TABLE-ZERO: SELF-IMPROVEMENT FOR COMPLEX TABLE REASONING WITHOUT ADDITIONAL HUMAN SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Complex table reasoning requires agents to combine evidence across multiple tables through multi-step reasoning, but high-quality training data for these tasks is costly to build. Self-improving reinforcement learning (RL) offers a promising alternative: a model generates its own training questions and learns from them without human labels. However, existing self-improving methods struggle in this setting: it is hard for a model to generate questions that require multiple tables, and the reference answers to such questions may be wrong. To address this, we introduce TABLE-ZERO, a self-improvement framework that uses the table collection itself to choose which tables each question combines and to compute its answer. TABLE-ZERO selects related tables through two intrinsic properties of tables, joinability and decomposability. A proposer writes a question and reference SQL over these tables, and executing the SQL supplies the reference answer. The proposer and the solver then improve in turn: the proposer is trained with DPO on its own checked questions, and the solver is trained with GRPO on the resulting tasks, for three iterations. Across seven datasets, TABLE-ZERO achieves an average exact match of 32.3% with *Qwen3-8B* and 37.0% with *Qwen3-14B*, compared with 20.4% and 24.3% for PROMPTING, respectively. At both model sizes, it exceeds the average accuracy of adapted self-improvement baselines and RL trained on human-annotated tasks.

1 INTRODUCTION

Table question answering has been a long-standing research topic in natural language processing, spanning questions over individual tables (Pasupat & Liang, 2015; Herzig et al., 2021b), text-to-SQL semantic parsing (Yu et al., 2018), and question answering over tables augmented with textual context (Chen et al., 2020; Zhao et al., 2022). With the growing complexity of application scenarios, recent benchmarks move toward complex table question answering, where a model must find relevant tables in a large collection, connect records across tables, and perform filtering, aggregation, and comparison (Pal et al., 2023b; Wu et al., 2025; Wang et al., 2026; Solodko et al., 2026). Such questions arise in many high-impact applications, including financial analysis, biomedical research, and public-sector decision-making (Chen et al., 2021c; Zhu et al., 2021; Zhao et al., 2022; Lee et al., 2022; Koretsky et al., 2026; Solodko et al., 2026). For example, local education authorities deciding where to allocate additional staff and safety resources may need to compare class sizes and reported incidents across schools and neighborhoods. Such an assessment requires linking school-location records with class-size statistics and incident reports (Wang et al., 2026). Current large language models (LLMs) still find these tasks challenging, particularly when they must retrieve relevant tables and reason across them (Wu et al., 2025).

Building training data for complex table question answering is difficult. Each training question needs a reliable reference answer, and creating one requires annotators to locate related tables, write a question that needs several dependent reasoning steps, and verify the answer against the underlying data (Wang et al., 2026). Self-improving reinforcement learning (RL) offers a way to avoid this manual work. In methods such as R-Zero (Huang et al., 2025), a model generates its own training questions and learns from them through RL, and related methods have shown promise in mathematical reasoning, coding, and agentic search (Zhao et al., 2025; Liu et al., 2025; Lu et al., 2025; Yue

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.

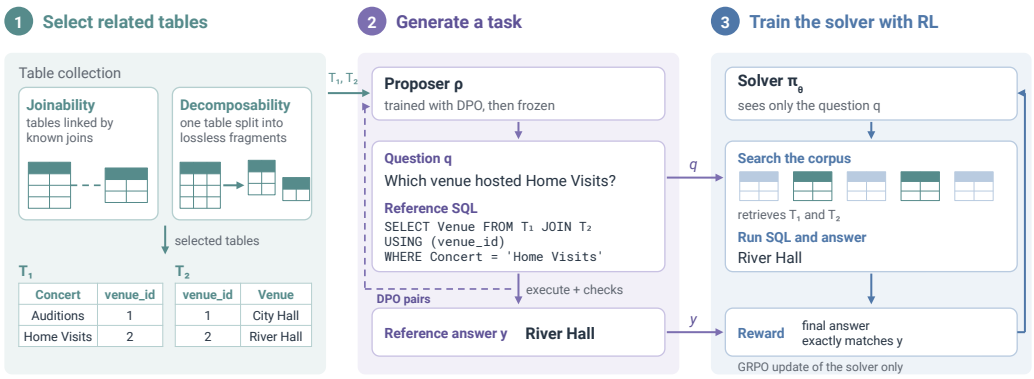


Figure 1: Overview of TABLE-ZERO. (1) Joinability or decomposability selects related tables. (2) The proposer writes a question and reference SQL; executing the SQL gives the reference answer, and checked proposals give DPO pairs for training the proposer. (3) The solver sees only the question, searches the tables and runs SQL, and is trained with RL.

et al., 2026). Tables appear well suited to this setting: their contents are structured, and the answer to a question over them can often be computed by executing a query.

However, existing self-improving methods struggle in this setting for three reasons. First, it is hard for a model to generate questions that require multiple tables. Second, the reference answers to such questions may be wrong, especially for multi-step questions, where each step is another chance for error. Third, training the proposer with GRPO, as these methods do, causes reward hacking: since each proposal is scored on its own, the proposer will raise its reward by repeating the same easy question. We observe the first two problems in a preliminary experiment that adapts DR. ZERO (Yue et al., 2026), a self-improving method for search agents, to MMQA with *Qwen3-8B*. Its trained proposer makes 120,000 generation attempts, but only 144 usable training questions remain, and 85 of them involve only one table. Most of the other questions are never answered correctly by the solver, which may reflect wrong reference answers (Section A).

To address this, we introduce TABLE-ZERO, a self-improvement framework for complex table reasoning that requires no additional human task annotation. As in prior self-improving methods, each model generates its own training questions: the proposer starts from the same model as the solver, and the two are trained in turn. The difference lies in how we use the environment. Prior methods use it to check generated tasks or to supply their content: Absolute Zero runs code to validate tasks and verify answers (Zhao et al., 2025), and SPICE generates questions from documents in a corpus (Liu et al., 2025). Our key idea is to also use the table collection to choose which tables each question must combine, so the proposer does not have to find related tables by trial and error. Two intrinsic properties of tables make this possible. *Joinability* lets us sample tables connected by known join relations, and *decomposability* lets us split one table into fragments that reconstruct it exactly. TABLE-ZERO works in three steps (Figure 1). First, it selects a set of tables using one of the two properties. Second, the proposer writes a question and reference SQL over these tables; executing the SQL gives the reference answer, and automatic checks decide whether to keep the task. Third, the solver learns through RL to retrieve relevant tables, combine their evidence, and compute the answer, with rewards for exact matches between its final answer and the reference answer.

We evaluate TABLE-ZERO with *Qwen3-8B* and *Qwen3-14B* on MMQA and six transfer datasets spanning table and passage reasoning (Table 1). On MMQA, exact match (EM) increases from 12.2% to 48.4% with *Qwen3-8B* and from 30.0% to 64.0% with *Qwen3-14B*. Across all seven datasets, TABLE-ZERO achieves the highest average EM among the evaluated methods: 32.3% and 37.0%, respectively, exceeding both adapted self-improvement baselines and Human Annotated RL (29.0% and 34.6%). Without further training, it improves over PROMPTING on all six transfer datasets at both model sizes.

2 Related Work

Table Reasoning. Prior work studies multi-step and multi-table reasoning over supplied sources (Wang et al., 2024; Pal et al., 2023a; Yang et al., 2025). Execution-based learning and synthetic task generation provide supervision for table reasoning and text-to-SQL (Yao et al., 2025; Zhong et al., 2020; Liu et al., 2022; Zhang et al., 2025). Open-domain table QA and multi-table retrieval extend this setting by requiring models to discover relevant sources (Herzig et al., 2021a; Chen et al., 2021a; 2025b; Agarwal et al., 2025). TABLE-ZERO uses Joinability and Decomposability to construct training tasks for agents to retrieve relevant tables, combine evidence, and derive answers.

Self-Improvement of Language Models. Prior work improves language models using model-generated reasoning traces and responses while retaining existing task supervision or human demonstrations (Zelikman et al., 2022; Singh et al., 2024; Chen et al., 2024). More recent methods generate training tasks through proposer–solver self-play, obtaining supervision through majority voting (Huang et al., 2025; Chen et al., 2025a), code execution (Zhao et al., 2025), or formal verification (Wilf et al., 2026). Related approaches construct tool-grounded tasks with executable verifiers (Zhou et al., 2025). External corpora provide another basis for task generation. SPICE (Liu et al., 2025) jointly trains a challenger that generates document-grounded questions and answers and a reasoner that answers without retrieval. Search Self-play trains a shared proposer–solver model using retrieval-assisted verification (Lu et al., 2025). Similarly, DR. ZERO (Yue et al., 2026) uses a search-equipped proposer to generate multi-hop questions and alternately improves the proposer and solver through reinforcement learning.

3 Method

Task formulation. We study complex table reasoning in the setting of multi-table question answering, where answering a question requires combining evidence across related tables through multiple reasoning steps (Wu et al., 2025). Given a question q and access to a table corpus $\mathcal{T}$, the solver retrieves relevant tables and executes SQL queries to produce an answer.

TABLE-ZERO improves this solver through reinforcement learning on automatically generated tasks, using final-answer exact-match rewards. A proposer initialized from the same model as the solver generates questions q and reference SQL p from training tables, so each model trains on its own questions; executing p supplies answer labels y for computing training rewards. The proposer and the solver are trained in turn for three iterations. In each iteration, the proposer is first updated with DPO, the updated proposer generates a task pool, and the solver is then trained with RL on that pool.

Self-improvement loop. Let $\rho^{(0)} = \pi^{(0)}$ be the initial model. For iterations $t = 1, 2, 3$,

$$\rho^{(t)} = \mathrm{DPO}\big(\rho^{(t-1)};\, \pi^{(t-1)}\big), \qquad \mathcal{D}^{(t)} \sim \rho^{(t)}, \qquad \pi^{(t)} = \mathrm{GRPO}\big(\pi^{(t-1)};\, \mathcal{D}^{(t)}\big). \tag{1}$$

The proposer learns from the current solver $\pi^{(t-1)}$ to write questions that the solver answers only some of the time (Equation (8)), and the solver then trains on these questions.

3.1 Task Construction

To generate training tasks for this self-improvement process, TABLE-ZERO exploits two intrinsic properties of tables: *joinability* and *decomposability* (Figure 2). These properties provide complementary ways to organize evidence across tables while retaining a basis for computing reference answers through SQL execution. Both routes record a required-source set G during task construction. Only q enters the solver's task prompt; reference SQL, answer labels, and G are withheld.

Joinability. Joinability is the ability to combine records across tables by matching values in corresponding columns (Deng et al., 2024). We use a *join graph* $\mathcal{J} = (\mathcal{T}, E)$, whose nodes are tables and whose edges record known join relations (Shanbhag & Sudarshan, 2014; Mohammad & Rezig, 2026). Each edge $e = (T_i, a, T_j, b)$ identifies columns a of T_i and b of T_j for an equality join:

$$T_i \bowtie_{T_i.a = T_j.b} T_j. \tag{2}$$

The operator $\bowtie$ combines rows whose specified column values match. We sample a connected set $S \subseteq \mathcal{T}$ of two or three tables; the proposer generates a question and reference SQL over S. These tables remain searchable, and G contains only those read by the accepted SQL. The reported graph

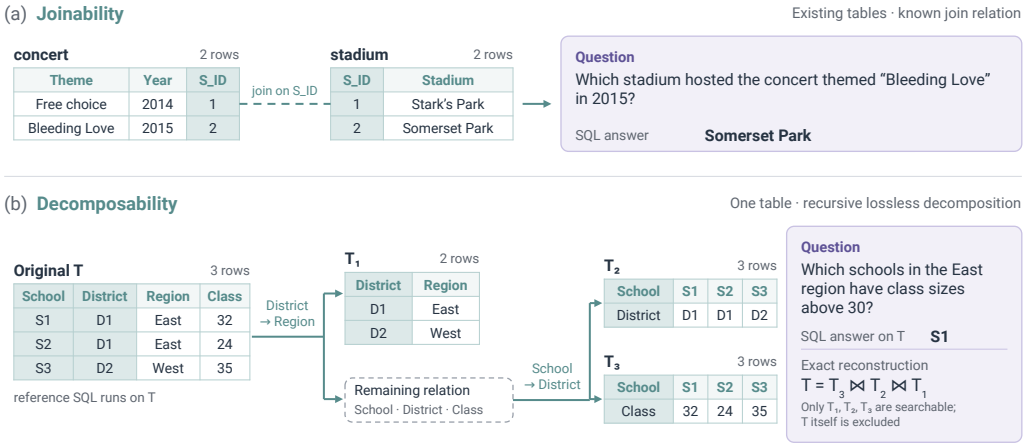


Figure 2: Task construction through (a) joinability and (b) decomposability. Panel (a) uses MMQA training records; (b) illustrates recursive FD-based decomposition with unequal fragment cardinalities. Reference SQL runs on the original table; only the final fragments are searchable in (b). The reported experiments use the shared-key specialization of decomposability.

is extracted from training SQL (Section 4.1), retaining this relational metadata while generating new tasks. Algorithm 1 and Section B.2 give the procedure.

Decomposability. We define decomposability through lossless relational decomposition (Silberschatz et al., 2019). Let $T(A)$ satisfy a functional dependency $X \to Y$, where $A = X \dot\cup Y \dot\cup Z$ partitions the attributes into disjoint sets. The dependency requires $t[X] = t'[X] \Rightarrow t[Y] = t'[Y]$ for all $t, t' \in T$ and permits the decomposition

$$T_1 = \pi_{X \cup Y}(T), \quad T_2 = \pi_{X \cup Z}(T), \qquad T = T_1 \bowtie_X T_2. \tag{3}$$

Here π denotes projection and $\bowtie_X$ denotes equality join on X. Projection can therefore produce fragments with different cardinalities. Recursive application yields a family $F = \{T_1, \ldots, T_m\}$ whose join reconstructs T (Figure 2 and Section B.3).

3.2 Training

In each iteration, the updated proposer builds a task pool using SQL execution and automatic checks, and the solver is trained on the accepted tasks (Equation (1)). The checks provide reproducible labels without establishing question–program faithfulness; filtering details and the question audit appear in Sections B.4 and D.

Final-answer reward. SQL execution supplies the reference answer y. We reward a tool-use trajectory τ_i when its final answer exactly matches this label:

$$r_i = \mathbf{1}[\mathrm{Match}(\mathrm{Final}(\tau_i), y)]\,. \tag{4}$$

The match compares complete typed rows, preserving cell types, column positions, and multiplicities, with row order enforced for order-sensitive tasks. The reward depends only on the final-answer match; it does not require a supporting SQL query or coverage of G.

Group-relative optimization. We use GRPO (Shao et al., 2024). For each task, the rollout policy π_{old} samples K trajectories using table search and SQL execution. We compute

$$\bar r = \frac{1}{K}\sum_{j=1}^{K} r_j, \qquad A_i = r_i - \bar r. \tag{5}$$

The baseline compares solutions to the same question. Following DAPO (Yu et al., 2025), we discard all-failure and all-success groups because $A_i = 0$ provides no relative learning signal. Retained groups contain both outcomes. Advantages are not divided by the group standard deviation.

Let $u_{i,t} = \pi_\theta(a_{i,t} \mid h_{i,t})/\pi_{\text{old}}(a_{i,t} \mid h_{i,t})$ be the probability ratio for generated token $a_{i,t}$ given interaction history $h_{i,t}$. The same trajectory advantage weights all L_i generated tokens. We minimize

$$\ell_{i,t} = -\min(u_{i,t}A_i, \text{clip}(u_{i,t}, 1-\epsilon_l, 1+\epsilon_h)A_i)\,,$$
$$\mathcal{L}(\theta) = \mathbb{E}\left[\frac{1}{K}\sum_{i=1}^{K}\frac{1}{L_i}\sum_{t=1}^{L_i}\ell_{i,t}\right] + \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{base}}). \tag{6}$$

The expectation is over retained task groups; losses are averaged within trajectories and then across trajectories. The thresholds ϵ_l and ϵ_h set asymmetric clipping ranges; the KL term regularizes the policy toward the base model.

Proposer training. We train the proposer ρ_ϕ with DPO (Rafailov et al., 2023) on its own checked proposals. Each generation context C is one construction unit from Section 3.1 (a joinable table set S with its join relations, or a fragment family F with its shared key), together with a requested operation and a few rare cell values that can anchor a question. Because C fixes the tables, valid proposals are frequent enough to form pairs, two proposals from the same C differ only in their question and SQL, and every check runs without human input.

For a proposal $x = (q, p) \sim \rho_\phi(\cdot \mid C)$, let $c_1(x), \ldots, c_M(x) \in \{0, 1\}$ be the checks (SQL execution, label audits that re-execute SQL variants, and use of the requested operation), and let $s(x)$ be the number of correct answers in K attempts ($K = 5$) by the current solver $\pi^{(t-1)}$. A proposal lies in the difficulty band if the solver answers it correctly at least once but not every time:

$$v(x) = \prod_{m=1}^{M} c_m(x), \qquad b(x) = \mathbf{1}\big[1 \le s(x) \le K-1\big]. \tag{7}$$

In each iteration, the proposer samples n proposals per context, and we pair proposals of the same context in two ways:

$$\mathcal{P} = \underbrace{\{(x^+, x^-) : v(x^+) = 1,\ v(x^-) = 0\}}_{\text{quality pairs}} \cup \underbrace{\{(x^+, x^-) : b(x^+) = 1,\ s(x^-) = K\}}_{\text{difficulty pairs}}. \tag{8}$$

A quality pair prefers a proposal that passes all checks over one that fails a check; a difficulty pair prefers a proposal in the band over one the solver always answers. With $r_\phi(x) = \beta \log \frac{\rho_\phi(x|C)}{\rho_{\text{ref}}(x|C)}$, we minimize

$$\mathcal{L}(\phi) = -\mathbb{E}_{(C,x^+,x^-)\sim\mathcal{P}}\Big[\log\sigma\big(r_\phi(x^+) - r_\phi(x^-)\big) + \frac{\alpha}{|x^+|}\log\rho_\phi(x^+ \mid C)\Big], \tag{9}$$

where log-probabilities sum over the proposer's tokens and the second term is a length-normalized likelihood on the preferred proposal ($\beta = 0.1$, $\alpha = 0.2$). In iteration t, the reference is the previous proposer, $\rho_{\text{ref}} = \rho^{(t-1)}$, and the updated proposer $\rho^{(t)}$ generates that iteration's task pool.

4 Experiments

4.1 Experimental Settings

Datasets & Models. We train on curated MMQA (Wu et al., 2025), a dataset for multi-hop reasoning across multiple relational tables, and evaluate transferability on six additional datasets. For table reasoning, NQ-Tables (Herzig et al., 2021b) contains questions answerable from Wikipedia tables, while Open-WikiTable (Kweon et al., 2023) requires operations such as aggregation, comparison, and sorting. For joint table and passage reasoning, OTT-QA (Chen et al., 2021b) combines Wikipedia tables and text, while LakeQA (Wang et al., 2026) requires discovering and combining evidence across a heterogeneous data lake. For passage reasoning, HotpotQA (Yang et al., 2018) requires multi-hop reasoning across Wikipedia passages, while MuSiQue (Trivedi et al., 2022) uses composed questions that require connected multi-hop reasoning. We initialize solvers from the post-trained *Qwen3-8B* and *Qwen3-14B* checkpoints (Qwen Team, 2025). For MMQA, we separate training and test sources by table-content family. Tables used for testing and their derived fragments are excluded from both task generation and solver training.

Baselines & Evaluation. All methods draw on the same training-side MMQA sources. For Table-Zero, each model writes its own questions with a proposer initialized from the same

Table 1: Exact match (%) on seven datasets. Avg. is the unweighted mean over datasets; Δ is the gain over the corresponding PROMPTING row, computed before rounding. Bold and underlined scores denote the best and second-best per model, including ties.

Method	Avg.	Table			Table + Passage		Passage		Δ
		MMQA	NQ-Tables	Open-WikiTable	OTT-QA	LakeQA	HotpotQA	MuSiQue	
Qwen3-8B									
+ PROMPTING	20.4	12.2	23.3	59.9	13.5	9.2	20.9	3.7	–
+ DR. ZERO	23.1	20.0	24.9	65.3	19.5	6.0	21.6	4.3	+2.7
+ SPICE	23.0	18.1	24.5	65.2	19.0	6.8	23.2	4.2	+2.6
+ TABLE-ZERO (w/o proposer training)	30.2	<u>43.3</u>	29.9	67.5	<u>24.0</u>	8.4	29.8	8.3	+9.8
+ TABLE-ZERO (GRPO)	<u>31.1</u>	42.0	**33.6**	**69.7**	23.4	**11.6**	29.9	7.7	<u>+10.7</u>
+ TABLE-ZERO (DPO)	**32.3**	**48.4**	<u>32.3</u>	67.9	**26.6**	<u>9.6</u>	**31.5**	**9.7**	**+11.9**
HUMAN ANNOTATED RL	29.0	37.1	30.3	<u>68.7</u>	22.2	5.2	<u>30.4</u>	<u>9.4</u>	+8.7
Qwen3-14B									
+ PROMPTING	24.3	30.0	26.8	60.0	17.5	7.6	23.4	4.8	–
+ DR. ZERO	28.8	49.5	25.8	63.1	20.4	12.0	25.2	5.5	+4.5
+ SPICE	28.4	47.7	28.6	60.0	17.6	15.6	24.3	5.2	+4.1
+ TABLE-ZERO (w/o proposer training)	35.1	62.7	29.4	70.7	<u>25.4</u>	15.6	<u>32.5</u>	<u>9.4</u>	+10.8
+ TABLE-ZERO (GRPO)	<u>35.3</u>	62.3	29.3	<u>71.9</u>	25.3	<u>17.6</u>	31.8	8.6	<u>+11.0</u>
+ TABLE-ZERO (DPO)	**37.0**	**64.0**	**30.0**	**72.5**	**27.8**	**18.4**	**34.9**	**11.5**	**+12.7**
HUMAN ANNOTATED RL	34.6	<u>63.0</u>	<u>29.6</u>	68.9	25.3	15.2	30.8	9.3	+10.3

model and trained with DPO (Equation (9)). Across all evaluated datasets, every solver uses the same tool interface: `search_value` to retrieve sources, `read` to inspect tables or passages, and `execute_sql` to query tables. For each dataset, we index its tables and passages, if applicable, for BM25 retrieval (Robertson & Zaragoza, 2009).

We compare TABLE-ZERO with PROMPTING, HUMAN ANNOTATED RL, and adaptations of DR. ZERO (Yue et al., 2026) and SPICE (Liu et al., 2025). PROMPTING uses the released checkpoint without additional training, while HUMAN ANNOTATED RL trains on 1,000 human-annotated tasks drawn from the training sources. Both DR. ZERO and SPICE train task proposers and filter generated tasks using solver feedback. We run each for three iterations, following their papers' setups. For each dataset, we estimate pass1 by averaging exact-match (EM) answer accuracy over eight rollouts per task. Evaluation prompts are in Section C.2.

4.2 OVERALL PERFORMANCE

Consistent gains across model sizes. Table 1 shows that TABLE-ZERO (DPO) achieves the highest average EM at both model sizes: 32.3% with *Qwen3-8B* and 37.0% with *Qwen3-14B*. These scores improve over PROMPTING by $+11.9\%$ and $+12.7\%$, respectively, from 20.4% and 24.3%. Compared with DR. ZERO, the strongest adapted self-improvement baseline by average EM at both sizes, the gains are $+9.2\%$ and $+8.2\%$. TABLE-ZERO (DPO) exceeds both adapted baselines on all seven datasets at both sizes. The largest gains over PROMPTING occur on MMQA: EM increases from 12.2% to 48.4% with *Qwen3-8B* and from 30.0% to 64.0% with *Qwen3-14B*. The corresponding TABLE-ZERO (GRPO) averages are 31.1% and 35.3%.

Transfer across evidence types. Without further training, TABLE-ZERO (DPO) improves over PROMPTING on all six transfer datasets with both *Qwen3-8B* and *Qwen3-14B*. Average EM over these six datasets rises from 21.8% to 29.6% and from 23.4% to 32.5%, respectively. Gains extend across table, joint table and passage, and passage reasoning. On HotpotQA, EM increases from 20.9% to 31.5% with *Qwen3-8B* and from 23.4% to 34.9% with *Qwen3-14B*; on MuSiQue, it increases from 3.7% to 9.7% and from 4.8% to 11.5%, respectively. The smallest transfer gain is on LakeQA with *Qwen3-8B* (9.6% versus 9.2%).

Comparison with annotated training. TABLE-ZERO (DPO) exceeds Human Annotated RL in average EM: 32.3% versus 29.0% with *Qwen3-8B* and 37.0% versus 34.6% with *Qwen3-14B*. With *Qwen3-8B*, it scores higher on six of seven datasets; Human Annotated RL performs better on

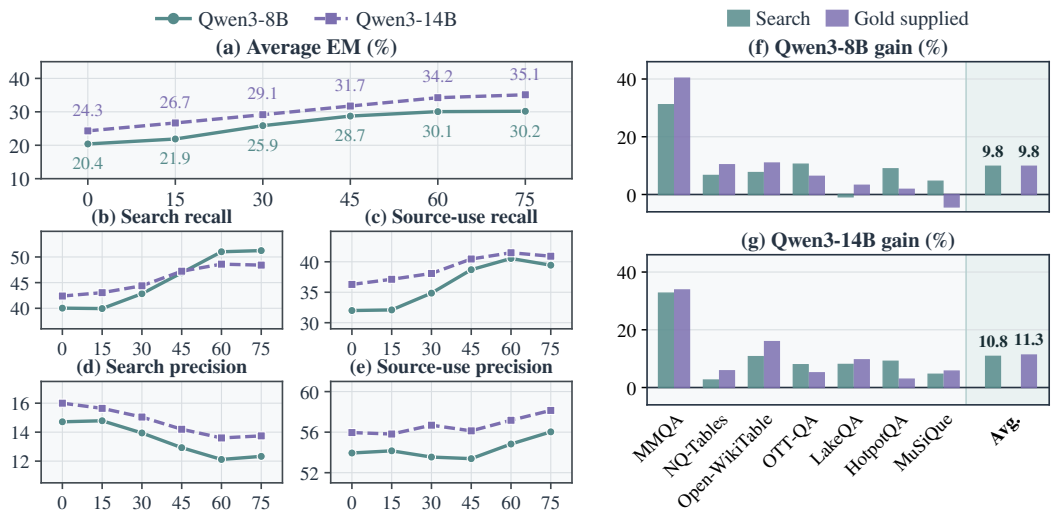


Figure 3: Training progress (left) and performance with supplied evidence (right) for TABLE-ZERO without further training using *Qwen3-8B* and *Qwen3-14B*. (a–e) Average EM and evidence metrics over seven datasets (%); step 0 is the initial checkpoint, and each curve represents one run. (f–g) EM gain of TABLE-ZERO over PROMPTING (%) with search versus supplied gold evidence. Avg. is the unweighted dataset mean; absolute scores are in Table 6.

Open-WikiTable (68.7% versus 67.9%). With *Qwen3-14B*, TABLE-ZERO (DPO) scores higher on all seven datasets.

4.3 IMPROVEMENT ANALYSIS

The analyses in this section use TABLE-ZERO without proposer training with *Qwen3-8B* and *Qwen3-14B*. Figure 3 combines accuracy and evidence metrics across checkpoints (a–e) with a comparison of TABLE-ZERO and PROMPTING under search and supplied gold evidence (f–g).

4.3.1 RQ1: WHICH CAPABILITIES IMPROVE DURING TRAINING?

Source discovery and use. We track TABLE-ZERO before training and every 15 updates through step 75, using the same evaluation questions within each run. Search recall measures whether retrieved sources cover all required labelled evidence; source-use recall requires all of it to be read or queried. Search precision measures the fraction of returned items that are labelled evidence, while source-use precision measures this fraction among used items. Figure 3(b–e) reports the unweighted average over all seven datasets. Full definitions and aggregation rules are in Section E.

Both runs improve evidence coverage, with the largest gains during steps 30–60. Over this interval, average search recall rises from 42.8% to 51.0% with *Qwen3-8B* and from 44.4% to 48.6% with *Qwen3-14B*. Source-use recall also increases, peaking at step 60 at 40.5% and 41.5%, respectively, before declining to 39.4% and 40.9% at step 75. Training thus improves both finding the required evidence and accessing it, but these gains do not continue at every checkpoint.

Precision develops differently. From PROMPTING to step 75, average search precision decreases from 14.7% to 12.3% with *Qwen3-8B* and from 16.0% to 13.7% with *Qwen3-14B*, so a smaller proportion of retrieved items is relevant. Source-use precision instead rises from 53.9% to 56.0% and from 56.0% to 58.1%, with most of the increase after step 45. Evidence coverage and precision therefore develop at different rates: evidence coverage improves mainly during the middle of training despite lower search precision, followed by more modest gains in the precision of source use.

Answering with supplied evidence. To assess performance when labelled evidence is available from the outset, we supply handles to the gold sources and evaluate PROMPTING and TABLE-ZERO on all seven datasets. All other evaluation settings match Table 1. The models still need to inspect these sources and produce an answer. On HotpotQA and MuSiQue, all four models begin by reading `t1`, average two to three reads per question, and make at most 0.05 searches per question, indicating that they use the supplied handles.

Figure 3(f–g) shows that the average advantage of TABLE-ZERO over PROMPTING is maintained when gold evidence is supplied. For *Qwen3-8B*, the average EM gain is $+9.8\%$ in both settings; with *Qwen3-14B*, it is $+10.8\%$ with search and $+11.3\%$ with supplied evidence. Thus, the overall advantage does not depend solely on discovering the relevant sources. With supplied evidence, gains persist on all five datasets involving tables, with the largest improvements on MMQA ($+40.3\%$ with *Qwen3-8B* and $+33.8\%$ with *Qwen3-14B*).

The passage-only results are more mixed: on HotpotQA, the advantage narrows from $+8.9\%$ to $+1.8\%$ with *Qwen3-8B* and from $+9.1\%$ to $+2.9\%$ with *Qwen3-14B* relative to the searching evaluation (Table 1). On MuSiQue, TABLE-ZERO is below PROMPTING by 4.3% with *Qwen3-8B* and above it by 5.7% with *Qwen3-14B*.

4.3.2 RQ2: HOW DOES TABLE-ZERO PERFORM AS TRAINING PROGRESSES?

Figure 3(a) reports average EM over the same seven datasets using the shared evaluation prompt. Both runs improve at every evaluated checkpoint and reach their highest observed average at step 75. For *Qwen3-8B*, average EM increases from 20.4% before training to 30.2%, with only a small gain from step 60 to step 75 (30.1% to 30.2%). The *Qwen3-14B* run improves from 24.3% to 35.1%, including an increase from 34.2% to 35.1% over the final interval. Thus, the *Qwen3-8B* run approaches a plateau over the observed checkpoints, while the *Qwen3-14B* run retains a larger gain late in training. Neither trajectory establishes whether gains would continue beyond 75 updates.

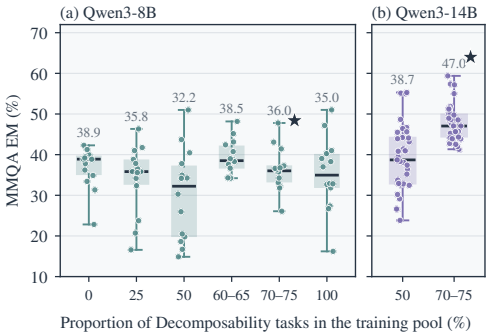


Figure 4: MMQA EM (%) of RL checkpoints by Decomposability proportion over five splits. Stars mark the TABLE-ZERO (DPO) checkpoints in Table 1.

Viewed together, panels (a–e) show that the accuracy gains during steps 30–60 coincide with increasing search and source-use recall. Over the final interval, however, EM rises while source-use recall falls at both sizes. Source-use precision instead increases from 54.8% to 56.0% with *Qwen3-8B* and from 57.2% to 58.1% with *Qwen3-14B*. Late accuracy gains thus coincide with more precise source use, even as complete-evidence use becomes less frequent. These aggregate curves do not isolate the causal contribution of each capability.

4.3.3 RQ3: WHAT IS THE QUALITY OF THE SELF-GENERATED TRAINING DATA?

We conducted a study surveying a senior database expert with more than 5 years of industry experience in database development to review 15 human-annotated tasks and 15 tasks proposed by TABLE-ZERO. Each item included a task, the supplied tables to answer it, and a reference answer; the order of the tasks is randomized and unknown to the surveyee. The expert rated clarity, answerability, and the correctness of the reference answer on three-point scales.

Table 2: Expert ratings of 15 human-annotated and 15 TABLE-ZERO tasks.

Criterion	Human-annotated	TABLE-ZERO
Clarity	2.87	2.93
Answerability	2.80	3.00
Answer correctness	3.00	3.00

Table 2 reports the expert's ratings. Mean clarity is 2.87 for human-annotated tasks and 2.93 for tasks proposed by TABLE-ZERO, while mean answerability is 2.80 and 3.00, respectively. Within this reviewed sample, some human-annotated tasks contain ambiguity or require unstated assumptions, which may introduce difficulty beyond the intended reasoning requirements. The higher ratings for TABLE-ZERO are consistent with its use of decomposition to construct clearly specified questions grounded in the supplied evidence. The questionnaire, rating anchors, and an example are in Section F.

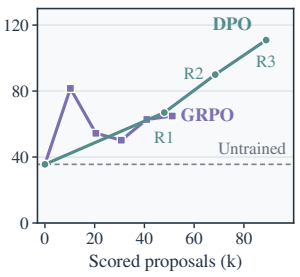
4.4 ABLATION STUDIES

We examine three design choices: the mix of the two construction routes, the training-source split, and proposer training.

(a) EM (%) by source fold

	s1	s2	s3	s4	s5	mean ± SD
Avg.	**29.6**	**30.0**	**30.3**	**30.2**	**28.6**	**29.7 ± 0.7**
MMQA	49.7	49.7	46.9	43.3	41.3	46.2 ± 3.8
OTT-QA	21.4	23.0	23.5	24.0	22.9	23.0 ± 1.0
NQ-Tables	29.8	29.2	30.3	29.9	28.3	29.5 ± 0.8
LakeQA	9.6	7.2	8.0	8.4	8.0	8.2 ± 0.9
Open-WikiTable	65.0	65.9	67.1	67.5	65.8	66.3 ± 1.0
HotpotQA	25.3	27.4	28.1	29.8	27.3	27.6 ± 1.6
MuSiQue	6.3	7.4	8.1	8.3	6.7	7.4 ± 0.9

(b) Usable tasks per 10k



(c) Seed tables covered (%)

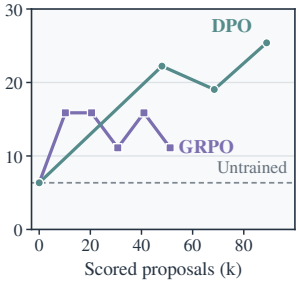


Figure 5: (a) TABLE-ZERO (DPO) with *Qwen3-8B* on each source fold; color shows the difference from the dataset's five-fold mean (teal above, violet below), and the right column gives mean ± SD (Section D.1). (b, c) Proposer training on held-out seed tables, against the number of tasks scored during training. A task is usable if it passes all checks, including the label audit, and falls in the solver's difficulty band. R1–R3: DPO rounds; dashed line: untrained proposer.

Effect of the Decomposability proportion. We first ask how the mix of the two construction routes affects training. Figure 4 compares RL checkpoints trained on pools with different proportions of Decomposability tasks, the rest being Joinability tasks (boxes: middle half; whiskers: full range; black line: median; 14 checkpoints per proportion for *Qwen3-8B* and 35 for *Qwen3-14B*). For *Qwen3-8B*, the median MMQA EM stays between 32.2% and 38.9% for proportions from 0% to 100%, and the ranges overlap, so the proportion has no clear effect across splits and updates. For *Qwen3-14B*, pools with 70–75% Decomposability tasks give a higher median than balanced pools (47.0% vs. 38.7%) and a narrower range.

Robustness across training-source splits. We assess TABLE-ZERO with *Qwen3-8B* across five MMQA train/test splits. Each split holds out a different group of table families, which are excluded from task generation and training (Section D). Each solver is evaluated on its held-out MMQA questions and the same six transfer datasets. Across the reported checkpoints, average EM ranges from 28.6% to 30.3%, with a mean of 29.7% and a standard deviation of 0.7% (Figure 5a). MMQA varies most (mean 46.2%, standard deviation 3.8%), while each transfer dataset has a standard deviation of at most 1.6%. Exact scores and interval details can be found in Section D.1.

Proposer training ablation. We next ask whether training the proposer improves the tasks it provides to the solver, comparing DPO with GRPO and with the untrained proposer. GRPO rewards each task by the checks it passes, with 32 samples per prompt; DPO uses the preference pairs in Equation (8). We count usable tasks per 10,000 proposals on held-out seed tables, where a usable task passes all checks, a rule-based audit of its label, and the difficulty band, and plot them against the number of tasks scored during training (Figure 5b). The untrained proposer yields 36. GRPO is volatile: it peaks at 82 after about 10,000 scored tasks and then falls to 50–65. DPO improves in every round, from 67 to 90 to 111, 3.1 times the untrained proposer. DPO also covers more held-out seed tables: 16 of 63, against 10 for GRPO and 4 for the untrained proposer (Figure 5c).

5 CONCLUSION AND LIMITATIONS

We introduced TABLE-ZERO for agent self-improvement in complex table reasoning. Joinability and decomposability support multi-table task construction, and SQL execution supplies reference answers for exact-match rewards. Across seven datasets, TABLE-ZERO reaches an average EM of 32.3% with *Qwen3-8B* and 37.0% with *Qwen3-14B*, above the adapted self-improvement baselines and Human Annotated RL.

The method is limited to relational tables, and the Joinability route uses relations extracted from training SQL, so it reduces but does not remove prior supervision. Execution gives reproducible labels but does not guarantee that the reference SQL matches the question; the question audit examines this (Section D). Next, we analyze proposer training in detail for TABLE-ZERO without further training. Future work should strengthen question–program verification, measure variation across training runs, and study proposer learning over more rounds.

AI USE STATEMENT

We used generative AI assistants for manuscript editing and for assistance with analysis, evaluation, and job-orchestration code. The vector overview figure was developed with AI assistance and revised against the implementation. These uses do not replace author review or independent result verification. A hosted language model (Claude Opus 5) also assisted the question pool's question–program faithfulness audit (Section D). This audit is separate from the human expert study in Section F. The research questions, framework, and experimental design are the authors' own. We take responsibility for the final text, claims, and artifacts, including those produced with generative AI assistance.

ETHICS STATEMENT

Experiments use publicly released benchmark data and open-weight proposer and solver models. The task-quality study involves one senior database expert, who rated 15 human-annotated and 15 generated tasks for clarity, answerability, and reference-answer correctness (Section F).

Generated questions and trained models can reproduce biases in the source data and amplify model errors. Execution-derived labels make program outputs auditable, but do not establish that a question has a unique interpretation or that its reference SQL is faithful to that interpretation. The dataset audit and expert judgments address aspects of task quality; neither guarantees the reliability of every generated task or model answer.

REPRODUCIBILITY STATEMENT

Sections B and 3 describe task construction and its deterministic checks. Section D documents MMQA curation and table-family separation. The settings and reported checkpoints for the current comparison are given in Section C.1; source-evidence metrics and the supplied-evidence evaluation are described in Section E. Section F describes the expert study, and Section C.4 states the limits of these experiments. The training-progress and source-evidence analyses cover one TABLE-ZERO training run per model size, using different MMQA folds.

We plan to release the task-construction code, generated task pools, evaluation code, and trained checkpoints with the camera-ready version.

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. *arXiv preprint arXiv:2511.00805*, 2025. URL https://arxiv.org/abs/2511.00805.

Lili Chen, Mihir Prabhudesai, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Self-Questioning Language Models. *arXiv preprint arXiv:2508.03682*, 2025a. URL https://arxiv.org/abs/2508.03682.

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 Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30298–30317, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1463. URL https://aclanthology.org/2025.acl-long.1463/.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Wang, and William W. Cohen. Open Question Answering over Tables and Text. In *The Ninth International Conference on Learning Representations*, 2021a. URL https://arxiv.org/abs/2010.10439.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Yang Wang, and William W. Cohen. Open question answering over tables and text. In *International Conference on Learning Representations*, 2021b.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3697–3711, 2021c.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

Yuhao Deng, Chengliang Chai, Lei Cao, Qin Yuan, Siyuan Chen, Yanrui Yu, Zhaoze Sun, Junyi Wang, Jiajun Li, Ziqi Cao, Kaisen Jin, Chi Zhang, Yuqing Jiang, Yuanfang Zhang, Yuping Wang, Ye Yuan, Guoren Wang, and Nan Tang. LakeBench: A benchmark for discovering joinable and unionable tables in data lakes. *Proceedings of the VLDB Endowment*, 17(8):1925–1938, 2024. doi: 10.14778/3659437.3659448.

Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Eisenschlos. Open domain question answering over tables via dense retrieval. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 512–519, Online, June 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.43. URL https://aclanthology.org/2021.naacl-main.43/.

Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Eisenschlos. Open domain question answering over tables via dense retrieval. In *Proceedings of NAACL-HLT*, pp. 512–519, 2021b. doi: 10.18653/v1/2021.naacl-main.43.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Zhu, Jiaxin Huang, Haitao Mi, and Dong Yu. R-Zero: Self-evolving reasoning LLM from zero data. *arXiv preprint arXiv:2508.05004*, 2025. ICLR 2026.

Mathew J. Koretsky, Maya Willey, Owen Bianchi, Chelsea X. Alvarado, Tanay Nayak, Nicole Kuznetsov, Sungwon Kim, Mike A. Nalls, Daniel Khashabi, and Faraz Faghri. BiomedSQL: Text-to-SQL for scientific reasoning on biomedical knowledge bases. In *Proceedings of the Conference on Language Modeling (COLM)*, 2026.

Sunjun Kweon, Yeonsu Kwon, Seonhee Cho, Yohan Jo, and Edward Choi. Open-WikiTable: Dataset for open domain question answering with complex reasoning over table. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8285–8297, 2023. doi: 10.18653/v1/2023.findings-acl.526.

Gyubok Lee, Hyeonji Hwang, Seongsu Bae, Yeonsu Kwon, Woncheol Shin, Seongjun Yang, Minjoon Seo, Jong-Yeup Kim, and Edward Choi. EHRSQL: A practical text-to-SQL benchmark for electronic health records. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2022.

Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. SPICE: Self-play in corpus environments improves reasoning. *arXiv preprint arXiv:2510.24684*, 2025.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. TAPEX: Table Pre-training via Learning a Neural SQL Executor. In *The Tenth International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2107.07653.

Hongliang Lu, Yuhang Wen, Pengyu Cheng, Ruijin Ding, Haotian Xu, Jiaqi Guo, Chutian Wang, Haonan Chen, Xiaoxi Jiang, and Guanjun Jiang. Search self-play: Pushing the frontier of agent capability without supervision. *arXiv preprint arXiv:2510.18821*, 2025. ICLR 2026.

Mir Mahathir Mohammad and El Kindi Rezig. Qualitative join discovery in data lakes using examples. *Proceedings of the ACM on Management of Data*, 4(1):68:1–68:28, 2026. doi: 10.1145/3786682.

Vaishali Pal, Andrew Yates, Evangelos Kanoulas, and Maarten de Rijke. MultiTabQA: Generating tabular answers for multi-table question answering. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6322–6334, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.348. URL https://aclanthology.org/2023.acl-long.348/.

Vaishali Pal, Andrew Yates, Evangelos Kanoulas, and Maarten de Rijke. MultiTabQA: Generating tabular answers for multi-table question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6322–6334, 2023b. doi: 10.18653/v1/2023.acl-long.348.

Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *Proceedings of ACL-IJCNLP*, pp. 1470–1480, 2015. doi: 10.3115/v1/P15-1142. URL https://aclanthology.org/P15-1142/.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

Anil Shanbhag and S. Sudarshan. Optimizing join enumeration in transformation-based query optimizers. *Proceedings of the VLDB Endowment*, 7(12):1243–1254, 2014. doi: 10.14778/2732977.2732997.

Zhihong Shao, Peiyi Wang, Qihao Zhu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. *Database System Concepts*. McGraw-Hill, 7 edition, 2019. URL https://www.db-book.com/.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2312.06585.

Michael Solodko, Steven Gong, Guangwei Yu, Satya Krishna Gorti, Jesse C. Cresswell, and Victor Zhong. LakeQuest: A three-domain benchmark for grounded question answering across data lakes. In *Proceedings of the Conference on Language Modeling (COLM)*, 2026. URL https://arxiv.org/abs/2607.12310.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Haonan Wang, Jiaxiang Liu, Yurong Liu, Austin Senna Wijaya, Tianle Zhou, Eden Wu, Yijia Chen, Wanting You, Reya Vir, Daniela Pinto Veizaga, Grace Fan, Yusen Zhang, Juliana Freire, and Eugene Wu. Lakeqa: An exploratory qa benchmark over a million-scale data lake. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. Chain-of-Table: Evolving Tables in the Reasoning Chain for Table Understanding. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2401.04398.

Alex Wilf, Pranjal Aggarwal, Bryan Parno, Daniel Fried, Louis-Philippe Morency, Paul Pu Liang, and Sean Welleck. Propose, solve, verify: Self-play through formal verification. In *Proceedings of the International Conference on Machine Learning*, 2026. arXiv:2512.18160.

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*, 2025. URL https://openreview.net/forum?id=GGlpykXDCa.

Zheyuan Yang, Lyuhao Chen, Arman Cohan, and Yilun Zhao. Table-R1: Inference-time scaling for table reasoning. *arXiv preprint arXiv:2505.23621*, 2025. EMNLP 2025 Main.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Zhewei Yao et al. Arctic-Text2SQL-R1: Simple rewards, strong reasoning in text-to-SQL. *arXiv preprint arXiv:2505.20315*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, et al. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, 2025.

Tao Yu, Rui Zhang, Kai Yang, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of EMNLP 2018*, 2018.

Zhenrui Yue, Kartikeya Upasani, Xianjun Yang, Suyu Ge, Shaoliang Nie, Yuning Mao, Zhe Liu, and Dong Wang. Dr. Zero: Self-evolving search agents without training data. *arXiv preprint arXiv:2601.07055*, 2026. COLM 2026.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022. arXiv:2203.14465.

Yuhao Zhang, Shaoming Duan, Jinhang Su, Chuanyi Liu, and Peiyi Han. SPFT-SQL: Enhancing large language model for text-to-SQL parsing by self-play fine-tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025. arXiv:2509.03937.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025. NeurIPS 2025.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 6588–6600, 2022. URL https://aclanthology.org/2022.acl-long.454.

Victor Zhong, Mike Lewis, Sida I. Wang, and Luke Zettlemoyer. Grounded adaptation for zero-shot executable semantic parsing. In *Proceedings of EMNLP 2020*, 2020. arXiv:2009.07396.

Yifei Zhou, Sergey Levine, Jason Weston, Xian Li, and Sainbayar Sukhbaatar. Self-Challenging Language Model Agents. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2506.01716.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pp. 3277–3287, 2021.

A PRELIMINARY EXPERIMENT WITH DR. ZERO

We adapt DR. ZERO (Yue et al., 2026) to task generation for complex table reasoning. Using *Qwen3-8B* on MMQA fold s3, we sample 120,000 proposals in three completed sampling rounds from the same trained proposer over 2,400 seed tables. Proposer checks, per-seed selection, and deduplication leave 1,327 unique candidate questions. Only 150 fall within the target difficulty band, defined as one to seven successful rollouts out of eight from a fixed judge solver; 1,137 receive no successful rollouts. Reserving six questions for validation leaves 144 training tasks, of which 85 involve only one table. In this experiment, a large sampling budget therefore yields a small training pool still dominated by single-table tasks. These observations motivate task construction that explicitly organizes evidence across related tables (Section 3.1). This preliminary study is separate from the baseline solver reruns in Table 1.

B METHOD AND OPTIMIZATION DETAILS

This appendix specifies task records, construction and verification procedures, and the solver optimization used in Section 3. Experimental settings are collected in Section C.

Algorithm 1 summarizes the full procedure.

Algorithm 1 Table-Zero self-improvement

Require: training tables $\mathcal{T}$, join edges E, proposer ρ, initial solver π_θ

for iteration $i = 1, 2, 3$ **do** ▷ proposer and solver improve in turn
  sample several (q, p) per prompt from ρ and run the checks below
  pair, within a prompt, a proposal that passes with one that fails, and
  one the current solver π_θ sometimes answers with one it always answers
  update ρ by DPO on these pairs (Equation (9)), with the current ρ as reference
  task pool $\mathcal{D} \leftarrow \emptyset$ ▷ generated with the updated ρ
  for each generation attempt with configured route r **do**
    if $r =$ JOINABILITY **then**
      sample two or three connected tables S from $(\mathcal{T}, E)$
      $C \leftarrow$ schemas, sampled rows, and join keys of S; $T_{\text{ref}} \leftarrow S$
      $\mathcal{T}_{\text{solver}} \leftarrow \mathcal{T}$
    else ▷ Decomposability: reported shared-key case
      choose $T(A) \in \mathcal{T}$ with unique key k ($k \to A$); partition $A \setminus \{k\}$ into $A_1, \ldots, A_m$
      $T_j \leftarrow \pi_{\{k\} \cup A_j}(T)$ for $j = 1, \ldots, m$; $F \leftarrow \{T_1, \ldots, T_m\}$, where $m \in \{2, 3\}$
      skip this attempt unless $T_1 \bowtie_k \cdots \bowtie_k T_m = T$ under SQL execution
      $C \leftarrow$ schema and sampled rows of T, plus fragment schemas; $T_{\text{ref}} \leftarrow \{T\}$
      $\mathcal{T}_{\text{solver}} \leftarrow (\mathcal{T} \cup F) \setminus \{T$ and same-base catalog variants$\}$
    end if
    $(q, p) \sim \rho(\cdot \mid C)$; execute p on T_{ref}
    $G \leftarrow$ tables read by p (Joinability), or fragments matched by its non-key columns
    if execution, coverage, answer-format, and task-quality checks pass (Section B.4) **then**
      re-execute p and serialize its complete typed result as y
      add $(q, y, G, \mathcal{T}_{\text{solver}})$ to $\mathcal{D}$
    end if
  end for
  for each solver update **do**
    sample tasks from $\mathcal{D}$ and K tool-use trajectories per task; prompt with q only
    compute final-answer exact-match rewards (Equation (4)); discard equal-reward groups
    update π_θ using group-relative advantages and the objective in Equation (6)
  end for
end for
return trained proposer ρ and solver π_θ

B.1 TASK RECORDS AND SOURCE ACCESS

Let $\mathcal{T}_{\text{tr}}$ denote a fold's training tables and E_{tr} a graph of column-pair relations between these tables. The graph is environment metadata; task generation does not require a collection of reference training programs as input. Each task record is (C_π, q, p, y, G, d), where C_π specifies source metadata and corpus exclusions, q is the question, p is the reference SQL, y is its execution result, and G is the required-source subset. Optional d is the current solver's empirical success rate. The proposer

ρ generates (q, p) from context C_ρ. The full tool environment $\mathcal{E}(C_\pi)$ enforces task-specific source exclusions; neither C_π nor G is an oracle prompt for the solver. Task generation is restricted to training tables and does not access development or test tables.

Apart from shared instructions and tool definitions, the solver policy π receives only q. It searches for candidate sources, reads tables or passages, and queries retrieved tables with SQL. A trajectory τ ends with a final answer. The current training reward checks this answer against y; source coverage is measured separately in the evidence analysis (Section E).

B.2 JOINABILITY: CONTEXT SAMPLING

The Joinability route takes a join graph $(\mathcal{T}_{\text{tr}}, E_{\text{tr}})$ and samples connected contexts (Algorithm 2). Each edge identifies two tables and their joining columns. Declared foreign keys can supply these relations without reference training SQL. We retain only edges whose endpoint tables belong to the training corpus. The reported experiments use the graph construction described in Section C.3; a fully schema-derived graph is an alternative input, not a reported ablation. We do not infer links from column names or similar cell values. To sample two or three connected tables, we downweight frequently chosen seed edges. With probability $1/2$, we add a third table through a uniformly sampled frontier edge, if one exists. Only seed selections update usage counts. The proposer receives schemas, the first three rows of each selected table, and the recorded join keys. These tables remain searchable in the full corpus; G contains only those actually read by the accepted reference SQL.

Algorithm 2 Sampling a context from the join graph

Require: training tables $\mathcal{T}_{\text{tr}}$, join edges E_{tr}
$E \leftarrow \{e \in E_{\text{tr}} : \text{both endpoint tables are in } \mathcal{T}_{\text{tr}}\}$
seed usage $u(e) \leftarrow 0$ for each $e \in E$
if $E = \emptyset$ **then**
  return no join-graph context
end if
For each context sample, retaining E and u:
sample seed $e_0 \sim \text{Cat}((1 + u(e))^{-1} : e \in E)$
$u(e_0) \leftarrow u(e_0) + 1$; $S \leftarrow \{e_0\}$; $V_S \leftarrow$ endpoint tables
while $|V_S| < 3$ and $\text{Bernoulli}(1/2) = 1$ **do**
  $B \leftarrow \{e \in E : e \text{ has exactly one endpoint table in } V_S\}$
  if $B = \emptyset$ **then**
    break
  end if
  sample e uniformly from B
  $S \leftarrow S \cup \{e\}$; add the new endpoint of e to V_S
end while
return schemas, first three rows, and recorded keys for (V_S, S)

B.3 DECOMPOSABILITY: CONSTRUCTION DETAILS

The Joinability route uses supplied relations between existing tables. The Decomposability route creates multi-table structure from a single table while preserving its values. Given a table with a verified unique key, we partition its non-key columns into two or three disjoint groups and copy the key into each fragment. We retain a split only if joining the fragments on this key reconstructs the source exactly, row by row, without missing or duplicate rows.

Functional-dependency basis. For a source instance T with columns A and verified unique key k, $k \to A$ means that agreement on k implies agreement on every column. Let $A_1, \ldots, A_m$ partition $A \setminus \{k\}$ and define $T_j = \pi_{\{k\} \cup A_j}(T)$. Each fragment retains the determinant k, so the lossless-join condition gives

$$T = T_1 \bowtie_k \cdots \bowtie_k T_m. \tag{10}$$

The shared key prevents combining values from different source rows. This is a key-based special case of FD-supported lossless decomposition; the evaluated construction does not discover arbitrary FDs or normalize schemas. We verify the key on the source instance and check reconstruction under the actual SQL semantics, including preservation of row multiplicities.

General FD decomposition. For a binary decomposition of an attribute set A into B and C, $B \cup C = A$, the condition

$$(B \cap C) \to B \quad \text{or} \quad (B \cap C) \to C \tag{11}$$

is sufficient for a lossless join (Silberschatz et al., 2019). The construction in Equation (3) takes $B = X \cup Y$ and $C = X \cup Z$, with $X \to Y$. To recurse, initialize $F_0 = \{T\}$ and replace a selected relation $R \in F_t$ using a dependency valid on that relation:

$$F_{t+1} = (F_t \setminus \{R\}) \cup \{\pi_{X \cup Y}(R), \pi_{X \cup Z}(R)\}. \tag{12}$$

Here X, Y, Z partition the attributes of R. Requiring $Y, Z \neq \emptyset$ ensures each child has fewer attributes than its parent; successive lossless steps preserve the reconstruction invariant $\bowtie F_t = T$. These relational identities use set semantics. An SQL realization must also preserve row multiplicities and check reconstruction with its actual join and null semantics.

Figure 2(b) illustrates two steps: `District` $\to$ `Region` first separates a two-row lookup from a three-row relation; `School` $\to$ `District` then decomposes the latter into two three-row relations. The resulting `SchoolClass`, `SchoolDistrict`, and `DistrictRegion` tables reconstruct the three-row original. The illustration uses synthetic records and describes the general construction, not an additional training condition. All reported decomposition runs use the shared-key specialization.

The proposer receives the original schema, its first three rows, and the fragment schemas. It generates (q, p) for the original table, on which execution supplies y. The solver searches a corpus containing the verified fragments; the original and its same-base catalog variants are excluded. We derive G by matching non-key column names in the SQL text to their fragments and require at least two such fragments. This lexical check does not establish that each fragment is semantically necessary. Lossless reconstruction preserves the source data and reference answer.

B.4 Task generation and verification details

The proposer jointly generates a question q and reference SQL p. Deterministic checks require successful execution, route-specific source coverage, and a small answer in an allowed format (Algorithm 3). They also filter question-shape and schema-style patterns and literal answer leakage. Both routes require a nonnumeric cell value of at least four characters in the question. This value must occur in at most three training tables for the Joinability route, or four tables in the catalog plus fragments for decomposition. Within each proposal run, normalized question and SQL duplicates are removed; corpus assembly further removes near-duplicate questions. The corpus builder re-executes accepted SQL and serializes its complete result rows as JSON, with row-order sensitivity set by each task's `keep_order` flag. SQL execution does not establish question–program faithfulness; the separate question audit is described in Section D.

Algorithm 3 Dual-route task generation

Require: training tables $\mathcal{T}_{\text{tr}}$, join edges E, proposer ρ; solver source access is governed by $\mathcal{E}(C_\pi)$
select route $r \in \{\text{JOINABILITY}, \text{DECOMPOSABILITY}\}$ for the task pool
if $r = \text{JOINABILITY}$ **then**
  $C_\rho \leftarrow \text{SAMPLEJOINGRAPHCONTEXT}(E, u)$ using Algorithm 2
  $T_{\text{ref}} \leftarrow$ selected tables; $C_\pi \leftarrow$ source metadata, no exclusions
else ▷ Decomposability
  split a keyed table T into fragments F; require exact reconstruction
  $C_\rho \leftarrow (T, \text{schemas}(F))$; $T_{\text{ref}} \leftarrow \{T\}$
  $C_\pi \leftarrow$ fragment metadata, excluding T and same-base catalog variants
end if
$(q, p) \sim \rho(\cdot \mid C_\rho)$; $z \leftarrow \text{EXECUTE}(p, T_{\text{ref}})$
$G \leftarrow$ tables read by p (join), or fragments matched by non-key columns (decomposition)
if (q, p, z, G) fails execution, coverage, answer, question, anchor, or duplicate checks **then**
  return REJECT
end if
$y \leftarrow \text{SERIALIZERESULTROWS}(z)$; standardize the answer-format instruction in q
return (r, C_π, q, p, y, G)

B.5 SOLVER OPTIMIZATION

Equations (4) to (6) define the reward, mean-centered advantage, and full clipped objective. The loss averages generated-token terms within each trajectory before averaging trajectories. Tool observations are not generated-token targets. Equal-reward groups are discarded, and advantages are not divided by their group standard deviation.

Current training conditions. TABLE-ZERO and Human Annotated RL combine online filtering with difficulty weighting of retained task groups. The adapted DR. ZERO and SPICE solver reruns retain their respective GRPO and Dr. GRPO recipes without these additions. Their rollout counts, common update budget, and reported checkpoints are summarized in Section C.1.

B.6 INTERPRETING DYNAMICALLY FILTERED TRAINING REWARD

DAPO-style dynamic filtering retains groups with nonzero within-group reward variation. Mean reward over retained groups is therefore conditional on this selection and is not an estimate of accuracy over the entire training pool. As the policy changes, different tasks can enter or leave the retained set. A flat retained-group reward curve can coexist with changing task coverage; held-out performance requires a separate, fixed evaluation protocol.

C EXPERIMENTAL SETTINGS

C.1 TRAINING AND REPORTED CHECKPOINTS

The main comparison uses post-trained *Qwen3-8B* and *Qwen3-14B* solvers (Qwen Team, 2025). Each model setting has one training run per method. The initial checkpoint is denoted step 0. All solver runs have a budget of 75 updates and save checkpoints every 15 updates. The interaction budget is eight turns and 8,192 tokens per trajectory. Generated pool sizes, proposer training, and total compute remain method-specific.

Training conditions. All methods access the same training-side MMQA sources within a model setting. PROMPTING is the released checkpoint without additional RL. HUMAN ANNOTATED RL trains on 1,000 annotated tasks. For TABLE-ZERO, each model generates its own tasks with a DPO-trained proposer initialized from the same model, and executing their reference SQL supplies the answer labels. The DR. ZERO and SPICE adaptations train proposers and filter tasks using solver feedback. Their solver reruns reuse the resulting task pools with GRPO and Dr. GRPO, respectively; DR. ZERO uses five rollouts per prompt and SPICE uses eight. TABLE-ZERO and Human Annotated RL use online group filtering and difficulty weighting, which are not added to the baseline solver reruns. Thus, these comparisons vary the task-construction and optimization recipes together, rather than isolating the task source under identical training compute.

Method	***Qwen3-8B***	***Qwen3-14B***
PROMPTING	0	0
DR. ZERO	30	75
SPICE	30	60
TABLE-ZERO (w/o proposer training)	75	75
TABLE-ZERO (GRPO)	75	75
TABLE-ZERO (DPO)	75	75
Human Annotated RL	60	75

Table 3: Training steps of the checkpoints evaluated in Table 1. The same checkpoints are used for all seven datasets within each row.

Table 3 records the reported checkpoints without assuming a common selection rule across methods. The TABLE-ZERO rows use the step-75 endpoints; the training-progress analysis separately reports every saved checkpoint. No transfer dataset is used for checkpoint selection, and transfer involves no further training.

System prompt

```
You are a tabular QA agent.
Answer the question by searching for relevant tables and passages, executing
grounded SQL, reading passages with read, and then returning exactly one
final answer in this form:
<final_answer>ANSWER_IN_THE_QUESTION_REQUESTED_FORMAT</final_answer>

search_value returns trajectory-local handles such as t1 and t2 for tables
and passages.  SQL may reference only the exact table handles currently
listed by the tool; a passage cannot be queried with SQL, read its text with
read(ref, 1, 1).  Copy those handles into SQL or read; never invent a handle
or use a remembered global table ID. Different handles with the same
source_name can be incompatible table variants, so choose by the displayed
columns and sample values.  Use only displayed column names.  If SQL fails
or its result is insufficient, correct it using the tool observations.  Once
a successful result answers the question, emit the requested final-answer
format and do not put the final answer inside a tool call.
```

User message template

```
Each answer row holds only the value or values the question asks for.  Reads
are limited, so read a table in blocks, read(ref, 1, 40), never one row at a
time; a passage is read with read(ref, 1, 1).  Return [] only when nothing
observed answers the question.  Return your answer as a JSON array of rows
inside <final_answer>...</final_answer>.  Include all requested answer
fields in the order requested by the question; preserve duplicate answer
rows.  Do not add column names, prose, or Markdown fences.  Use [] for an
empty result.  Row order is not graded; column order and duplicate
multiplicity are graded.  <QUESTION>
```

Figure 6: Shared solver evaluation prompts. The user message concatenates the guidance, answer-format instructions, and the evaluation question, represented by `<QUESTION>`.

C.2 Shared tools, prompt, and scoring

All solvers use `search_value`, `read`, and `execute_sql`. Each dataset has its own BM25 index of tables and, where available, passages (Robertson & Zaragoza, 2009). Search returns source handles, read exposes their contents, and SQL queries retrieved tables. The shared interface permits the same solver to operate on table, mixed, and passage datasets; their corpora and evidence annotations remain dataset-specific.

Figure 6 gives the prompt and user-message used by the baselines and the diagnostic analyses: Table-Zero (DPO) with *Qwen3-8B* and *Qwen3-14B*. In this setting, the system prompt is identical across datasets and methods. Each question is sent in one user message, with the shared guidance first, followed by the answer-format instructions and the question.

Dataset	Evidence type	Questions
MMQA	Table	300
NQ-Tables	Table	966
Open-WikiTable	Table	2,000
OTT-QA	Table + Passage	2,214
LakeQA	Table + Passage	250
HotpotQA	Passage	2,000
MuSiQue	Passage	2,417

Table 4: Evaluation sets used in the current comparison and training curves. MMQA uses the 300 held-out questions for the corresponding source partition. Question sets remain fixed across methods and checkpoints within each model setting.

For MMQA, strict EM requires a complete match to the typed reference rows, preserving cell types, column positions, and multiplicities, with row order enforced for order-sensitive tasks. Pass@1 is the mean EM over eight rollouts per question and then over questions; it is not the fraction of questions solved by at least one rollout.

C.3 JOIN-GRAPH PROVENANCE

The Joinability route uses a graph constructed by extracting alias-resolved column equalities from existing training SQL. The builder records distinct table–column endpoint pairs and predicate occurrence counts without executing the source queries, then restricts the graph to the training tables. Occurrence counts do not affect context sampling. This graph supplies relational metadata; the proposer generates new questions and reference programs, and executing those programs supplies their labels. The construction therefore retains inherited SQL-derived metadata. A graph built solely from schema declarations is an alternative input, not a completed ablation in the current experiments.

C.4 INTERPRETATION AND SCOPE

The main comparison describes the model setting. *Qwen3-8B* and *Qwen3-14B* use different MMQA source partitions, so their differences do not isolate model size. The additional *Qwen3-8B* study in Figure 5a covers five source partitions; its mean weights all folds equally. MMQA test questions differ across folds, while transfer questions remain fixed. Fold s3 uses step 60 and the others use step 75, so the spread does not isolate training-seed variability. The figure's nominal t intervals assume independent fold scores without adjusting for overlapping training sources; their coverage is not established (Section D.1). Fold-matched baseline results are not provided for this additional study. A shared update count does not imply equal numbers of generated trajectories, task-pool sizes, or total compute. The comparisons with DR. ZERO and SPICE concern the adaptations described above and vary task generation together with solver optimization. Reported checkpoint steps also differ across methods. The current results do not estimate variation over independent training seeds or task-pool draws.

Training curves use fixed evaluation sets, making changes across steps comparable within a run. However, aggregate evidence metrics do not identify the causal contribution of retrieval, source selection, or answer generation to EM. The supplied-evidence intervention and the small expert study provide complementary evidence with their own limitations (Sections E.3 and F).

D MMQA PROVENANCE AND CONTENT-LEVEL ISOLATION

Curated question pool. The MMQA pool contains 1,513 questions retained by a three-stage audit: solve-and-compare with the production executor, a uniform re-review, and an outcome-blind consistency review. The audit retains 1,398 questions and recovers 115 initially flagged questions; it excludes 294 questions with recorded reasons. Retained questions and reference SQL are unchanged, apart from removing a legacy answer instruction. The audit includes model-assisted judgments of question–program faithfulness. These judgments are separate from the deterministic execution and structural checks used in task construction: a successful SQL execution establishes the program's output, but does not establish that the program answers the question as intended.

Table-family separation. Tables are grouped into 267 families using row content, including content-duplicate clusters, reviewed pairs, co-references, and paraphrases. Training and test sources are separated by table family. Fragments, the join graph, and the training corpus are constructed from the training side of each fold. An ancestry check excludes held-out tables from fragment parents, catalog tables, and graph nodes. Exact and near-duplicate question checks supplement this table-family separation. Held-out table content and its derived fragments are excluded from task generation and solver training; held-out schemas may still resemble training schemas.

Scope of the audit. The audit reduces known reference-data problems but does not certify question–program faithfulness. In the recorded audit, solve-and-compare alone identified about 12% of the concerns later confirmed by people, and blind review identified about half. Residual ambiguity and question–program mismatches can therefore remain. This dataset audit is separate from the expert evaluation of 15 annotated and 15 generated tasks reported in Section F.

D.1 Results across source partitions

Table 5 reports the five *Qwen3-8B* settings shown in Figure 5a. Each uses a different MMQA training-source partition. The MMQA test questions also differ across folds, whereas each transfer dataset uses the same question set for all five solvers. Fold s3 uses its step-60 checkpoint, and the other folds use step 75. The observed spread therefore reflects the reported source partitions and checkpoints, rather than isolating training-seed variability. Avg. is the unweighted mean of seven dataset scores within each fold; its summary statistics are computed from these five means before rounding.

Table 5: *Qwen3-8B* results across five MMQA source partitions. Scores and summary statistics are on the EM (%) scale. Fold s3 uses step 60; the other folds use step 75. Means, standard deviations, and interval bounds are computed from the supplied scores before rounding.

Dataset	**s1**	**s2**	**s3**	**s4**	**s5**	**Mean**	**SD**	**Nominal 95% t interval**
MMQA	49.7	49.7	46.9	43.3	41.3	46.18	3.79	[41.48, 50.88]
NQ-Tables	29.8	29.2	30.3	29.9	28.3	29.50	0.78	[28.53, 30.47]
Open-WikiTable	65.0	65.9	67.1	67.5	65.8	66.26	1.02	[64.99, 67.53]
OTT-QA	21.4	23.0	23.5	24.0	22.9	22.96	0.98	[21.75, 24.17]
LakeQA	9.6	7.2	8.0	8.4	8.0	8.24	0.88	[7.15, 9.33]
HotpotQA	25.3	27.4	28.1	29.8	27.3	27.58	1.62	[25.57, 29.59]
MuSiQue	6.3	7.4	8.1	8.3	6.7	7.36	0.86	[6.29, 8.43]
Avg.	29.59	29.97	30.29	30.17	28.61	29.73	0.68	[28.89, 30.56]

Circles show individual fold scores at 50% opacity; the open circle denotes s3. Diamonds mark the arithmetic mean, and bars show the nominal 95% Student t interval

$$\bar{x} \pm t_{0.975,4} \frac{s}{\sqrt{5}}, \qquad t_{0.975,4} = 2.7764,$$

where s is the sample standard deviation with denominator $5 - 1$. This calculation treats the five reported scores as independent, identically distributed observations and does not adjust for overlapping training sources or the checkpoint difference. Its nominal 95% coverage is therefore not established for this experiment. The intervals do not isolate uncertainty from question sampling or independent retraining.

E Evidence and training-progress analysis

E.1 Evidence matching and aggregation

Figure 3(a–e) tracks Table-Zero without further training with *Qwen3-8B* and *Qwen3-14B* at steps 0, 15, 30, 45, 60, and 75. Panel (a) reports average EM; panels (b–e) report search recall, source-use recall, search precision, and source-use precision. Each checkpoint uses the fixed question sets and decoding settings in Table 4. Each plotted mean gives equal weight to all seven datasets, including MMQA.

Labelled evidence. The analysis includes both tables and passages wherever the dataset provides their labels. A table is used if the solver reads it or references it in a successful SQL query; a passage is used if the solver reads it. Article-level annotations match any passage from the annotated article. For OTT-QA, any labelled answer passage satisfies the passage requirement when a passage is required. Consequently, these measures assess coverage of the dataset's evidence labels, not whether every retrieved or read source is semantically necessary to derive the answer.

Complete-evidence recall. Search recall is the fraction of trajectories for which the union of search results covers all required labelled evidence. Source-use recall applies the same completeness test to sources used through read or SQL. Both are all-required-evidence success rates: partially covering the gold set does not receive fractional credit.

Precision. Search precision is the fraction of returned items that match labelled evidence, pooling item counts over all searches. Source-use precision first computes the fraction of used items that

match labelled evidence within a trajectory, then averages over trajectories that use at least one item. The eligible trajectories can therefore differ across checkpoints and methods. Search precision and source-use precision also weight observations differently. Neither should be combined with complete-evidence recall to form F1: item-based precision and all-evidence recall have different units and denominators.

Coverage and uncertainty. Only 82.1% of HotpotQA questions and 72.2% of MuSiQue questions have every labelled article represented in the index. Evidence recall is thus limited by corpus coverage as well as solver behavior. The source analysis uses 2,000 paired bootstrap resamples of questions for comparisons with step 0, retaining all eight rollouts when an MMQA question is sampled. These comparisons cover question-sampling uncertainty, not independent training runs, and do not test changes between adjacent checkpoints. The figure shows point estimates without uncertainty bands because confidence intervals for the seven-dataset averages are not available.

E.2 Accuracy curves and numerical provenance

The curves use per-dataset scores before display rounding. EM is taken from the supplied checkpoint-accuracy results, and evidence metrics are taken from the checkpoint-level source report. The source reports differ for one EM entry: *Qwen3-8B* on MuSiQue at step 30 is 0.046 in the accuracy results and 0.045 in the evidence report. Panel (a) retains 0.046 consistently with the accuracy series; panels (b–e) use only the evidence metrics. This difference changes the seven-dataset EM mean by approximately 0.014% and does not change the displayed one-decimal value.

E.3 Evaluation with supplied gold evidence

For PROMPTING and the diagnostic runs described above, we supply handles to labelled gold sources at the start of each trajectory. The questions, checkpoints, three-sentence prompt, decoding, scoring, and budgets match the corresponding original rows of Table 1; only the supplied handles change. Search remains available. The solver must still inspect the sources and produce the answer, so the intervention removes the need to discover the supplied handles without supplying a reasoning trace or answer.

On HotpotQA and MuSiQue, all four model conditions begin with a read of `t1`, average two to three reads per question, and make at most 0.05 searches per question. This behavior supports interpreting the scores as performance with the supplied handles. Table 6 reports the absolute scores corresponding to Figure 3(f–g).

Table 6: Exact match (%) with gold-source handles supplied; variants are identified in the method rows. Avg. is the unweighted mean across seven datasets; Δ is the gain over the corresponding Qwen row, computed before rounding. Bold denotes the higher score within each model size.

		Table			Table + Passage		Passage		
Method	**Avg.**	**MMQA**	**NQ-Tables**	**Open-WikiTable**	**OTT-QA**	**LakeQA**	**HotpotQA**	**MuSiQue**	Δ
Qwen3-8B	35.7	39.6	34.1	59.2	21.6	14.0	48.1	**33.3**	–
+ TABLE-ZERO (DPO)	**45.5**	**79.9**	**44.4**	**70.1**	**27.9**	**17.2**	**49.9**	29.0	**+9.8**
Qwen3-14B	37.3	48.0	41.8	51.9	23.0	17.2	51.0	27.9	–
+ TABLE-ZERO (DPO)	**48.5**	**81.8**	**47.6**	**67.8**	**28.1**	**26.8**	**53.9**	**33.6**	**+11.3**

Supplying evidence does not define a strict upper bound on searching performance: Open-WikiTable EM decreases for both initial models and for TABLE-ZERO with *Qwen3-14B*. A changed initial evidence context can alter the trajectory even when other settings are fixed. The comparison therefore tests whether gains persist with labelled sources available, rather than measuring a pure reasoning effect. Averages and differences use the supplied three-decimal scores; paired confidence intervals are unavailable.

F Expert evaluation of training tasks

Review protocol. The study in Section 4.3.3 includes 15 human-annotated tasks and 15 tasks proposed by TABLE-ZERO, presented in randomized order to one database expert with more than five

years of industry experience. The questionnaire presents each item with its question, two or three complete tables labelled A, B, and C, and a reference answer. The reviewer was instructed to assess the question as written using only these tables, without supplying missing assumptions. A separate practice item preceded the 30 study items and was excluded from the reported ratings.

The reviewer was asked to rate clarity and answerability and select the smallest sufficient set of tables before reading the reference answer, then rate reference-answer correctness. Whether an item requires at least two tables can be derived from the selected set. Each rating allows one response; *Cannot assess* is available when the item cannot be evaluated, for example because material is unreadable, and must be selected alone for the table-set question. Skipped items remain unrated; *Cannot assess* is not a numeric score. Missing necessary information receives Answerability 1, and substantive ambiguity that changes the answer receives Reference-answer correctness 1. The instructions ask the reviewer to ignore harmless formatting differences in the reference answer.

Table 7: Three-point rating anchors from the expert questionnaire.

Criterion	**1**	**2**	**3**
Clarity	Major ambiguity or missing conditions	Minor wording issues needing clarification	Clear and sufficiently specified
Answerability	Necessary information is unavailable	Answering needs an unstated assumption	The available information is sufficient
Reference-answer correctness	Incorrect, or depends on substantive ambiguity that changes the answer.	Partially correct: contains errors, omissions or incorrect extra answer elements.	Fully correct: correct and complete.

Questionnaire example. Table 8 reproduces study item Q01, including all rows of both supplied tables and its reference answer. The completed questionnaire records a score of 3 for each criterion and selects both tables as the smallest sufficient set.

Table 8: Actual questionnaire item Q01. The reference answer was placed after the clarity, answerability, and table-selection questions.

Question. How many distinct codes exist for skills with the description TV, Video?

Table A (3 rows)

skill_id	skill_description
1	Electrical
2	Mechanical
3	TV, Video

Table B (3 rows)

skill_id	skill_code
1	ELEC
2	MECH
3	TV

Reference answer. 1

These ratings describe a small sample evaluated by one expert. They do not estimate agreement between reviewers or establish the quality of every task in either training pool.

G COMPARISON OF TASK CONSTRUCTION AND TRAINING SIGNALS

Table 9: Task construction and training signals in related methods. Question sources describe the training procedure, not whether a model has previously received human supervision. Execution-based filtering and answer verification are distinct from the source of training targets.

Method	Training questions	Training signal	RL
STaR (Zelikman et al., 2022)	Annotated questions	Rationales filtered by reference answers	✗
GAZP (Zhong et al., 2020)	Generated from sampled SQL	SQL targets; execution-based cycle filtering	✗
SPFT-SQL (Zhang et al., 2025)	Annotated and generated	SQL targets; execution-based verification	✗
Arctic-Text2SQL-R1 (Yao et al., 2025)	Annotated and synthetic	SQL execution agreement and validity	✓
Table-R1-Zero (Yang et al., 2025)	Annotated questions	Task-specific rewards against references	✓
R-Zero (Huang et al., 2025)	Model-generated	Majority-vote answers	✓
Absolute Zero (Zhao et al., 2025)	Model-generated	Code execution	✓
SPICE (Liu et al., 2025)	Corpus-conditioned generation	Answers extracted from documents	✓
Search Self-play (Lu et al., 2025)	Search-based generation	Answer matching; RAG verification	✓
DR. ZERO (Yue et al., 2026)	Search-based generation	Proposer answers; solver feedback	✓
TABLE-ZERO	Generated from relational tables	SQL-derived labels; final-answer exact match	✓