hope this work supports further development of the algorithmic and systems foundations of rVecDBs, where learned semantic structure informs both data modeling and query processing.

ACKNOWLEDGMENTS

This project was supported in part by the NSF grant 2348919, CloudBank computation (NSF ACCESS allocation CIS251215), and an Amazon Research Award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the supporting organizations.

REFERENCES

[1] Shuai Bai, Keqin Chen, Xuejing Liu, et al. 2025. Qwen2.5-VL Technical Report. *arXiv preprint arXiv:2502.13923* (2025).

[2] Usha Bhalla, Alex Oesterling, Suraj Srinivas, Flavio P. Calmon, and Himabindu Lakkaraju. 2024. Interpreting CLIP with Sparse Linear Concept Embeddings (SpLiCE). In *Advances in Neural Information Processing Systems*, Vol. 37. https://doi.org/10.52202/079017-2678

[3] Yuzheng Cai, Jiayang Shi, Yizhuo Chen, and Weiguo Zheng. 2024. Navigating Labels and Vectors: A Unified Approach to Filtered Approximate Nearest Neighbor Search. *Proceedings of the ACM on Management of Data* 2, 6 (2024), 1–27. https://doi.org/10.1145/3698822

[4] Yanqi Chen, Xiao Yan, Alexandra Meliou, and Eric Lo. 2025. DiskJoin: Large-scale Vector Similarity Join with SSD. *Proceedings of the ACM on Management of Data* 3, 6 (2025), 1–27.

[5] Yannis Chronis, Helena Caminal, Yannis Papakonstantinou, Fatma Özcan, and Anastasia Ailamaki. 2025. Filtered Vector Search: State-of-the-art and Research Opportunities. *Proceedings of the VLDB Endowment* 18, 12 (2025), 5488–5492. https://doi.org/10.14778/3750601.3750700

[6] Eric D. Crandall, Rachel H. Toczydlowski, Libby Liggins, Ann E. Holmes, Maryam Ghoojaei, Michelle R. Gaither, Briana E. Wham, Andrea L. Pritt, Cory Noble, Tanner J. Anderson, Randi L. Barton, Justin T. Berg, Sofia G. Beskid, Alonso Delgado, Emily Farrell, Nan Himmelsbach, Samantha R. Queeno, Thienthanh Trinh, Courtney Weyand, Andrew Bentley, John Deck, Cynthia Riginos, Gideon S. Bradburd, and Robert J. Toonen. 2023. Importance of Timely Metadata Curation to the Global Surveillance of Genetic Diversity. *Conservation Biology* 37, 4 (2023), e14061. https://doi.org/10.1111/cobi.14061

[7] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic Entailment Cones for Learning Hierarchical Embeddings. In *Proceedings of the 35th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research)*, Vol. 80. PMLR, 1646–1655.

[8] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. 2023. ImageBind one embedding space to bind them all. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 15180–15190.

[9] Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, Amit Singh, and Harsha Vardhan Simhadri. 2023. Filtered-DiskANN: Graph Algorithms for Approximate Nearest Neighbor Search with Filters. In *Proceedings of the ACM Web Conference 2023*. Association for Computing Machinery, 3406–3416. https://doi.org/10.1145/3543507.3583552

[10] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2015), 19:1–19:19.

[11] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *International Conference on Learning Representations (ICLR)*.

[12] Alexa Iancu, Fatemeh Rahimi, Thomas Ganslandt, and Christian Gulden. 2026. Assessing Imaging Metadata Quality in Radiology: Investigating Discrepancies Between PACS and RIS Data. *Studies in Health Technology and Informatics* 336 (2026), 1420–1424. https://doi.org/10.3233/SHTI260443

[13] Daniel Kang, John Guibas, Peter D. Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2022. TASTI: Semantic Indexes for Machine Learning-based Queries over Unstructured Data. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22)*. ACM, 1934–1947. https://doi.org/10.1145/3514221.3517897

[14] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised Contrastive Learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*. 18661–18673.

[15] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. 2020. Concept Bottleneck Models. In *Proceedings of the 37th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research)*, Vol. 119. PMLR, 5338–5348.

[16] Weize Kong, Swaraj Khadanga, Cheng Li, Shaleen Kumar Gupta, Mingyang Zhang, Wensong Xu, and Michael Bendersky. 2022. Multi-Aspect Dense Retrieval. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 3178–3186. https://doi.org/10.1145/3534678.3539137

[17] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. 2022. Matryoshka Representation Learning. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*.

[18] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*. 9459–9474.

[19] Mocheng Li, Xiao Yan, Baotong Lu, Yue Zhang, James Cheng, and Chenhao Ma. 2025. Attribute filtering in approximate nearest neighbor search: An in-depth experimental study. *Proceedings of the ACM on Management of Data* 3, 6 (2025), 1–26.

[20] Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Zou. 2022. Mind the Gap: Understanding the Modality Gap in Multi-modal Contrastive Representation Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[21] Yanjun Lin, Kai Zhang, Zhenying He, Yinan Jing, and X Sean Wang. 2025. Survey of filtered approximate nearest neighbor search over the vector-scalar hybrid data. *arXiv preprint arXiv:2505.06501* (2025).

[22] Yao Lu, Aakanksha Chowdhery, Srikanth Kandula, and Surajit Chaudhuri. 2018. Accelerating Machine Learning Inference with Probabilistic Predicates. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18)*. ACM, 1493–1508. https://doi.org/10.1145/3183713.3183751

[23] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[24] Milvus. [n.d.]. Multi-Vector Hybrid Search. Milvus Documentation. https://milvus.io/docs/multi-vector-search.md Accessed: November 5, 2026.

[25] Maximilian Nickel and Douwe Kiela. 2017. Poincaré Embeddings for Learning Hierarchical Representations. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. 6338–6347.

[26] James Jie Pan, Jianguo Wang, and Guoliang Li. 2024. Survey of vector database management systems: J. Jie et al. *The VLDB Journal* 33, 5 (2024), 1591–1615.

[27] Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. 2025. Semantic Operators and Their Optimization: Enabling LLM-Based Data Processing with Accuracy Guarantees in LOTUS. *Proceedings of the VLDB Endowment* 18, 11 (2025), 4171–4184. https://doi.org/10.14778/3749646.3749685

[28] Liana Patel, Peter Kraft, Carlos Guestrin, and Matei Zaharia. 2024. ACORN: Performant and Predicate-Agnostic Search Over Vector Embeddings and Structured Data. *Proceedings of the ACM on Management of Data* 2, 3, Article 120 (2024), 27 pages. https://doi.org/10.1145/3654923

[29] pgvector Contributors. [n.d.]. pgvector: Open-Source Vector Similarity Search for Postgres. GitHub repository. https://github.com/pgvector/pgvector Accessed: November 5, 2026.

[30] Jason Priem, Heather Piwowar, and Richard Orr. 2022. OpenAlex: A fully-open index of scholarly works, authors, venues, institutions, and concepts. *arXiv preprint arXiv:2205.01833* (2022).

[31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*. 8748–8763.

[32] Dominique G. Roche, Loeske E. B. Kruuk, Robert Lanfear, and Sandra A. Binning. 2015. Public Data Archiving in Ecology and Evolution: How Well Are We Doing? *PLOS Biology* 13, 11 (2015), e1002295. https://doi.org/10.1371/journal.pbio.1002295

[33] Lütfi Kerem Şenel, İhsan Utlu, Furkan Şahinuç, Haldun M. Ozaktas, and Aykut Koç. 2021. Imparting interpretability to word embeddings while preserving semantic structure. *Natural Language Engineering* 27, 6 (2021), 721–746. https://doi.org/10.1017/S1351324920000315

[34] Amanpreet Singh, Mike D'Arcy, Arman Cohan, Doug Downey, and Sergey Feldman. 2023. SciRepEval: A Multi-Format Benchmark for Scientific Document Representations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[35] Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard Hovy. 2018. SPINE: SParse Interpretable Neural Embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32. https://doi.org/10.1609/aaai.v32i1.11935

[36] Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao. 2023. EVA-CLIP: Improved Training Techniques for CLIP at Scale. *arXiv preprint arXiv:2303.15389* (2023).

[37] Xiaojie Sun, Keping Bi, Jiafeng Guo, Sihui Yang, Qishen Zhang, Zhongyi Liu, Guannan Zhang, and Xueqi Cheng. 2024. A Multi-Granularity-Aware Aspect Learning Model for Multi-Aspect Dense Retrieval. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. Association for Computing Machinery, 674–682. https://doi.org/10.1145/3616855.3635770

[38] Matthieu Tehenan. 2025. Semantic Geometry of Sentence Embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2025*. Association for Computational Linguistics, 11993–12004. https://doi.org/10.18653/v1/2025.findings-emnlp.641

[39] Frederick Tung and Greg Mori. 2019. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1365–1374.

[40] Grant Van Horn, Elijah Cole, Sara Beery, Kimberly Wilber, Serge Belongie, and Oisin Mac Aodha. 2021. Benchmarking Representation Learning for Natural World Image Collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12884–12893.

[41] Andreas Veit, Serge Belongie, and Theofanis Karaletsos. 2017. Conditional Similarity Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 830–838. https://doi.org/10.1109/CVPR.2017.193

[42] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-Embeddings of Images and Language. In *International Conference on Learning Representations (ICLR)*.

[43] Jesse Vig, Shilad Sen, and John Riedl. 2012. The Tag Genome: Encoding Community Knowledge to Support Novel Interaction. *ACM Transactions on Interactive Intelligent Systems* 2, 3 (2012), 13:1–13:44.

[44] 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 '21)*. ACM, 2614–2627. https://doi.org/10.1145/3448016.3457550

[45] Mengzhao Wang, Lingwei Lv, Xiaoliang Xu, Yuxiang Wang, Qiang Yue, and Jiongkang Ni. 2023. An Efficient and Robust Framework for Approximate Nearest Neighbor Search with Attribute Constraint. In *Advances in Neural Information Processing Systems*, Vol. 36. 15738–15751. https://doi.org/10.52202/075280-0692

[46] Yile Wang, Zhanyu Shen, and Hui Huang. 2025. LDIR: Low-Dimensional Dense and Interpretable Text Embeddings with Relative Representations. In *Findings of the Association for Computational Linguistics: ACL 2025*. Association for Computational Linguistics, 14397–14409. https://doi.org/10.18653/v1/2025.findings-acl.743

[47] 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. *Proceedings of the VLDB Endowment* 13, 12 (2020), 3152–3165. https://doi.org/10.14778/3415478.3415541

[48] Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. 2019. Zero-Shot Learning – A Comprehensive Evaluation of the Good, the Bad and the Ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 9 (2019), 2251–2265.

[49] Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-Pack: Packed Resources For General Chinese Embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 641–649.

[50] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. 2022. Lit: Zero-shot transfer with locked-image text tuning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 18102–18112.

[51] Chaoji Zuo, Miao Qiao, Wenchao Zhou, Feifei Li, and Dong Deng. 2024. SeRF: Segment Graph for Range-Filtering Approximate Nearest Neighbor Search. *Proceedings of the ACM on Management of Data* 2, 1, Article 69 (2024), 26 pages. https://doi.org/10.1145/3639324