

Fine-Grained Table Retrieval Through the Lens of Complex Queries

Wojciech Kosiuk¹, Xingyu Ji², Yeounoh Chung³, Fatma Özcan³, Madelon Hulsebos^{4*}

¹University of Amsterdam, ²UC Berkeley, ³Google, ⁴CWI

Abstract

Enabling question answering over tables and databases in natural language has become a key capability in the democratization of insights from tabular data sources. These systems first require retrieval of data that is relevant to a given natural language query, for which several methods have been introduced. In this work we present and study Decomposition-based Connectivity-aware Table Retrieval (DCTR), a table retrieval mechanism combining fine-grained typed query decomposition with schema connectivity signals for open-domain question answering over relational databases. We evaluate the effectiveness of the two mechanisms through the lens of *retrieval complexity* which we measure along the axes of query- and data complexity. Our analyses over industry-aligned benchmarks show that typed decomposition improves robustness for highly compositional queries, while connectivity-aware retrieval is most useful when schema structure provides informative join signals.

1 Introduction

Open-domain question answering and text-to-SQL over large heterogeneous table collections face a fundamental retrieval challenge: users rarely know which tables are relevant, and required join paths cannot be discerned from the query alone (Chen et al., 2025c). Ambiguous attribute names and inconsistent terminology create mismatches between natural-language queries and schema organization (Lei et al., 2020). Multi-constraint queries allow for multiple plausible interpretations (Gomm et al., 2025), making it difficult to identify correct tables before SQL generation. Retrieval errors propagate to downstream SQL generation, and are compounded when varying data model designs admit multiple valid SQL structures for the same query (Fürst et al., 2024),

where even reliably evaluating query correctness remains challenging (Kim et al., 2024). Moreover, while recent methods devise query decomposition for table retrieval, single-hop retrieval using full-query embeddings is a common default (Chen et al., 2025b; Zou et al., 2025), motivating a close examination of the robustness of single-hop retrieval for complex compositional queries.

Despite recent progress in table retrieval mechanisms, single-vector table embeddings cannot represent multi-constraint relevance signals (Weller et al., 2025) and degrade when query semantics diverges from schema conventions (Chen et al., 2025b; Lei et al., 2020). Industry workloads intensify these challenges: queries are verbose and compositional, databases span hundreds of tables with unaligned semantics, and relevant data is distributed across many join-connected tables. While existing studies surface patterns in retrieval performance across methods and datasets (Ji et al., 2025), deeper insights are lacking regarding such complex retrieval characteristics.

Focusing on complex retrieval settings, we study Decomposition-based Connectivity-aware Table Retrieval (DCTR), which combines 1) typed query decomposition for fine-grained query-schema alignment using multi-vector embeddings, and 2) schema-connectivity grouping for surfacing join-related tables beyond semantic similarity. We evaluate DCTR on industry-aligned benchmarks for open-domain analytical tabular question answering and show that decomposition is most useful for long, compositional queries and smaller embedding models. Finally, we formalize retrieval complexity along the query- and data dimensions, and analyze the influence of query signal-to-noise ratios, and schema heterogeneity and connectivity. Our findings surface the usefulness and limitations of fine-grained query decomposition and global connectivity-awareness in complex retrieval settings involving highly compositional queries and

*Correspondence to: madelon@cwi.nl

large relational databases.

2 Related Work

2.1 Text-to-SQL and Schema Linking

Schema linking aligns natural-language queries with schema elements at varying granularities. RASL (Eben et al., 2025) introduces multi-stage retrieval for databases by decomposing schemas into semantic units. Nahid et al. (Nahid et al., 2025) propose a context-aware bidirectional schema retrieval approach that links queries to tables and columns using LLM-guided selection. Recent work questions explicit linking: Maamari et al. (Maamari et al., 2024) argue that high-capacity LLMs identify relevant schema despite distracting context, while RESDSQL (Li et al., 2023a) frames linking as ranking-enhanced encoding decoupled from SQL generation. These works generally assume a known data context and limited retrieval scope, providing a narrow view of the open-domain setting in complex data context.

2.2 Retrieval in Open-Domain Text-to-SQL

Open-domain QA faces ambiguous intent and heterogeneous corpora. MURRE (Zhang et al., 2024) performs iterative multi-hop retrieval with query rewriting which is complemented by data augmentation in Abacus-SQL (Xu et al., 2025), while LinkAlign (Wang et al., 2025) scales schema linking to large databases via multi-round query rewriting and response filtering. EnrichIndex (Chen et al., 2025b) and Pneuma (Balaka et al., 2025) show the value of enriching retrieval indices with LLM-generated metadata, while GenEdit (Maamari et al., 2025) integrates domain knowledge in SQL generation through specialized operators and user feedback. Complementary techniques include graph-based hierarchical retrieval (Zou et al., 2025) and weak supervision (Liang et al., 2025). To address the large scale of databases, R2D2 (Bodensohn and Binnig, 2024) retrieves table segments instead of full tables. JAR (Chen et al., 2025c) introduces join-aware reranking to balance table-query relevance with table-table compatibility, which was adapted for iterative retrieval by Boutaleb et al.. We build on these retrieval principles, but introduce *global* connectivity-aware retrieval to surface relevant tables beyond semantic similarity, and typed query decomposition for fine-grained schema alignment. Our use of FK-based schema graphs shares struc-

tural motivation with Relational Graph Transformers (Dwivedi et al., 2026), though that work targets node-level prediction rather than table retrieval. We analyze the effectiveness through the lens of complex queries and databases providing a novel view of performance in complex settings.

3 Problem Setting

Open-Domain Table Retrieval Let $T = \{T_1, \dots, T_n\}$ denote the set of tables in a relational database, where each table T_i consists of a set of columns $C(T_i) = \{c_1, \dots, c_m\}$, possibly connected through foreign-key relationships. In an open-domain setting, a natural-language query q expresses an information need from which the data scope needs to be inferred for retrieval of relevant tables from a large topic-diverse set of tables. The ground-truth set of relevant tables for q is denoted as $T^*(q)$. The table retrieval task aims to return a ranked list $R_k(q)$ of top- k candidate tables such that capped recall (CR) (Thakur et al., 2021) is maximized: $\text{CR@k}(q) = \frac{|R_k(q) \cap T^*(q)|}{\min(k, |T^*(q)|)}$.

Retrieval Complexity The composition of queries and data that is typically faced in real-world retrieval tasks for end-to-end question answering stretches the complexity levels of what existing evaluations surface. We characterize retrieval complexity along two dimensions: *query complexity* and *data complexity*. We term *query complexity* as *the complexity induced by how the query is formulated in terms of semantic density and functional composition* (Gomm et al., 2025). We term *data complexity* as *the complexity resulting from how the data is structured within a database due to database normalization and schema size* (Chen et al., 2025c). These complexity dimensions guide our analysis of fine-grained retrieval for complex question answering tasks.

4 Fine-Grained Table Retrieval

We describe the Decomposition and Connectivity-aware Table Retrieval method (DCTR) that is tailored for complex retrieval tasks over two dimensions: 1) functional query decomposition for fine-grained schema alignment, and 2) global connectivity aware retrieval for recovering relevant tables beyond the semantic similarity scope.

Before queries are executed, an index of the data is built first. To support scalable retrieval over large schemas with thousands of tables and a proportional number of columns, we construct

two dense indices: one encoding table names and one encoding column names. Retrieval is performed by comparing embeddings of query elements against embeddings of these different schema elements.

4.1 DCTR Retrieval Pipeline

DCTR comprises a retrieval pipeline instrumented with fine-grained typed component extraction from queries, component-wise retrieval, join-graph construction, and FK-based expansion to account for global connectivity beyond semantic similarity (Fig. 1).

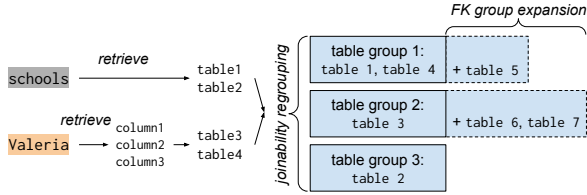


Figure 1: Overview of DCTR with component-wise retrieval, group formation, and FK-group expansion.

Typed Query Decomposition in Semantic Units

In an open-domain setting, natural language queries reflect constraints over different entities, attributes, and operations (Gomm et al., 2025). To align these with concrete schema elements and operators, we first perform a *typed query decomposition* step that segments an incoming query q into atomic *semantic units*, forming the basis for both retrieval and downstream task execution (e.g. SQL generation). The decomposition is done using a language model with a fixed few-shot instruction prompt and decoding configuration to ensure consistent behavior across datasets (Appendix A). Since the prompt receives only the query text and no schema information, decomposition cost is independent of database size. DCTR classifies each *semantic unit* as one of three component types:

- **Schema components**, denoting candidate tables or columns names and are matched to database schemas during retrieval.
- **Value components**, corresponding to entities or literals that act as query filters.
- **Aggregator components**, which specify aggregation or comparison operations (e.g., average, min, max).

For example, the query “What was the average sale of Luka Dončić jerseys in 2025?” is decomposed into schema components **jersey** and **sale**, value components **Luka Dončić** and

2025, and aggregator component **average**. In the retrieval phase, only schema and value components are used while the aggregator components benefits the downstream tasks, e.g. text-to-SQL.

First Pass Retrieval In the first stage of retrieval, each query component is processed independently for each database. For each component, we perform dense similarity search against both the table and column indices, retrieving the candidates per component. This initial retrieval produces unranked lists of tables and columns. The results from all components and all databases are then merged into a single candidate set. Columns retrieved from the column index are mapped back to their parent tables, ensuring that all relevant tables are included in the set that is forwarded to the graph-based grouping stage.

Schema Graph and Group Construction As discussed in (Chen et al., 2025c), joinable tables are relevant to consider but may not surface in a first-pass retrieval stage, which is conditioned on semantic similarity alone, requiring multi-hop reasoning over join paths. Given the set of candidate tables retrieved in the first pass, we represent each database schema as an undirected graph, where nodes correspond to tables and edges represent foreign-key (FK) relationships.

We restrict this graph to the first-pass candidate tables and the FK relationships between them. Each connected component defines a *table group*. By global FK-expansion, we then include all FK-connected tables with any table in the group to recover multi-hop join context.

Group Scoring Each table group is scored based on its coverage of the extracted query components. For each component, DCTR first computes the cosine similarity between its embedding and the embeddings of all tables in the group.

We introduce a parameter $vote_k$ to cap the number of tables included in the scoring of a table group. The resulting *group coverage score* is obtained by summing the similarity scores of the top $vote_k$ tables per component across all components: $group_cov_score(T, C) = \sum_{j=1}^{|C|} \text{top}_{vote_k}(\{\text{sim}(t_1, C_j), \dots, \text{sim}(t_n, C_j)\})$. Higher group coverage score indicates that the tables in the group collectively cover more of the query’s components, and therefore are more likely to contain the information needed to answer the query accurately. In the current formulation, all

components contribute equally to the group score. Because generic components (e.g., "id") produce uniformly high similarity across groups, they contribute little discriminative signal. Disambiguation is driven by specific components (e.g., "customer", "transaction date") that differentiate groups in the scoring.

Final Retrieval The final retrieval stage selects the most relevant tables by retaining the top n_groups scoring table groups per database. Within each selected group, the $vote_k$ tables with the highest individual coverage scores are selected as the final candidate set. These two hyperparameters control the breadth and granularity of the retrieval, allowing the pipeline to balance recall and precision depending on query complexity and schema size.

5 Experimental Setup

5.1 Method and Baselines Implementation

Retrieval Implementation We implement the retrieval backend using Qdrant with two collections. Schema representations are indexed separately: table identifiers are stored in `table_embeddings`, while column identifiers are stored in `column_embeddings`. For query decomposition we use `gpt-4.1-mini`; switching to `gemini-2.5-pro` changes capped recall by only 0.007 on average. For text-to-SQL evaluation we use `gemini-2.5-pro`.

Hyperparameters DCTR introduces three parameters: $vote_k$, the number of in-group candidates selected per component during group scoring; n_groups , the number of table subgroups retained per database for downstream ranking; and $expand_groups$, whether to expand high-scoring subgroups via join paths in the schema graph.

Dense Retrieval Baseline We compare our method against the common dense retrieval baseline (Karpukhin et al., 2020). Given a natural-language query q , the baseline embeds the full query as a single vector and tables are represented by an embedding of their name, column names, and a sample of data values, in line with the strongest method from the TARGET benchmark (Ji et al., 2025). Retrieval is performed based on cosine similarity across table embeddings and the given query embedding.

Embedding Models All retrieval components use dense text embeddings for query units and schema elements. We evaluate the following three models covering different capacity–efficiency trade-offs. **Stella-large** (Zhang et al., 2025): a high-capacity model (400M parameters, 1024-dim) designed for fine-grained semantic representation. **BGE-small** (Xiao et al., 2024): a lightweight general-purpose model (33M parameters, 384-dim) serving as an efficiency-oriented baseline. **E5-small** (Wang et al., 2024): A lightweight model (33M parameters, 384-dim) trained with weakly supervised contrastive objectives for scalable retrieval.

5.2 Datasets and Metrics

Datasets We evaluate on three text-to-SQL benchmarks accessed via TARGET (Ji et al., 2025), selected to mirror the friction points of industry retrieval: large schemas, ambiguous column naming, compositional queries, and multi-level nested joins. Together they span enterprise settings, stretching both query and data complexity (Sec. 5.2). **BEAVER** (Chen et al., 2025a): an enterprise benchmark from real data warehouses, with large schemas (77.2 tables per database on average) and verbose queries (31.9 words on average) involving multi-table joins and aggregations. **FIBEN** (Sen et al., 2020): a finance-domain benchmark with 300 natural-language questions over a single dense schema of 152 tables, requiring reasoning over domain-specific jargon and complex nested queries. **BIRD** (Li et al., 2023b): a cross-domain benchmark spanning 11 databases with smaller but well-connected schemas (6.8 tables, 1.4 FKs per table). We report results on the test sets of BEAVER and FIBEN, and the validation set of BIRD. Full dataset characteristics are presented in Table 7 (Appendix D).

Retrieval Metrics We evaluate performance using Capped Recall $RC@k$ for $k \in \{5, 10, 25\}$ across all embedding models and benchmarks (Sec. 3). First-stage candidate breadth is fixed to 30 and the remaining hyperparameters values are reported per experiment. Results are reported over multiple runs to account for stochasticity from LLM query decomposition. The dense baseline is deterministic and shows no variance.

Complexity Measures Following our notion of retrieval complexity as defined in Sec. 3, we investigate retrieval performance under varying degrees

of *query complexity* and *data complexity*, as typical industry workloads stretch these attributes. For each query, we capture attributes outlined below and analyze capped recall@25, aggregated across datasets and embedding models.

For *query complexity*, we measure 1) the length of the query in number of tokens, and 2) the number of extracted functional query components. For *data complexity*, we measure the total number of tables connected through FKs with the gold tables involved in a query reflecting the degree of schema normalization and connectivity. Distributional statistics of the datasets along these dimensions are presented in Appendix D.

6 Results

6.1 Overall Retrieval Performance

Tables 1-3 report capped Recall@ k for $k \in \{5, 10, 25\}$ across datasets, embedding models, and retrieval settings. For small scope ($k = 5$), low settings for vote_k and n_{groups} perform best. As k increases, moderate vote_k and n_{groups} improve recall in several settings, with the largest gains on BEAVER and for smaller embedding models. Notably, smaller embedding models benefit most from DCTR, narrowing the gap with a high-capacity baseline model. Connectivity-aware retrieval is particularly beneficial for larger schemas, as seen on BEAVER. Good priors for DCTR settings are proportional to the retrieval scope: for small k (e.g. 5) increasing vote_k or n_{groups} is ineffective, while for larger k , i.e. 10 or 25, setting vote_k and n_{groups} to 2 and 5, respectively, yields consistent gain in recall (App. B).

Group expansion shows dataset-dependent behavior. It improves recall on BIRD (+3-5%), particularly at larger k , but degrades performance on FIBEN and BEAVER (up to 30%). This aligns with structural differences across benchmarks (App. D). That is, BIRD databases are small yet well-connected (6.8 tables, 1.4 FKs per table), making expansion effective for recovering non-semantically similar but relevant tables. In contrast, FIBEN and BEAVER contain substantially larger schemas (152 and 77.5 tables/DB), yielding more candidate tables that might get truncated by the k -capped evaluation. A dynamic value for k would be viable in practice to fully utilize global FK-expansion. Specifically, on BEAVER the same three hub tables dominate the top-3 in 7 of 10 failure cases (Appendix E), in-

dicating that FK-expansion should be conditioned on schema density. We therefore treat expansion as an optional connectivity signal whose utility depends on schema topology, not as a default improvement over non-expanded retrieval.

Table 1: Capped Recall@5 (mean $\pm$ std) across datasets and models. Bold values indicate the best result per dataset and embedding model.

Setting	Model	BEAVER	FIBEN	BIRD
Baseline	stella	0.213	0.302	0.901
	bge-small	0.181	0.168	0.809
	e5-small	0.154	0.025	0.739
DCTR $\text{vote}_k = 1$, $n_{\text{groups}} = 1$	stella	0.231 $\pm_{.004}$	0.317 $\pm_{.001}$	0.834 $\pm_{.001}$
	bge-small	0.223 $\pm_{.005}$	0.294 $\pm_{.003}$	0.826 $\pm_{.002}$
	e5-small	0.199 $\pm_{.009}$	0.257 $\pm_{.004}$	0.817 $\pm_{.007}$

Table 2: Capped Recall@10 (mean $\pm$ std) across datasets and models. Bold values indicate the best result per dataset and embedding model.

Setting	Model	BEAVER	FIBEN	BIRD
Baseline	stella	0.277	0.413	0.961
	bge-small	0.231	0.262	0.909
	e5-small	0.175	0.140	0.850
DCTR $\text{vote}_k = 2$, $n_{\text{groups}} = 2$	stella	0.313 $\pm_{.007}$	0.396 $\pm_{.006}$	0.899 $\pm_{.004}$
	bge-small	0.326 $\pm_{.002}$	0.331 $\pm_{.006}$	0.893 $\pm_{.003}$
	e5-small	0.273 $\pm_{.007}$	0.346 $\pm_{.004}$	0.882 $\pm_{.006}$
DCTR $\text{vote}_k = 2$, $n_{\text{groups}} = 2$, expand_groups	stella	0.073 $\pm_{.014}$	0.202 $\pm_{.002}$	0.936 $\pm_{.003}$
	bge-small	0.095 $\pm_{.001}$	0.091 $\pm_{.014}$	0.942 $\pm_{.001}$
	e5-small	0.109 $\pm_{.001}$	0.078 $\pm_{.008}$	0.904 $\pm_{.005}$

Table 3: Capped Recall@25 (mean $\pm$ std) across datasets and models. Bold values indicate the best result per dataset and embedding model.

Setting	Model	BEAVER	FIBEN	BIRD
Baseline	stella	0.413	0.584	0.986
	bge-small	0.350	0.461	0.976
	e5-small	0.281	0.283	0.949
DCTR $\text{vote}_k = 5$, $n_{\text{groups}} = 5$	stella	0.435 $\pm_{.002}$	0.513 $\pm_{.007}$	0.971 $\pm_{.002}$
	bge-small	0.429 $\pm_{.003}$	0.448 $\pm_{.001}$	0.972 $\pm_{.002}$
	e5-small	0.408 $\pm_{.008}$	0.516 $\pm_{.003}$	0.965 $\pm_{.001}$
DCTR $\text{vote}_k = 5$, $n_{\text{groups}} = 5$, expand_groups	stella	0.228 $\pm_{.001}$	0.259 $\pm_{.001}$	0.991 $\pm_{.000}$
	bge-small	0.232 $\pm_{.002}$	0.127 $\pm_{.014}$	0.988 $\pm_{.002}$
	e5-small	0.228 $\pm_{.001}$	0.011 $\pm_{.003}$	0.975 $\pm_{.001}$

6.2 Retrieval Complexity Analysis

Query complexity. Figure 2 shows the retrieval performance against *query length* in number of tokens. We find that DCTR consistently outperforms the dense baseline for longer queries, with the gap widening beyond 40 tokens. For smaller

embedding models, in particular, the baseline is more sensitive to noise in the query than DCTR illustrating the effectiveness of fine-grained query decomposition for complex queries.

Figure 3 reinforces this finding along the axis of query compositionality: while both methods perform comparably for queries with few components, the baseline’s recall degrades steadily as compositionality increases, regardless of model capacity, whereas DCTR remains stable. High-capacity models are more robust to irrelevant noise (Fig. 2) but cannot resolve highly composite queries, confirming the need for component-level retrieval.

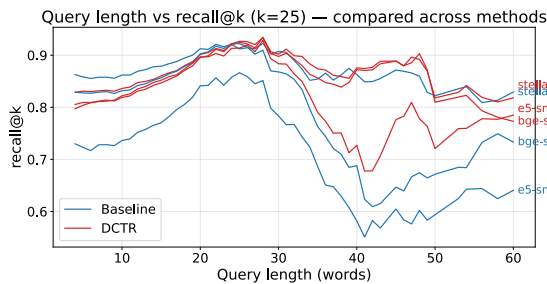


Figure 2: Query length vs recall@25, aggregated across datasets. Comparison through three embedding models, shows that DCTR outperforms the baseline, especially for longer queries (≥ 40 tokens).

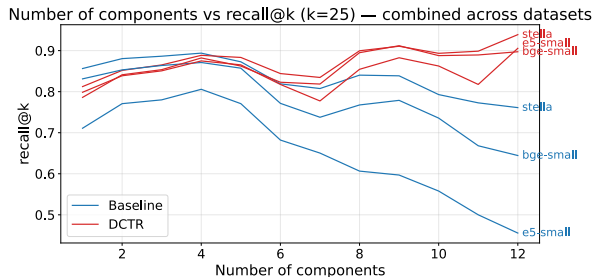


Figure 3: Number of components vs Recall@25 across datasets. Comparison through three embedding models shows that DCTR is robust for compositional queries while a single-vector baseline does not resolve compositional queries regardless of model capacity.

Data complexity. We assess data complexity via the connectivity of gold tables, measured by their number of join relationships within the database. Figure 4 illustrates that DCTR achieves stronger performance on queries involving densely connected tables, indicating that connectivity-aware grouping improves retrieval of joinable table sets.

6.3 Downstream Text-to-SQL Performance

We assess downstream utility by comparing DCTR (*stella* model, $k \in \{5, 10, 25\}$) against a long-context baseline that passes the full schema

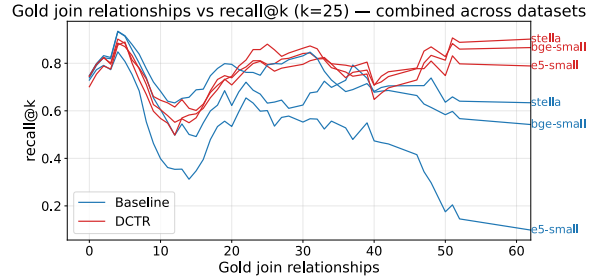


Figure 4: Number of tables connected to gold tables vs recall@25, combined across datasets, for different embedding models. DCTR improves more on queries where gold tables are densely connected.

to Gemini (Chung et al., 2025), using execution accuracy (Table 4). DCTR improves execution accuracy by 3% on BEAVER and 5% on FIBEN while matching the baseline on BIRD. The performance over k varies per dataset and correlates with retrieval difficulty: BIRD achieves peak accuracy at small k where recall is high, FIBEN requires moderate k , and BEAVER, where retrieval is most challenging, benefits from larger k .

Table 4: Downstream SQL execution accuracy of DCTR against the long-context baseline.

Setting		BEAVER	FIBEN	BIRD
Baseline		0.2730	0.1370	0.6630
DCTR	k=5	0.2150	0.0970	0.6656
	k=10	0.2400	0.1870	0.6630
	k=25	0.2980	0.1270	0.6623

7 Conclusion

The emergence of natural language interfaces to large-scale databases amplifies the importance of retrieval mechanisms that are robust to complex query- and data characteristics. In this context, we introduce Decomposition-based Connectivity-aware Table Retrieval (DCTR), which yields recall improvements over common single-vector dense retrieval baselines in several complex retrieval settings. Analysis of retrieval complexity along the axes of query- and data complexity surfaces that typed query decomposition is key in addressing highly compositional queries that require fine-grained multi-table schema alignment, while global connectivity-aware retrieval uncovers relevant tables beyond semantically similar tables in densely connected relational databases. The modular group scoring in DCTR naturally accommodates extensions such as component weighting, schema-density aware expansion, user-specific priors, and multi-turn refinement of decomposed components, which we leave for future work.

Limitations

Despite its effectiveness, DCTR has several limitations. Its performance depends on accurate query decomposition as errors may propagate to retrieval, and gains diminish for simple, single-table queries where added reasoning incurs unnecessary overhead.

Connectivity-aware retrieval further assumes an informative relational structure. In weakly connected, de-normalized, or poorly specified schemas, global connectivity signals may offer limited advantage over strong dense baselines.

The method also introduces additional cost due to query decomposition and multi-stage retrieval, which may affect latency-sensitive deployments. Finally, evaluation on industry-representative benchmarks may not fully reflect the diversity of real-world schema designs and query distributions.

References

- Muhammad Imam Luthfi Balaka, David Alexander, Qiming Wang, Yue Gong, Adila Krisnadhi, and Raul Castro Fernandez. 2025. Pneuma: Leveraging llms for tabular data representation and retrieval in an end-to-end system. *arXiv preprint arXiv:2504.09207*.
- Jan-Micha Bodensohn and Carsten Binnig. 2024. [Re-thinking table retrieval from data lakes](#). In *Proceedings of the Seventh International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, aiDM 2024, Santiago, Chile, 14 June 2024*, pages 2:1–2:5. ACM.
- Allaa Boutaleb, Bernd Amann, Rafael Angarita, and Hubert Naacke. Exploring multi-table retrieval through iterative search. In *EurIPS 2025 Workshop: AI for Tabular Data*.
- Peter Baile Chen, Fabian Wenz, Yi Zhang, Devin Yang, Justin Choi, Nesime Tatbul, Michael Cafarella, Çağatay Demiralp, and Michael Stonebraker. 2025a. [Beaver: An enterprise benchmark for text-to-sql](#). *Preprint*, arXiv:2409.02038.
- Peter Baile Chen, Tomer Wolfson, Michael Cafarella, and Dan Roth. 2025b. [Enrichindex: Using llms to enrich retrieval indices offline](#). *Preprint*, arXiv:2504.03598.
- Peter Baile Chen, Yi Zhang, and Dan Roth. 2025c. [Is table retrieval a solved problem? exploring join-aware multi-table retrieval](#). *Preprint*, arXiv:2404.09889.
- Yeounoh Chung, Gaurav T. Kakkar, Yu Gan, Brenton Milne, and Fatma Özcan. 2025. [Is long context all you need? leveraging llm’s extended context for nl2sql](#). *Proceedings of the VLDB Endowment*, 18(8):2735–2747.
- Vijay Prakash Dwivedi, Sri Jaladi, Yangyi Shen, Federico López, Charilaos I. Kanatsoulis, Rishi Puri, Matthias Fey, and Jure Leskovec. 2026. [Relational graph transformer](#). *Preprint*, arXiv:2505.10960.
- Jeffrey Eben, Aitzaz Ahmad, and Stephen Lau. 2025. [Rasl: Retrieval augmented schema linking for massive database text-to-sql](#). *Preprint*, arXiv:2507.23104.
- Jonathan Fürst, Catherine Kosten, Farhad Nooralahzadeh, Yi Zhang, and Kurt Stockinger. 2024. [Evaluating the data model robustness of text-to-sql systems based on real user queries](#). *Preprint*, arXiv:2402.08349.
- Daniel Gomm, Cornelius Wolff, and Madelon Hulsebos. 2025. Are we asking the right questions? on ambiguity in natural language queries for tabular data analysis. In *AI for Tabular Data workshop at EurIPS 2025*.
- Xingyu Ji, Parker Glenn, Aditya G. Parameswaran, and Madelon Hulsebos. 2025. [Target: Benchmarking table retrieval for generative tasks](#). *Preprint*, arXiv:2505.11545.
- 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](#). *Preprint*, arXiv:2004.04906.
- Heegy Kim, Taeyang Jeon, Seunghwan Choi, Seungtaek Choi, and Hyunsouk Cho. 2024. [Flex: Expert-level false-less execution metric for reliable text-to-sql benchmark](#). *Preprint*, arXiv:2409.19014.
- 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)*, pages 6943–6954, Online. Association for Computational Linguistics.
- Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. [Resdsq: Decoupling schema linking and skeleton parsing for text-to-sql](#). *Preprint*, arXiv:2302.05965.
- Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023b. [Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls](#). *Preprint*, arXiv:2305.03111.
- Hsing-Ping Liang, Che-Wei Chang, and Yao-Chung Fan. 2025. [Improving table retrieval with question generation from partial tables](#). *Preprint*, arXiv:2508.06168.

Karime Maamari, Fadhil Abubaker, Daniel Jaroslawicz, and Amine Mhedhbi. 2024. [The death of schema linking? text-to-sql in the age of well-reasoned language models.](#) *Preprint*, arXiv:2408.07702.

Karime Maamari, Connor Landy, and Amine Mhedhbi. 2025. [Genedit: Compounding operators and continuous improvement to tackle text-to-sql in the enterprise.](#) In *Proceedings of the 11th Conference on Innovative Data Systems Research (CIDR)*, Amsterdam, The Netherlands.

Md Mahadi Hasan Nahid, Davood Rafiei, Weiwei Zhang, and Yong Zhang. 2025. [Rethinking schema linking: A context-aware bidirectional retrieval approach for text-to-sql.](#) *Preprint*, arXiv:2510.14296.

Jaydeep Sen, Chuan Lei, Abdul Quamar, Fatma Özcan, Vasilis Efthymiou, Ayushi Dalmia, Greg Stager, Ashish Mittal, Diptikalyan Saha, and Karthik Sankaranarayanan. 2020. [Athena++: natural language querying for complex nested sql queries.](#) *Proc. VLDB Endow.*, 13(12):2747–2759.

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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. [Text embeddings by weakly-supervised contrastive pre-training.](#) *Preprint*, arXiv:2212.03533.

Yihan Wang, Peiyu Liu, and Xin Yang. 2025. [Linkalign: Scalable schema linking for real-world large-scale multi-database text-to-sql.](#) *arXiv preprint arXiv:2503.18596*.

Orion Weller, Michael Boratko, Iftekhar Naim, and Jinhyuk Lee. 2025. [On the theoretical limitations of embedding-based retrieval.](#) *Preprint*, arXiv:2508.21038.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muenighoff, Defu Lian, and Jian-Yun Nie. 2024. [C-pack: Packed resources for general chinese embeddings.](#) *Preprint*, arXiv:2309.07597.

Keyan Xu, Dingzirui Wang, Xuanliang Zhang, Qingfu Zhu, and Wanxiang Che. 2025. [Abacus-sql: A text-to-sql system empowering cross-domain and open-domain database retrieval.](#) *arXiv preprint arXiv:2504.09824*.

Dun Zhang, Jiacheng Li, Ziyang Zeng, and Fulong Wang. 2025. [Jasper and stella: distillation of sota embedding models.](#) *Preprint*, arXiv:2412.19048.

Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. 2024. [Murre: Multi-hop table retrieval with removal](#)

for open-domain text-to-sql. *arXiv preprint arXiv:2402.10666*.

Jiaru Zou, Dongqi Fu, Sirui Chen, Xinrui He, Zihao Li, Yada Zhu, Jiawei Han, and Jingrui He. 2025. [Gtr: Graph-table-rag for cross-table question answering.](#) *arXiv preprint arXiv:2504.01346*.

Appendix

A Query Decomposition Prompt

We use a fixed instruction prompt to extract schema-related query components. The prompt is shown below:

```
---Goal---
Given a natural language query and a
→ list of component types, identify
→ all schema
element components from the text.

---Output---
For each identified component, extract
→ the following information:
- component_name: Name of the component,
→ use exact phrase from the input
→ query.
  Keep each as short and concise as
  → possible.
- component_description: Short
→ description of the component's
→ attributes and any
potential context that might be
→ helpful for understanding the role
→ of the component
within the query.

---Schema Element Component
→ Description---
The specific data schema element (e.g.
→ table or column names) directly
→ referenced
in the query. Think about which parts of
→ the query are most likely to have
→ counterparts
in the available data, and explain why.
→ Be as concise as possible, and shave
→ any
constraints or filtering phrases
→ surrounding the schema element
→ component.

Note: Exclude phrases that are more
→ likely to refer to specific cell
→ values rather than
schema elements. Only focus on elements
→ most likely to have matching schema
→ entries.
```

```
---Examples---
{few_shot_examples}
```

```
---Real Data---
Input Query:
{query}
```

```
Output:
```

B Ablation Study

We study the effect of key hyperparameters (n_groups , $vote_k$, and $expand_groups$) on retrieval performance. Since datasets differ in database count, schema size, and inter-table connectivity, hyperparameter selection is essential for balancing coverage and precision. We focus on $k = 25$, where the effects of grouping and voting are most pronounced and easier to observe.

Results. Table 5 summarizes the ablation results. Across datasets, increasing n_groups and $vote_k$ consistently improves recall up to a saturation point, beyond which gains diminish or disappear. For $k = 25$, optimal performance is typically achieved with n_groups and $vote_k$ in the range of 5–6, while larger values yield no further improvement.

On BEAVER and FIBEN, enabling group expansion substantially degrades performance across all configurations. In contrast, BIRD benefits from group expansion in several settings, often reaching near-perfect recall. These trends are consistent across embedding models, indicating that hyperparameter behavior is largely independent of the embedding choice and instead driven by dataset characteristics.

Ablation results for $k=5,10$ For $k \in 5,10$, we evaluate larger hyperparameter configurations (n_groups , $vote_k$), with settings 1–3 considered for $k = 5$ and 1–4 for $k = 10$, each evaluated both with and without $expand_groups$.

Results in Table 6 show the effect of hyperparameter choices for $k = 5$ and $k = 10$, using only *stella-large*, as no significant differences were observed across models.

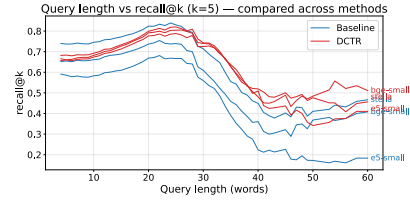
The optimal (n_groups , $vote_k$) configuration varies by dataset. For BEAVER, the best performance is achieved with (1, 1) at $k = 5$ and (2, 2) at $k = 10$. In FIBEN, (1, 1) yields the highest recall for both $k = 5$ and $k = 10$. For BIRD, (3, 3) provides the best results at $k = 5$. Note that BIRD is evaluated on a 200-query subsample, so these optimal values differ when considering the full dataset.

C Results for Detailed Per-Query Analysis for $k=5,10$

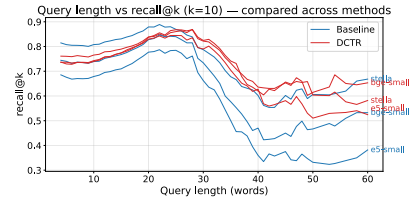
Figures [5, 6, 7, 8] report per-query analyses for both query and data complexity at $k = 5$ and $k = 10$.

In addition to the results discussed in the main text, Figure 7 presents recall as a function of the number of gold tables. As expected, retrieval difficulty increases with the number of required tables, leading to consistent recall degradation. Our method follows a similar trend to the dense baseline, indicating that large multi-table requirements remain challenging.

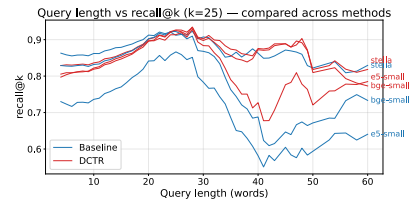
The remaining analyses replicate the trends observed in the main section. On the query complexity side, performance improves with increasing query length and number of extracted components. On the data complexity side, the method is more robust for queries involving densely connected tables, yet performance deteriorates when a large number of gold tables must be jointly retrieved.



(a) CappedRecall@5



(b) CappedRecall@10



(c) CappedRecall@25

Figure 5: Query length vs recall@ k , combined across datasets. Comparison of three embedding models for the proposed method versus the baseline. Across all values of k , the proposed method consistently outperforms the baseline, especially for longer queries (≥ 40 tokens).

D Datasets and Query distribution

Figure 9 shows the distribution of query and data complexity attributes across the evaluated queries. Most queries are shorter than 40 tokens, contain

Dataset ($n_{\text{groups}}, \text{vote}_k$)		Embedding Model					
		expand_groups=False			expand_groups=True		
		stella	bge-small	e5-small	stella	bge-small	e5-small
BEAVER	(3,3)	0.3901	0.3869	0.3782	0.2231	0.2338	0.2155
	(4,4)	0.4106	0.4073	0.3933	0.2306	0.2392	0.2392
	(5,5)	0.4332	0.4321	0.4106	0.2284	0.2306	0.2284
	(6,6)	0.4504	0.4332	0.4138	0.2285	0.2349	0.2371
	(7,7)	0.4569	0.4353	0.3966	0.2381	0.2403	0.2371
FIBEN	(3,3)	0.4526	0.3890	0.4091	0.2324	0.1149	0.1227
	(4,4)	0.4769	0.4051	0.4508	0.2524	0.1271	0.1044
	(5,5)	0.5152	0.4491	0.5178	0.2594	0.1366	0.1131
	(6,6)	0.5466	0.4700	0.5535	0.2672	0.1271	0.1262
	(7,7)	0.5648	0.4795	0.5614	0.2715	0.1393	0.1184
BIRD (200 queries)	(3,3)	0.9214	0.9337	0.9509	1.0000	1.0000	0.9951
	(4,4)	0.9582	0.9754	0.9730	1.0000	1.0000	0.9975
	(5,5)	0.9754	0.9705	0.9902	1.0000	1.0000	0.9951
	(6,6)	0.9754	0.9803	0.9926	1.0000	1.0000	0.9877
	(7,7)	0.9754	0.9779	0.9926	1.0000	0.9926	0.9975

Table 5: Hyperparameter ablation results. Capped Recall@25 for different $(n_{\text{groups}}, \text{vote}_k)$ settings.

Dataset ($n_{\text{groups}}, \text{vote}_k$)		Recall@5		Recall@10	
		False	True	False	True
BEAVER	(1,1)	0.2333	0.0317	0.2575	0.0830
	(2,2)	0.1993	0.0294	0.3308	0.0873
	(3,3)	0.1880	0.0294	0.3114	0.0862
	(4,4)			0.2931	0.0916
FIBEN	(1,1)	0.4626	0.1737	0.4743	0.1766
	(2,2)	0.3763	0.1694	0.4034	0.2066
	(3,3)	0.3477	0.1604	0.4368	0.1988
	(4,4)			0.4359	0.1918
BIRD (200 queries)	(1,1)	0.8354	0.7641	0.8452	0.9165
	(2,2)	0.8697	0.8010	0.9042	0.9754
	(3,3)	0.8747	0.7740	0.9140	0.9754
	(4,4)			0.9435	0.9779

Table 6: Hyperparameter ablation results for stella-large embedding model. Capped Recall@5 and @10 are reported for different $(n_{\text{groups}}, \text{vote}_k)$ settings, with *expand_groups* enabled (True) or disabled (False).

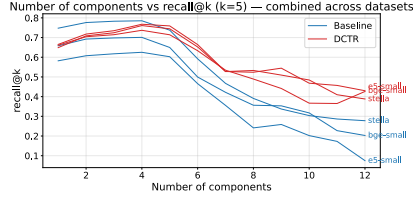
Metric	BEAVER	BIRD	FIBEN
Queries	209	1534	300
Unique DBs	6	11	1
Avg query (w)	31.9	14.6	12.3
Avg SQL (tok)	152.3	25.4	61.9
Tables	463	75	152
Columns	4238	798	373
Avg Tab./DB	77.2	6.8	152.0
Avg Col./Tab.	9.2	10.6	2.5
Avg FK/Tab.	1.1	1.4	1.0

Table 7: Benchmark datasets characteristics.

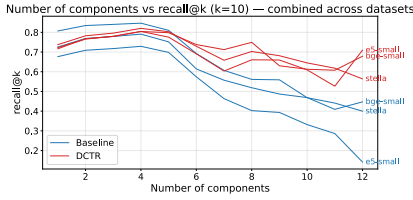
per database than BIRD, making retrieval considerably more challenging in these enterprise-scale settings. In contrast, BIRD exhibits the highest number of foreign keys per database, a structural property that better aligns with our connectivity-aware design and increases the likelihood of benefiting from group expansion.

1-6 extracted components, require 1-4 gold tables, and involve 1-10 join relationships among those tables. These distributions contextualize the per-query analyses in the main paper and clarify the typical structural characteristics of the evaluated workloads.

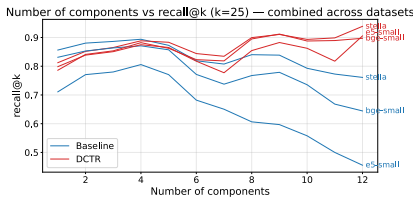
Additionally, Table 7 summarizes benchmark characteristics. BEAVER and FIBEN contain substantially more tables and higher average tables



(a) CappedRecall@5

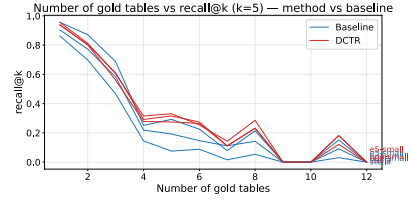


(b) CappedRecall@10

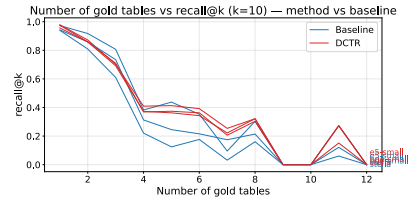


(c) CappedRecall@25

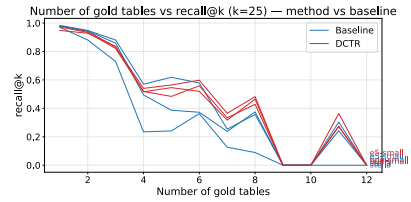
Figure 6: Number of components vs recall@k, combined across datasets. Comparison of three embedding models for the proposed method versus the baseline. Our method consistently performs better than the baseline for queries with more components. The proposed method consistently outperforms the baseline as the number of query components increases. At $k=25$ (Figure 6c), our method maintains stable performance for highly compositional queries, whereas the baseline exhibits a performance drop.



(a) CappedRecall@5

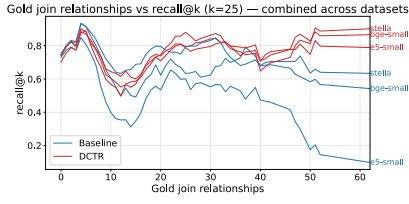


(b) CappedRecall@10



(c) CappedRecall@25

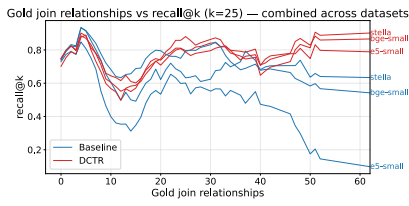
Figure 7: Number of tables gold tables vs recall@k, combined across datasets. Comparison of three embedding models for the proposed method versus the baseline. Both methods exhibit a degradation in capped recall as there the number of gold tables increases. The proposed method behaves similarly to the baseline, indicating limited advantage when there is a lot of relevant tables to retrieve.



(a) CappedRecall@5

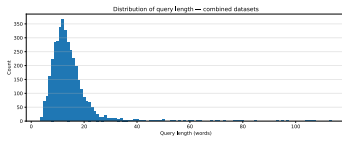


(b) CappedRecall@10

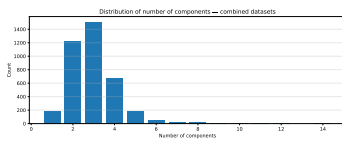


(c) CappedRecall@25

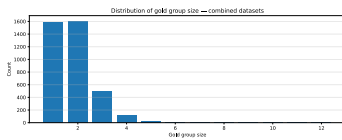
Figure 8: Number of gold tables join relationships vs recall@k, combined across datasets. Comparison of three embedding models for the proposed method versus the baseline. DCTR improves performance on queries for which gold tables are densely connected.



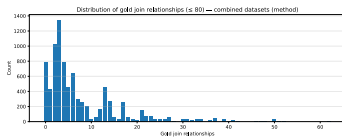
(a) Distribution of query length



(b) Distribution of number of extracted schema/value components



(c) Distribution of average group size of gold tables



(d) Distribution of number of join relationships in gold tables

Figure 9: Distribution of analyzed properties. Results combined for all datasets.

E FK-Expansion Failure Cases

Table 8 presents examples where FK-expansion degrades retrieval on BEAVER. Hub tables such as `SPACE_UNIT` and `LIBRARY_SUBJECT_OFFERED` are connected to many tables via foreign keys. When expansion is enabled, these hub nodes pull in large numbers of unrelated tables, displacing gold tables under the fixed k budget.

#	Query (abridged)	Gold tables	Top-3 after expansion
1	For each course, provide room number, building name, area, organization name, room usage, term code, course level, total subjects and units.	COURSE_CATALOG_ SUBJECT_OFFERED, FAC_ROOMS, FAC_BUILDING, FAC_MAJOR_USE, FAC_BUILDING_ADDRESS, SIS_COURSE_DESCRIPTION (6 tables)	SPACE_UNIT, LIBRARY_SUBJECT_ OFFERED, SPACE_FLOOR
2	Show unique activity titles, locations, term start date, and supervisor name for all independent activities.	IAP_SUBJECT_DETAIL, IAP_SUBJECT_PERSON, ACADEMIC_TERMS_ALL, IAP_SUBJECT_SESSION (4 tables)	SPACE_UNIT, LIBRARY_SUBJECT_ OFFERED, SPACE_FLOOR
3	For each building key, what is the building name and the number of courses offered by the Center for International Studies?	CIS_COURSE_CATALOG, COURSE_CATALOG_ SUBJECT_OFFERED, FCLT_ROOMS, FCLT_BUILDING (4 tables)	FAC_BUILDING, FAC_ROOMS, FCLT_ORGANIZATION
4	For each course, list its title, building name, duration, and a running average of duration partitioned by academic year.	ACADEMIC_TERMS, COURSE_CATALOG_ SUBJECT_OFFERED, FCLT_ROOMS, FCLT_BUILDING (4 tables)	SPACE_UNIT, LIBRARY_SUBJECT_ OFFERED, SPACE_FLOOR
5	For each course, list its name, building name, start date, previous and next course names, access level, and assignable area.	COURSE_CATALOG_ SUBJECT_OFFERED, ACADEMIC_TERMS, FCLT_ROOMS, FCLT_BUILDING (4 tables)	SPACE_UNIT, LIBRARY_SUBJECT_ OFFERED, SPACE_FLOOR
6	Provide information about all physical hosts.	RESOURCE_PROVIDERS, ALLOCATIONS, INVENTORIES, COMPUTE_NODES (4 tables)	PORTS, SUBNETROUTES, SUBNETPOOLPREFIXES
7	For each sponsor of independent activity periods, list sponsor name, number of sessions, total enrollment, min/max fee.	IAP_SUBJECT_SESSION, IAP_SUBJECT_DETAIL, IAP_SUBJECT_SPONSOR (3 tables)	SPACE_UNIT, LIBRARY_SUBJECT_ OFFERED, SPACE_FLOOR
8	What is the term code, term description, total IAP sessions, total fee collected, and min/max enrollment per term?	IAP_SUBJECT_SESSION, IAP_SUBJECT_DETAIL, ACADEMIC_TERMS_ALL (3 tables)	SPACE_UNIT, LIBRARY_SUBJECT_ OFFERED, SPACE_FLOOR
9	For each term, list term code, description, whether current, and total number of CIS course types.	ACADEMIC_TERMS_ALL, ACADEMIC_TERM_PARAMETER, CIS_COURSE_CATALOG (3 tables)	SPACE_UNIT, LIBRARY_SUBJECT_ OFFERED, SPACE_FLOOR
10	Group biology and chemistry courses by current status and department; list max/avg duration and units with subtotals.	SUBJECT_SUMMARY, ACADEMIC_TERMS_ALL, SUBJECT_OFFERED (3 tables)	SIS_COURSE_ DESCRIPTION, ACADEMIC_TERMS, FCLT_ORGANIZATION

Table 8: FK-expansion failure cases on BEAVER. In 7 of 10 cases, the same three hub tables (SPACE_UNIT, LIBRARY_SUBJECT_OFFERED, SPACE_FLOOR) dominate the top-3, displacing all gold tables under the fixed k .