PROSE: Provenance-Aware Hybrid Retrieval over Enterprise Schema Catalogs

Alex Liu*
Korso, Inc.
United States
alex@korsoai.com

Daichi Hiraoka*
Korso, Inc.
United States
daichi@korsoai.com

Martin Pan*
Korso, Inc.
United States
martin@korsoai.com

Abstract

Enterprise agents fail when they cannot map natural-language intents to executable enterprise schema elements. In multi-ERP settings, this grounding step is hard due to metadata heterogeneity, source conflicts, multilingual labels, and tenant-specific vocabulary. We present PROSE, a systems architecture for schema grounding that combines (i) an eight-entity canonical schema catalog with explicit provenance, (ii) hybrid lexical-semantic retrieval with reciprocal rank fusion and alias-aware score shaping, and (iii) deterministic source-precedence merge policies. PROSE is implemented on a multi-tenant NestJS/PostgreSQL stack with pgvector and row-level security. The retrieval path fuses BM25 and vector candidates, supports language fallback, and performs provenance-aware canonicalization. We report implementation-grounded verification evidence: 7 test suites and 252 passing tests covering retrieval, merge logic, quality gates, and adapter conformance. These counts measure implementation conformance to the architecture's stated invariants; they are not a measurement of retrieval quality. We also specify a pre-registered benchmark protocol for Recall@10, MRR@10, and nDCG@10 with multilingual and alias-heavy slices, and the harness that implements it. **This manuscript reports no retrieval-quality results.** The judged query set the protocol requires has not been built, and no Recall, MRR, or nDCG value appears anywhere in this paper. Section 7 therefore states a measurement plan that has not been executed, not an evaluation that has; readers should treat the retrieval hypotheses as specified and falsifiable rather than as supported. What this paper offers is an architecture, a determinism argument for its merge procedure, executed implementation verification, and a pre-registered evaluation protocol. The contribution is a deployable information-management system, not a new embedding model.

CCS Concepts

• **Information systems** → **Information extraction**; *Data mining*; *Database query processing*.

Keywords

schema retrieval, enterprise search, reciprocal rank fusion, provenance-aware ranking, ERP integration, multi-tenant knowledge systems

1 Introduction

Enterprise AI agents increasingly depend on operational data systems rather than static document corpora. For these agents, the critical first step is *schema grounding*: mapping a natural-language intent (for example, "overdue purchase order amount") to concrete provider entities and fields. Every downstream capability (query generation, tool invocation, workflow execution, write-back) is conditioned on this mapping being correct. A grounding error is not a ranking inconvenience; it silently changes the semantics of the action the agent then performs. The difficulty is easy to underestimate, because the target space looks like a scale problem and is not one. A single SAP S/4HANA installation exposes far more dictionary tables and CDS views than any agent prompt can hold, as does an Odoo tenant's model list, but the operative difficulty lies elsewhere: the *descriptions* of those targets are redundant, inconsistent, incomplete, and unequally trustworthy at the same time. In real enterprise settings this task is difficult for three reasons.

(1) **Metadata heterogeneity.** The same business concept is represented differently across ERPs (SAP, Odoo, others), and even within one ERP channel (SQL dictionary vs. OData service metadata).
(2) **Source conflict and trust asymmetry.** Labels can disagree across authoritative and heuristic sources. Selecting by score alone may surface low-authority fallback labels over high-authority system labels.
(3) **Tenant language and vocabulary drift.** Users query in different languages and tenant-specific business terms ("client", "plant", local abbreviations) that may not appear in canonical ERP metadata.

These constraints make naive semantic search or lexical-only schema matching brittle. They also expose a gap between many research prototypes and operational requirements: deterministic provenance handling, tenant isolation, and graceful degradation when one retrieval branch fails.

To make the gap concrete, consider a German-speaking buyer who asks for "*lieferant konto*" (vendor account). A dense retriever trained on general text may rank an English-labelled customer table above the correct vendor master, because the multilingual signal is weak and "account" is lexically overloaded. A lexical retriever fails differently: under strict conjunctive matching the query returns nothing at all, because no single catalog label contains both tokens. Suppose both branches are fixed and the correct target is retrieved. A third failure now becomes visible: the label the system *shows* the agent may come from a heuristic knowledge-file enrichment with locally high confidence rather than from the authoritative SAP data-dictionary text table, so the agent reasons over a plausible-sounding but non-canonical field name. These three failures (semantic drift, lexical collapse, and provenance inversion) are independent, and a system that fixes only one of them is not usable.

We present PROSE (**Pro**venance-aware **S**chema r**E**trieval), a deployed architecture for schema grounding in a multi-ERP agent

*All authors contributed equally as first authors.

platform. The contribution is not a new embedding model. It is a systems design that tightly couples hybrid retrieval with explicit provenance semantics and operational safeguards.

Our main contributions are:

C1: Canonical, provenance-first schema representation. We formalize an eight-entity schema catalog (`models`, `fields`, `relationships`, `capabilities`, `enum_values`, `labels`, `aliases`, `ingestion_runs`) with source and confidence contracts on enrichable entities (§4).

C2: Hybrid retrieval with deterministic fusion and tenant vocabulary control. We combine BM25 and pgvector candidates via reciprocal rank fusion (RRF) [5], inject alias-aware score boosts, and preserve multilingual fallback behavior at query time (§5.2).

C3: Deterministic provenance merge ladders. Instead of single-score arbitration, we apply source precedence ladders for labels, capabilities, and relationships, with confidence only as a secondary tie-breaker (§5.3).

C4: Operational verifiability. We implement explicit quality gates and adapter conformance checks, plus fail-open/fail-safe retrieval paths (for example BM25 fallback when embeddings or vector operators fail), and report code-level verification evidence (§6, §7.1).

C5: A pre-registered evaluation protocol and runnable harness. We define the query-to-schema retrieval task, its qrels construction procedure, its metric definitions, and a five-step ablation ladder in advance of measurement, and we have implemented an executable harness that computes the metrics (§7). The harness is not distributed with this manuscript; it is intended to accompany a subsequent code release, and §7 specifies it in enough detail to be reimplemented from the paper alone.

1.1 Research Contributions vs. Systems Contributions

This paper makes two kinds of claim, and they rest on different evidence. We separate them here so that neither is read as support for the other.

A *systems* contribution asserts that a particular arrangement of known components is buildable and operable under stated constraints and that it holds stated invariants; its evidence is design rationale, those invariants, and executed verification. A *research* contribution asserts that a stated hypothesis about retrieval behavior holds; its evidence is measurement against baselines on a defined task. PROSE makes contributions of both kinds.

C1, C3, and C4 are systems contributions. Their evidence is the canonical model and its invariants (§4), the determinism argument for the merge procedure (§5.3, Prop. 5.1), and executed verification suites (§7.1). These claims are checkable by inspection rather than by a leaderboard: determinism, tenant isolation, and graceful degradation are stated precisely enough to be falsified.

C2 and C5 are research contributions, and we state their hypotheses explicitly so that the protocol in §7 can refute them:

- **H1 (fusion).** On enterprise schema-target retrieval, rank-based fusion of a lexical and a dense branch dominates either branch alone, measured by nDCG@10.
- **H2 (alias shaping).** A constant additive boost applied to tenant-alias matches improves nDCG@10 on the alias-heavy slice without degrading it on the metadata-rich slice.
- **H3 (provenance ordering).** Resolving surfaced labels by source ladder rather than by confidence changes the top-1 surfaced label on a measurable fraction of queries, and the ladder ordering is preferred by annotators on those queries.

This manuscript reports the systems evidence and reports no measurement of H1–H3. The judged query set required to test them has not been built, so no retrieval metric, ablation, or significance result appears anywhere in this paper. We report the absence rather than substitute a plausible number for a measured one, and §9 states what this costs the paper. §7 states the protocol under which H1–H3 will be tested; C5 is a claim about that protocol's precision, not about its outcome.

2 Problem Formulation

Given a natural-language query q, tenant t, provider family p, and language preference tuple $L = (\ell_{user}, \ell_{tenant}, \mathsf{en})$, we need to return a ranked list of schema targets $R = \{(m_i, f_i)\}_{i=1}^{k}$ such that:

(1) **Relevance**: R contains high-likelihood mappings for user intent.
(2) **Trust consistency**: canonical labels prioritize high-authority sources.
(3) **Tenant adaptation**: tenant vocabulary can affect ranking without mutating canonical source truth.
(4) **Operational robustness**: failures in one retrieval branch do not collapse the tool path.

This differs from generic document retrieval because candidate identity is schema-keyed ($provider\_model, provider\_field$), and post-retrieval canonicalization must obey source governance.

We make the object of retrieval precise. Let $\mathcal{T}_{t,p}$ be the set of schema targets visible to tenant t for provider family p. A target is either a model-level target $(m, \varnothing)$ or a field-level target (m, f). Each target carries a set of *label records* $\Lambda(m, f) = \{(\text{text}, \ell, \sigma, c)\}$, where ℓ is a language tag, σ is a source class, and $c \in [0, 1]$ is a source-local confidence. Retrieval is therefore a two-stage problem rather than one:

Stage 1 (ranking).
Produce an ordering over $\mathcal{T}_{t,p}$ from evidence spread across many label records per target.

Stage 2 (canonicalization).
For each returned target, choose exactly one label record to surface to the agent.

Standard ad-hoc retrieval collapses these stages, because a document is its own canonical representation. Here they must be separated, and this separation is the source of most of the design in §5. Ranking wants *all* evidence, including low-authority enrichments and tenant slang, because more surface forms means higher recall. Canonicalization wants only *authoritative* evidence, because the surfaced string is what the agent will treat as the field's meaning. Optimizing a single score for both objectives produces the provenance inversion described in §1: a heuristic label with locally high confidence wins arbitration and becomes the agent's ground truth.

Two further properties belong to the problem statement rather than to the solution, because an architecture that satisfies (1)–(4) while violating either one is not deployable. First, *isolation*: no query issued in the context of tenant t may observe a target or label record belonging to $t' \neq t$. Second, *availability under partial failure*: the grounding call is on the critical path of an interactive agent, so a failure in the embedding service must degrade result quality rather than return an error.

3 Related Work

Schema matching and integration. Classical schema matching surveys highlight the challenge of structural and semantic heterogeneity across enterprise systems [26]. Our work is aligned with this tradition but targets agent-time grounding with online retrieval constraints rather than offline one-shot schema alignment. Cupid [19] and the ten-year retrospective of generic matching [1] establish the standard decomposition into linguistic, structural, and constraint-based matchers, and the data integration literature formalizes mediated schemas and mappings over them [6]. Two differences from that line matter here. First, schema matching produces a *correspondence* between two schemas and is evaluated as a matching problem, whereas we produce a *ranking* over targets in response to an ad-hoc natural-language query and are evaluated as a retrieval problem. Second, schema matching typically assumes a single trusted description per element, whereas our central difficulty is that one element carries many descriptions of differing authority.

Text-to-SQL and schema linking. Schema linking, the alignment of question tokens to tables and columns, is a well-studied subproblem of text-to-SQL, from Seq2SQL [35] and Spider [32] through relation-aware encoders [31] and analyses of how much of parser accuracy schema linking actually explains [14]. LLM-era systems make the dependency starker: decomposed prompting [24] and benchmark studies on large, dirty databases [8, 16] both report that retrieving the right subset of schema into the prompt is a dominant error source. PROSE addresses exactly this retrieval step, but decoupled from any particular parser and under two constraints those systems do not carry: targets come from multiple heterogeneous providers rather than one database, and the surfaced description of a target must be provenance-governed because it is consumed by a downstream generative model.

Hybrid sparse-dense retrieval and rank fusion. BM25 remains a strong lexical baseline [27], while ANN vector search via HNSW enables scalable semantic retrieval [20]. RRF has been repeatedly shown to be robust for combining rankers without delicate calibration [5]. We adopt these primitives but add provenance-aware post-resolution and tenant alias semantics specific to enterprise schema discovery. Dense dual encoders are competitive on open-domain tasks [13], and subsequent analysis of fusion functions for hybrid retrieval characterizes when rank-based fusion should be preferred to score normalization [2]. Zero-shot benchmarks such as BEIR [30] and reproducibility toolkits such as Pyserini [18] established the methodological expectation that hybrid systems be reported against both single-branch baselines; the protocol in §7 follows that convention. To our knowledge the interaction between rank fusion and a downstream deterministic canonicalization step has not been studied; it is the subject of H3.

Tool and API grounding for LLM agents. Retrieval-augmented generation [15] and tool-using models [28] both reduce to a retrieval problem when the tool or API space is large: Gorilla [23] and ToolLLM [25] retrieve over thousands of API specifications before generation. Schema grounding is the same shape of problem with a harsher failure mode. An incorrectly retrieved API usually produces a visible error; an incorrectly retrieved schema field produces a syntactically valid query with wrong semantics. This asymmetry is why we invest in deterministic canonicalization rather than relying on the language model to notice that a label is untrustworthy.

Data catalogs, discovery, and provenance. Enterprise metadata management systems such as Goods [9], Aurum [7], and Data Tamer [29] address dataset-level discovery and curation at scale, and table-oriented discovery work targets union and join relatedness over lakes [22, 34]. Semantic type detection [11] and pretrained entity matching [17] supply the kind of enrichment signal our catalog ingests. Provenance has a precise database-theoretic foundation [3, 4] and a standard interchange model [21]. Our use of provenance is deliberately narrower. We do not track derivation lineage for query results; we attach a source class and a source-local confidence to each label record, and use a total order over source classes as the arbitration rule. The novelty we claim is not the provenance model but its placement: inside the retrieval path, as the canonicalization step, rather than alongside it as offline lineage metadata.

Knowledge systems in production. Knowledge graphs are now common enterprise infrastructure [10], and compound AI systems emphasize orchestration over single-model improvements [33]. Our contribution sits at this systems boundary: robust schema grounding as a prerequisite for downstream KG or workflow reasoning.

4 System Overview

4.1 Catalog Architecture

PROSE separates provider-specific extraction from shared retrieval and merge logic. Provider adapters (for example SAP HANA/OData, Odoo) emit canonical entities; downstream persistence and search are ERP-agnostic.

The layering in Figure 1 encodes one architectural rule: *no provider-specific logic exists below the adapter boundary.* Adapters may know that SAP dictionary text tables are named `DD03T` or that Odoo exposes `ir.model.fields`; the persistence, retrieval, merge, and tool layers may not. Provider knowledge enters the lower layers only as data: as the `source` value on a label record, and as an entry in the source precedence order of §5.3. This is what makes onboarding a new ERP an additive change: a new adapter plus new entries in the precedence order, with no change to retrieval.

4.2 Canonical Data Model

The core schema catalog uses eight entities: `schema_models`, `schema_fields`, `schema_relationships`, `schema_capabilities`, `schema_enum_values`, `schema_labels`, `schema_aliases`, and

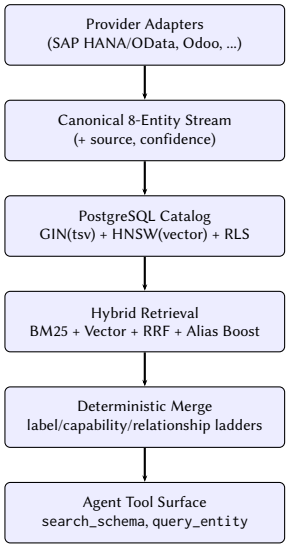

Figure 1: PROSE pipeline from provider extraction to agent-time schema grounding.

`schema_ingestion_runs`. This representation evolved from a prior flat catalog row into an explicit entity split.

The `schema_labels` table stores both a generated `tsvector` and a `vector(1536)` embedding, enabling lexical and semantic retrieval on the same logical row. All schema tables are tenant-isolated with RLS, and every policy is predicated on a tenant identifier held in a session-scoped configuration setting that the application establishes per connection.

The entity split is what allows Stage 1 and Stage 2 of §2 to be served by different tables. `schema_models` and `schema_fields` carry structural identity; `schema_labels` carries the many-to-one descriptive evidence used for ranking and canonicalization; `schema_aliases` carries tenant-scoped surface forms that must influence ranking but must never be surfaced as a canonical label. `schema_capabilities` records what an adapter can actually do with a target (filterable, sortable, writeable, and similar), which the agent needs in order to know whether a correctly grounded field is usable. `schema_enum_values` records closed domains, and `schema_ingestion_runs` records the provenance of the ingestion event itself, so that any row can be traced to the run that produced it.

4.3 Provenance Contract

Every enrichable row carries:

- `source`: origin class (for example `DD03T`, `OData_sap:label`, `knowledge_file`, `tenant_glossary`),
- `confidence`: source-local confidence in $[0, 1]$.

Importantly, confidence is not globally dominant. Selection first respects source ladders, then confidence, then stable input order. This prevents low-authority rows with high local confidence from overriding high-authority system metadata.

The reason confidence cannot be globally dominant is that confidences from different sources are not commensurable. A dictionary text table reports, in effect, "this is the label," and any confidence attached to it is an artifact of our ingestion. A heuristic enrichment reports "I believe this is the label with probability c," where c is calibrated, if at all, against that heuristic's own distribution. Comparing the two numerically is a category error, and one that fires often: heuristics are most confident where they are most fluent, not where they are most correct. Ordering by source class first makes the comparison well-posed: confidence is only ever compared *within* a source class, where it is meaningful.

5 Retrieval and Merge Methods

5.1 Hybrid Candidate Generation

Given query q, provider p, and language tuple $(\ell_u, \ell_t, \texttt{en})$, we generate:

- lexical top-k candidates from `tsv @@ plainto_tsquery`,
- vector top-k candidates from cosine distance over `embedding`.

In the current implementation $k = 20$ per branch. For long natural-language prompts we add a bounded OR-tsquery fallback to avoid strict-AND failure collapse.

The OR-fallback addresses the lexical collapse described in §1. PostgreSQL's `plainto_tsquery` conjoins terms, so a verbose intent ("show me the vendor account number on open purchase orders") matches no single catalog label and the lexical branch contributes nothing at all. Rather than always disjoining, which floods the pool with weak matches, we retry with a bounded `to_tsquery` disjunction only when the conjunctive query returns fewer results than the branch limit. The bound matters: an unbounded disjunction over a long query makes the lexical branch's rank ordering nearly uninformative, which in turn makes its RRF contribution noise.

5.2 Reciprocal Rank Fusion and Alias-Aware Reweighting

For each grouped key (m, f):

$$s_{\text{rrf}}(m, f) = \frac{\mathbf{1}[r_b \neq \varnothing]}{K + r_b} + \frac{\mathbf{1}[r_v \neq \varnothing]}{K + r_v}, \tag{1}$$

with $K = 60$, BM25 rank r_b, and vector rank r_v.

If any tenant alias row matches the target key, a constant boost is applied:

$$s(m, f) = s_{\text{rrf}}(m, f) + \lambda \cdot \mathbf{1}[\text{alias\_match}], \tag{2}$$

where $\lambda = 0.5$ in the current system. This enables tenant vocabulary injection without rewriting canonical labels.

Two aspects of Eq. (1) are load-bearing for this application. First, it consumes *ranks*, not scores. BM25 scores and cosine similarities live on incomparable scales, and their distributions shift with tenant catalog size and with embedding model version; any score-normalizing fusion therefore requires per-tenant recalibration that we would have to maintain across model upgrades [2]. Rank-based fusion is invariant to both. Second, the indicator terms mean a target present in only one branch is scored on that branch alone rather than penalized by a sentinel rank. This is what makes single-branch degradation graceful: if the vector branch returns nothing, Eq. (1) reduces exactly to a lexical-only ranking of the surviving pool, with no discontinuity in the scoring function.

Grouping requires care. Candidates are label records, but the unit of retrieval is a target, so several records may map to the same (m, f). We take the best (smallest) rank per branch per target rather than summing over records, so that a target with many redundant labels is not advantaged over a target with one authoritative label. Without this, targets in well-enriched provider areas would systematically outrank equally relevant targets in sparsely enriched ones, an artifact of enrichment coverage rather than of relevance.

The additive form of Eq. (2) is chosen over multiplicative shaping for interpretability under fusion. Since RRF contributions are bounded by $1/(K+1)$ per branch, a constant λ has a directly readable meaning: with $K = 60$, $\lambda = 0.5$ exceeds the maximum attainable s_{rrf} value, so an alias match promotes a target above all non-alias targets while preserving the relative order *within* the alias-matched set and *within* the non-matched set. λ is thus not a tuned weight but a switch between two regimes, and it is the parameter H2 is designed to test. The current value was chosen for this property rather than fitted to data; §9 treats that as a limitation.

5.3 Deterministic Provenance Merge

Rows are then resolved by source ladders. For labels, the current precedence is:

$$\texttt{DD03T} > \texttt{DD02T} > \texttt{CDS\_annotation} > \texttt{OData\_sap:label} > \texttt{DD04T} > \texttt{CDS} > \texttt{DDIC} > \texttt{knowledge\_file}. \tag{3}$$

Tenant glossary entries are represented as aliases and do not overwrite canonical labels.

Two further ladders govern the other enrichable entities:

$$\texttt{OData\_\$metadata} > \texttt{catalogservice} > \texttt{probe} > \texttt{knowledge\_file}, \tag{4}$$

$$\texttt{DDIC} > \texttt{OData\_NavigationProperty} > \texttt{CDS} > \texttt{inferred}, \tag{5}$$

for capabilities and relationships respectively. The shape is the same in all three cases and reflects a single principle: prefer the artifact that the source system itself uses to operate, then declarative service metadata, then anything we inferred.

This resolves two common operational issues:

(1) **Authoritative-vs-fallback conflicts**: low-tier fallback labels remain useful for recall but do not become canonical when high-tier evidence exists.
(2) **Language consistency**: language filtering occurs before ladder selection, with explicit fallback chain control.

Language filtering must precede ladder selection, not follow it. If the ladder ran first, a query in German against a target whose highest-tier label exists only in English would surface the English label and silently drop a lower-tier German one, even though the German label is the one the user can read. Running language selection first over the ordered preference tuple $(\ell_u, \ell_t, \texttt{en})$, then the ladder within the selected language, yields the intended semantics: the best available authority *in the most preferred language that has any label at all.*

Proposition 5.1 (Determinism). *For a fixed catalog state, tenant, and language preference tuple, the merge procedure returns the same label record for a target regardless of the order in which label records are supplied.*

Argument. Selection is a lexicographic minimum over the key (language rank, source rank, −confidence, stable input key). The first three components are total orders on finite domains, and the fourth is a total order on the natural key of the label record, which is unique by the adapter-conformance invariants of §6.3. A lexicographic minimum over a total order is unique and independent of input order. □

Determinism is not a stylistic preference here. Because the surfaced label becomes part of an LLM prompt, a non-deterministic merge would make agent behavior irreproducible for reasons invisible in the agent's own trace, and would make regression testing of the grounding layer impossible. Proposition 5.1 is the property that the `catalog-merge` suite of §7.1 is written to check.

5.4 Failure Handling

If embedding generation fails or vector branch execution errors, the system retries lexical-only search. This preserves availability and avoids hard failure on transient embedding/vector issues. Because of the indicator structure of Eq. (1), this degraded mode is not a separate code path with its own ranking semantics: it is the same scoring function evaluated on a pool where $r_v = \varnothing$ for every candidate. The reverse degradation (vector-only, when the lexical branch yields an empty pool even after OR-fallback) follows symmetrically.

5.5 End-to-End Retrieval Procedure

(1) Build lexical and vector candidate pools (top-k each, currently $k = 20$).
(2) Join alias matches by $(target\_model, target\_field)$ in the active language set.
(3) Group candidates by $(provider\_model, provider\_field)$.
(4) Compute fused score $s(m, f)$ via RRF plus alias boost.
(5) Resolve surfaced label by deterministic source ladder in selected language.
(6) Sort by score, then stable key order, and return top-N.

The order of steps 4 and 5 is deliberate. Canonicalization is applied only to targets that survived ranking, which is what allows the two stages of §2 to use different evidence policies without either compromising the other: ranking sees every label record including low-authority enrichments, and canonicalization sees only the ladder.

5.6 Complexity

Let k_b, k_v be lexical and vector branch limits. Candidate fusion and group scoring are linear in $O(k_b + k_v)$ after index retrieval. With fixed branch caps (20 each), runtime is dominated by indexed retrieval rather than merge-time compute. Merge adds $O(|\Lambda|)$ per returned target for the lexicographic minimum, where $|\Lambda|$ is the number of label records on that target, with no sorting required. The end-to-end cost is therefore one GIN probe, one HNSW probe, one alias lookup, and a bounded amount of application-level work independent of catalog size.

6 Implementation and Operational Controls

6.1 Persistence and Indexing

PostgreSQL stores the catalog with:

- `GIN` index over generated `tsvector` for lexical retrieval,
- `HNSW` index over `vector(1536)` for semantic retrieval,

- tenant RLS policies on every schema catalog table.

Co-locating both indexes on `schema_labels` is a deliberate choice against the more common design of a separate vector store. It has three consequences. The two branches see exactly the same row population, so a fusion result can never be explained by index skew. Tenant isolation is enforced once, by row-level security in the database, rather than separately in two systems that can drift; there is no second copy of tenant data outside the RLS perimeter. And ingestion is transactional: a label and its embedding become visible together, so the catalog is never in a state where lexical retrieval can find a target that vector retrieval cannot. The cost is that we inherit PostgreSQL's ANN characteristics rather than a specialized engine's, which we revisit in §9.

6.2 Quality Gates

Post-ingestion quality checks include:

- minimum model count (default 10),
- minimum label coverage (default 0.70),
- minimum embedding coverage (default 1.00),
- discovery query checks (provider defaults such as "purchase order header", "customer master", "material master").

Gate failures are recorded rather than crashing ingestion orchestration, preserving run observability.

The embedding-coverage gate is set to 1.00 rather than to a tolerance because partial embedding coverage is the failure mode that hybrid retrieval hides best: the lexical branch keeps answering, aggregate quality degrades smoothly, and nothing alerts. Requiring total coverage converts a silent quality regression into a visible ingestion signal. The discovery-query checks play the same role at the semantic level: they are canaries, not benchmarks, and they use provider-default strings so that a gate failure indicates broken ingestion rather than a hard query.

6.3 Adapter Conformance

A shared conformance harness enforces 10 invariants across adapters, including natural-key uniqueness, language-tag validity, capability shape constraints, provenance presence, tombstone consistency, and leakage checks for credentials and interpolated SQL fragments. Because every adapter is checked against the same invariants, the retrieval and merge layers can assume them; the conformance harness is what makes the "no provider logic below the adapter boundary" rule enforceable rather than aspirational.

7 Verification Evidence and Evaluation Protocol

The evidence in this paper is implementation-grounded rather than benchmark-leaderboard oriented. This section does two separate things, and the distinction is the most important thing to carry out of the paper. §7.1 reports evidence that exists today: executed implementation-verification suites from the production codebase. §7.2 through §7.6 state a *protocol* for measuring retrieval quality, and that protocol has not been run.

We are explicit about the gate between the two. Nothing in §7.2 through §7.6 has been executed: the judged query set those subsections require has not been sampled, authored, pooled, or annotated, and no run files have been produced for any variant. Consequently this paper contains no Recall@10, MRR@10, or nDCG@10 value, no significance test, no confidence interval, and no serving-latency percentile, and it contains none of these because none has been measured rather than because they were omitted for space. We have not filled any of them from the four-query pilot fixture, which is degenerate, nor from any estimate, which would be fabrication.

Suite Group	Suites	Tests
Retrieval + merge + gate	3	90
Adapter ingesters	4	162
Total	**7**	**252**

Table 1: Executed verification suites, counted in test suites and in individual test cases. All 252 tests passed. These counts measure implementation conformance to the invariants of §5; they are not retrieval measurements and no retrieval metric is derived from them.

What the protocol subsections are for is precision. The metrics, their cutoff, the qrels construction procedure, the annotation and adjudication rules, the ablation ladder, and the analysis plan are all fixed here, in advance of the numbers, so that the eventual measurement is a test rather than a search over reporting choices. A reader who requires measured retrieval gains should stop at §7.1 and treat the rest of this section as a specification.

7.1 Verification Evidence

We executed the following suites in the backend workspace:

- core retrieval, merge, and gate suites: `schema-search`, `catalog-merge`, `quality-gate`;
- adapter suites: `odoo-catalog`, `sap/hana-ingester`, `sap/odata-ingester`, `sap/sap-catalog-ingester`.

Key checked behaviors include:

- exact RRF arithmetic and per-branch contribution handling,
- alias-boost ranking effects,
- deterministic label ladder ordering and tie-breaks,
- parameterized SQL contracts and tenant scoping setup,
- failover to BM25-only mode on embedding/vector failures,
- conformance invariants across heterogeneous adapters.

Table 1 establishes that the invariants of §5 hold in the implementation: the fusion arithmetic of Eq. (1), the determinism of Prop. 5.1, the degradation path, and the adapter contract. It establishes nothing about retrieval quality. Test counts are not retrieval results and we do not present them as such; they are the evidence base for the systems contributions C1, C3, and C4 of §1.1, and for those only.

7.2 Task, Metrics, and Definitions

The benchmark task is *query-to-schema-target retrieval*. A system receives (q, t, p, L) as in §2 and returns a ranked list of targets; it is scored against graded relevance judgments over targets. Let Q be the query set, G_q the set of targets with relevance $\mathrm{rel}(q, \cdot) > 0$, and $\pi_q(i)$ the target at rank i of the system's run for q. The three metrics, all at cutoff $k = 10$ and all in $[0, 1]$, are defined as:

$$\text{Recall@}k = \frac{1}{|Q|} \sum_{q \in Q} \frac{|\{i \le k : \text{rel}(q, \pi_q(i)) > 0\}|}{|G_q|}, \tag{6}$$

$$\text{MRR@}k = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{\min\{i \le k : \text{rel}(q, \pi_q(i)) > 0\}}, \tag{7}$$

$$\text{nDCG@}k = \frac{1}{|Q|} \sum_{q \in Q} \frac{\text{DCG}_q\text{@}k}{\text{IDCG}_q\text{@}k}, \tag{8}$$

where the reciprocal rank is 0 if no relevant target appears in the top k, and

$$\text{DCG}_q\text{@}k = \sum_{i=1}^{k} \frac{2^{\text{rel}(q,\pi_q(i))} - 1}{\log_2(i+1)}, \tag{9}$$

with $\text{IDCG}_q\text{@}k$ computed by Eq. (9) over the relevance values of G_q sorted descending and truncated at k [12]. Four conventions are fixed here because they materially change reported values and are frequently left implicit. Recall@k uses $|G_q|$, the full judged relevant set, as denominator, so it is capped below 1 when $|G_q| > k$. Queries in Q with no judgments are excluded rather than scored as 0. Ties in the fused score are broken by the stable key order of §5, step 6, so a run is a total order and no metric depends on an arbitrary tie resolution. Targets returned by a system but absent from the qrels are treated as rel = 0, so the protocol is a closed-pool evaluation and comparisons are valid only across systems whose candidates were included in the judging pool. Eqs. (6)–(9) are implemented by the metrics script of the harness described in §7.5, which also fixes a line-oriented JSON contract for queries, qrels, and system runs. That harness is not distributed with this manuscript, so the equations above, together with the four conventions just stated, are the normative definition here.

7.3 External Benchmark Protocol

For comparative evaluation, the protocol specifies a query-to-schema retrieval benchmark with held-out gold mappings:

(1) Build query sets from operational prompts and analyst-authored intents.
(2) Label one-or-more gold (*provider_model*, *provider_field*) targets per query.
(3) Compare BM25-only, vector-only, and full PROSE fusion variants.
(4) Report Recall@10, MRR@10, and nDCG@10.
(5) Add robustness slices: multilingual, alias-heavy, and sparse-label queries.

This protocol is directly supported by the deployed architecture and can be artifactized without changing retrieval logic.

The query set has not been built. It will be constructed in three passes. *Operational sampling* will draw candidate queries from logged agent schema-grounding calls, deduplicated by normalized surface form and stratified by provider family so that no single tenant dominates Q. *Analyst authoring* will add intents for business concepts that appear in the catalog but not in the log, so that Q is not biased toward what the current system already answers well. That selection effect would inflate every variant of §7.4 equally while specifically understating the `alias-heavy` and `multilingual` slices. *Slice assignment* will label each query with zero or more slices: `multilingual` (query language ≠ `en`), `alias-heavy` (intent uses tenant vocabulary absent from canonical labels), `sparse-label` (gold target has fewer label records than the catalog median), and `metadata-rich` (the complement).

Judgments will be collected as follows. For each query, a pool will be formed from the union of the top-50 results of every variant in the ladder of §7.4, so that no variant is advantaged by having contributed the pool. Two annotators with ERP domain experience will independently assign graded relevance on $\{0, 1, 2\}$, where 2 denotes the target a correct query would use, 1 denotes a target that is topically correct but would require additional joins or is a near-duplicate view of the correct target, and 0 denotes irrelevant. A third annotator will adjudicate disagreements. Annotators will judge (q, target) pairs in randomized order, without the identity of the producing variant. Inter-annotator agreement before adjudication will be reported as Cohen's κ, computed over the graded scale on the pre-adjudication judgments. Because no annotation round has been run, we report no agreement figure here; the protocol fixes how it will be computed, not what it is.

A small pilot fixture exists alongside the harness, and it is *not* this query set. It contains 4 queries and 4 gold targets and exists only to exercise the harness end to end, not to measure anything; any metric computed on it is degenerate and is reported nowhere in this paper. When the judged query set is built it will be characterized by four descriptors, none of which we can state today because the set does not exist: its size $|Q|$, targeted at no fewer than 200 queries; the number of tenants sampled; the provider families covered; and the total number of graded judgments collected.

7.4 Ablation Ladder and Baselines

The protocol specifies five variants, each adding one mechanism to the previous, so that every architectural claim in §5 is attributable to a single row transition:

(1) `bm25_only`: lexical branch alone, with OR-fallback.
(2) `vector_only`: dense branch alone.
(3) `bm25_vector_rrf`: Eq. (1). Tests H1 against rows 1–2.
(4) `+ alias boost`: Eq. (2). Tests H2 against row 3, jointly on the `alias-heavy` and `metadata-rich` slices.
(5) `+ provenance ladder`: Eqs. (3)–(5) applied at canonicalization. Tests H3 against row 4.

Rows 1–2 are the standard single-branch baselines for hybrid retrieval [13, 27]. The protocol also specifies an oracle upper bound on the judging pool, namely the best achievable ranking of the pooled candidates, which separates ranking error from candidate-generation error.

Because the ladder changes canonicalization rather than ordering, row 5 will be scored under a modified relevance function in which a target counts as relevant only if the surfaced label is judged faithful to it. This is the only point in the protocol where the metric definition differs across rows, so row 4 will be reported under both relevance functions and the change remains auditable.

The planned significance analysis is a two-sided paired permutation test on per-query nDCG@10 with 10^4 permutations, Holm-corrected across the four row transitions, reported with effect sizes and 95% bootstrap confidence intervals rather than with p-values

alone. The harness implements this analysis. Its metrics script exposes per-query nDCG@10 and provides the two-sided paired permutation test, the percentile bootstrap interval, Holm correction across the four row transitions, and a paired effect size, all under a fixed seed so that a reported p-value and interval are reproducible from the same run files. What is still absent is not the machinery but the data: no judged query set exists, so there is nothing to run the analysis on, and no significance claim about H1, H2, or H3 is made here.

What will be reported, and what is absent here. This paper presents no results table, because there are no results. We state exactly what the tables will contain so that the reporting commitment is on the record. The primary report will be one row per ladder variant, plus a pool-oracle row, with three columns: Recall@10, MRR@10, and nDCG@10, each a mean over queries at cutoff $k = 10$, dimensionless and in $[0, 1]$, computed as in §7.2. Each of the four row transitions will additionally carry a Holm-corrected permutation-test p-value, an effect size, and a 95% bootstrap interval on the nDCG@10 difference. The secondary report will be one row per robustness slice (`multilingual`, `alias-heavy`, `sparse-label`, `metadata-rich`) with the number of judged queries in the slice and nDCG@10 for the RRF variant and for the full system, so that H2 is decided by the joint movement of the `alias-heavy` and `metadata-rich` rows rather than by a single aggregate.

None of these quantities has been measured, so none is reported, and nothing has been estimated, extrapolated, simulated, or carried over from the pilot fixture to stand in for them.

7.5 Evaluation Harness

We have implemented a runnable harness for the protocol above, comprising:

- a line-oriented JSON data specification for queries, qrels, and system runs;
- a metrics script for Recall@10, MRR@10, and nDCG@10 as defined in §7.2;
- slice-aware reporting by language, alias load, and metadata sparsity;
- the paired analysis of §7.4: permutation test, bootstrap interval, Holm correction, and paired effect size, seeded for reproducibility.

This makes the protocol directly executable for ablations and regression tracking once a judged query set exists. The harness is deliberately decoupled from the serving stack: it consumes run files, so a variant may be produced by the deployed system, by a research prototype, or by a third party, and the comparison remains valid.

We state its availability precisely, because the rest of this section depends on it. The harness is internal at the time of writing and is not included with this manuscript, so a reader cannot currently obtain or execute it; it is intended to accompany a subsequent code release, together with the judged query set if that set proves releasable. We therefore make no reproducibility claim for it here, and nothing in this paper rests on the reader having it: the task, the metric definitions of §7.2, the data contract above, the ablation ladder, and the analysis plan are stated in the text so that the protocol can be reimplemented independently, which is what §9 means by reproducible in procedure but not in data.

7.6 Deployment Measurements

Serving-side measurements answer a different question from retrieval quality and are therefore specified separately. The protocol fixes the following four instruments. None of them has been built, so this paper reports no value for any of them:

- End-to-end latency of the `search_schema` call, in milliseconds, at the 50th, 95th, and 99th percentiles.
- Per-branch latency decomposition in milliseconds: GIN probe, HNSW probe, alias join, and merge.
- Frequency of the degraded lexical-only path, as a fraction of calls over a stated window.
- Catalog scale at measurement time, counted in targets, label records, and tenants.

We do not report a production operating window, active-user count, or escalation rate, because that instrumentation does not exist yet; asserting any of these would be fabrication. A request-latency percentile is available from platform telemetry, but it aggregates every interactive API path and is therefore not a measurement of `search_schema`; we exclude it rather than present it as one.

8 Discussion

Where the contribution sits. The contribution lies in information management under operational constraints: heterogeneous metadata integration, hybrid retrieval, deterministic conflict resolution, and auditable multi-tenant query infrastructure. These are central concerns of information and knowledge management, but often underrepresented in work that focuses only on model quality.

Practical novelty. Three details matter in practice:

(1) **Provenance ladders over global confidence**: avoids subtle trust regressions.
(2) **Alias layer separated from canonical labels**: allows tenant terminology adaptation without contaminating source truth.
(3) **Graceful retrieval degradation**: preserves agent operability when one retrieval modality fails.

Generality beyond ERP. Nothing below the adapter boundary is ERP-specific. The pattern applies to any setting where retrieval targets carry multiple descriptions of unequal authority: rank over all descriptive evidence, canonicalize over authoritative evidence only, and keep tenant vocabulary in a separate layer that shapes ranking but not canonical truth. Tool and API catalogs assembled from vendor specifications plus scraped documentation have exactly this structure, as do data-lake catalogs combining declared schemas with inferred semantic types [11]. Transferring PROSE to such a setting requires supplying a total order over that domain's source classes and an adapter that emits the eight-entity stream; it does not require changing retrieval or merge.

Lessons from building the system. Two lessons were not obvious to us in advance. First, the failures that cost us most were not ranking failures but *canonicalization* failures, and they are invisible to retrieval-quality metrics: the right target was returned at rank 1 with a misleading label, and the agent then reasoned over

a field it had misunderstood. H3 is stated as a distinct hypothesis with its own relevance function for this reason, rather than folded into nDCG. Second, hybrid retrieval conceals ingestion defects. A healthy lexical branch masks a broken vector branch and vice versa, so we had to make coverage a hard gate rather than a dashboard, and to make the degraded path emit an explicit signal rather than succeed silently.

9 Limitations

We state the limitations in descending order of severity, beginning with the most consequential.

The retrieval hypotheses are unmeasured. This is the paper's principal limitation. H1–H3 are specified but not tested. The judged query set of §7 has not been built, so this manuscript reports no ablation and no slice breakdown at all; we report nothing rather than populate them from the 4-query pilot fixture, which would be meaningless, or from any estimate, which would be fabrication. The blocker here is data rather than tooling: the harness now implements the significance and confidence-interval machinery specified in §7.4, so that analysis is ready to run and has nothing to run on until the judged query set is built. Current evidence is engineering-verification heavy and benchmark-light. Consequently the paper's research claims (C2, C5) should be read as stated-and-falsifiable rather than as supported, and the 252 passing tests of §7.1 should not be read as standing in for them. A reader who requires measured retrieval gains should not be persuaded by this manuscript.

No external benchmark exists for this task. A public or reproducible anonymized enterprise schema benchmark would strengthen cross-system comparison. Query-to-schema-target retrieval over multi-provider enterprise catalogs has no public dataset, because the catalogs are proprietary and the tenant vocabulary that makes the task hard is exactly the part that cannot be released. Our protocol is therefore reproducible in *procedure* but not in *data*, and constructing a releasable benchmark is the highest-value follow-on work.

Hyperparameters are set by design argument, not by search. $K = 60$ follows the RRF literature [5], and $\lambda = 0.5$ and $k = 20$ were chosen for the structural properties described in §5.2 rather than fitted. We have not established that they are near-optimal, and a fixed λ cannot adapt to query classes where alias evidence should be weaker than a strong lexical match. The alias boost is in addition currently a constant; future work can learn adaptive boost schedules by query class.

The source precedence order is hand-authored. Eqs. (3)–(5) encode expert judgment about which SAP and OData artifacts are authoritative. The orderings are auditable and were validated against operational conflicts, but they are not learned, and extending to a new ERP requires a domain expert to extend the order. We do not currently have a procedure for detecting that a hand-authored order is wrong other than observing downstream grounding errors.

Scope of the evaluated system. Verification covers SAP (HANA, OData, catalog service) and Odoo adapters. Claims of ERP-agnosticism rest on the adapter conformance contract rather than on having onboarded a third family, and the catalog is a single-region PostgreSQL deployment, so we make no claim about ANN behavior at catalog sizes beyond those we operate.

Ethics and data handling. The system processes enterprise metadata (table, field, and relationship descriptions), not customer business records. Tenant isolation is enforced by row-level security on every catalog table, and adapter conformance includes leakage checks for credentials and interpolated SQL fragments. Tenant glossaries are tenant-scoped and never propagate to another tenant's canonical labels. Residual risk remains where a tenant's alias vocabulary itself encodes commercially sensitive terminology; such aliases sit inside the same RLS perimeter as catalog rows, and no cross-tenant model is trained on them.

10 Conclusion

We presented PROSE, a provenance-aware hybrid schema retrieval architecture for enterprise agent grounding. By combining BM25 and vector retrieval with deterministic source-aware merge policies and tenant alias semantics, the system provides deterministic and auditable schema grounding in heterogeneous ERP settings. The central design claim is that schema grounding must be treated as two problems rather than one: ranking over all descriptive evidence, and canonicalization over authoritative evidence only. Separating them is what makes provenance enforceable inside a retrieval path rather than bolted onto it. The implementation is production-integrated and verified at the systems level by the suites of §7.1. The retrieval hypotheses H1–H3 are specified, pre-registered, and not yet measured, and we have reported no number for any of them rather than an estimated one. This manuscript is therefore best read as an architecture together with the evaluation protocol that would test it, and we regard executing that protocol as the necessary next step rather than as optional future work. We believe this class of work, bridging retrieval theory and operational information management, belongs in the literature on applied information management.

GenAI Usage Disclosure

Generative-AI tools were used for language editing and manuscript drafting support. All technical claims and equations were validated against executable scripts and source code artifacts by the authors. No GenAI tool was used to generate or alter empirical metric values, and this manuscript reports no empirical retrieval metric.

References

[1] Philip A. Bernstein, Jayant Madhavan, and Erhard Rahm. 2011. Generic Schema Matching, Ten Years Later. *Proceedings of the VLDB Endowment* 4, 11 (2011), 695–701.

[2] Sebastian Bruch, Siyu Gai, and Amir Ingber. 2024. An Analysis of Fusion Functions for Hybrid Retrieval. *ACM Transactions on Information Systems* 42, 1 (2024).

[3] Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. 2001. Why and Where: A Characterization of Data Provenance. In *Proceedings of the 8th International Conference on Database Theory (ICDT)*.

[4] James Cheney, Laura Chiticariu, and Wang-Chiew Tan. 2009. Provenance in Databases: Why, How, and Where. *Foundations and Trends in Databases* 1, 4 (2009), 379–474.

[5] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 758–759.

[6] AnHai Doan, Alon Halevy, and Zachary Ives. 2012. *Principles of Data Integration*. Morgan Kaufmann.

[7] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A Data Discovery System. In *Proceedings of the 34th IEEE International Conference on Data Engineering (ICDE)*.

[8] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *Proceedings of the VLDB Endowment* 17, 5 (2024).

[9] Alon Halevy, Flip Korn, Natalya F. Noy, Christopher Olston, Neoklis Polyzotis, Sudip Roy, and Steven Euijong Whang. 2016. Goods: Organizing Google's Datasets. In *Proceedings of the 2016 ACM SIGMOD International Conference on Management of Data (SIGMOD)*.

[10] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. Knowledge Graphs. *Comput. Surveys* 54, 4 (2021), 1–37.

[11] Madelon Hulsebos, Kevin Hu, Michiel Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çağatay Demiralp, and César Hidalgo. 2019. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

[12] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.

[13] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[14] Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the Role of Schema Linking in Text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[15] 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 (NeurIPS)*.

[16] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLM Already Serve as a Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*.

[17] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *Proceedings of the VLDB Endowment* 14, 1 (2020), 50–60.

[18] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

[19] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. 2001. Generic Schema Matching with Cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB)*. 49–58.

[20] Yu A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (2020), 824–836.

[21] Luc Moreau, Paolo Missier, et al. 2013. PROV-DM: The PROV Data Model. W3C Recommendation.

[22] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table Union Search on Open Data. *Proceedings of the VLDB Endowment* 11, 7 (2018), 813–825.

[23] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large Language Model Connected with Massive APIs. arXiv:2305.15334.

[24] Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[25] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. In *International Conference on Learning Representations (ICLR)*.

[26] Erhard Rahm and Philip A. Bernstein. 2001. A survey of approaches to automatic schema matching. *The VLDB Journal* 10, 4 (2001), 334–350.

[27] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.

[28] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[29] Michael Stonebraker, Daniel Bruckner, Ihab F. Ilyas, George Beskales, Mitch Cherniack, Stanley B. Zdonik, Alexander Pagan, and Shan Xu. 2013. Data Curation at Scale: The Data Tamer System. In *Proceedings of the 6th Biennial Conference on Innovative Data Systems Research (CIDR)*.

[30] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*.

[31] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

[32] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[33] Matei Zaharia, Omar Khattab, Lingjiao Chen, Jared Quincy Davis, Heather Miller, Christopher Potts, James Zou, Michael Carbin, Jonathan Frankle, Naveen Rao, and Ali Ghodsi. 2024. The Shift from Models to Compound AI Systems. BAIR Blog.

[34] Yi Zhang and Zachary G. Ives. 2020. Finding Related Tables in Data Lakes for Interactive Data Science. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD)*.

[35] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. arXiv:1709.00103.