GRIPS: Grounded Pre-SQL Signature for Structured Data Reasoning

Anonymous ACL submission

Abstract

Language models struggle with structured-data reasoning not because they cannot produce SQL, but because they misalign the question with the schema before SQL generation begins. These pre-SQL grounding errors, such as selecting the wrong target field, filter, join path, or schema slice, propagate into downstream query generation and execution. We introduce GRIPS (*Grounded Pre-SQL Signature*), a three-stage neurosymbolic framework that separates question-to-schema grounding from SQL generation. GRIPS first constructs a deterministic, LLM-free grounding record that resolves the question against the schema; this record then conditions SQL generation, and a deterministic executor returns the final answer. We evaluate GRIPS on TempTabQA-C and BIRD, covering both semi-structured temporal tables and fully relational databases. A failure audit of SQL Static shows that pre-computation grounding errors dominate on both benchmarks, accounting for 53.9% of failures on TempTabQA-C and 64.7% on BIRD. Across two model families, GRIPS-Adaptive improves over SQL Static in 18 of 20 comparisons, with an average gain of 11.0 REMS points, while reducing token usage by 47.8%. These results suggest that making grounding explicit before SQL generation improves accuracy, efficiency, and debuggability.

1 Introduction

Large language models (LLMs) have made reasoning over structured data look deceptively simple: provide a table or database schema, ask a question, and let the model answer. In practice, however, structured-data reasoning requires several decisions before any computation can be performed. The system must determine what the question refers to in the schema, identify the target field, resolve values and filters, recover relevant tables or schema slices, and decide which operation should be applied. If these pre-SQL decisions are wrong, errors carry over to downstream reasoning. This is why models that perform strongly in free-form reasoning remain brittle on structured-data QA across both semi-structured temporal tables and fully relational databases (Zhou et al., 2024; Shankarampeta et al., 2025; Gruber et al., 2025; Li et al., 2023; Osés Grijalba et al., 2025; Du et al., 2025).

Recent work has addressed parts of this bottleneck from three broad directions. The first and largest line of work targets SQL generation quality, focusing on schema linking and schema retrieval (Gao et al., 2024; Wang et al., 2025; Shi et al., 2025; Glass et al., 2025; Nahid et al., 2025; Eben et al., 2025; Sharma et al., 2025; Zhang et al., 2025). The second direction targets inference efficiency through cost-aware routing (Zhu et al., 2025; Yi and Chen, 2025; Shao et al., 2025). The third direction targets tabular reasoning for QA, improving prompting and execution strategies to recover final answers - though these methods have largely been evaluated on semi-structured tables and do not generalize readily to fully relational settings (Kulkarni et al., 2025; Deng et al., 2025a; Rajgaria et al., 2025). Crucially, these advances optimize one component at a time and none addresses the full pipeline from grounding through SQL generation to final answer recovery (Wu et al., 2024; Yang et al., 2025).

In this work, we investigate the pipeline end-to-end, starting with the pre-SQL condition, focusing on what inputs are available and where LLMs fail before generation even begins. Using the strongest recent neurosymbolic baseline (symbolic procedures externalize grounding and execution, while the neural model is reserved for constrained SQL synthesis), we evaluate on both structured (BIRD) and semi-structured (TempTabQA-C) datasets and find that the model's inability to correctly map the question onto the schema is the dominant error source.

Seeing this consistent pattern across both settings, we introduce GRIPS (*Grounded Pre-SQL Signature*), a robust and efficient framework that addresses this bottleneck with the core idea: ground first, then generate. Rather than asking the model to infer schema grounding and write SQL simultaneously, GRIPS makes the pre-SQL grounding state explicit and inspectable. Our evaluation is organized around three research questions:

RQ1: Where do current structured-data reasoning baselines fail, and do these failures occur before computation across both semi-structured temporal tables and relational databases?

RQ2: Does explicit pre-SQL grounding improve end-to-end answer accuracy across heterogeneous question types, counterfactual perturbations, table-size variation, and relational database queries?

RQ3: Can explicit grounding reduce irrelevant schema context and token usage without sacrificing answer accuracy?

Our main contributions are as follows:

1. We provide a cross-benchmark failure analysis of SQL-based structured-data reasoning, showing that pre-computation grounding failures dominate across both TempTabQA-C and BIRD.

2. We introduce GRIPS, a three-stage neurosymbolic framework that constructs an explicit, deterministic, LLM-free grounding record before SQL generation.

3. We show that separating grounding from generation improves both accuracy and efficiency. GRIPS-Adaptive improves over SQL Static in 18 of 20 comparisons, with an average gain of 11.0 REMS points, while reducing average token usage by 47.8%.

2 Related Work

SQL generation quality. The dominant line of work improves SQL generation by refining schema linking through semantic retrieval (Wang et al., 2025), pseudo-SQL abstraction (Shi et al., 2025), agent-driven schema discovery (Ziyang Wang, 2026), graph-based pathfinding (Safdarian et al., 2025), and token-level linking (Glass et al., 2025; Nahid et al., 2025; Li et al., 2023; Gao et al., 2024). All rely on an LLM at the grounding step and optimize for SQL correctness rather than final answer accuracy. To our knowledge, no prior method resolves grounding deterministically before generation while targeting end-to-end answer correctness across both structured and semi-structured settings, which is what GRIPS attempts.

Inference efficiency. Cost-aware methods reduce SQL generation overhead through complexity routing (Zhu et al., 2025), tree-guided decoding (Sharma et al., 2025), execution-bootstrapped selection (Zhang et al., 2025), and multi-agent dispatch (Shao et al., 2025; Yi and Chen, 2025), but typically require learned routers or dataset-specific supervision. GRIPS aims to pursue efficiency differently: by eliminating the LLM from the grounding step entirely, the symbolic stage naturally produces compact, question-specific prompts, reducing average token usage significantly without any routing supervision.

Tabular reasoning for QA. Work on semi-structured table QA improves answer recovery through symbolic intermediate representations (Kulkarni et al., 2025), temporal prompting (Deng et al., 2025b), and adaptive prompting (Rajgaria et al., 2025); robustness benchmarks confirm persistent brittleness under temporal and structural variation (Shankarampeta et al., 2025; Zhou et al., 2024; Gruber et al., 2025). These methods improve execution or prompting but do not address the upstream grounding step, and largely remain within semi-structured settings. We build directly on Kulkarni et al. (2025) and extend the grounding bottleneck analysis to fully relational databases, showing the same failure pattern holds in both settings.

3 Diagnosing Pre-SQL Failures in Structured-Data Reasoning

Before introducing GRIPS, we first ask where current structured-data reasoning pipelines fail. Our goal is not to re-rank existing methods, but to localize the dominant source of error. We focus this analysis on SQL Static, the strongest non-adaptive symbolic baseline from prior temporal tabular reasoning work (Kulkarni et al., 2025). SQL Static is a useful diagnostic target because it already constrains the model to generate executable SQL from schema context and fixed demonstrations. If such a pipeline fails, the error can often be traced to a specific stage: the model may have selected the wrong schema element, applied the wrong operation, or failed to produce executable SQL.

Example 1: TempTabQA-C, Hard

C: Darrell Pace
Personal Information: Birth Date = 1956-10-23
Olympic Games: Individual MedalGold | 1976 | Montreal; MedalGold | 1984 | Los Angeles
Pan American Games: Team MedalGold | 1979 | San Juan; MedalGold | 1983 | Caracas
Pan American Games: Individual MedalGold | 1983 | Caracas; MedalSilver | 1979 | San Juan
Q: How many medals did Darrell Pace win in his/her twenties?
A: 6

Example 2: BIRD, *california_schools*

E: "between 1/1/1980 and 12/31/1980" means the year = 1980.
Q: What is the average number of test takers from Fresno schools that opened between 1/1/1980 and 12/31/1980?
A: 137.89

Table 1: Examples from TempTabQA-C and BIRD.

3.1 Datasets

Our analysis spans two structurally different benchmarks, with examples shown in Table 1.

TempTabQA-C (Gupta et al., 2023) is a temporal table QA benchmark built from Olympic and athletics infoboxes. It provides seven controlled evaluation splits: three difficulty levels (*Easy*(732), *Medium*(507), *Hard*(719), the original unmodified split (*Orig.*(578)), a counterfactual variant with perturbed table content (*CFact.*)(699), and two table-size variants (*Small*(855), *Large*(538)). Together these splits test robustness across question difficulty, distributional shift, and structural variation.

BIRD (Li et al., 2023) is a text-to-SQL benchmark spanning eight real-world relational databases. For answer-level evaluation, we execute the human-authored gold SQL queries to derive the reference outputs. Each question has a gold SQL query, an evidence hint, and one of three hardness labels: *Simple*(92), *Moderate*(92), or *Challenging*(92). We use the Mini-Dev split throughout and refer to it as BIRD.

3.2 Analysis

We run SQL Static (Kulkarni et al., 2025) on development-style evaluation slices (20 per split) from each benchmark and retain only the failed predictions. Each instance is manually inspected using the full question, schema or table context, optional evidence, generated SQL, executed result, and gold answer. Failure is defined at the final-answer level: on TempTabQA-C via REMS; on BIRD when the executed SQL does not produce the gold answer. Each failure receives one primary

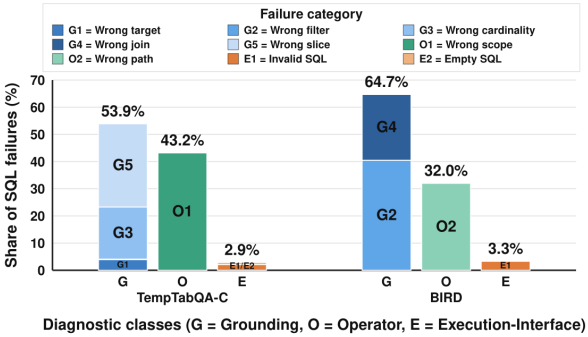


Figure 1: SQL Static failure breakdown by diagnostic class and subcategory. TempTabQA-C grounding errors concentrate on schema-slice and cardinality (G5/G3); BIRD on filter and join grounding (G2/G4).

Type	Example Question	Root Cause
Wrong target (G)	*Which was the most current medal win for Lin Sang?*	Returns a partial event string instead of the correct format; target field never resolved.
Wrong scope (O)	*How many times did Ilya Shymanovich win an International medal in the European Championships (SC) before 2019?*	Schema slice identified but temporal operator applied incorrectly; produces 0 instead of gold count 4.
Empty SQL (E)	*What is the shortest time span (in years) within which Abhinav Bindra won gold, silver, and bronze medals in the same format across any tournament?*	Empty SQL produced; pipeline fails before any executable query is formed.

Table 2: One representative failure per diagnostic class from SQL Static. Full subcategory examples in Appendix A.

diagnostic label structured around a single causal question: does the error arise *before* computation, *during* computation, or at the *execution interface*?

We group failures into three diagnostic classes. **Grounding (G)** errors occur before computation begins. In these cases, the model fails to correctly resolve the question against the schema. Examples include selecting the wrong target field, grounding a value to the wrong column, using the wrong filter, choosing the wrong schema slice, or following the wrong join path. **Operator (O)** errors occur after the relevant schema region is approximately identified, but the wrong computation is applied. Examples include incorrect temporal scope, aggregation, ranking, comparison, or path-based selection. **Execution-Interface (E)** errors occur when the pipeline fails to produce a usable query or answer. Examples include invalid SQL, empty SQL,

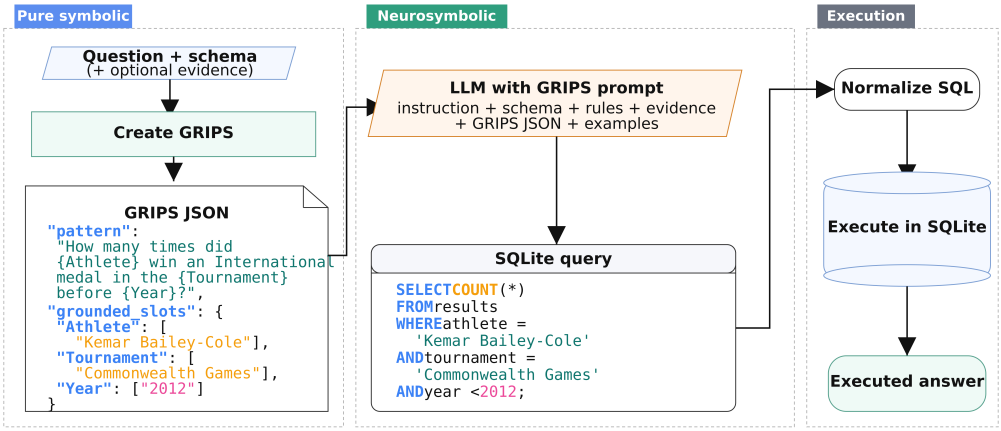


Figure 2: The GRIPS Framework

or malformed output.

This taxonomy intentionally abstracts away from dataset-specific labels. TempTabQA-C and BIRD differ in structure, but both require the system to ground the question, apply the correct operation, and produce an executable query. The three-way taxonomy therefore allows us to compare failure composition across benchmarks.

3.3 Results and Key Takeaway

As shown in Figure 1, Grounding is the largest failure class on both benchmarks: 53.9% of failures on TempTabQA-C and 64.7% on BIRD arise before any reasoning step, with Operator errors secondary (43.2%, 32.0%) and Execution-Interface failures small (2.9%, 3.3%). Table 2 gives one representative case per class; in all three, the SQL is syntactically valid yet wrong because the error occurs upstream when the question is resolved against the schema. This consistent pattern across both a semi-structured temporal benchmark and a fully relational one motivates GRIPS's core design: resolving question-to-schema grounding deterministically before any LLM call.

4 The GRIPS Framework

We introduce GRIPS, Grounded Pre-SQL Signature, a three-stage neurosymbolic framework for structured-data reasoning. Given a question, schema or table context, and optional evidence, GRIPS separates the task into: (1) deterministic grounding, (2) LLM-based SQL generation, and (3) deterministic execution. Figure 2 illustrates this pipeline.

The key idea is to separate what the question refers to from how the SQL should be written. Standard prompting pipelines ask the LLM to infer schema elements, values, filters, joins, and SQL in a single step. GRIPS instead resolves the question-to-schema mapping first and passes this grounding record to the LLM before SQL generation.

4.1 Stage I: Deterministic Grounding

Stage I constructs a question-specific grounding object as a JSON record. This object makes pre-SQL decisions explicit and contains: the original question, a slotized pattern, grounded slots, a structure profile, and a compressed grounding signature.

For TempTabQA-C, we ground the question against the temporal infobox schema using slot types such as `Athlete`, `Tournament`, `Format`, `MedalColor`, and `Year`; the structure profile records answer-shape cues such as `total_placeholder_count`, `distinct_slot_type_count`, `answer_shape`, and `requirement`. For BIRD, we lift the same idea to the relational level, grounding to `Table`, `Column`, `Value`, and `Date`; the structure profile records implicated tables, grounded column and value counts, grounding concentration, filter selectivity, target determinacy, and foreign-key path length when applicable.

The construction procedure is deterministic and uses no gold SQL, labeled training data, or additional LLM calls. It builds alias maps over schema names, values, dates, and evidence; grounds question spans against those aliases; normalizes and deduplicates matches; and replaces grounded spans with typed placeholders. The result is a compact representation of the question's grounded structure before SQL generation.

GRIPS is schema-aware rather than benchmark-identical: the same operations are reused across datasets, while the slot labels and schema inven-

GRIPS Prompt Template

```
# Instruction
Write one SQLite query for the question.
Output SQL only.

# Schema
[CREATE TABLE ...]

# Rules
- SQLite only; output one SQL statement
- Return only the requested field(s)
- [Dataset-specific rules, e.g.,
Medal types: MedalGold, MedalSilver,
MedalBronze
Event = Tournament + Format + Year
BIRD: use provided evidence hint]

# GRIPS JSON
{
"question": "...",
"pattern": "...",
"grounded_slots": {...},
"structure_profile": {...}
}

# Examples
Ex1: {Q1} → SQL: {SQL1}
Ex2: {Q2} → SQL: {SQL2}

# Question
{Question}
# Target
SQL:
```

Figure 3: The GRIPS prompt template.

tories vary by data representation. Appendix B provides full construction details.

The compressed signature records the structural pattern of the grounded question: which slot types appear, how many grounded entities or values are present, whether slots repeat, and what answer shape is implied. The signature serves two purposes. First, it conditions SQL generation by making the question-to-schema mapping visible to the LLM. Second, in GRIPS-Adaptive, it acts as a retrieval key for selecting structurally similar demonstrations from the development pool.

4.2 Stage II: GRIPS SQL Generation

In Stage II, the LLM generates SQL conditioned on the GRIPS grounding object. The prompt includes the task instruction, schema block, dataset-specific rules, optional evidence, GRIPS JSON, demonstrations, and target question. The model is instructed to output exactly one SQLite query.

We evaluate two variants.
(1) **GRIPS-Static** uses the GRIPS JSON with a fixed set of demonstrations, isolating the effect of explicit grounding.
(2) **GRIPS-Adaptive**, our main method, uses the same GRIPS JSON but dynamically selects demonstrations using the compressed signature, testing the added value of structurally matched examples.

Thus, GRIPS-Static measures the contribution of explicit pre-SQL grounding, while GRIPS-Adaptive measures the additional contribution of signature-based retrieval. The full prompt template and benchmark-specific instantiations are in Figure 3.

4.3 Stage III: Deterministic Execution

In Stage III, the generated SQL is normalized and executed in SQLite. GRIPS returns the executed result as the final answer rather than the raw model output. Keeping execution fixed across methods isolates the effect of grounding and demonstration selection. The design also improves debuggability: errors can be traced to grounding, retrieval, SQL generation, or execution, rather than being hidden inside a single end-to-end prompt.

GRIPS assumes that relevant schema elements and values can be recovered through aliasing, normalization, evidence matching, and graph traversal. This is reasonable for the benchmarks studied here, but may be harder for schemas with incomplete metadata or undocumented aliases, which is left for future work.

5 Experimental Setup

We experiment with two models: GPT-5-nano with default reasoning effort and temperature (OpenAI, 2025), and Llama-3.3-70B-Instruct (Grattafiori et al., 2024). All methods are evaluated on all 10 splits on both benchmarks, 140 experiments in total.

5.1 Baselines

We compare against seven baselines spanning direct-answer and SQL-based approaches, following the evaluation protocol of Kulkarni et al. (2025). The first six are direct-answer baselines, whereas the last one is a neurosymbolic method.

- **Direct Static**: fixed few-shot, free-text answer from raw context.
- **CLEAR** (Kulkarni et al., 2025): row extraction, question decomposition, answer synthesis.
- **CoT** (Wei et al., 2022; Kojima et al., 2022): explicit reasoning steps before answering.
- **PlanSolve** (Wang et al., 2023): reasoning plan followed by step-by-step calculation.

Model	Method	**TempTabQA-C** (REMS ↑)							**BIRD** (REMS ↑)		
		Easy	Medium	Hard	Orig.	CFact.	Small	Large	Simple	Moderate	Chall.
GPT-5-nano	Direct Static	91.47	63.72	56.44	70.77	59.03	80.10	60.90	7.61	3.26	3.26
	CLEAR	71.58	57.04	38.13	54.29	45.47	56.34	46.80	0.00	1.09	0.00
	CoT	76.73	61.74	46.81	59.44	50.25	66.77	50.99	1.63	2.17	1.09
	PlanSolve	78.46	61.11	45.68	61.04	51.15	68.85	53.18	0.00	2.17	1.09
	PoT	5.12	2.38	3.45	4.76	2.39	6.20	1.49	2.17	2.17	1.09
	Faithful-CoT	36.37	25.97	23.11	27.37	20.28	37.20	15.43	2.17	1.74	0.00
	SQL Static	77.94	63.31	60.71	64.71	60.20	75.87	66.81	**64.93**	43.13	30.92
Ours	GRIPS-Static	88.87	73.37	**62.68**	76.73	64.66	77.39	78.72	60.74	39.76	33.11
	GRIPS-Adaptive	**93.24**	**80.28**	58.48	**80.10**	**76.82**	**89.55**	**85.22**	61.67	**46.10**	**37.45**
Llama-70B	Direct Static	83.46	65.01	44.20	62.78	45.82	77.56	55.13	6.34	5.01	7.09
	CLEAR	72.79	61.33	42.33	61.70	48.11	68.87	46.76	0.00	2.17	3.26
	CoT	90.86	72.16	52.79	75.54	54.38	83.27	59.12	0.00	2.17	1.09
	PlanSolve	89.79	72.64	50.76	72.72	58.20	81.08	59.61	1.09	3.26	1.09
	PoT	14.75	0.25	6.82	6.23	6.22	6.78	6.88	3.82	4.35	1.63
	Faithful-CoT	14.75	0.00	6.54	7.44	6.51	8.30	7.43	3.26	7.61	1.82
	SQL Static	76.98	65.78	50.84	64.19	59.73	70.39	65.38	61.52	44.34	32.91
Ours	GRIPS-Static	93.38	61.01	**61.72**	74.49	73.10	78.16	65.35	59.05	45.42	35.20
	GRIPS-Adaptive	**94.61**	**77.02**	56.46	**78.05**	**76.86**	**83.49**	**78.44**	**63.15**	**62.51**	**41.72**

Table 3: REMS ↑ across TempTabQA-C and BIRD. EMS results follow the same pattern, shown in Appendix C.2.

- **PoT** (Chen et al., 2022): Python program over table contents to compute the answer.
- **Faithful-CoT** (Lyu et al., 2023): explicit decomposition before program generation.
- **SQL Static** (Kulkarni et al., 2025): SQL from raw schema with six fixed in-context examples; our direct point of comparison.

5.2 Evaluation Metrics

Following prior temporal tabular reasoning work, we report three metrics. (i) **EMS** (Exact Match Score) checks whether the predicted answer is character-for-character identical to the gold answer. (ii) **REMS** (Relaxed Exact Match Score) applies surface normalization before comparison, allowing minor formatting differences to count as correct. (iii) **Avg. Token Count** measures the average tokens consumed per query across input and output, serving as a direct proxy for inference cost.

6 Results and Discussion

We evaluate GRIPS against seven baselines across TempTabQA-C and BIRD using two model families, GPT-5-nano and Llama-70B. Table 3 reports answer accuracy using REMS, while Table 6 reports average token usage.

Across both datasets, the main pattern is consistent: GRIPS improves structured-data reasoning by making pre-SQL grounding explicit. GRIPS-Adaptive is the strongest overall method, improving over SQL Static in 18 of 20 comparisons with an average gain of 11.0 REMS points. At the same time, it reduces average token usage by 47.8% relative to SQL Static.

6.1 RQ1: Where Do Baselines Fall Short?

As shown in Table 3, direct prompting methods such as CoT, PlanSolve, PoT, Faithful-CoT and Direct Static perform unevenly across difficulty splits. On TempTabQA-C, for instance, CoT drops from 76.73 to 46.81 (GPT-4o-mini) and from 90.86 to 52.79 (Llama-70B) between Easy and Hard, a pattern that is consistent across methods. This suggests that neither step-by-step reasoning nor direct table querying reliably handles harder compositional grounding problems.

SQL Static performs better than many prompting-only methods, because it constrains the model to produce executable SQL, however, Table 4 shows that it still fails in systematic ways. In the TempTabQA-C example, SQL Static incorrectly splits "European Championships (SC)" into separate fields, treating part of the entity as a format constraint, producing the wrong answer. GRIPS instead preserves the phrase as the correct entity and recovers the correct count. In the BIRD example, SQL Static queries 'UserDisplayName' from the wrong table, while GRIPS identifies the correct join path and returns the correct answer.

The error categories in Figure 4 further show that SQL Static failures are often grounding failures rather than purely SQL execution failures. On TempTabQA-C, GRIPS rescues 24% of SQL

Benchmark	Question	SQL Static failure	GRIPS correction
TempTabQA-C	*How many times did Ilya Shymanovich win an International medal in the European Championships (SC) before 2019?*	Splits into `t.name = 'European Championships'` and `f.name = 'SC'`; incorrect format constraint; returns 0.	Preserves as `t.name = 'European Championships (SC)'`; removes format filter; recovers count 4.
BIRD	*Name the user that commented 'thank you user93!'*	Queries `UserDisplayName` from `comments`; column does not exist— wrong target.	Joins via `C.UserId = U.Id`; selects `U.DisplayName`; returns *Preets*.

Table 4: Representative failure cases showing what Stage I resolves before the LLM writes SQL. Red = SQL Static Failure; green = GRIPS correction. Full queries in Appendix C.4.

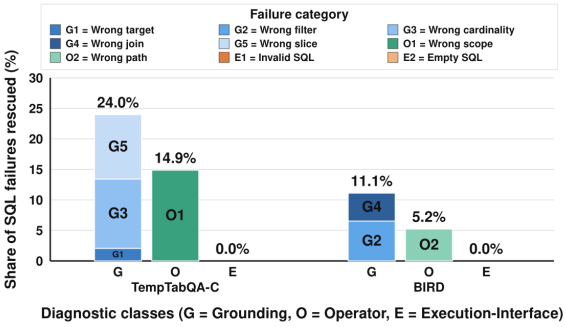


Figure 4: Share of SQL Static failures rescued by GRIPS, by diagnostic class and subcategory.

Static failures from grounding-related categories and about 15% from operator-related categories. On BIRD, it rescues 11% from grounding-related categories and about 5% from operator/path-related categories. This means the baseline often fails because it selects the wrong target, filter, cardinality, join, path, or scope. This suggests that the baseline methods are not just weak at final answer generation, but they often begin from the wrong interpretation of the question.

6.2 RQ2: Does Explicit Grounding Improve Accuracy?

GRIPS-Adaptive achieves the best REMS in 17 of 20 groups and improves over SQL Static, the strongest prior neurosymbolic baseline, in 18 of 20 groups, with an average gain of 11.0 REMS points. It is particularly effective when grounding ambiguity is high: large tables, counterfactual perturbations, and relational queries requiring non-trivial joins. Table 5 decomposes the gains: grounding alone (SQL Static → GRIPS-Static) contributes +5.1 REMS, and adding adaptive retrieval (GRIPS-Static → GRIPS-Adaptive) contributes another +5.9, confirming that both components are necessary.

The broader takeaway is that instead of asking the model to identify evidence, resolve constraints, and write SQL all at once, GRIPS separates these steps, reducing error propagation. Another benefit is that GRIPS improves debuggability because intermediate grounding decisions can be inspected.

Step	Component	Accuracy ↑	Efficiency ↓	ΔREMS
SS → GS	Grounding	15/20	20/20	+5.1
GS → GA	Retrieval	18/20	17/20	+5.9

Table 5: Ablation study of GRIPS component: SS = SQL Static; GS = GRIPS-Static; GA = GRIPS-Adaptive

Robustness across splits. On TempTabQA-C, GRIPS-Adaptive is strongest across all three axes — difficulty, counterfactual perturbation, and table size. The largest gains over SQL Static appear on large-table (+18.4 REMS on average) and counterfactual splits (+16.6 REMS), consistent with grounding being most valuable when context is broader or noisier. One exception: GRIPS-Static outperforms GRIPS-Adaptive on the Hard split for both models. We attribute this to retrieval sparsity — Hard questions are structurally rare in the development pool, so signature-matched examples are less reliable; fixed demonstrations prove more stable under this condition.

On BIRD, GRIPS-Adaptive is best on Moderate and Challenging for both models, improving over SQL Static by up to 18.2 REMS points. The one exception is BIRD Simple under GPT-5-nano, where SQL Static is marginally stronger: when a query can be answered from a near-direct schema match, full-schema exposure is already sufficient and GRIPS has little room to help. Its advantage appears once queries require more selective grounding — exactly where it improves on harder splits.

Model effects. GPT-5-nano and Llama-70B have different strengths. On TempTabQA-C, GPT-5-nano often has stronger Direct Static and GRIPS-Adaptive scores. Thus the method is not simply model-dependent in one direction. GRIPS transfers across both, but model strengths vary by dataset and split. Llama-70B is stronger in some CoT /

Model	Method	TempTabQA-C avg. tokens ↓							BIRD avg. tokens ↓		
		Easy	Medium	Hard	Orig.	CFact.	Small	Large	Simple	Moderate	Chall.
GPT-5-nano	Direct Static	2799	3098	3838	3373	4559	2539	4614	2885	2919	2875
	CLEAR	4824	5418	6811	5794	7066	4599	7423	3459	3492	3454
	CoT	2459	2878	3957	3333	4606	2327	4722	2683	2709	2661
	PlanSolve	2410	2812	3908	3237	4444	2238	4661	2589	2602	2551
	PoT	3921	4462	5924	4956	6176	3464	6674	2882	3160	3019
	Faithful-CoT	3763	4485	5725	4775	6114	3311	6717	2733	2893	2761
	SQL Static	3937	4329	5122	4719	4719	4334	4505	2227	2833	3220
Ours	GRIPS-Static	2299	2664	3338	3212	3272	3010	2978	1475	1660	1872
	GRIPS-Adaptive	1513	1989	2221	2110	2224	1895	2049	2012	2471	3077
Llama-70B	Direct Static	1597	1595	1667	1576	1763	1398	2004	2532	2559	2657
	CLEAR	2272	2370	2544	2408	2669	2107	2916	3239	3276	3220
	CoT	1423	1501	1680	1537	1838	1265	2065	2610	2574	2536
	PlanSolve	1336	1402	1612	1458	1729	1192	1964	2438	2506	2432
	PoT	1563	1647	1863	1686	2102	1349	2399	2576	2628	2596
	Faithful-CoT	1651	1762	1980	1805	2176	1445	2465	2594	2702	2653
	SQL Static	2918	3017	3080	3003	3017	2974	2977	1193	1241	1256
Ours	GRIPS-Static	1293	1131	2047	2525	1824	1504	1953	1170	1219	1238
	GRIPS-Adaptive	634	803	996	814	823	775	801	1039	1133	1203

Table 6: Average total token count per query across all splits (↓ lower is better).

PlanSolve settings. The BIRD gains from GRIPS-Adaptive over GRIPS-Static are larger for Llama.

6.3 RQ3: Does Grounding Reduce Token Usage?

Avg. Token Count measures provider-reported total tokens (input prompt and completion) for the final answer-generation run. It counts only content inserted into the prompt, such as schema, evidence, GRIPS JSON, and demonstrations, and excludes local retrieval pools, database files, signature construction, example scoring, and SQL execution. Because GPT-5-nano and Llama-70B use different tokenizers and provider accounting, token counts should be compared within model/provider rather than across them.

As Table 6 shows, GRIPS-Adaptive substantially reduces the token usage across both datasets, in 17 of 20 groups, with an average reduction of 47.8% relative to SQL Static. This efficiency gain comes from narrowing the context before final generation. Rather than passing large amounts of schema or evidence into the model, GRIPS-Adaptive retrieves the most relevant information after grounding the question. The system therefore spends fewer tokens on irrelevant context and avoids long reasoning traces that do not improve correctness.

Overall, the results suggest that better decomposition can improve both performance and cost, as many baselines use more tokens without consistently matching GRIPS-Adaptive's accuracy.

Scaling across difficulty and datasets. GRIPS-Adaptive reduces token counts by 55–78% versus SQL Static across Easy/Medium/Hard splits on TempTabQA-C for both model families, as it queries only a compact grounded schema slice rather than the full schema. On BIRD — a stricter setting with larger relational schemas — Llama-70B shows that GRIPS-Adaptive is simultaneously the cheapest and most accurate method, reducing tokens from 1,193–1,256 (SQL Static) to 1,039–1,203 while improving REMS across all three difficulty levels. Under GPT-5-nano on BIRD, GRIPS-Static is cheaper than GRIPS-Adaptive (e.g., 1,475 vs. 2,012 on Simple), suggesting adaptive selection over-expands on large schemas for weaker models, yet GRIPS-Adaptive still undercuts SQL Static on all splits and improves accuracy on Moderate and Challenging.

7 Conclusion and Future Work

We introduced GRIPS, a neurosymbolic framework that makes pre-SQL grounding explicit before generation. GRIPS-Adaptive improves over SQL Static in 18 of 20 comparisons (+11.0 REMS, −47.8% tokens), showing that explicit grounding improves accuracy and efficiency simultaneously. Future work will extend to noisier schemas and investigate transfer to knowledge graph QA, where the same upstream bottleneck arises before SPARQL generation.

Limitations

GRIPS's symbolic grounding relies on heuristic alias maps and schema normalization rules that may not transfer cleanly to noisier schemas or paraphrastic queries. At the same time, the procedure is schema-aware rather than benchmark-specialized: the same operations—alias mapping, span grounding, normalization, and optional graph traversal—are reused across settings, while only the schema inventory changes. Our failure taxonomy involves manual interpretation, and our evaluation covers two benchmark families — TempTabQA-C and BIRD — which do not exhaust the broader space of real-world tabular settings. BIRD results should be interpreted with caution, as prior work has documented noisy questions and incorrect gold SQL in some domains (Wretblad et al., 2024), which can affect benchmark reliability.

Ethics Statement

This work studies question answering over benchmarked structured datasets and relational databases; it does not involve human-subject experimentation or user deployment. The main risk is over-reliance on generated answers or SQL in settings where correctness matters. Although symbolic grounding improves interpretability and reduces some error modes, incorrect grounding can still produce plausible but false outputs. We therefore do not recommend use of the method without task-level verification in high-stakes domains such as medicine, law, finance, or public administration. Users should also respect the licensing and access conditions of the underlying datasets and database resources.

References

Wenhu Chen, Xueguang Ma, Xueguang Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Irwin Deng, Kushagra Dixit, Dan Roth, and Vivek Gupta. 2025a. Enhancing temporal understanding in LLMs for semi-structured tables. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4951–4970, Albuquerque, New Mexico. Association for Computational Linguistics.

Irwin Deng, Kushagra Dixit, Dan Roth, and Vivek Gupta. 2025b. Enhancing temporal understanding in llms for semi-structured tables. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4936–4955.

Yufeng Du, Minyang Tian, Srikanth Ronanki, Subendhu Rongali, Sravan Babu Bodapati, Aram Galstyan, Azton Wells, Roy Schwartz, Eliu A Huerta, and Hao Peng. 2025. Context length alone hurts LLM performance despite perfect retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 23281–23298, Suzhou, China. Association for Computational Linguistics.

Jeffrey Eben, Aitzaz Ahmad, and Stephen Lau. 2025. Rasl: Retrieval augmented schema linking for massive database text-to-sql. *Preprint*, arXiv:2507.23104.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-SQL empowered by large language models: A benchmark evaluation. *Proceedings of the VLDB Endowment*, 17(5):1132–1145.

Michael Glass, Mustafa Eyceoz, Dharmashankar Subramanian, Gaetano Rossiello, Long Vu, and Alfio Gliozzo. 2025. Extractive schema linking for text-to-sql. *Preprint*, arXiv:2501.17174.

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

Raphael Gruber, Abdelrahman Abdallah, Michael Färber, and Adam Jatowt. 2025. Complextempqa: A 100m dataset for complex temporal question answering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 9100–9112.

Vivek Gupta, Pranshu Kandoi, Mahek Vora, Shuo Zhang, Yujie He, Ridho Reinanda, and Vivek Srikumar. 2023. Temptabqa: Temporal question answering for semi-structured tables. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2431–2453.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Atharv Kulkarni, Kushagra Dixit, Vivek Srikumar, Dan Roth, and Vivek Gupta. 2025. Llm-symbolic integration for robust temporal tabular reasoning. *arXiv preprint arXiv:2506.05746*.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chen-Chuan Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLM already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. In *Advances in Neural Information Processing Systems*, volume 36.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *The 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (IJCNLP-AACL 2023)*.

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

OpenAI. 2025. Gpt-5 technical report. Technical report, OpenAI.

Jorge Osés Grijalba, L. Alfonso Ureña-López, Eugenio Martínez Cámara, and Jose Camacho-Collados. 2025. SemEval-2025 task 8: Question answering over tabular data. In *Proceedings of the 19th International Workshop on Semantic Evaluation (SemEval-2025)*, pages 2512–2522, Vienna, Austria. Association for Computational Linguistics.

Abhishek Rajgaria, Kushagra Dixit, Mayank Vyas, Harshavardhan Kalalbandi, Dan Roth, and Vivek Gupta. 2025. No universal prompt: Unifying reasoning through adaptive prompting for temporal table reasoning. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 2800–2821, Mumbai, India. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics.

AmirHossein Safdarian, Milad Mohammadi, Ehsan Jahanbakhsh, Mona Shahamat Naderi, and Heshaam Faili. 2025. SchemaGraphSQL: Efficient schema linking with pathfinding graph algorithms for text-to-SQL on large-scale databases. *arXiv preprint arXiv:2505.18363*.

Abhilash Shankarampeta, Harsh Mahajan, Tushar Kataria, Dan Roth, and Vivek Gupta. 2025. TRANSIENTTABLES: Evaluating LLMs' reasoning on temporally evolving semi-structured tables. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6526–6544, Albuquerque, New Mexico. Association for Computational Linguistics.

Zhihui Shao, Shubin Cai, Rongsheng Lin, and Zhong Ming. 2025. Enhancing text-to-sql with question classification and multi-agent collaboration. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4340–4349.

Chetan Sharma, Ramasuri Narayanam, Soumyabrata Pal, Kalidas Yeturu, Shiv Kumar Saini, and Koyel Mukherjee. 2025. TTD-SQL: Tree-guided token decoding for efficient and schema-aware sql generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1287–1298, Suzhou, China. Association for Computational Linguistics.

Jie Shi, Bo Xu, Jiaqing Liang, Yanghua Xiao, Jia Chen, Chenhao Xie, Peng Wang, and Wei Wang. 2025. Gensql: Efficient text-to-sql by bridging natural language question and database schema with pseudo-schema. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3794–3807.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*.

Yihan Wang, Peiyu Liu, and Xin Yang. 2025. 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*, pages 977–991.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Niklas Wretblad, Fredrik Riseby, Rahul Biswas, Amin Ahmadi, and Oskar Holmström. 2024. Understanding the effects of noise in text-to-sql: An examination of the bird-bench benchmark. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 356–369.

Zijun Wu, Bingyuan Liu, Ran Yan, Lei Chen, and Thomas Delteil. 2024. Reducing distraction in long-context language models by focused learning. *Preprint*, arXiv:2411.05928.

Minglai Yang, Ethan Huang, Liang Zhang, Mihai Surdeanu, William Yang Wang, and Liangming Pan. 2025. How is LLM reasoning distracted by irrelevant context? an analysis using a controlled benchmark. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13340–13358, Suzhou, China. Association for Computational Linguistics.

Jiawen Yi and Guo Chen. 2025. SKT-SQL: Multi-decoupling and skeleton prompt framework for text-to-SQL on small-scale LMs. *Journal of Intelligent Information Systems*, 63:1779–1805.

Jipeng Zhang, Haolin Yang, Kehao Miao, Ruiyuan Zhang, Renjie Pi, Jiahui Gao, and Xiaofang Zhou. 2025. ExeSQL: Self-taught text-to-SQL models with execution-driven bootstrapping for sql dialects. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 24305–24326, Suzhou, China. Association for Computational Linguistics.

Wei Zhou, Mohsen Mesgar, Heike Adel, and Annemarie Friedrich. 2024. Freb-tqa: A fine-grained robustness evaluation benchmark for table question answering. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2479–2497.

Yizhang Zhu, Runzhi Jiang, Boyan Li, Nan Tang, and Yuyu Luo. 2025. EllieSQL: Cost-efficient text-to-SQL with complexity-aware routing. In *Conference on Language Modeling (COLM)*.

Zhenbiao Cao Xiaojin Zhang Zhongyu Wei Pei Fu Zhenbo Luo Wei Chen Xiang Bai Ziyang Wang, Yuanlei Zheng. 2026. AutoLink: Autonomous schema exploration and expansion for scalable schema linking in text-to-SQL at scale. In *Proceedings of the 40th AAAI Conference on Artificial Intelligence*. ArXiv:2511.17190.

A Failure Analysis

We audit SQL Static failures to identify where structured-data reasoning breaks down before introducing GRIPS. The protocol, taxonomy, and empirical composition are described below.

A.1 Failure Analysis Protocol

We run SQL Static on development-style evaluation slices from each benchmark and manually inspect each failed instance using the full question, schema or table context, optional evidence, generated SQL, executed result, and gold answer. Each failure receives one primary diagnostic label structured around a single causal question: does the error arise before computation, during computation, or at the execution interface?

A.2 Nine-Way Failure Taxonomy

Table 7 lists one representative instance per subcategory, covering all nine non-zero failure subtypes observed across TempTabQA-C and BIRD. Each example is drawn directly from the audited SQL Static runs.

A.3 Empirical Failure Composition

Figure 5 shows the class distribution after re-auditing SQL Static errors across all splits. Grounding dominates on both benchmarks, confirming the pattern reported in Section 3.3.

B GRIPS Construction Details

This section describes how the GRIPS JSON is constructed deterministically for each benchmark, illustrated in Figure 6, without LLM calls, gold SQL, or manual annotation.

B.1 Algorithm

Algorithm 1 summarizes the routine: we build alias maps over schema names, values, dates, and evidence; ground question spans against these aliases; merge evidence-level matches when available; normalize duplicates; and replace grounded mentions with typed placeholders to produce the slotized pattern. We then derive the structure profile and serialize the full object into a compressed grounding signature. The slot labels differ because the schemas differ, but the construction operations are shared: alias expansion, span matching, normalization, deduplication, and structure profiling.

B.2 TempTabQA-C

For TempTabQA-C, the symbolic stage grounds the question against a fixed temporal infobox schema using regex- and cue-based rules over five slot types: `Athlete`, `Tournament`, `Format`, `MedalColor`, and `Year`. Years are extracted through explicit year patterns; medal colors from a controlled inventory; formats through distance-format expressions; tournament spans through capitalized multi-word patterns; and athlete names through cue-based fallback matching. Overlapping candidates are deduplicated so that no span is assigned to conflicting slots. The question is then rewritten into a slotized pattern by replacing grounded spans with typed placeholders, and structural counts — total placeholder count, distinct slot-type count, repeated slots, and tuple-style requirement labels (*singlet*, *doublet*, *triplet*) — are derived deterministically.

B.3 BIRD

Thus, the procedure is not benchmark-specific in the sense of encoding task-specific reasoning modules; it is a schema-aware deterministic grounding pipeline instantiated over different schema inventories. For BIRD, the same principle is applied at the relational-schema level, grounding the question to `Table`, `Column`, `Value`, and `Date` objects. Schema metadata — table names, column names, and foreign-key structure — is loaded and alias maps are built over normalized forms to support spacing, inflectional, and orthographic variation. Value grounding handles quoted strings, numeric spans, categorical codes, and date expressions. The structural profile records implicated tables, grounded column and value counts, target determinacy, grounding concentration, filter selec-

Code	Failure type	Example	Dataset	Observation / Root cause
G1	Wrong target	*Which was the most current medal win for Lin Sang?*	TempTabQA-C	Returns a partial event description instead of the correct answer form; target field never resolved.
G2	Wrong filter	*State the names and full communication address of high schools in Monterey which has more than 800 free or reduced price meals for ages 15–17?*	BIRD	Grounds `Monterey` to `City` instead of `County`; query filtered on wrong attribute.
G3	Wrong cardinality	*In which year(s) did Kylie Masse win medals in the World Championships (LC)?*	TempTabQA-C	Question is set-valued but system emits no final answer despite a valid multi-year `SELECT DISTINCT` query.
G4	Wrong join	*What is the telephone number for the school with the lowest average score in reading in Fresno Unified?*	BIRD	Joins through `frpm` instead of the minimal gold join path; incorrect relational grounding.
G5	Wrong slice	*How many International medals did Abhinav Bindra win in the 2002?*	TempTabQA-C	Counts over an incorrectly grounded evidence slice; returns 0 instead of gold answer 2.
O1	Wrong temporal scope	*How many times did Ilya Shymanovich win an International medal in the European Championships (SC) before 2019?*	TempTabQA-C	Schema slice largely identified but temporal operator applied incorrectly; returns 0 instead of 4.
O2	Wrong path	*For the branch located in south Bohemia with the biggest number of inhabitants, what is the percentage of male clients?*	BIRD	Follows wrong structural path for target branch selection; incorrect maximum-based condition used.
E1	Invalid SQL	*Count the events where Natalia Valeeva won gold, then a lower medal, then gold again in consecutive iterations.*	TempTabQA-C	References non-existent column `m2.tournament_id`; execution fails.
E2	Empty SQL	*What is the shortest time span (in years) within which Abhinav Bindra won gold, silver, and bronze medals in the same format across any tournament?*	TempTabQA-C	System produces `empty_sql`; pipeline fails before any executable query is formed.

Table 7: Representative failures from audited SQL Static runs, aligned with the nine non-zero subcategories in Figure 1.

Algorithm 1 GRIPS JSON Construction

Require: Question q; schema or table context $\mathcal{S}$; optional evidence e
Ensure: GRIPS JSON z
$\mathcal{A} \leftarrow \textsc{BuildAliasMaps}(\mathcal{S}, e)$
$M_q \leftarrow \textsc{GroundQuestion}(q, \mathcal{A})$
if e is available **then**
  $M_e \leftarrow \textsc{GroundEvidence}(e, \mathcal{A})$
else
  $M_e \leftarrow \emptyset$
end if
$M \leftarrow \textsc{DeduplicateNormalize}(M_q \cup M_e)$
$p \leftarrow \textsc{SlotPattern}(q, M_q)$
$\phi \leftarrow \textsc{StructureProfile}(q, M, \mathcal{S})$
$z \leftarrow \{\texttt{question} : q, \texttt{pattern} : p, \texttt{slots} : M, \texttt{profile} : \phi\}$
$z.\texttt{signature} \leftarrow \textsc{Compress}(z)$
return z

tivity, and shortest foreign-key path length, all derived through deterministic graph traversal.

B.4 What the Symbolic Layer Contributes

The GRIPS JSON is not an answer, a chain-of-thought, or an executable program. It externalizes the question-to-schema mapping that would otherwise remain implicit inside the prompt, directly targeting the dominant upstream failure identified in Section 3.3. Figure 6 shows one real signature from each benchmark.

C Additional Results

C.1 Full Baseline Results

Tables 8–11 report REMS and EMS for all baselines across all splits of TempTabQA-C and BIRD. EMS follows the same pattern as REMS throughout; both are included here for completeness.

C.2 Results with All Baselines

We report the full appendix results for all baselines, including REMS/EMS and token count (TC), in addition to the focused main-paper comparisons.

C.3 Overall Accuracy–Efficiency

Table 12 reports weighted overall REMS, average token count, and REMS per 1k tokens across both benchmarks. REMS/1k Tok jointly captures accuracy and efficiency on a single frontier metric.

C.4 Extended Rescue Examples

This section extends the two main-paper cases with two additional failure–correction pairs, one per

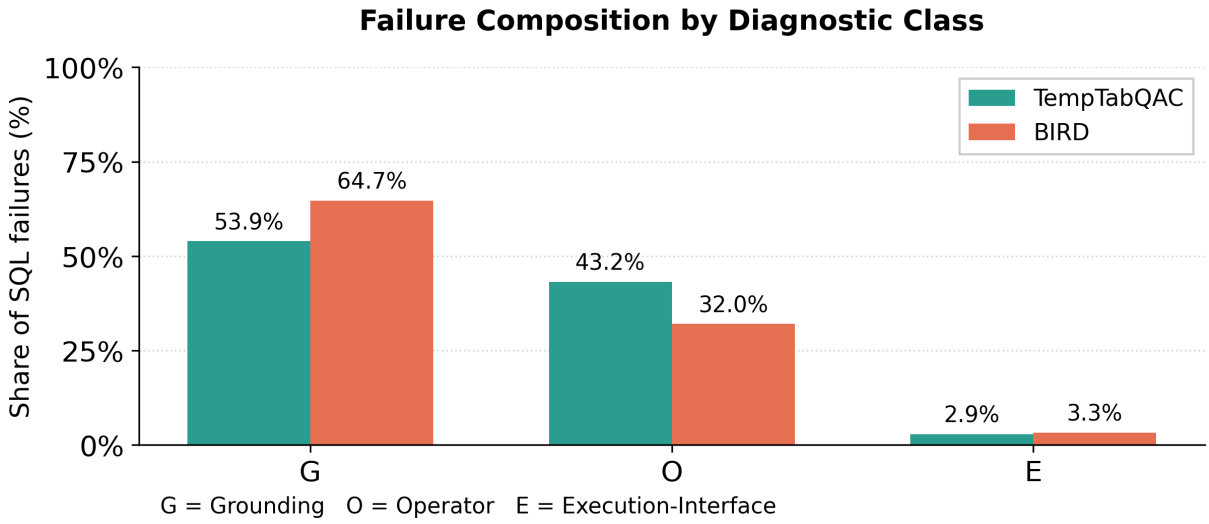


Figure 5: Failure composition by diagnostic class after re-auditing SQL Static errors on TempTabQA-C (Easy/Medium/Hard) and BIRD (Simple/Moderate/Challenging).

benchmark, covering a different failure subtype in each.

TempTabQA-C — Full SQL. *Question: How many times did Ilya Shymanovich win an International medal in the European Championships (SC) before 2019?*

SQL Static (returns 0).

```
SELECT COUNT(*) AS medal_count
FROM Medal m
JOIN Format f ON m.format_id = f.format_id
JOIN Tournament t ON f.tournament_id = t.
    tournament_id
JOIN Athlete a ON t.athlete_id = a.athlete_id
WHERE a.name = 'Ilya Shymanovich'
  AND t.name = 'European Championships'
  AND f.name = 'SC'
  AND m.year < 2019
  AND m.type IN ('MedalGold', 'MedalSilver', '
    MedalBronze');
```

GRIPS Adaptive (returns 4).

```
SELECT COUNT(m.medal_id) AS total_medals
FROM Medal m
JOIN Format f ON m.format_id = f.format_id
JOIN Tournament t ON f.tournament_id = t.
    tournament_id
JOIN Athlete a ON t.athlete_id = a.athlete_id
WHERE a.name = 'Ilya Shymanovich'
  AND t.name = 'European Championships (SC)'
  AND m.year < 2019;
```

BIRD — Full SQL. *Question: Name the user that commented 'thank you user93!'*

SQL Static (no answer).

```
SELECT UserDisplayName
FROM comments
WHERE Text = 'thank you user93!'
LIMIT 1;
```

GRIPS Static and GRIPS Adaptive (returns *Preets*).

```
SELECT U.DisplayName
FROM comments C
JOIN users U ON C.UserId = U.Id
WHERE C.Text = 'thank you user93!'
LIMIT 1;
```

D AI Disclosure

AI-based assistants were used for limited writing, coding, and figure-preparation support during the development of this project. All experiments, annotations, technical claims, and final manuscript decisions were reviewed and validated by the authors.

TempTabQA-C GRIP SIGNATURE

```
question:
  How many times did Kemar Bailey-Cole win
an
  International medal in the Commonwealth
Games
  before 2012?
pattern:
  How many times did {Athlete} win an
  International medal in the {Tournament}
  before {Year}?
grounded_slots:
  Athlete = Kemar Bailey-Cole
  Tournament = Commonwealth Games
  Year = 2012
structure_profile:
  total_placeholder_count = 3
  distinct_slot_type_count = 3
  answer_shape = count
  requirement = triplet
```

BIRD GRIP SIGNATURE

```
question:
  What is the ratio of customers who pay
in EUR
  against customers who pay in CZK?
pattern:
  What is the ratio of customers who pay
in
  {Value} against customers who pay in
{Value}?
grounded_slots:
  Table = customers
  Column = Currency
  Value = EUR; Value = CZK
structure_profile:
  implicated_tables = [customers]
  grounded_column_count = 1
  grounded_value_count = 2
  grounding_concentration = high
  filter_selectivity = narrow
  target_determinacy = inferred_clear
```

Figure 6: Examples of real GRIP signatures from TempTabQA-C and BIRD. Each signature records the original question, its slotized pattern, grounded schema elements, and a compact structure profile before direct answering or SQL generation.

Model	Method	Easy		Medium		Hard	
		REMS/EMS	TC	REMS/EMS	TC	REMS/EMS	TC
GPT-5-nano	Direct Static	91.47 / 90.57	2799	63.72 / 61.54	3098	56.44 / 55.22	3838
	CLEAR	71.58 / 70.77	4824	57.04 / 54.44	5418	38.13 / 37.27	6811
	CoT	76.73 / 75.68	2459	61.74 / 58.97	2878	46.81 / 45.48	3957
	PlanSolve	78.46 / 77.73	2410	61.11 / 58.78	2812	45.68 / 44.37	3908
	PoT	5.12 / 5.05	3921	2.38 / 2.17	4462	3.45 / 3.34	5924
	Faithful-CoT	36.37 / 35.66	3763	25.97 / 24.46	4485	23.11 / 22.39	5725
	SQL Static	77.94 / 77.73	3937	63.31 / 63.31	4329	60.71 / 60.64	5122
Ours	GRIPS-Static	88.87 / 87.57	2299	73.37 / 72.98	2664	**62.68 / 62.59**	3338
	GRIPS-Adaptive	**93.24 / 92.35**	1513	**80.28 / 80.28**	1989	58.48 / 58.28	2221
Llama-70B	Direct Static	83.46 / 82.79	1597	65.01 / 56.61	1595	44.20 / 42.56	1667
	CLEAR	72.79 / 72.13	2272	61.33 / 57.59	2370	42.33 / 41.72	2544
	CoT	90.86 / 89.89	1423	72.16 / 67.65	1501	52.79 / 51.60	1680
	PlanSolve	89.79 / 89.34	1336	72.64 / 68.05	14017	50.76 / 49.65	1612
	PoT	14.75 / 14.62	1563	0.25 / 0.20	1647	6.82 / 6.68	1863
	Faithful-CoT	14.75 / 14.21	1651	0.00 / 0.00	1762	6.54 / 5.98	1980
	SQL Static	76.98 / 76.78	2918	65.78 / 64.89	3017	50.84 / 49.79	3080
Ours	GRIPS-Static	93.38 / 93.17	1293	61.01 / 58.58	1131	**61.72 / 60.08**	2047
	GRIPS-Adaptive	**94.61 / 94.40**	634	**77.02 / 76.33**	803	56.46 / 54.94	996

Table 8: TempTabQA-C Easy, Medium, and Hard split accuracy and token usage with all baselines.

Model	Method	**Simple**		**Moderate**		**Challenging**	
		REMS/EMS	TC	REMS/EMS	TC	REMS/EMS	TC
GPT-5-nano	Direct Static	7.61 / 6.52	2885	3.26 / 3.26	2919	3.26 / 2.17	2875
	CLEAR	0.00 / 0.00	3459	1.09 / 1.09	3492	0.00 / 0.00	3454
	CoT	1.63 / 1.09	2683	2.17 / 2.17	2709	1.09 / 1.09	2661
	PlanSolve	0.00 / 0.00	2589	2.17 / 2.17	2602	1.09 / 1.09	2551
	PoT	2.17 / 1.09	2882	2.17 / 2.17	3160	1.09 / 1.09	3019
	Faithful-CoT	2.17 / 2.17	2733	1.74 / 1.09	2893	0.00 / 0.00	2761
	SQL Static	**64.93 / 58.70**	2227	43.13 / 34.78	2833	30.92 / 21.74	3220
Ours	GRIPS-Static	60.74 / 54.35	1475	39.76 / 33.70	1660	33.11 / 27.17	1872
	GRIPS-Adaptive	61.67 / 55.43	2012	**46.10 / 40.22**	2471	**37.45 / 26.09**	3077
Llama-70B	Direct Static	6.34 / 5.43	2532	5.01 / 4.35	2559	7.09 / 4.35	2657
	CLEAR	0.00 / 0.00	3239	2.17 / 2.17	3276	3.26 / 3.26	3220
	CoT	0.00 / 0.00	2610	2.17 / 2.17	2574	1.09 / 1.09	2536
	PlanSolve	1.09 / 0.00	2438	3.26 / 3.26	2506	1.09 / 1.09	2432
	PoT	3.82 / 3.26	2576	4.35 / 4.35	2628	1.63 / 0.00	2596
	Faithful-CoT	3.26 / 2.17	2594	7.61 / 7.61	2702	1.82 / 1.09	2653
	SQL Static	61.52 / 55.43	1193	44.34 / 34.78	1241	32.91 / 21.74	1256
Ours	GRIPS-Static	59.05 / 53.26	1170	45.42 / 33.70	1219	35.20 / 25.00	1238
	GRIPS-Adaptive	**63.15 / 55.43**	1039	**62.51 / 50.00**	1133	**41.72 / 30.43**	1203

Table 9: BIRD Simple, Moderate, and Challenging split accuracy and token usage with all baselines.

Model	Method	Original		Counterfactual	
		REMS/EMS	TC	REMS/EMS	TC
GPT-5-nano	Direct Static	70.77 / 69.55	3373	59.03 / 57.31	4559
	CLEAR	54.29 / 53.11	5794	45.47 / 44.28	7066
	CoT	59.44 / 57.61	3333	50.25 / 48.34	4606
	PlanSolve	61.04 / 59.34	3237	51.15 / 49.20	4444
	PoT	4.76 / 4.50	4956	2.39 / 2.32	6176
	Faithful-CoT	27.37 / 26.47	4775	20.28 / 19.25	6114
	SQL Static	64.71 / 64.71	4719	60.20 / 60.20	4719
Ours	GRIPS-Static	76.73 / 76.47	3212	64.66 / 63.95	3272
	GRIPS-Adaptive	**80.10 / 79.76**	2110	**76.82 / 75.82**	2224
Llama-70B	Direct Static	62.78 / 59.86	1576	45.82 / 41.82	1763
	CLEAR	61.70 / 59.86	2408	48.11 / 45.30	2669
	CoT	75.54 / 73.36	1537	54.38 / 51.81	1838
	PlanSolve	72.72 / 70.76	1458	58.20 / 55.72	1729
	PoT	6.23 / 6.06	1686	6.22 / 6.22	2102
	Faithful-CoT	7.44 / 6.75	1805	6.51 / 6.37	2176
	SQL Static	64.19 / 63.67	3003	59.73 / 59.08	3017
Ours	GRIPS-Static	74.49 / 72.66	2525	73.10 / 72.53	1824
	GRIPS-Adaptive	**78.05 / 76.99**	814	**76.86 / 75.39**	823

Table 10: TempTabQA-C Original vs. CounterFactual accuracy and token usage with all baselines.

Model	Method	Small		Large	
		REMS/EMS	TC	REMS/EMS	TC
GPT-5-nano	Direct Static	80.10 / 78.71	2539	60.90 / 58.74	4614
	CLEAR	56.34 / 55.44	4599	46.80 / 45.17	7423
	CoT	66.77 / 65.38	2327	50.99 / 49.26	4722
	PlanSolve	68.85 / 67.49	2238	53.18 / 51.49	4661
	PoT	6.20 / 6.08	3464	1.49 / 1.49	6674
	Faithful-CoT	37.20 / 36.02	3311	15.43 / 15.06	6717
	SQL Static	75.87 / 75.79	4334	66.81 / 66.54	4505
Ours	GRIPS-Static	77.39 / 75.79	3010	78.72 / 77.70	2978
	GRIPS-Adaptive	**89.55 / 88.54**	1895	**85.22 / 83.83**	2049
Llama-70B	Direct Static	77.56 / 74.39	1398	55.13 / 50.93	2004
	CLEAR	68.87 / 67.37	2107	46.76 / 44.98	29164
	CoT	83.27 / 81.52	1265	59.12 / 57.06	20654
	PlanSolve	81.08 / 79.88	11925	59.61 / 57.81	1964
	PoT	6.78 / 6.78	1349	6.88 / 6.32	2399
	Faithful-CoT	8.30 / 8.07	1445	7.43 / 6.88	2465
	SQL Static	70.39 / 69.79	2974	65.38 / 64.87	2977
Ours	GRIPS-Static	78.16 / 76.37	1504	65.35 / 63.75	1953
	GRIPS-Adaptive	**83.49 / 82.22**	775	**78.44 / 77.51**	801

Table 11: TempTabQA-C small-table vs. large-table accuracy and token usage with all baselines.

Dataset	Model	Method	REMS	Tokens	REMS/1k Tok
BIRD	GPT-5-nano	Direct Static	4.71	2893.3	1.63
		CLEAR	0.36	3468.4	0.10
		CoT	1.63	2684.2	0.61
		PlanSolve	1.09	2580.7	0.42
		PoT	1.81	3020.4	0.60
		Faithful-CoT	1.30	2795.8	0.47
		SQL Static	46.33	2760.2	17.99
	Ours	GRIPS-Static	44.54	1669.3	27.60
		GRIPS-Adaptive	48.41	2520.0	20.49
	Llama-70B	Direct Static	6.15	2582.8	2.38
		CLEAR	1.81	3244.8	0.56
		CoT	1.09	2573.6	0.42
		PlanSolve	1.81	2458.7	0.73
		PoT	3.27	2600.2	1.26
		Faithful-CoT	4.23	2649.7	1.59
		SQL Static	46.26	1230.0	37.83
	Ours	GRIPS-Static	46.56	1209.4	38.71
		GRIPS-Adaptive	55.79	1125.0	50.21
TempTabQA-C	GPT-5-nano	Direct Static	69.87	3492.0	21.66
		CLEAR	53.00	5916.2	9.51
		CoT	59.46	3407.7	19.57
		PlanSolve	60.47	3325.4	20.46
		PoT	3.88	4995.3	0.89
		Faithful-CoT	27.35	4886.3	6.35
		SQL Static	67.67	4519.0	15.18
	Ours	GRIPS-Static	74.63	2971.8	25.84
		GRIPS-Adaptive	80.68	1990.0	41.78
	Llama-70B	Direct Static	64.27	1622.3	39.83
		CLEAR	59.69	2406.7	25.43
		CoT	73.23	1550.3	49.33
		PlanSolve	71.90	1467.5	51.27
		PoT	7.36	1714.9	4.48
		Faithful-CoT	7.86	1815.0	4.54
		SQL Static	65.25	2992.3	21.81
	Ours	GRIPS-Static	73.42	1742.3	45.24
		GRIPS-Adaptive	79.56	801.3	99.43

Table 12: Weighted overall accuracy and efficiency across TempTabQA-C and BIRD. REMS/1k Tok measures answer accuracy per thousand tokens consumed, combining accuracy and efficiency into a single frontier metric.