HCVR: HYPOTHESIZE, COMPOSE, AND VERIFY FOR PRECISE MULTI-TABLE RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Text-to-SQL is increasingly deployed over a *pool* of tables, a data-lake-style repository that consolidates many source systems without database boundaries or declared foreign keys. Answering a query then starts with retrieval, which must find both the tables that hold the requested information and the tables needed to join them. Pooled-table retrievers to date rank candidates by how well each table matches the query, and a match alone cannot tell whether a table adds information or a connection that the tables already selected do not. We instead ask what role each table plays in answering the query: a table is a *provider* if it supplies one information requirement, a *connector* if it joins providers that would otherwise remain unjoined, and is excluded if it serves neither role. Under this criterion, competing providers of the same requirement and competing connectors of the same providers are resolved rather than all retained. We propose HCVR (**H**ypothesize, **C**ompose, and **V**erify for multi-table **R**etrieval), a training-free framework that treats every candidate as a hypothesis about its role. HCVR parses the query into typed requirements and retrieves provider candidates for each; it composes providers by bounded beam search and adds the connectors between them along minimum-hop paths in an inferred join graph; and it declares a role absent only after cross-verifying the composed set against an independent LLM selection. The whole procedure takes two to three LLM calls per query. On pooled BIRD, Spider, and MMQA with two retrieval LLMs, HCVR exceeds the best baseline in every setting by 10.7 to 31.2 precision points and 8.0 to 25.5 F1 points, while the fraction of queries whose gold tables are all retrieved stays within 3.9 points of the best baseline. With the retrieved tables frozen, the execution accuracy of two downstream SQL-generation LLMs improves on every corpus, by up to 11.8 points.

1 INTRODUCTION

A text-to-SQL system answers a natural-language query by executing SQL over a database, yet real deployments increasingly place tables from many source systems into one corpus, of which only a few are relevant to any query (Wang et al., 2025). We call such a corpus a table *pool*: its tables are indexed together, but their source-database boundaries and declared foreign keys are unavailable (Zhu et al., 2019; Nargesian et al., 2019; Cong et al., 2022). Answering a query over a pool therefore starts with a *retrieval* stage that must find the tables that hold the requested information and the tables needed to join them. This stage bounds everything downstream: a missing table makes correct SQL impossible, and every extra table offers the *SQL generator*, the LLM that writes the SQL from the retrieved tables, wrong joins and columns to choose from (Soliman et al., 2026).

The pooled-table retrievers evaluated so far, JAR, ARM, REAR, and CORE-T (Chen et al., 2024; 2025; Agarwal et al., 2026; Soliman et al., 2026), formulate selection as a matching problem: *how well does each table match the query?* They score each table for semantic relevance to the query and for join compatibility with other tables, and return the top-scoring tables by combinatorial optimization, encoder reranking, or LLM selection. These scores do not explicitly assess whether a table is still needed given what the other selected tables already cover, so a well-scoring table may remain even when it provides no additional information or necessary connection. Consider "Which courses is Alice enrolled in?" over a pooled university schema (Figure 1(a)). The query needs `student`, `course`, and `enrollment`, which joins them. A score-based retriever retains three further tables:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

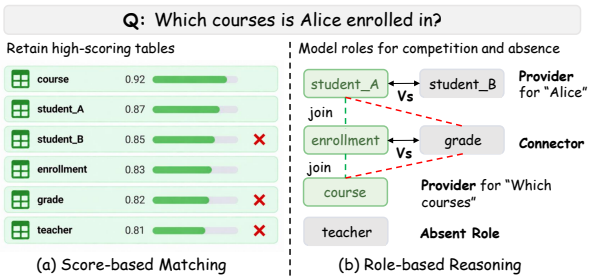


Figure 1: Score-based matching versus role-based reasoning for "Which courses is Alice enrolled in?". (a) Matching asks how well each table matches the query and retains `student_B`, `grade`, and `teacher` next to the three needed tables. (b) Role reasoning asks what role each table plays, keeps one provider per requirement and one connector per join, and drops tables with no role.

a second `student` table from another source system, because it is equally relevant; `grade`, because it joins `student` and `course` just as well; and `teacher`, because it is tied to `course`. All three match, yet none is needed: the first three tables already supply every piece of information and every join the answer requires. This is the rule rather than a corner case: CORE-T, the strongest of these retrievers overall, returns 37 to 62% unused tables per query across three pooled corpora and two retrieval LLMs (Table 1).

We take a different perspective and ask ***what role does this table play in answering the query?*** A table is a *provider* if it supplies an *information requirement*, a piece of information the query asks for or constrains; it is a *connector* if it joins providers that would otherwise remain unjoined; a table that does neither has no place in the answer. Roles resolve the example directly (Figure 1(b)). The two `student` tables compete for the same requirement, so only one is kept. `grade` and `enrollment` compete to connect the same providers, so a single connector suffices and the better-matching `enrollment` is kept. `teacher` plays neither role and is removed.

Deciding roles is harder than scoring matches because no role is a property of a table by itself: the provider role depends on how the query is read, the connector role on which other tables are selected, and the absent role on evidence that retrieval cannot see. **C1: Ambiguity of the provider role.** A query states its information requirements in natural language, not in terms of the tables that hold them, and some requirements are implicit: "Alice" calls for a table of student names although neither a table nor an attribute is named. Concept–attribute pairs (Chen et al., 2024) and natural-language sub-queries (Chen et al., 2025) carry no type stating how a piece of information participates in the query, so a filter, an aggregate, and a requested attribute over the same entity are not told apart when candidate tables are sought for them. A requirement that is not aligned with a table is then served by the wrong table or by none. **C2: Entanglement of the connector role.** Whether a table is a connector depends on the whole provider set and on the join graph, and neither is given. Provider combinations grow exponentially with the number of requirements, and joins must be inferred from schemas, so the inferred graph can be incomplete or wrong and a connector can be missed or a false one added. **C3: Unverifiability of the absent role.** A table that plays neither role must be removed, yet no evidence available at retrieval time can certify that a role is truly absent rather than missed: complete requirements, declared foreign keys, and execution results are all out of reach. Rechecking a selection with the evidence that produced it cannot expose that evidence's own errors, so a second judgement must rest on evidence the first did not use.

HCVR (**H**ypothesize, **C**ompose, and **V**erify for multi-table **R**etrieval) follows this perspective: relevance still proposes and ranks candidates, but every candidate enters as a hypothesis about its role, and only tables whose role survives are returned. Its three stages meet the three challenges in turn. **Hypothesize** makes provider ambiguity explicit (C1): an LLM parses the query into typed *entity–predicate–attribute* triples, one per information requirement and aligned with how schemas organize entities and attributes, and dense retrieval proposes candidates for each, so every table is proposed as the provider of a named requirement next to its alternatives. **Compose** untangles the connector role (C2): a bounded beam search selects one provider per requirement, and minimum-hop paths over an inferred join graph add a connector only where the chosen providers would otherwise remain unjoined. **Verify** requires two independent judgements before a role is declared absent (C3): the composed set is compared with an LLM selection that sees neither the search's assignments nor its ranks, disagreements are reconciled by their set relation, and a dedicated deletion review is invoked when the LLM selection strictly extends the composed set.

We evaluate HCVR on pooled BIRD (Li et al., 2023b), Spider (Yu et al., 2018), and MMQA (Wu et al., 2025) with two retrieval LLMs against six retrieval baselines. In all six settings HCVR attains the highest precision and F1, exceeding the best baseline by 10.7 to 31.2 and 8.0 to 25.5 points, while its perfect recall (the fraction of queries whose gold tables are all retrieved) is the highest in three settings and within 3.9 points of the best elsewhere. With the retrieved tables frozen, it also yields the highest SQL execution accuracy on every corpus with both SQL generators.

Our contributions are:

1. **A role-based criterion for pooled-table retrieval.** Every returned table must provide an information requirement or connect providers; relevance ranks the alternatives for each role but does not decide membership.
2. **HCVR, a training-free procedure for assigning and checking roles.** Hypothesize makes provider ambiguity explicit, Compose searches jointly for providers and connectors, and Verify declares a role absent only after two independent judgements are reconciled.
3. **Evidence that precise retrieval helps SQL generation without sacrificing coverage.** Across three pooled corpora, two retrieval LLMs, and two SQL generators, HCVR returns the smallest table sets in every setting while attaining the highest F1, and execution accuracy rises by 0.5 to 11.8 points.

2 Related work

Closed-book table retrieval. Most table retrieval for text-to-SQL assumes a *closed-book* setting in which the target database and its schema graph are given, so the task reduces to schema linking within one database (Wang et al., 2020; Li et al., 2023a; Talaei et al., 2024), often with dataset-specific training that weakens transfer to unseen databases (Hao et al., 2025). Neither assumption holds over a pool, where tables from heterogeneous sources are indexed without database boundaries or declared foreign keys, the *open-book* setting of Soliman et al. (2026); HCVR targets this setting without training. Appendix A surveys schema linking, question decomposition, data discovery, and verification in more detail.

Score-based multi-table retrieval. Score-based pooled-table retrievers decide membership by score, without explicitly assessing whether a table is still needed given the requirements and joins already covered by the other selected tables. JAR and ARM combine relevance and joinability scores by mixed-integer programming over inferred join edges (Chen et al., 2024; 2025); REAR replaces the program with cross-encoder reranking (Agarwal et al., 2026); CORE-T adds an LLM selector guided by these scores (Soliman et al., 2026). All four return the top-scoring set without checking whether two retained tables serve the same requirement or the same join, so competing providers, competing connectors, and role-less tables are retained together. HCVR instead makes the role of each table explicit, so that competing providers and connectors are resolved and role-less tables are dropped, and it cross-verifies the LLM selection rather than trusting it directly.

3 HCVR: Hypothesize, Compose, and Verify

HCVR reduces retrieval to three role decisions: which tables can provide each requirement (Hypothesize), which providers and connectors form a joinable set (Compose), and whether a table outside that set truly has no role (Verify); Figure 2 traces them on the running example.

3.1 Problem Definition and Preliminaries

Problem definition. Given a natural-language query q and a table pool $\mathcal{T} = \{t_1, \ldots, t_N\}$, the goal is to retrieve a compact subset $F(q) \subseteq \mathcal{T}$ containing the tables required for downstream text-to-SQL generation. Source-database identifiers and declared foreign-key relations are unavailable. Each retrieved table must be a provider or a connector as defined in Section 1.

Offline preprocessing. Following CORE-T (Soliman et al., 2026), we generate a brief purpose description for each table with an LLM and combine it with the table name and schema to form

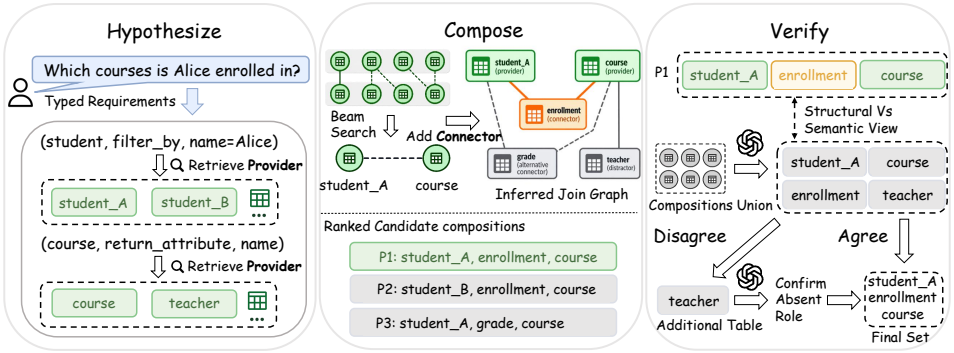


Figure 2: Overview of HCVR on "Which courses is Alice enrolled in?". **Hypothesize** extracts typed requirements and retrieves candidate providers for each; **Compose** searches over provider assignments and adds connectors along an inferred join graph, producing ranked compositions $P_1, P_2, \ldots$; **Verify** compares P_1 with an independent LLM selection over the union of compositions and resolves the absent role (here, `teacher`) by cross-verification.

its textual representation. Let f denote the shared embedding model that maps textual query representations and table representations into one embedding space, and let $s(x,t) = \cos(f(x), f(t))$ denote their semantic relevance. For each table pair (t,u) we also compute a compatibility score $c(t,u) \in [0,1]$ that estimates their joinability from column-level value overlap and header similarity, restricted to column pairs that satisfy uniqueness and containment conditions (Appendix B.2).

3.2 Hypothesize: Identifying Provider Candidates

Hypothesize turns the query into provider-role hypotheses: an LLM extracts one typed *entity–predicate–attribute* triple per information requirement, at a granularity a single table can provide, and each requirement then retrieves its own candidate providers.

Requirement representation. Each information requirement is a triple $r_i = \langle e_i, \rho_i, a_i \rangle$: the entity e_i is the object or domain the requirement concerns, the attribute a_i is the information sought or constrained, with explicit values and conditions attached, and the predicate ρ_i states how that information participates in the query. The entity–attribute pair is the unit of alignment with relational schemas, the entity corresponding to a table and the attribute to a column. Keyword-level fragments would leave generic attributes such as *name* without their entity, and entity-only fragments would merge information about one entity that is spread over several tables. When an entity or attribute is implicit, the LLM infers it from context: in Figure 2, the requirement on Alice becomes $(\textit{student}, \textit{filter_by}, \textit{name} = \textit{Alice})$, although neither *student* nor *name* is mentioned.

Requirement coverage. The predicate ρ_i is drawn from a typed inventory with two groups, *constraint predicates* for filtering, grouping, aggregation, ordering, and existence conditions, and *answer predicates* for the entities, attributes, or quantities the query returns (Appendix B.3). The type is what separates a filter, an aggregate, and a requested attribute over the same entity; dropping it gives the untyped variant of Section 4.3. Because extraction can still omit information, the original query is retained as a *global requirement* alongside the extracted ones.

Provider hypothesis formation. Each requirement r_i is rendered as a retrieval fragment x_i that preserves its entity, attribute, predicate, and conditions, and is embedded with the same model f as the tables. Tables are ranked by $s(x_i, t)$ and the top K are kept:

$$C_i = \mathrm{TopK}_{t \in \mathcal{T}}\, s(x_i, t), \qquad \mathcal{H}_i = \{(r_i, t) : t \in C_i\}. \tag{1}$$

Each pair (r_i, t) is the hypothesis that table t provides requirement r_i. Hypotheses for one requirement are competing providers; hypotheses across requirements are complementary, and one table may provide several requirements. The global requirement uses the query itself as its fragment, $x_g = q$, so its candidates reflect whole-query relevance; Compose treats it as one more requirement.

3.3 COMPOSE: COMBINING PROVIDERS AND INFERRING CONNECTORS

Compose selects one provider per requirement and adds the connectors those providers need, although the connector role depends on a provider set not yet known and the joins must be inferred (C2).

Bounded hypothesis search. A complete composition selects one hypothesis from each $\mathcal{H}_i$. Rather than enumerating up to K^n combinations, we run beam search of width K, so the beam can hold every candidate of a single requirement while the search cost grows linearly in n. With requirements indexed $1, \ldots, n$, a partial assignment after the first d requirements is $a_d = \{(r_i, t_i)\}_{i=1}^{d}$ with $(r_i, t_i) \in \mathcal{H}_i$; it induces the table set $T(a_d) = \{t_i : i \leq d\}$, in which a table providing several requirements appears once. We score a partial assignment by requirement-level relevance ℓ and whole-query relevance g, combined with a weight $\alpha \in [0, 1]$:

$$S(a_d) = \alpha\, \ell(a_d) + (1-\alpha)\, g(a_d), \qquad \ell(a_d) = \frac{1}{d} \sum_{i=1}^{d} s(x_i, t_i), \qquad g(a_d) = \frac{1}{|T(a_d)|} \sum_{t \in T(a_d)} s(q, t). \tag{2}$$

The two terms guard against different failures: requirement-level relevance alone admits tables that match a fragment lexically but not the query's context, whereas whole-query relevance alone favors the tables most representative of the query and overlooks providers of specific requirements such as a filter condition (Section 4.3).

State compression. Because one table can provide several requirements, different assignments often induce the same table set and differ only in which requirement each table is credited with. Keeping them separately would spend beam slots on duplicates, so before applying the beam constraint we group assignments by induced table set and keep the highest-scoring one per set. The beam then holds K distinct table combinations rather than K relabelings of a few.

Connector inference. The providers of a completed assignment need not be joinable directly, so we construct a query-specific *inferred join graph*. Let $C_q = \bigcup_i C_i$ be the union of provider candidates; we add their one-hop neighbors whose compatibility reaches a threshold τ ($\tau = 0.3$) and define $\mathcal{G}_q = (V_q, \mathcal{E}_q)$ as

$$V_q = C_q \cup \{u \in \mathcal{T} : \exists\, t \in C_q,\ c(t, u) \geq \tau\}, \qquad \mathcal{E}_q = \{(t, u) : t, u \in V_q,\ c(t, u) \geq \tau\}. \tag{3}$$

For each completed assignment the providers $T(a_n)$ are terminals in $\mathcal{G}_q$; we connect reachable terminals by minimum-hop paths of at most $h = 5$ hops and include the intermediate tables as connectors. We minimize hops rather than maximize compatibility so that each connection crosses as few inferred edges as possible, each of which can be wrong (C2); alternative paths of equal length yield separate compositions, so no composition carries two connector sets for the same terminals. Because inferred edges can also be missing, terminals that cannot be connected are left unconnected, and the composition is kept rather than discarded.

Ranking composed table sets. Completed compositions are scored with Equation 2, where the whole-query term now averages over providers and connectors alike; compositions that yield the same table set are merged, and the top K are retained as $\mathcal{P} = (P_1, \ldots, P_J)$ with $J \leq K$. The highest-ranked composition P_1 is passed to Verify.

3.4 VERIFY: CROSS-VERIFICATION FOR THE ABSENT ROLE

The composed set P_1 inherits any omission in requirement extraction and any error in join inference, so a table missing from P_1 is either irrelevant or was missed for one of these reasons, and P_1 alone cannot tell which. Verify resolves this by comparing P_1 with a semantic selection that reuses neither the requirement assignments nor the ranking that produced P_1 (Algorithm 1).

Independent semantic judgement. Let $U = \bigcup_{j=1}^{J} P_j$ be the union of the retained compositions. An LLM selects a required set $R \subseteq U$ (SELECT) from the query, the textual representations of the tables in U, and, as in CORE-T's selector, their pairwise compatibility records (Figure 4); it sees neither the requirement assignments, nor the composition scores, nor which tables form P_1.

Algorithm 1 Verify: cross-verification for the absent role. Input: query q, union U of the retained compositions, top composition P_1. SELECT and REVIEW are LLM calls; ADD is deterministic.

$R \leftarrow \textsc{Select}(q, U)$ ▷ semantic selection
if $R = P_1$ **then return** R ▷ agree
if $P_1 \subsetneq R$ **then**
  if $\textsc{Review}(q, R, P_1) = \texttt{DELETE}$ **then return** P_1
  return R
if $R \subsetneq P_1$ **then return** P_1
if $R \cap P_1 \neq \varnothing$ **then return** $\textsc{Add}(R \cup P_1, U)$ ▷ overlap
return $\textsc{Add}(R, U)$ ▷ disjoint

procedure $\textsc{Add}(F, U)$ ▷ inferred joins
  $F' \leftarrow F$
  for all $t \in F$ **do**
    $u^\star \leftarrow \arg\max_{u \in U \setminus F'} c(t, u)$
    if $c(t, u^\star) \geq \tau$ **then** $F' \leftarrow F' \cup \{u^\star\}$
  end for
  return F'
end procedure

Table 1: Multi-table retrieval on pooled BIRD, Spider, and MMQA with two retrieval LLMs. $\bar{G}$ and $\bar{T}$: mean gold and retrieved tables per query; P and F1 are averaged over queries; PR: perfect recall, the fraction of queries whose retrieved set contains all gold tables. Best in bold, second best underlined; "–": not supported by the baseline's release (Appendix D).

	BIRD ($\bar{G} = 1.95$)				Spider ($\bar{G} = 1.51$)				MMQA ($\bar{G} = 2.20$)			
Method	$\bar{T}\downarrow$	P↑	F1↑	PR↑	$\bar{T}\downarrow$	P↑	F1↑	PR↑	$\bar{T}\downarrow$	P↑	F1↑	PR↑
Llama-3.1-8B-Instruct												
DR@5	5.0	34.9	49.4	83.0	5.0	29.4	43.9	95.6	5.0	30.8	42.6	50.3
ReAct	4.8	43.1	55.6	86.1	4.5	39.7	52.2	92.8	6.1	29.6	40.4	53.1
JAR@5	5.0	35.5	50.1	86.0	5.0	29.6	44.2	96.7	–	–	–	–
ARM	5.3	40.4	53.5	**90.9**	–	–	–	–	–	–	–	–
REAR	5.0	36.1	51.0	88.3	5.0	28.2	42.1	90.8	5.0	28.4	39.2	47.8
CORE-T	4.1	50.0	62.3	90.0	4.2	40.2	53.8	96.6	5.1	38.4	49.2	61.3
HCVR	**3.2**	**64.4**	**72.2**	87.0	**2.4**	**71.4**	**79.3**	**97.0**	**4.2**	**49.1**	**57.2**	**63.5**
Qwen2.5-7B-Instruct												
DR@5	5.0	35.2	49.8	84.2	5.0	29.6	44.1	**96.4**	5.0	31.1	43.0	51.0
ReAct	3.3	65.6	71.6	81.3	2.7	67.6	75.5	93.4	4.9	42.1	50.4	55.4
JAR@5	5.0	35.5	50.1	86.0	5.0	29.5	44.1	96.3	–	–	–	–
ARM	3.5	59.2	68.3	84.6	–	–	–	–	–	–	–	–
REAR	5.0	36.1	51.0	**88.3**	5.0	28.2	42.1	90.8	5.0	28.4	39.2	47.8
CORE-T	3.1	63.1	72.3	87.0	3.0	54.4	66.8	94.9	4.1	48.3	56.4	59.4
HCVR	**2.3**	**80.5**	**83.1**	84.6	**1.8**	**86.3**	**89.2**	93.0	**3.2**	**60.5**	**64.9**	**60.4**

Resolving the absent role. Algorithm 1 decides the final set F by the set relation between R and P_1. If the two agree, the set is final. If $P_1 \subsetneq R$, the extra tables in R either recover roles that extraction or join inference missed or are tables that the semantic selection over-selected. A dedicated LLM review (REVIEW, Figure 5) answers `DELETE` only if every extra table lacks a unique contribution and every requirement remains served by P_1; then P_1 is returned, and otherwise R is kept. If $R \subsetneq P_1$, we keep P_1, because the tables it holds beyond R are the providers or connectors the composition needs to stay joinable. If the two overlap partially, each may hold roles the other lacks, so we keep $R \cup P_1$ and apply *inferred-join addition* (ADD) within U: each retained table is augmented with its most compatible neighbor whose compatibility reaches τ, which recovers connectors between the selected tables. If the two are disjoint, their disagreement lends the composition no support, so we start from R and apply the same addition, because R was not built to be joinable.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and metrics. We evaluate on pooled BIRD (Li et al., 2023b), Spider (Yu et al., 2018), and MMQA (Wu et al., 2025) under CORE-T's pooled protocol (Soliman et al., 2026): 75, 81, and 710 tables and 1,534, 1,034, and 1,105 queries (Appendix C). We report precision (P) and F1 averaged over queries, perfect recall (PR), the fraction of queries whose retrieved set contains all gold tables, and the mean number of retrieved tables per query ($\bar{T}$). For downstream text-to-SQL we

Table 2: Execution accuracy (%) of two SQL generators on the tables retrieved with Llama-3.1-8B-Instruct. all: $\mathrm{EX}_{\mathrm{all}}$ (all queries); PR: $\mathrm{EX}_{\mathrm{PR}}$ (fraction of all queries with perfect-recall retrieval and correct execution). Oracle uses gold tables; best retriever in bold.

	Llama-3.2-3B-Instruct						GPT-4o-mini					
	BIRD		Spider		MMQA		BIRD		Spider		MMQA	
Retriever	all	PR	all	PR	all	PR	all	PR	all	PR	all	PR
DR@5	4.0	3.9	13.9	13.9	2.1	1.8	40.0	37.7	65.4	64.7	31.7	27.8
ReAct	10.8	10.4	20.0	20.0	2.5	2.2	42.5	40.9	63.7	63.4	33.1	28.7
JAR@5	8.3	8.1	14.4	14.4	–	–	41.6	40.4	66.0	65.7	–	–
ARM	5.1	5.0	–	–	–	–	42.3	41.5	–	–	–	–
REAR	10.0	9.8	29.6	29.3	8.3	6.9	42.3	40.6	62.3	61.7	29.0	25.2
CORE-T	16.6	16.2	34.5	34.1	18.8	15.8	43.0	41.6	66.7	66.2	35.8	33.3
HCVR	**18.2**	**17.9**	**46.3**	**46.1**	**20.6**	**19.0**	**43.5**	**42.5**	**68.1**	**67.8**	**38.7**	**35.8**
Oracle	27.8	27.8	58.9	58.9	45.4	45.4	50.7	50.7	71.8	71.8	65.7	65.7

report execution accuracy over all queries ($\mathrm{EX}_{\mathrm{all}}$) and $\mathrm{EX}_{\mathrm{PR}}$, the fraction of all queries for which retrieval had perfect recall and the generated SQL executed correctly; the difference $\mathrm{EX}_{\mathrm{all}} - \mathrm{EX}_{\mathrm{PR}}$ is the accuracy obtained despite incomplete retrieval.

Baselines. We compare with six multi-table retrieval methods: dense retrieval of the top five tables (DR@5) (Karpukhin et al., 2020); ReAct (Yao et al., 2023), which retrieves through interleaved reasoning and actions; JAR with five returned tables (JAR@5) (Chen et al., 2024) and ARM (Chen et al., 2025), which select tables by combinatorial optimization; REAR (Agarwal et al., 2026), which reranks with an encoder; and CORE-T (Soliman et al., 2026), which selects tables with an LLM and adjusts the selection by compatibility. Baseline scores are those published in CORE-T's evaluation suite under the same pooled protocol, which uses the same embedding model and enriched table representations as HCVR; our own CORE-T runs reproduce its published PR within 0.0 to 4.5 points (Appendix D). JAR and ARM are reported only on the corpora their releases support (Appendix D).

Implementation details. UAE-Large-V1 (Li & Li, 2024) is the shared embedding model, $K = 10$ candidates are kept per requirement, $\alpha = 0.5$, and all LLM calls use temperature 0. Retrieval is run with Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and Qwen2.5-7B-Instruct (Qwen Team et al., 2025). For SQL generation we freeze the Llama-based retrieval outputs and generate with Llama-3.2-3B-Instruct and GPT-4o-mini (OpenAI, 2024). All numbers come from a single deterministic run (temperature 0) per setting over 1,534, 1,034, and 1,105 queries; paired significance tests against reproduced CORE-T are reported in Appendix E.9. Remaining settings are listed in Appendix B.

4.2 Main Results

HCVR retrieves the most precise sets without giving up coverage. HCVR has the highest precision and F1 in all six settings of Table 1, exceeding the best baseline in each by 10.7 to 31.2 precision points and 8.0 to 25.5 F1 points. Against our reproduced CORE-T, every precision and F1 gain is significant (Holm-adjusted $p < 0.001$, paired permutation tests), whereas PR is significantly lower only on BIRD with Llama and significantly higher only on MMQA with Llama (Appendix E.9). Its sets are also the smallest and the closest to gold size: with Qwen it returns 2.3, 1.8, and 3.2 tables per query against 1.95, 1.51, and 2.20 gold tables, where CORE-T returns 3.1, 3.0, and 4.1. Compactness does not come from aggressive pruning: PR is the highest in three settings and within 3.9 points of the best baseline in the other three. ReAct under Qwen shows the trade-off HCVR avoids; it reaches the second-best precision on BIRD with 3.3 tables but the lowest PR of all LLM-based methods (81.3), whereas HCVR is both smaller (2.3) and more complete (84.6). The cost is 2.5 to 2.8 LLM calls per query; total tokens are 2.3 to 2.7 times CORE-T's and about a third below ReAct's (Appendix E.11).

Explicit roles, not a stronger scorer, account for the gain. HCVR shares the baselines' embedding model and table representations, so the difference lies in how membership is decided: score-

Table 3: Composition quality before verification with Llama-3.1-8B-Instruct. Exact@1 / Exact@10: gold set equals P_1 / appears among the at most ten retained compositions; $\overline{|U|}$: mean union size; PR_U: gold set within U. Higher is better except for $\overline{|U|}$; best in bold, second best underlined.

	BIRD				Spider				MMQA			
Variant	Exact@1	Exact@10	$\overline{\lvert U\rvert}\downarrow$	$\mathrm{PR}_U\uparrow$	Exact@1	Exact@10	$\overline{\lvert U\rvert}\downarrow$	$\mathrm{PR}_U\uparrow$	Exact@1	Exact@10	$\overline{\lvert U\rvert}\downarrow$	$\mathrm{PR}_U\uparrow$
C1: Requirement formulation (provider ambiguity)												
JAR-style decomposition	28.4	53.0	8.7	**94.5**	44.7	70.3	8.8	98.1	14.0	29.5	10.2	64.9
Sub-query decomposition	31.7	69.3	7.7	93.7	49.5	86.6	8.0	98.1	15.4	45.2	8.5	63.3
Untyped decomposition	28.5	61.5	8.1	93.0	46.9	75.2	8.3	97.9	15.8	35.7	9.5	65.6
C2: Composition (connector entanglement)												
w/o state compression	33.6	61.9	**5.6**	86.5	55.2	76.6	**5.2**	93.8	18.6	39.1	7.2	60.1
w/o connector inference	33.1	**78.8**	6.2	92.4	53.5	**88.6**	6.7	**98.3**	17.1	**47.1**	**7.0**	62.8
Requirement-only scoring	**34.5**	61.3	9.1	91.5	50.5	77.4	8.7	96.8	15.5	31.6	10.5	61.6
Whole-query-only scoring	21.6	70.0	7.2	92.2	46.3	86.9	7.5	97.5	9.5	44.3	8.1	63.9
HCVR (full)	33.8	72.0	7.6	94.4	**55.3**	86.9	8.0	**98.3**	**18.8**	46.2	8.5	**67.1**

based methods (DR, JAR, REAR) return a fixed number of top-ranked tables, which caps their precision at 28.2 to 36.1 points, while LLM-based methods (ReAct, ARM, CORE-T) keep roles implicit in the LLM reasoning and either leave them unverified or rely on brittle checks without explicit role grounding. HCVR names each role, resolves its competing candidates, and drops tables that hold none. Table 9 in Appendix E.1 supports this role-based criterion: Exact, which requires the retrieved set to match the gold tables exactly, improves across all datasets and table-count groups under both models. For single-table queries, excluding tables without a necessary role is reflected in pooled Exact increasing from 0.7% to 38.7% with Llama and from 6.5% to 64.6% with Qwen, while PR remains nearly unchanged. For queries requiring at least three tables, pooled PR rises from 58.6% to 64.7% and from 45.0% to 55.2%, consistent with preserving complementary providers and the connectors needed to join them.

Precise retrieval carries through to SQL execution. With the retrieved tables frozen, HCVR yields the highest $\mathrm{EX}_{\mathrm{all}}$ and $\mathrm{EX}_{\mathrm{PR}}$ in all six corpus–generator settings of Table 2. The gain is largest on Spider with Llama-3.2-3B-Instruct (11.8 $\mathrm{EX}_{\mathrm{all}}$ points) and ranges from 0.5 to 2.9 points with GPT-4o-mini; it is significant on Spider and MMQA with both generators (McNemar $p \leq 0.0126$) (Appendix E.9). The smaller Qwen-based retrieval table sets also yield consistent improvements over CORE-T, with $\mathrm{EX}_{\mathrm{all}}$ gains of 4.1–4.2 points on BIRD and MMQA with Llama-3.2-3B-Instruct and 3.1–3.2 points with GPT-4o-mini, including significant gains on BIRD with both generators ($p < 0.001$; Appendix E.7 and E.9). The gain comes from the tables HCVR leaves out rather than the tables it adds. On BIRD its PR is 3.0 points below CORE-T's (Table 1), yet its execution accuracy is higher with both generators; on Spider, where the two have nearly equal PR, the gain with Llama-3.2-3B-Instruct is the largest of all. Nearly all of HCVR's correct executions come from queries with complete retrieval ($\mathrm{EX}_{\mathrm{all}} - \mathrm{EX}_{\mathrm{PR}}$ is at most 2.9 points), and the gold-table oracle still leads by 3.7 to 27.0 points, so retrieval remains the bottleneck.

4.3 Ablation Study

Each ablation removes the mechanism aimed at one challenge, so the cost of provider ambiguity (C1), connector entanglement (C2), and the unverifiable absent role (C3) is measured separately; all runs use Llama-3.1-8B-Instruct. For C1 and C2 we evaluate the compositions before verification, which isolates them from the later LLM decisions. Exact@1 is the fraction of queries whose top composition P_1 equals the gold set; Exact@10 the fraction whose gold set appears among the at most ten retained compositions; $\overline{|U|}$ the mean size of the union U passed to Verify; and PR_U the fraction of queries whose gold set lies within U. A compact U eases semantic selection, while a high PR_U lets Verify still recover the gold set; final retrieval results for the same variants are in Appendices E.2 and E.3. For C3 we evaluate the final sets with P, F1, PR, and $\bar{T}$.

C1, provider ambiguity: typed requirements give providers the right granularity. Table 3 compares three alternative formulations: JAR-style concept–attribute decomposition (Chen et al., 2024), sub-query decomposition as used by ARM baseline (Chen et al., 2025), and untyped decomposition, which keeps free-form requirements but drops the typed triple. Typed triples give the best

Table 4: Verification ablations on the final retrieved sets with Llama-3.1-8B-Instruct. Direct P_1 refinement and R + LLM refinement: one additional LLM call that revises P_1 (respectively R) over the candidates in U (Figure 6). Higher is better except for $\bar{T}$; best in bold, second best underlined.

	BIRD				Spider				MMQA			
Variant	$\bar{T}$	P	F1	PR	$\bar{T}$	P	F1	PR	$\bar{T}$	P	F1	PR
P_1 only	**1.6**	**82.0**	69.6	43.6	**1.4**	**85.2**	**79.8**	67.6	**2.2**	**61.5**	54.8	28.4
R only	3.3	60.0	69.9	**87.1**	2.9	55.0	66.9	<u>95.9</u>	4.0	48.4	<u>56.5</u>	<u>61.5</u>
Direct P_1 refinement	<u>2.4</u>	<u>68.3</u>	<u>71.9</u>	67.5	<u>2.1</u>	66.1	72.5	81.4	<u>2.7</u>	<u>52.8</u>	55.8	42.0
R + LLM refinement	2.5	66.0	70.5	69.4	2.2	63.2	69.7	79.3	2.8	<u>52.8</u>	56.2	43.3
HCVR (full)	3.2	64.4	**72.2**	<u>87.0</u>	2.4	<u>71.4</u>	<u>79.3</u>	**97.0**	4.2	49.1	**57.2**	**63.5**

Exact@1 of the four formulations on all three corpora and raise Exact@10 over untyped decomposition by 10.5, 11.7, and 10.5 points on BIRD, Spider, and MMQA, so the type, not decomposition alone, is what the untyped variant lacks. JAR-style decomposition trails by 16.6 to 19.0 Exact@10 points. Sub-query decomposition comes within 0.3 to 2.7 Exact@10 points, yet after verification it loses 6.7, 7.2, and 10.3 PR points to typed triples (Appendix E.2).

C2, connector entanglement: composition keeps distinct alternatives and completes their connectors. Removing state compression leaves Exact@1 almost unchanged but lowers PR_U by 7.9, 4.5, and 7.0 points, because relabelings of one table set crowd out genuinely different compositions. After verification this variant has higher precision and F1 on BIRD (70.6 and 74.7 against 64.4 and 72.2) but loses 8.3 to 11.2 PR points (Appendix E.3). Since correct executions almost all come from queries with complete retrieval (Table 2), the PR loss is the one that reaches the SQL. Removing connector inference raises Exact@10, since the retained compositions then differ in providers rather than in connector paths, but lowers Exact@1 on all corpora and PR_U on BIRD and MMQA, and the final sets lose 6.9 to 10.5 PR points. Each scoring term alone fails in the way predicted in Section 3.3: requirement-only scoring enlarges U (9.1 against 7.6 tables on BIRD) without raising PR_U, and whole-query-only scoring drops Exact@1 by 9.0 to 12.2 points because providers of specific requirements are outranked by broadly relevant tables.

C3, the absent role: cross-verification resolves it better than either view alone. Table 4 separates the two views. Disagreement is the common case: with Llama, $P_1 \subsetneq R$ holds for 70 to 82% of queries and the two agree on only 9 to 12% (Appendix E.6), so the deletion review decides most queries. P_1 alone is the most precise but cannot tell a missed role from an absent one: its PR is 43.6, 67.6, and 28.4 on BIRD, Spider, and MMQA. R alone is a CORE-T-style selector applied to U instead of to the top dense-retrieval candidates, and it already exceeds CORE-T by 7.3 to 13.1 F1 (Table 1); the candidate space built by Hypothesize and Compose is therefore a large part of the gain. Without explicit roles, however, R keeps redundant tables, and its precision trails the full method by 4.4, 16.4, and 0.7 points. A further LLM pass that refines either view over U does not combine the precision of P_1 with the coverage of R, because it cannot expose the omissions of the view it starts from. Both refinements land between the extremes, with PR of 67.5 to 81.4 on BIRD and Spider and 42.0 to 43.3 on MMQA. Reconciling the two views yields the best F1 on BIRD and MMQA and the best PR on Spider and MMQA. On Spider it trails P_1 alone by 0.5 F1 while adding 29.4 PR points; over R alone it adds 0.7 to 12.4 F1, with higher PR on Spider and MMQA and 0.1 points lower on BIRD. Plain intersection, union, or unconditional deletion each give up one side of this balance (Appendix E.5).

5 Conclusion

We asked which role each table plays, provider, connector, or none, rather than how well it matches, and built HCVR to assign those roles and to check the absent one without training, at two to three LLM calls per query. HCVR attains the highest precision and F1 on pooled BIRD, Spider, and MMQA under both retrieval LLMs at competitive perfect recall, and its more precise sets raise downstream execution accuracy in every setting.

AI USE STATEMENT

LLMs are components of the proposed method (offline table descriptions, requirement extraction, semantic selection, and deletion review) and serve as the SQL generators; this use is part of the methodology and is fully described in Sections 3 and 4, with all prompts in Appendix F. In preparing this work, we used generative AI coding assistants to help implement and debug the retrieval pipeline and evaluation scripts; the resulting code was verified by the authors against the specification in Section 3 and Appendix B, and all reported numbers were produced by running it. We did not use generative AI tools to propose the research idea, to design the method or the experiments, to formulate claims, or to interpret the results; generating synthetic data, developing theoretical models, and writing proofs are not applicable to this work. Additionally, we used generative AI tools to edit the manuscript for readability and to check the consistency of reported numbers across sections; every edit was reviewed by the authors against the source tables. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses only publicly available text-to-SQL benchmarks (BIRD, Spider, and MMQA) under their licenses and involves no human subjects or personal data. Retrieval errors can propagate to generated SQL; HCVR reduces this risk by returning fewer irrelevant tables, but generated SQL should still be validated before use in decision-making.

REPRODUCIBILITY STATEMENT

The method is fully specified in Section 3 and Algorithm 1. Hyperparameters, the compatibility computation, and the predicate inventory are given in Appendix B, prompts in Appendix F, dataset statistics and the pooled protocol in Appendix C, and baseline sources in Appendix D. All LLM calls use temperature 0, and all numbers come from a single deterministic run per setting. The code will be released publicly upon acceptance.

REFERENCES

Rishita Agarwal, Himanshu Singhal, Peter Baile Chen, Manan Roy Choudhury, Dan Roth, and Vivek Gupta. REaR: Retrieve, expand and refine for effective multitable retrieval. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 39360–39374, 2026.

Allaa Boutaleb, Bernd Amann, Rafael Angarita, and Hubert Naacke. Exploring multi-table retrieval through iterative search. In *EurIPS 2025 Workshop: AI for Tabular Data*, 2025. URL https://openreview.net/forum?id=d64whXpOgf.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. LGESQL: Line graph enhanced text-to-SQL model with mixed local and non-local relations. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 2541–2555, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.198. URL https://aclanthology.org/2021.acl-long.198/.

Peter Baile Chen, Yi Zhang, and Dan Roth. Is table retrieval a solved problem? exploring join-aware multi-table retrieval. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 2687–2699, 2024.

Peter Baile Chen, Yi Zhang, Mike Cafarella, and Dan Roth. Can we retrieve everything all at once? ARM: An alignment-oriented LLM-based retrieval method. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30298–30317, 2025.

Tianji Cong, James Gale, Jason Frantz, HV Jagadish, and Çağatay Demiralp. Warpgate: A semantic join discovery system for cloud data warehouses. *arXiv preprint arXiv:2212.14155*, 2022.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3563–3578, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.212. URL https://aclanthology.org/2024.findings-acl.212/.

Yuyang Dong, Chuan Xiao, Takuma Nozawa, Masafumi Enomoto, and Masafumi Oyamada. DeepJoin: Joinable table discovery with pre-trained language models. *Proceedings of the VLDB Endowment*, 16(10):2458–2470, 2023.

Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. Aurum: A data discovery system. In *2018 IEEE 34th International conference on data engineering (ICDE)*, pp. 1001–1012. IEEE, 2018.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-SQL in cross-domain database with intermediate representation. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4524–4535, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1444. URL https://aclanthology.org/P19-1444/.

Zhifeng Hao, Junqi Huang, Shaobin Shi, Ruichu Cai, and Boyan Xu. GenLink: Generation-driven schema-linking via multi-model learning for text-to-SQL. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 29892–29905, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1518. URL https://aclanthology.org/2025.emnlp-main.1518/.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. RESDSQL: Decoupling schema linking and skeleton parsing for text-to-SQL. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 13067–13075, 2023a.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. Can LLM already serve as a database interface? a big bench for large-scale database grounded text-to-SQLs. *Advances in Neural Information Processing Systems*, 36:42330–42357, 2023b.

Xianming Li and Jing Li. AoE: Angle-optimized embeddings for semantic textual similarity. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1825–1839, 2024.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4870–4888, 2020.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-Refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Jayant Madhavan, Phil Bernstein, and Erhard Rahm. Generic schema matching with cupid. Technical Report MSR-TR-2001-58, August 2001. URL https://www.microsoft.com/en-us/research/publication/generic-schema-matching-with-cupid/.

Fatemeh Nargesian, Erkang Zhu, Ken Q Pu, and Renée J Miller. Table union search on open data. *Proceedings of the VLDB Endowment*, 11(7):813–825, 2018.

Fatemeh Nargesian, Erkang Zhu, Renée J Miller, Ken Q Pu, and Patricia C Arocena. Data lake management: challenges and opportunities. *Proceedings of the VLDB Endowment*, 12(12):1986–1989, 2019.

OpenAI. Gpt-4o mini: Advancing cost-efficient intelligence, 2024. URL https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

Mohammadreza Pourreza and Davood Rafiei. DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. *Advances in neural information processing systems*, 36:36339–36348, 2023.

Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Hassan Soliman, Vivek Gupta, Dan Roth, and Iryna Gurevych. CORE-T: COherent REtrieval of tables for text-to-SQL. *arXiv preprint arXiv:2601.13111*, 2026.

Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. CHESS: Contextual harnessing for efficient SQL synthesis. *arXiv preprint arXiv:2405.16755*, 2024.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7567–7578, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.677. URL https://aclanthology.org/2020.acl-main.677/.

Yihan Wang, Peiyu Liu, and Xin Yang. LinkAlign: Scalable schema linking for real-world large-scale multi-database text-to-SQL. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 977–991, 2025.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198, 04 2020. ISSN 2307-387X. doi: 10.1162/tacl_a_00309. URL https://doi.org/10.1162/tacl_a_00309.

Jian Wu, Linyi Yang, Dongyuan Li, Yuliang Ji, Manabu Okumura, and Yue Zhang. MMQA: Evaluating LLMs with multi-table multi-hop complex questions. In *International Conference on Learning Representations*, volume 2025, pp. 48626–48643, 2025.

Mohamed Yakout, Kris Ganjam, Kaushik Chakrabarti, and Surajit Chaudhuri. Infogather: Entity augmentation and attribute discovery by holistic matching with web tables. In *ACM SIGMOD Conference*, January 2012.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 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*, pp. 3911–3921, 2018.

Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. MURRE: Multi-hop table retrieval with removal for open-domain text-to-SQL. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 5789–5806, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.386/.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*, 2023.

Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J Miller. Josie: Overlap set similarity search for finding joinable tables in data lakes. In *Proceedings of the 2019 International Conference on Management of Data*, pp. 847–864, 2019.

A More Related Work

Schema Linking and Schema Context Selection. Schema linking grounds natural-language questions in database elements, while schema encoding models the relationships needed for SQL generation. Representative approaches use intermediate representations (Guo et al., 2019), relation-aware attention (Wang et al., 2020), joint question–schema representations (Lin et al., 2020), and graph structures (Cao et al., 2021) to support this grounding. RESDSQL separates schema linking from skeleton parsing through ranking-enhanced encoding and skeleton-aware decoding (Li et al., 2023a), while CHESS includes schema selection to reduce the context supplied to SQL generation (Talaei et al., 2024). These approaches primarily operate within a supplied target database. LinkAlign extends schema linking to large multi-database environments by addressing database retrieval and schema-item grounding (Wang et al., 2025). Our setting requires selection directly from a pooled corpus without source-database identifiers or declared foreign keys.

Question Decomposition and Requirement Representation. Decomposing complex questions or generation tasks into intermediate steps is a recurring strategy in question answering and semantic parsing. QDMR represents questions as ordered natural-language operations (Wolfson et al., 2020), least-to-most prompting solves simpler subproblems sequentially (Zhou et al., 2023), and DIN-SQL decomposes SQL generation into stages (Pourreza & Rafiei, 2023). For table retrieval, JAR decomposes questions into concepts and attributes (Chen et al., 2024), while ARM explicitly addresses the mismatch between query decomposition and the organization of available data through information and structure alignment (Chen et al., 2025). These studies motivate decompositions that account for how information is represented and distributed. HCVR uses typed entity–predicate–attribute requirements specifically to construct table-level provider hypotheses. The entity preserves the context of an attribute, and the predicate distinguishes constraints from requested outputs.

Iterative Retrieval and Multi-Table Composition. Retrieval can be refined through reasoning–action interaction, successive query reformulation, and incremental set construction. ReAct provides a general framework for interleaving reasoning with actions (Yao et al., 2023); MURRE applies multi-hop retrieval to open-domain text-to-SQL by removing already covered information from the question (Zhang et al., 2025). Greedy join-aware retrieval incrementally balances query relevance, coverage, and joinability, providing an efficient alternative to MIP-based selection (Boutaleb et al., 2025). These methods already recognize that useful table sets depend on both information coverage and inter-table relationships. HCVR makes the corresponding roles explicit: bounded beam search explores alternative provider assignments, and minimum-hop paths introduce connectors when the selected providers would otherwise remain disconnected. Its distinction therefore concerns how table contributions are represented and checked, rather than the use of search or structural evidence alone.

Data Discovery, Schema Matching, and Joinability. Discovering useful information across heterogeneous tables has been studied through data discovery, entity augmentation, union search, and joinable-table search. Aurum organizes relationships among data sources to support discovery (Fernandez et al., 2018); InfoGather uses holistic matching of web tables for entity augmentation and attribute discovery (Yakout et al., 2012). Table union search identifies tables with compatible columns for combining rows (Nargesian et al., 2018), whereas DeepJoin learns column representations for joinable-table discovery (Dong et al., 2023). Schema matching provides complementary techniques for identifying correspondences between independently designed schemas (Rahm & Bernstein, 2001; Madhavan et al., 2001). These tasks provide relevant foundations for discovering candidate relationships in a pool, but a correspondence or potential join does not by itself establish that a table is needed for a particular question. HCVR reasons about this additional decision by evaluating provider alternatives against information requirements and connector roles relative to the selected providers. Joinability supplies structural evidence for composition, while table membership also depends on the contribution to the complete information need.

Verification and Conservative Table Selection. Feedback and verification have been used to revise model outputs and improve subsequent decisions. Self-Refine alternates self-generated feedback with refinement (Madaan et al., 2023); Reflexion maintains linguistic feedback across trials (Shinn et al., 2023); and Chain-of-Verification separately answers verification questions before re-

vising an initial response (Dhuliawala et al., 2024). Verification also appears in retrieval: ARM checks coverage and connectivity within alignment drafts and aggregates selections across reasoning beams (Chen et al., 2025). HCVR focuses on a specific uncertainty in pooled-table retrieval: a table omitted by composition may be unnecessary, or its contribution may have been missed because of incomplete requirements or inaccurate join inference. It compares the composed set with a semantic selection that receives candidate evidence but not the composition's provider assignments or ranked table sets. When the semantic selection strictly contains the composed set, a dedicated deletion review checks whether the additional tables can be removed without losing required information or connections. This makes absent-role assessment a concrete verification target and supports conservative revision before downstream SQL generation.

B More Implementation Details

B.1 Experimental Configuration

Table 5 summarizes the default configuration across BIRD, Spider, and MMQA. The requirement-level candidate budget and beam width share the same parameter K.

Table 5: Default experimental configuration across the three corpora.

Component or parameter	Setting
Embedding model	UAE-Large-V1
Retrieval-stage LLMs	Llama-3.1-8B-Instruct; Qwen2.5-7B-Instruct
SQL generation models	Llama-3.2-3B-Instruct; GPT-4o-mini
Maximum sample rows per table	5
Provider candidate budget and beam width	$K = 10$
Requirement-level relevance weight	$\alpha = 0.5$
Compatibility threshold	$\tau = 0.3$
Graph expansion depth	1 hop
Maximum connection path length	$h = 5$ hops
Generation temperature	0
Retrieval-stage output limit	8,192 tokens

B.2 Offline Processing

Following CORE-T (Soliman et al., 2026), we generate a short purpose description for each table using its name, schema, and up to five sample rows. The description summarizes the represented entities, attributes, and information granularity. The enriched representation is encoded with UAE-Large-V1 and indexed for retrieval.

Compatibility computation. For columns a and b from different tables, let V_a and V_b denote their distinct non-null values. Value overlap is

$$\mathrm{jac}(a, b) = \frac{|V_a \cap V_b|}{|V_a \cup V_b|}. \tag{4}$$

Header similarity combines lexical matching ex and embedding-based semantic similarity sem; header embeddings encode the table and column names:

$$\mathrm{name}(a, b) = 0.5\ \mathrm{sem}(a, b) + 0.5\ \mathrm{ex}(a, b). \tag{5}$$

Let $\mathrm{uniq}(a)$ indicate whether non-null values in a are unique, and $\mathrm{sub}(a, b)$ indicate whether one column's value set is contained in the other's. Eligible pairs satisfy

$$\mathrm{valid}(a, b) = \big[\max\{\mathrm{uniq}(a), \mathrm{uniq}(b)\} = 1\big] \wedge \big[\mathrm{sub}(a, b) = 1\big]. \tag{6}$$

Column compatibility is

$$s_{\mathrm{col}}(a, b) = \mathbf{1}[\mathrm{valid}(a, b)]\,(0.5\ \mathrm{jac}(a, b) + 0.5\ \mathrm{name}(a, b))\,. \tag{7}$$

Both mixtures use weights of 0.5. With $C(t)$ denoting the columns of table t, table compatibility is

$$c(t, u) = \max_{a \in C(t),\, b \in C(u)} s_{\text{col}}(a, b). \tag{8}$$

Pairs without eligible column matches receive zero compatibility. The maximizing column pair identifies candidate join columns. The processed representations, scores, and join columns are cached for online use.

B.3 Typed Requirements and Predicate Inventory

Requirement extraction produces two lists, `constraint` and `answer`. Each item contains a triple, a short explanation of its meaning, and a natural-language retrieval fragment. The triple's subject represents a semantic entity, while its object represents an attribute, measure, or entity with any associated condition. This representation approximates the entity–attribute organization of relational schemas at the semantic level.

Table 6 summarizes the predicate inventory, including filtering, grouping, ordering, aggregation, existence, entity and attribute retrieval, counting, and extremum queries.

Table 6: Predicate inventory for typed requirement extraction. Conditions remain attached to the object. Examples use semantic entities and attributes rather than physical schema identifiers.

Predicate	Role	Example triple
Constraint predicates		
`filter_by`	Attribute condition	(student, filter_by, age {numeric $>$ 18})
`group_by`	Grouping attribute	(movie, group_by, genre)
`order_by`	Ordering or selection evidence	(product, order_by, price {max})
`aggregate_by`	Aggregation evidence	(order, aggregate_by, amount {sum})
`exists`	Existence or minimum count	(employee, exists, permit {min_count $\geq$ 2})
Answer predicates		
`return_entity`	Requested entity	(student, return_entity, student)
`return_attribute`	Requested attribute	(school, return_attribute, name)
`count_entity`	Requested entity count	(school, count_entity, school)
`find_max_by`	Requested maximum target	(product, find_max_by, price)
`find_min_by`	Requested minimum target	(product, find_min_by, price)

Different attributes normally form separate requirements, while conditions on the same attribute may remain together. Existence constraints are explicitly extracted when the question requires a related entity or a minimum count, because either entity involved may have no other information requirement that would otherwise identify it as a provider. Ordinary relationships are left to connector inference.

B.4 Conservative Deletion

Our deletion policy is deliberately conservative: uncertainty about a table's contribution favors retention. When $P_1 \subsetneq R$, deletion review assesses the proposed removal $R \setminus P_1$. It identifies the sources that remain for the question's requirements and examines each proposed table's unique contribution, including any necessary connection.

The proposed removal is accepted only when every proposed table is judged unnecessary and every requirement has an identified remaining source. If any source is missing or any table's contribution remains uncertain, HCVR retains R. It also retains R when the review contains conflicting judgements, such as marking a table for deletion while identifying a necessary contribution from it, or issuing a `DELETE` decision while marking an individual table as `KEEP`. A `KEEP` decision returns R directly.

This conservative design places the burden of evidence on deletion: a table is not removed merely because the composed hypothesis omits it, which limits recall loss from unsupported deletions (Appendix E.5 quantifies the effect against Always KEEP and Always DELETE).

C Dataset Details

We follow CORE-T's pooled evaluation setup for BIRD, Spider, and MMQA detailed in Table 7. BIRD and Spider use their development queries; MMQA uses the same stratified subset of 1,105 queries. The table corpus for each benchmark is searched as a single pool.

Table 7: Dataset statistics after pooling. #Empty: zero-row tables; Rows/T and Cols/T: mean rows and columns per table; EG: queries with an empty gold execution result; $\bar{G}$: mean gold tables per query.

Dataset	#DB	#Tab	#Empty	Rows/T	Cols/T	#Q	EG (%)	$\bar{G}$
BIRD	11	75	0	52,437	10.6	1,534	0.0	1.95
Spider	20	81	0	6,665	5.4	1,034	4.7	1.51
MMQA	1	710	0	1,547	5.1	1,105	10.9	2.20

Differences among the corpora. BIRD has the largest mean table size and column count in this evaluation. Spider has fewer gold tables per query, making unnecessary additions particularly visible in exact set match. MMQA has a substantially larger table pool and more gold tables per query on average, increasing the need to preserve complementary evidence. We report results by gold-table count in Appendix E.1; the groups 1, 2, and ≥ 3 are mutually exclusive.

D Baseline Sources and Reporting Scope

Experimental comparisons and baseline sources. For the main comparisons, we use the baseline results reported in CORE-T's common evaluation suite (Soliman et al., 2026), preserving the original model configurations and dataset settings. We additionally reproduce CORE-T to obtain the per-query outputs required by our subsequent analyses. Table 8 shows that the reproduced retrieval results remain broadly consistent with the published ones: precision and F1 differ by at most 4.2 and 3.2 points, while PR differs by at most 4.5 points. The reproduced Qwen runs select slightly fewer tables, yielding higher precision and F1 but somewhat lower PR, consistent with a modest precision–coverage trade-off. Overall, these discrepancies are small, and we retain the published CORE-T results in the main comparison tables to maintain consistency with the original evaluation. Our reproduced per-query outputs are used only where such outputs are required: Appendix E.1 for retrieval comparisons stratified by the number of gold tables, Appendix E.7 for Text-to-SQL evaluation with reproduced Qwen-based retrieval, and Appendix E.9 for paired significance testing.

Table 8: Reproduced CORE-T table-retrieval results. $\bar{T}$ denotes the average number of retrieved tables and PR denotes perfect recall.

Retriever	Dataset	$\bar{T}\downarrow$	P↑	F1↑	PR↑
Llama-3.1-8B-Instruct	BIRD	4.3	50.6	62.4	90.0
	Spider	3.9	42.7	56.0	95.1
	MMQA	5.1	38.8	48.7	59.9
Qwen2.5-7B-Instruct	BIRD	2.9	64.6	72.8	82.5
	Spider	2.7	58.0	70.0	93.7
	MMQA	3.7	52.5	59.2	57.1

Unavailable baseline results. Following the reporting scope of CORE-T (Soliman et al., 2026), we report JAR results on BIRD and Spider, and ARM results on BIRD. For JAR, the release provides runnable scripts and tuned hyperparameters for BIRD and Spider, but does not fully specify the tuning procedure needed for consistent adaptation to other benchmarks. For ARM, the released pipeline relies on dataset-specific intermediate artifacts, while the details required to generate these artifacts for additional benchmarks are incomplete. Accordingly, JAR results on MMQA and ARM results on Spider and MMQA are left unreported.

Table 9: Exact set match and perfect recall by gold-table count. Scores are percentages, and N is the number of queries. Pooled results combine queries from all three datasets.

		Llama-3.1-8B-Instruct				Qwen2.5-7B-Instruct			
		Exact ↑		PR ↑		Exact ↑		PR ↑	
Gold tables	N	CORE-T	HCVR	CORE-T	HCVR	CORE-T	HCVR	CORE-T	HCVR
BIRD									
1	362	0.3	**23.5**	**98.3**	97.5	3.9	**54.4**	98.3	**98.9**
2	936	13.8	**25.0**	**91.5**	85.9	32.3	**55.4**	84.1	**86.0**
≥ 3	236	18.2	**37.7**	71.6	**75.4**	25.0	**36.0**	51.7	**57.2**
All	1534	11.3	**26.6**	**90.0**	87.0	24.4	**52.2**	82.5	**84.6**
Spider									
1	575	1.0	**48.3**	97.7	**98.1**	8.0	**70.8**	**97.4**	97.2
2	393	12.2	**36.1**	94.4	**96.9**	29.0	**69.7**	**94.7**	89.6
≥ 3	66	36.4	**65.2**	75.8	**87.9**	21.2	**66.7**	56.1	**77.3**
All	1034	7.5	**44.8**	95.1	**97.0**	16.8	**70.1**	**93.7**	93.0
MMQA									
1	4	0.0	**25.0**	**100.0**	**100.0**	25.0	**100.0**	**100.0**	**100.0**
2	879	5.6	**12.9**	64.8	**67.7**	18.2	**31.9**	62.6	**63.7**
≥ 3	222	6.8	**22.5**	39.6	**46.4**	10.4	**26.1**	34.7	**46.4**
All	1105	5.8	**14.8**	59.9	**63.5**	16.7	**31.0**	57.1	**60.4**
Pooled									
1	941	0.7	**38.7**	**98.0**	97.9	6.5	**64.6**	97.8	**97.9**
2	2208	10.2	**22.1**	**81.4**	80.6	26.1	**48.6**	77.4	**77.8**
≥ 3	524	15.6	**34.7**	58.6	**64.7**	18.3	**35.7**	45.0	**55.2**
All	3673	8.6	**28.2**	82.4	**82.8**	20.0	**50.9**	78.0	**79.7**

Configuration and prompts. The experimental configuration and predicate inventory are provided in Appendices B.1 and B.3. The prompts are presented in Appendix F.

E More Experimental Results and Analysis

E.1 Detailed Comparison with CORE-T

We present detailed comparisons with CORE-T, the strongest baseline overall, examining exact table-set recovery and complete coverage across different numbers of gold tables. Table 9 reports Exact and perfect recall (PR) for three mutually exclusive groups: queries requiring one, two, or at least three tables, together with results over all queries. The pooled results combine queries from all three datasets. All scores are computed from query-level counts and rounded directly to one decimal place.

Exact recovery improves across datasets and query sizes. Across both retrieval models, HCVR achieves higher Exact than CORE-T on all three datasets and in every cardinality group. Over the pooled queries, Exact increases from 8.6% to 28.2% and from 20.0% to 50.9%, respectively. The largest pooled Exact gains occur on one-table queries, where the scores increase from 0.7% to 38.7% and from 6.5% to 64.6%, while PR remains nearly unchanged. This pattern indicates that the additional exact matches in this group primarily reflect the exclusion of unnecessary tables. For queries requiring at least three tables, Exact also increases from 15.6% to 34.7% and from 18.3% to 35.7%, showing that the improvements extend to larger table combinations.

Recall gains are largest for queries requiring at least three tables. Across all six model–dataset settings, the group requiring at least three tables obtains the largest PR improvement among the three cardinality groups. Over its 524 pooled queries, PR increases from 58.6% to 64.7% and from 45.0% to 55.2%. These gains are more pronounced than those for one-table and two-table queries. This pattern matches the motivation for explicit provider and connector modeling: answering a question over several tables requires preserving both the complementary information sources and

the connections among them, and the simultaneous gains in Exact and PR show that role-based retrieval does so for larger table combinations rather than only pruning small ones.

E.2 Table Selection Results for Requirement Formulation

Table 10 supplements the composition-level analysis in the main text with final table-retrieval results.

Table 10: Final table-retrieval results for different requirement formulations with Llama-3.1-8B-Instruct.

Representation	P ↑	R ↑	F1 ↑	PR ↑	Exact ↑	$\bar{T}$ ↓
BIRD						
HCVR	64.4	**92.7**	72.2	**87.0**	26.6	3.2
Untyped decomposition	63.2	89.7	69.8	79.7	24.6	3.2
JAR-style decomposition	58.5	92.3	67.0	84.9	21.1	3.7
Sub-query decomposition	**66.4**	90.0	**72.5**	80.3	**28.7**	**3.0**
Spider						
HCVR	**71.4**	**98.1**	**79.3**	**97.0**	**44.8**	**2.4**
Untyped decomposition	63.3	94.1	71.4	90.9	34.5	2.7
JAR-style decomposition	62.6	95.9	71.3	93.3	32.5	2.8
Sub-query decomposition	68.2	93.9	75.2	89.8	39.1	**2.4**
MMQA						
HCVR	49.1	**76.4**	**57.2**	**63.5**	**14.8**	4.2
Untyped decomposition	44.7	74.0	52.6	58.1	12.6	4.7
JAR-style decomposition	40.2	73.3	48.8	57.2	10.0	5.3
Sub-query decomposition	**49.4**	71.2	55.4	53.2	14.6	**4.0**

Final retrieval broadly agrees with composition quality. The final retrieval results follow the overall composition-quality trends reported in the main text. HCVR achieves higher F1 and Exact than untyped decomposition and JAR-style decomposition on all three datasets, together with the highest PR among the compared formulations. The correspondence is not uniform across every metric: sub-query decomposition attains slightly higher F1 on BIRD, but lower PR. These supplementary results support the consistency between the quality of the retained table compositions and that of the final retrieved table sets.

E.3 Table Selection Results for Composition Variants

Table 11 supplements the composition-level analysis in the main text with final table-retrieval results.

The full composition configuration preserves the most complete coverage after verification. HCVR achieves the highest R and PR on all three datasets. Removing state compression or connector inference, or using either relevance signal alone, yields lower final coverage. Although some variants attain higher precision or F1, the advantage of the full configuration is consistent in retaining all required tables. These supplementary results support the main-text conclusions regarding the importance of composition diversity, connector inference, and complementary relevance signals for complete table retrieval.

E.4 Effect of the Global Requirement

Table 12 isolates the effect of retaining the original question as a global requirement. The variant without this requirement removes whole-question retrieval as a provider hypothesis while retaining whole-question relevance scoring in Compose.

The global requirement provides a fallback and preserves the original query semantics. The two configurations produce broadly comparable composition and candidate-set quality, with small

Table 11: Final table-retrieval results for composition variants with Llama-3.1-8B-Instruct.

Variant	P ↑	R ↑	F1 ↑	PR ↑	Exact ↑	$\bar{T}$ ↓
BIRD						
HCVR	64.4	**92.7**	72.2	**87.0**	26.6	3.2
w/o state compression	**70.6**	87.3	**74.7**	75.8	**34.0**	**2.7**
w/o connector inference	67.7	89.6	73.4	80.1	28.2	2.8
Requirement-only scoring	58.7	91.6	67.4	83.2	22.6	3.7
Whole-query-only scoring	68.7	87.5	73.2	76.6	29.4	2.8
Spider						
HCVR	71.4	**98.1**	**79.3**	**97.0**	**44.8**	2.4
w/o state compression	**72.0**	93.6	78.1	88.7	44.5	**2.2**
w/o connector inference	68.4	93.2	75.4	88.1	39.7	2.4
Requirement-only scoring	63.1	94.8	71.4	90.2	35.0	2.8
Whole-query-only scoring	69.0	92.5	75.2	87.3	39.1	2.4
MMQA						
HCVR	49.1	**76.4**	57.2	**63.5**	14.8	4.2
w/o state compression	**52.4**	70.3	**57.3**	52.3	**20.9**	**3.7**
w/o connector inference	50.2	71.0	56.4	53.0	16.1	**3.7**
Requirement-only scoring	41.0	72.4	49.5	54.9	10.0	5.0
Whole-query-only scoring	49.3	70.1	54.9	52.4	14.2	3.9

Table 12: Effect of the global requirement before verification with Llama-3.1-8B-Instruct.

Variant	Exact@1 ↑	Exact@10 ↑	$\overline{\lvert U \rvert}$ ↓	PR_U ↑
BIRD				
HCVR	33.8	72.0	**7.6**	94.4
w/o global requirement	**36.0**	**72.7**	7.8	**94.5**
Spider				
HCVR	55.3	86.9	**8.0**	98.3
w/o global requirement	**55.6**	**87.7**	8.3	**98.5**
MMQA				
HCVR	18.8	46.2	**8.5**	**67.1**
w/o global requirement	**20.4**	**47.1**	8.6	66.0

differences in Exact@10, union size, and PR_U. We retain the global requirement as a conservative design for two situations. First, it provides a retrieval input when extraction yields no usable requirements. Second, it preserves the original question as complementary evidence when extracted requirements omit important information or deviate substantially from the intended semantics. Its role is therefore to maintain access to the original information need when the initial extraction is incomplete or unreliable. The global requirement is thus included as a robustness-oriented design choice.

E.5 Additional Verification Ablations

Table 13 reports additional ablations of deletion review, set combination, and inferred-join addition; a recall-oriented variant of the deletion policy is evaluated separately in Appendix E.8. When $P_1 \subsetneq R$, Always KEEP retains R without review, whereas Always DELETE accepts the proposed deletion and returns P_1. The variant without inferred-join addition omits the final connection-recovery operation when the two sets partially overlap without containment or are disjoint. The set-combination controls directly return $P_1 \cap R$ or $P_1 \cup R$. The R + inferred joins control augments the independently selected semantic table set R with potential connectors, whereas $(P_1 \cup R)$ + inferred joins first merges the semantic and composed table sets and then adds potential connectors. Both use the inferred-join addition applied in Verify's partial-overlap and disjoint cases, aiming to form a connected table set.

Table 13: Additional verification ablations with Llama-3.1-8B-Instruct.

Variant	P ↑	R ↑	F1 ↑	PR ↑	Exact ↑	$\bar{T}$ ↓
BIRD						
HCVR (full)	64.4	92.7	72.2	87.0	26.6	3.2
Always KEEP	58.9	94.5	69.3	90.0	19.9	3.4
Always DELETE	81.8	68.8	70.1	47.3	33.4	1.7
w/o inferred-join addition	64.7	92.6	**72.5**	86.7	26.7	3.1
$P_1 \cap R$	**83.3**	65.1	69.8	41.9	**35.3**	**1.4**
$P_1 \cup R$	59.0	94.4	69.5	89.8	19.6	3.4
R + inferred joins	51.3	94.9	63.5	90.3	12.1	4.0
$(P_1 \cup R)$ + inferred joins	50.4	**95.6**	62.9	**92.0**	11.0	4.1
Spider						
HCVR (full)	71.4	98.1	79.3	97.0	44.8	2.4
Always KEEP	53.8	98.5	66.2	97.4	16.3	3.1
Always DELETE	84.3	81.7	79.6	70.1	54.9	1.5
w/o inferred-join addition	71.6	97.9	79.5	96.7	44.9	2.4
$P_1 \cap R$	**86.8**	79.5	**80.2**	66.8	**57.1**	**1.3**
$P_1 \cup R$	54.0	98.5	66.5	97.4	16.3	3.0
R + inferred joins	46.2	98.0	59.5	96.6	8.7	3.6
$(P_1 \cup R)$ + inferred joins	45.5	**98.9**	59.1	**98.0**	8.5	3.7
MMQA						
HCVR (full)	49.1	76.4	57.2	63.5	14.8	4.2
Always KEEP	46.4	77.0	55.6	64.5	12.0	4.4
Always DELETE	61.6	58.6	55.8	33.9	19.0	2.7
w/o inferred-join addition	49.5	75.9	**57.4**	62.7	14.9	4.1
$P_1 \cap R$	**64.0**	52.5	55.3	26.7	**20.7**	**1.9**
$P_1 \cup R$	46.5	76.6	55.7	63.7	11.7	4.3
R + inferred joins	42.0	76.3	52.2	63.8	7.7	4.7
$(P_1 \cup R)$ + inferred joins	40.2	**77.7**	51.0	**65.2**	6.5	5.0

Deletion review balances redundancy reduction and complete coverage. Compared with Always KEEP, HCVR improves precision, F1, and Exact on all three datasets while returning smaller table sets, with modest decreases in PR. Always DELETE achieves higher precision and Exact, but its PR falls to 47.3%, 70.1%, and 33.9% on BIRD, Spider, and MMQA. These results support reviewing proposed deletions to reduce redundancy while limiting the loss of complete coverage.

Direct set combination trades coverage for compactness. Intersection produces the smallest outputs and the highest precision and Exact, but reduces PR to 41.9%, 66.8%, and 26.7% on the three datasets. Requiring agreement between the two sets can discard necessary providers or connectors identified by only one of them, leaving information requirements unmet or required connections unsupported. Union preserves tables identified by either set, including complementary providers and connectors, but also retains redundant alternatives and tables serving neither role. Compared with HCVR, it achieves slightly higher PR while lowering precision, F1, and Exact on every dataset. These results support checking each table's provider or connector role rather than determining membership solely by agreement or inclusion across the two sets.

Unconditional inferred-join addition increases redundancy. The R + inferred joins control combines semantic provider selection with connection recovery, but yields lower precision, F1, and Exact than HCVR on all three datasets. Adding inferred joins to $P_1 \cup R$ increases PR relative to the union alone, while enlarging the outputs and reducing F1 and Exact. A table may be compatible with a selected provider without being necessary to connect the providers required by the query. These results support applying inferred-join addition selectively, using the relation between the selected sets and support from the retained compositions to limit unnecessary additions.

Selective inferred-join addition provides a modest coverage safeguard. Removing inferred-join addition from HCVR reduces PR from 87.0% to 86.7% on BIRD, from 97.0% to 96.7% on Spider, and from 63.5% to 62.7% on MMQA, while slightly improving precision and F1. Its selec-

Table 14: Distribution of the set relation between P_1 and R before verification (% of queries). Llama and Qwen denote Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct; the five relations are mutually exclusive.

	Llama			Qwen		
Relation	BIRD	Spider	MMQA	BIRD	Spider	MMQA
$P_1 = R$	9.3	11.6	8.8	32.5	56.5	30.8
$P_1 \subsetneq R$	81.2	81.6	70.2	55.6	37.8	39.3
$R \subsetneq P_1$	2.3	1.8	3.8	6.5	4.1	13.4
Partial overlap	6.3	4.7	15.8	4.4	1.5	16.0
Disjoint	1.0	0.2	1.4	1.0	0.1	0.5

tive use therefore provides a modest gain in complete coverage at the cost of retaining additional tables, consistent with its role as a conservative recovery operation.

E.6 Verification Relations and Composition Quality

Two further analyses examine how often Verify is exercised and how much of the remaining error lies in composition ranking rather than in the candidate compositions themselves.

Strict containment of the composition in the semantic selection is the dominant relation, so the deletion review decides most queries. Table 14 reports the distribution of the five mutually exclusive set relations between the top composition P_1 and the independent semantic selection R before verification. With Llama-3.1-8B-Instruct, $P_1 \subsetneq R$ holds for 81.2%, 81.6%, and 70.2% of queries on BIRD, Spider, and MMQA, and exact agreement for another 9.3%, 11.6%, and 8.8%; with Qwen2.5-7B-Instruct, agreement rises to 32.5%, 56.5%, and 30.8% and strict containment falls to 55.6%, 37.8%, and 39.3%. In every setting the semantic selection retains all composed tables for at least 70% of queries, and the two are disjoint for at most 1.4%. Because R sees neither the requirement assignments nor the ranking, this agreement is independent support that role-based composition recovers the tables a query-level reading also selects; the difference lies almost entirely in the extra tables that R keeps, which is exactly the case the deletion review adjudicates.

Oracle selection reveals substantial ranking headroom within the retained compositions. To measure the quality of the retained compositions independently of ranking and verification, we select for each query the retained composition that is best under the lexicographic order $\mathrm{PR} > \mathrm{F1} > \mathrm{P}$ against the gold set, and evaluate it directly. Table 15 reports this oracle selection together with the mean original rank $\bar{r}$ of the selected composition, where rank 1 is P_1. The best retained compositions reach F1 of 91.5, 95.8, and 73.6 with Llama and 91.6, 96.7, and 75.3 with Qwen on BIRD, Spider, and MMQA, with 2.2, 1.7, and 2.6 tables per query with Llama and 2.2, 1.6, and 2.6 with Qwen, and their mean rank is 1.8 to 2.9 in lists of at most ten. Against the final results in Table 1, the gap is small in PR (97.0 versus 98.2 on Spider and 63.5 versus 65.5 on MMQA with Llama) and larger in precision and Exact, so complete coverage is largely preserved and the remaining loss comes from tables that a better-ranked composition would have excluded. The single-composition oracle is not an upper bound on Verify, however: on BIRD the final output contains every gold table for 27 queries with Llama and 17 with Qwen for which no single retained composition does, because semantic selection can combine complementary tables from different retained compositions, even when no single composition contains the entire gold set. Selecting among the retained compositions with inference-time signals before cross-verification is therefore the most direct route to closing the remaining gap.

E.7 Text-to-SQL with Qwen-Based Retrieval

Qwen2.5-7B-Instruct produces smaller retrieved table sets than Llama-3.1-8B-Instruct, motivating an additional evaluation of their downstream utility. Table 16 compares HCVR with CORE-T, the strongest retrieval baseline overall, using Qwen-based retrieval outputs.

Table 15: Quality and original rank of the best retained composition under oracle selection (perfect recall, then F1, then precision), evaluated without verification. $\bar{T}$: mean output size; $\bar{r}$: mean rank of the selected composition (rank 1 is P_1).

Dataset	Model	P ↑	R ↑	F1 ↑	PR ↑	Exact ↑	$\bar{T}$ ↓	$\bar{r}$ ↓
BIRD	Llama	89.7	96.4	91.5	92.2	72.0	2.2	2.6
	Qwen	90.6	95.8	91.6	90.4	71.6	2.2	2.5
Spider	Llama	94.4	99.1	95.8	98.2	86.9	1.7	1.9
	Qwen	95.6	99.2	96.7	98.4	89.2	1.6	1.8
MMQA	Llama	72.3	79.0	73.6	65.5	46.2	2.6	2.9
	Qwen	73.7	80.7	75.3	68.1	48.0	2.6	2.8

Table 16: Text-to-SQL results with Qwen2.5-7B-Instruct retrieval. Oracle uses gold table sets and is shown as a reference.

	BIRD		Spider		MMQA	
Retriever	$\mathrm{EX}_{\mathrm{all}}$	$\mathrm{EX}_{\mathrm{PR}}$	$\mathrm{EX}_{\mathrm{all}}$	$\mathrm{EX}_{\mathrm{PR}}$	$\mathrm{EX}_{\mathrm{all}}$	$\mathrm{EX}_{\mathrm{PR}}$
Llama-3.2-3B-Instruct						
CORE-T	16.0	15.4	42.1	41.9	20.1	18.0
HCVR	**20.1**	**18.8**	**51.6**	**51.1**	**24.3**	**22.1**
Oracle	27.8	27.8	58.9	58.9	45.4	45.4
GPT-4o-mini						
CORE-T	41.1	39.2	65.3	64.6	36.4	32.9
HCVR	**44.2**	**42.0**	**66.5**	**65.8**	**39.6**	**36.5**
Oracle	50.7	50.7	71.8	71.8	65.7	65.7

Qwen-based retrieval yields higher overall accuracy than CORE-T. HCVR achieves higher scores on both metrics across all three datasets with both SQL generators. This trend is consistent with the improvements observed with Llama-based retrieval in the main text. The improvements range from 1.2 to 9.5 percentage points in $\mathrm{EX}_{\mathrm{all}}$ and from 1.2 to 9.2 points in $\mathrm{EX}_{\mathrm{PR}}$.

E.8 A Recall-Oriented Configuration

The default policy keeps R unchanged when $P_1 \subsetneq R$ and the deletion is rejected. For schemas in which even R omits a connector, or when a small model returns an inconsistent review that falls back to R, a recall-oriented variant instead applies inferred-join addition to R in this case, the same operation Algorithm 1 uses when the two sets are disjoint. Table 17 compares the two configurations.

Unconditional recovery buys 0.2 to 1.3 PR points at a cost of 3.9 to 6.5 precision points. The recall-oriented variant raises PR in all six settings, by 0.2 to 1.3 points, while returning 0.2 to 0.4 more tables per query and losing 3.9 to 6.5 precision points together with F1 and Exact. The added tables therefore recover some missing connectors but more often add redundancy, which is why the default applies inferred-join addition only when the two views overlap partially or not at all. The variant remains available when complete coverage outweighs compactness.

E.9 Statistical Significance Tests

We test the differences between HCVR and CORE-T with paired tests over the per-query predictions of both methods, using our reproduced CORE-T runs (Appendix D) because per-query outputs are required. Retrieval comparisons cover precision, F1, PR, and Exact for both retrieval LLMs on the three corpora: precision and F1 use two-sided paired sign-permutation tests with 100,000 permutations, and PR and Exact use exact two-sided McNemar tests. Holm–Bonferroni correction is applied jointly across these 24 comparisons, and adjusted p-values below 0.05 are regarded as significant. For text-to-SQL we test $\mathrm{EX}_{\mathrm{all}}$ with exact two-sided McNemar tests over all queries at a nominal threshold of 0.05; $\mathrm{EX}_{\mathrm{PR}}$ is not tested separately, since under the definition of Section 4.1

Table 17: Default and recall-oriented configurations. The recall-oriented variant applies inferred-join addition to R whenever a proposed deletion is rejected. Best in bold, second best underlined within each pair.

Dataset	Configuration	P ↑	F1 ↑	PR ↑	Exact ↑	$\bar{T}$ ↓
Llama-3.1-8B-Instruct						
BIRD	Default	**64.4**	**72.2**	87.0	**26.6**	**3.2**
	Recall-oriented	58.3	67.7	**88.3**	20.5	3.6
Spider	Default	**71.4**	**79.3**	97.0	**44.8**	**2.4**
	Recall-oriented	66.8	75.3	**97.2**	40.6	2.8
MMQA	Default	**49.1**	**57.2**	63.5	**14.8**	**4.2**
	Recall-oriented	45.2	54.2	**63.9**	11.4	4.5
Qwen2.5-7B-Instruct						
BIRD	Default	**80.5**	**83.1**	84.6	**52.2**	**2.3**
	Recall-oriented	74.0	79.0	**85.7**	41.2	2.7
Spider	Default	**86.3**	**89.2**	93.0	**70.1**	**1.8**
	Recall-oriented	82.0	86.2	**94.1**	62.3	2.0
MMQA	Default	**60.5**	**64.9**	60.4	**31.0**	**3.2**
	Recall-oriented	56.6	62.4	**61.4**	25.1	3.5

Table 18: Paired retrieval comparisons between HCVR and reproduced CORE-T. Entries are Holm-adjusted p-values across the 24 comparisons; arrows mark significant increases (↑) or decreases (↓) for HCVR at the 0.05 level.

Dataset	P	F1	PR	Exact
Llama-3.1-8B-Instruct				
BIRD	< 0.001 ↑	< 0.001 ↑	0.0099 ↓	< 0.001 ↑
Spider	< 0.001 ↑	< 0.001 ↑	0.0540	< 0.001 ↑
MMQA	< 0.001 ↑	< 0.001 ↑	0.0126 ↑	< 0.001 ↑
Qwen2.5-7B-Instruct				
BIRD	< 0.001 ↑	< 0.001 ↑	0.0540	< 0.001 ↑
Spider	< 0.001 ↑	< 0.001 ↑	0.5104	< 0.001 ↑
MMQA	< 0.001 ↑	< 0.001 ↑	0.0540	< 0.001 ↑

it differs from $\mathrm{EX}_{\mathrm{all}}$ only by the queries answered correctly despite incomplete retrieval, at most 2.9–3.1 points for HCVR.

Precision, F1, and Exact improve significantly in all six settings; PR does not move uniformly. All 18 precision, F1, and Exact comparisons in Table 18 remain significant after Holm correction ($p_{\mathrm{adj}} < 0.001$). PR is significantly higher on MMQA with Llama ($p_{\mathrm{adj}} = 0.0126$), significantly lower on BIRD with Llama ($p_{\mathrm{adj}} = 0.0099$), and not significantly different in the other four settings. The statistical support therefore concerns precision, F1, and exact recovery, while complete coverage is statistically on par with CORE-T except for the two settings named.

Table 19: Paired significance tests of $\mathrm{EX}_{\mathrm{all}}$ against CORE-T. Values are unadjusted two-sided McNemar p-values; ↑ indicates a significant improvement at the nominal 0.05 level.

Retrieval model	SQL generator	BIRD	Spider	MMQA
Llama-3.1-8B	Llama-3.2-3B	0.1292	5.70×10^{-8} ↑	0.0080 ↑
	GPT-4o-mini	0.4149	0.0118 ↑	0.0126 ↑
Qwen2.5-7B	Llama-3.2-3B	4.41×10^{-5} ↑	1.00×10^{-11} ↑	0.0006 ↑
	GPT-4o-mini	0.0008 ↑	0.2414	0.0113 ↑

Overall execution gains are significant in nine of twelve settings. Table 19 shows that HCVR significantly improves $\mathrm{EX}_{\mathrm{all}}$ over CORE-T in most evaluated settings. With Llama-based retrieval,

Table 20: Search-budget experiments with Llama-3.1-8B-Instruct. $\overline{|U|}$ denotes mean union size.

	BIRD				Spider				MMQA			
K	Ex@1	Ex@K	$\overline{\lvert U\rvert}\downarrow$	PR_U	Ex@1	Ex@K	$\overline{\lvert U\rvert}\downarrow$	PR_U	Ex@1	Ex@K	$\overline{\lvert U\rvert}\downarrow$	PR_U
1	**35.1**	35.1	**2.3**	62.3	50.5	50.5	**1.9**	76.7	15.5	15.5	**3.1**	38.6
3	34.5	56.4	4.0	83.4	54.4	73.2	3.6	91.7	18.4	31.0	4.7	53.2
5	34.4	64.5	5.1	89.7	54.4	79.4	5.1	95.2	**19.1**	37.8	6.0	58.5
8	34.0	70.0	6.7	93.3	55.0	84.5	7.0	97.8	**19.1**	44.1	7.5	64.1
10	33.8	**72.0**	7.6	**94.4**	**55.3**	**86.9**	8.0	**98.3**	18.8	**46.2**	8.5	**67.1**

improvements are significant on Spider and MMQA with both SQL generators, while the BIRD differences are not significant. Qwen-based retrieval provides complementary evidence: improvements are significant on all three datasets with Llama-3.2-3B-Instruct and on BIRD and MMQA with GPT-4o-mini. The only nonsignificant Qwen-based comparison is GPT-4o-mini on Spider.

E.10 Parameter Experiments

Larger search budgets improve complete coverage. We vary $K \in \{1, 3, 5, 8, 10\}$ with $\alpha = 0.5$. Here, K specifies the number of candidate tables retrieved per requirement, the beam size, and the maximum number of retained compositions, reflecting the breadth of retrieval and composition search. Table 20 reports composition quality and candidate-union coverage before verification.

Increasing K consistently improves Ex@K and PR_U on all three datasets, while changes in Ex@1 are smaller and are not uniformly positive. This indicates that additional search budget primarily helps retain alternative compositions and preserve the tables needed for complete coverage. The improvement comes with a larger candidate union. We choose $K = 10$ to prioritize perfect recall before verification: among the evaluated settings, it achieves the highest PR_U on all three datasets, providing more complete candidate evidence for subsequent selection. For $K \geq 3$, Ex@1 remains relatively stable, varying by less than one percentage point on each dataset. This suggests that our role-aware composition ranking maintains stable top-ranked selection quality as the search space expands, while the additional budget primarily improves the coverage of retained alternatives.

Balanced relevance weights support both ranking and coverage. We vary the requirement-level relevance weight α from 0 to 1 in increments of 0.1, with $1 - \alpha$ assigned to whole-query relevance and $K = 10$. Table 21 reports the resulting composition quality and candidate-union coverage.

Table 21: Relevance-weight experiments with Llama-3.1-8B-Instruct and $K = 10$. All scores are percentages; $\overline{|U|}$ denotes mean union size.

	BIRD				Spider				MMQA			
α	Ex@1	Ex@10	$\overline{\lvert U\rvert}\downarrow$	PR_U	Ex@1	Ex@10	$\overline{\lvert U\rvert}\downarrow$	PR_U	Ex@1	Ex@10	$\overline{\lvert U\rvert}\downarrow$	PR_U
0.0	21.6	70.0	**7.2**	92.2	46.3	86.9	**7.5**	97.5	9.5	44.3	**8.1**	63.9
0.1	24.3	70.7	7.3	93.0	48.4	87.3	**7.5**	97.9	11.2	45.3	**8.1**	64.6
0.2	26.9	71.1	7.4	93.5	50.1	**87.5**	7.6	98.1	12.5	46.0	8.2	64.9
0.3	29.4	71.6	7.4	94.0	51.5	87.1	7.7	98.1	13.8	**46.5**	8.2	65.6
0.4	31.6	**72.0**	7.5	94.3	54.0	87.0	7.9	98.2	16.1	**46.5**	8.3	66.3
0.5	33.8	**72.0**	7.6	**94.4**	55.3	86.9	8.0	98.3	18.8	46.2	8.5	**67.1**
0.6	36.2	**72.0**	7.8	**94.4**	55.7	86.5	8.2	**98.6**	19.8	45.5	8.6	66.9
0.7	37.2	71.4	7.9	94.1	57.0	86.1	8.3	98.5	**21.2**	43.9	8.8	66.9
0.8	**38.5**	69.4	8.2	94.3	**57.2**	85.0	8.4	98.2	20.7	41.7	9.2	66.5
0.9	38.3	66.2	8.5	93.7	54.0	82.8	8.6	97.6	17.6	38.5	9.6	65.7
1.0	34.5	61.3	9.1	91.5	50.5	77.4	8.7	96.8	15.5	31.6	10.5	61.6

As the requirement-level weight increases, U progressively expands across all three datasets. This signal favors providers that closely match individual requirements, even when those matches are weakly related to the complete query. With insufficient whole-query guidance, more such tables

can enter the retained compositions, enlarging U without improving complete coverage. At high requirement-level weights, both Ex@10 and PR_U decline despite the larger unions, suggesting that additional tables do not compensate for missing suitable providers.

Conversely, shifting the weight toward whole-query relevance produces smaller unions but increasingly favors tables that match the question's dominant topic. Providers for less prominent yet necessary requirements, such as a specific filtering condition or requested attribute, may then receive insufficient weight. Their omission can reduce PR_U even when the retained tables are broadly relevant to the question. Balancing the two signals helps preserve providers for specific requirements while limiting tables that match those requirements in isolation but deviate from the complete query.

We choose $\alpha = 0.5$ to give equal weight to matching individual information requirements and maintaining alignment with the complete query. This choice is supported by the relatively stable Ex@10 and PR_U for $\alpha \in [0.4, 0.6]$, where both metrics vary by at most one percentage point on each dataset. Thus, $\alpha = 0.5$ provides a balanced default within a stable performance range.

Complete coverage remains stable across compatibility thresholds. We vary the compatibility threshold $\tau \in \{0.3, 0.5, 0.7\}$ with $K = 10$ and $\alpha = 0.5$. The threshold determines which table pairs can be connected during graph construction and composition: only edges with compatibility scores of at least τ are retained. Table 22 reports the resulting composition quality and candidate-union coverage before verification.

Table 22: Compatibility-threshold experiments with Llama-3.1-8B-Instruct.

	BIRD				Spider				MMQA			
τ	Ex@1	Ex@10	$\overline{\lvert U\rvert} \downarrow$	PR_U	Ex@1	Ex@10	$\overline{\lvert U\rvert} \downarrow$	PR_U	Ex@1	Ex@10	$\overline{\lvert U\rvert} \downarrow$	PR_U
0.3	**33.8**	72.0	7.6	94.4	**55.3**	86.9	8.0	98.3	**18.8**	46.2	8.5	**67.1**
0.5	32.8	74.4	7.6	**94.6**	54.0	**88.1**	**6.9**	98.3	18.7	**46.8**	8.3	66.1
0.7	**33.8**	**78.9**	**6.2**	94.1	54.0	86.8	7.1	**98.5**	16.7	43.8	**8.2**	66.0

Across the evaluated thresholds, PR_U varies by only 0.5, 0.2, and 1.1 percentage points on BIRD, Spider, and MMQA. Complete coverage is therefore relatively insensitive to the threshold within this range. Composition exactness shows more dataset-dependent changes: a stricter threshold improves Ex@10 on BIRD, whereas $\tau = 0.7$ reduces both Ex@1 and Ex@10 on MMQA. Increasing the threshold generally produces smaller unions, but does not consistently improve composition quality.

We choose $\tau = 0.3$ as a recall-oriented default. A lower threshold preserves more potential connections between providers, reducing the risk of excluding a useful connection solely because its compatibility score is modest. This leaves more alternatives available for subsequent composition and selection.

E.11 Token Consumption

Table 23 compares token consumption with Llama-3.1-8B-Instruct. Table 24 summarizes online calls for methods that require LLM participation. We use *small* for query decomposition or requirement extraction and *large* for table-selection-related processing following CORE-T, including deletion review. These labels distinguish call functions rather than token counts.

Table 23: Token consumption with Llama-3.1-8B-Instruct, in millions (M).

	BIRD			Spider			MMQA		
Method	Input ↓	Output ↓	Total ↓	Input ↓	Output ↓	Total ↓	Input ↓	Output ↓	Total ↓
ARM	51.7	**0.7**	52.4	–	–	–	–	–	–
ReAct	43.5	1.1	44.5	24.2	**0.7**	24.9	26.6	**0.8**	27.5
CORE-T	**10.8**	1.7	**12.5**	**4.8**	1.1	**6.0**	**5.7**	1.2	**7.0**
HCVR	26.4	2.1	28.5	15.0	1.3	16.4	16.4	1.7	18.1

HCVR balances retrieval quality and token consumption. HCVR uses 28.5M, 16.4M, and 18.1M tokens on BIRD, Spider, and MMQA. These totals cover requirement extraction, selection, and deletion review. The resulting token budgets are lower than ReAct's on all three datasets and ARM's on BIRD, but higher than CORE-T's. Together with the retrieval results, this reflects a trade-off: HCVR invests additional calls in modeling information requirements and reviewing deletions to improve table-set quality, while keeping token consumption below iterative retrieval and multi-draft selection in these comparisons.

Conditional review reduces unnecessary inference. HCVR uses one small call for requirement extraction and one large call for independent selection. A second large call performs deletion review only when required by the relation between the two selected table sets. Deletion review adds an average of 0.8 calls per query with Llama and 0.5 with Qwen, giving 2.8 and 2.5 calls per query. Compared with reviewing every query, this conditional design avoids unnecessary review calls and their token overhead.

Table 24: Online LLM calls for methods requiring LLM participation.

Method	LLM calls/query	Rationale
ReAct	Iteration-dependent	Alternates reasoning and retrieval, with further calls as the search proceeds.
JAR	1 small	Decomposes the query to guide join-aware combinatorial selection.
ARM	1 small + 3 large	Performs query alignment and generates multiple table-set drafts for aggregation.
CORE-T	1 large	Selects tables in one LLM pass, followed by compatibility-based adjustment.
HCVR	1 small + 1–2 large	Extracts requirements, independently selects tables, and reviews proposed deletions when needed.

F PROMPTS OF HCVR AND BASELINES

Figures 3–5 show the prompts of HCVR's three LLM calls: typed requirement extraction (Hypothesize), independent semantic selection (SELECT in Algorithm 1), and conservative deletion review (REVIEW). The semantic selection prompt is adapted from CORE-T's table-selection prompt (Soliman et al., 2026); it receives the retained union U in place of the dense top-K candidates together with their compatibility records, and it exposes neither the requirement assignments nor the composition ranking. The deletion-review decision is accepted only if its final action is consistent with its requirement-level and table-level assessments; an inconsistent or unparseable review falls back to keeping R (Appendix B.4). Figure 6 presents the shared table-set refinement prompt used in the direct-refinement controls, with P_1 or the independent semantic selection R supplied as the initial table set and U as the candidate set.

Figures 7 and 8 show the JAR-style and sub-query decomposition prompts used as requirement formulations in the C1 ablation; each extracted component becomes a retrieval fragment, after which the same embedding model, candidate budget, join graph, and composition procedure apply, so these controls isolate the decomposition style and are not the published JAR and ARM retrievers. Figure 9 presents the untyped fragment extraction prompt, which expresses information requirements as natural-language search fragments without typed triples. For readability, some few-shot demonstrations and reasoning templates are omitted from the displayed prompts.

Typed requirement extraction HCVR

Role

You are a SQL schema analyst. Extract information-need triples for multi-table retrieval from the question.

Input

Question: {question}

Task

Produce two groups of requirements:
- constraint: one local scope, condition, grouping, ordering, aggregation, or existence need on a semantic entity.
- answer: the final entity, attribute, or quantity requested.

Represent each requirement as (subject, predicate, object). Use a semantic entity as the subject. Keep explicit values and conditions attached to the constraint object. Preserve the owner entity in the retrieval fragment, especially for generic attributes such as name or address. Keep ranking evidence separate from the entity or attribute being returned.

Use the typed predicates in the inventory. Do not invent physical table names or split a broad attribute into finer attributes unless the question requires them. Extract explicit existence needs, but do not add relationship-only triples merely to anticipate joins.

Output

Return a JSON object with constraint and answer arrays. Each item contains triple, rationale, and semantic_fragment.

Example

Question: Which schools in Boston have more than 500 students?

```
{
  "constraint": [
    {"triple": "(school, filter_by, city {text in Boston})",
     "rationale": "The school is in Boston.",
     "semantic_fragment": "schools located in Boston"},
    {"triple": "(school, filter_by, enrollment {numeric > 500})",
     "rationale": "The school has more than 500 students.",
     "semantic_fragment": "schools with enrollment above 500"}
  ],
  "answer": [
    {"triple": "(school, return_entity, school)",
     "rationale": "Return the matching schools.",
     "semantic_fragment": "schools"}
  ]
}
```

Figure 3: HCVR prompt for typed requirement extraction (Hypothesize).

Independent semantic selection HCVR

Role

```
You are a SQL schema analyst. Select a comprehensive table set that is relevant to the
question and can be joined to answer it.
```

Input

```
Question: {question}
Candidate tables: {indexed_tables_in_U}
  Each table includes its name, schema, purpose, and sample rows.
Compatibility evidence: {scores_and_best_join_columns}
```

Task

```
Identify the entities, requested values, conditions, and relationships needed by the
question. Assess table relevance from the visible schema and samples. Inspect the supplied
compatibility evidence and check that the join columns make semantic sense.

Form candidate groups that provide coherent, connected evidence. Select the group that best
supports the complete question. Keep potentially necessary tables when uncertain. Remove a
table only when it is clearly irrelevant or incompatible.
```

Output interface

```
Use the supplied table indices consistently. Return valid JSON with this structure:
```

```
{
  "overall_reasoning": "...",
  "group_formation": {
    "reasoning": "...",
    "groups_formed": [
      {"group_index": 0, "table_indices": [0, 1],
       "group_description": "..."}
    ]
  },
  "group_selection": {
    "selected_group_index": 0, "reasoning": "...",
    "group_analysis": [{"group_index": 0, "reasoning": "..."}]
  }
}
```

Figure 4: HCVR prompt for independent semantic selection (SELECT).

Conservative deletion review HCVR

Role

```
You are a conservative table-removal judge. Decide whether the proposed tables can all be
removed while the remaining tables still support the complete question.
```

Input

```
Question: {question}
Before removal: {R}
After removal: {P1}
Proposed removals: {R_minus_P1}
Table schemas, samples, and column index: {visible_evidence}
Compatibility evidence: {possible_joins}
```

Task

```
1. List every requested value and condition, including grouping, ordering, aggregation,
   counting population, and relationships. For each, cite a table and column that remain
   after removal. Use MISSING when no remaining source is visible.

2. Assess each proposed table's unique required contribution. Use DELETE only if that
   contribution is NONE. Otherwise name the contribution, or use UNCERTAIN, and choose KEEP.
   Preserve necessary lookup and relationship tables and bridges. Similar names or
   compatibility scores do not prove redundancy.

3. Choose DELETE only if every table verdict is DELETE and no requirement has a MISSING
   source. Otherwise choose KEEP. Never invent a remaining column to justify deletion.
```

Output

```
{
  "question_requirements": [
    {"requirement": "...",
     "source_after_removal": "remaining_table.column or MISSING"}
  ],
  "table_assessments": [
    {"table_index": 0,
     "table_name": "...",
     "unique_required_contribution": "NONE or contribution or UNCERTAIN",
     "verdict": "DELETE or KEEP"}
  ],
  "final_action": "DELETE or KEEP"
}
```

Figure 5: HCVR prompt for conservative deletion review (REVIEW).

Table-set refinement ABLATION

Role

```
You are a database table-set refinement analyst.
```

Input

```
Question: {question}
Candidate table information: {tables_content}
Initial table set: {initial_tables}
Compatibility relationships: {compatibility_content}
```

Task

```
Given a user question, a set of candidate database tables, compatibility relationships, and
one explicitly marked initial table set, return the final table set needed to answer the
question.

You may:
- remove unnecessary tables from the initial set;
- add missing information tables or relationship/bridge tables from the candidate set;
- remove and add tables in the same decision;
- return the initial set unchanged when no correction is needed.

Never select a table outside the candidate set. Include every table needed for requested
values, filters, grouping, ordering, aggregation, counting population, named-value lookup,
and joins. Compatibility only indicates possible joinability; it does not prove relevance.
Prefer the smallest sufficient set, but never remove a table when doing so would lose
required information or a required connection.
```

Output

```
`final_table_indices` must be a non-empty array of unique integer table indices.

Return only valid JSON with exactly these fields and in this order:

{
  "reasoning": "brief evidence-based explanation",
  "final_table_indices": [0, 1]
}
```

Figure 6: Shared table-set refinement prompt applied to P_1 or R.

JAR-style decomposition BASELINE

Task

Decompose the question into non-composite noun concepts and attributes. If a concept is uncertain, provide the attribute. Wrap each component in <sub_c></sub_c> and terminate with <FIN></FIN>.

Input

Question: {question}

Example

Question: What is the aircraft name for flight number 99?

```
<sub_c>aircraft:name</sub_c>
<sub_c>flight:number</sub_c>
<FIN></FIN>
```

Figure 7: Prompt for JAR-style concept–attribute decomposition, used as a requirement formulation in the C1 ablation.

Sub-query decomposition BASELINE

Task

Decompose the user question into simpler subquestions that together cover the information needed to answer it.

Input

User question: {question}

Output

Return a JSON object containing a non-empty subquestions array.

Example

Question: What are the names of students who joined a club in 2022?

```
{
  "subquestions": [
    "Which students joined a club?",
    "Which memberships began in 2022?",
    "What are the names of those students?"
  ]
}
```

Figure 8: Prompt for sub-query decomposition, used as a requirement formulation in the C1 ablation.

Untyped fragment extraction ABLATION

Task

Given a question, write a few short search fragments that describe information needs and could help find the database tables needed to answer it.

Do not write Python code, pseudocode, or an algorithm. Return only one JSON object.

Output

Return only JSON in this form:

```
{"fragments": ["fragment 1", "fragment 2"]}
```

Example 1

Question: Which singers released albums in 2010?

Output:

```
{"fragments": ["singers", "albums released in 2010"]}
```

Example 2

Question: Which stations have more than 20 docks?

Output:

```
{"fragments": ["stations with more than 20 docks"]}
```

Example 3

Question: Which students joined clubs in 2022?

Output:

```
{"fragments": ["students", "clubs joined in 2022"]}
```

Input

Question: {question}

Figure 9: Prompt for untyped fragment extraction.