Schema Guidance as Managed Context for Enhanced Agentic SQL Generation in Clinical Databases

Anonymous Author(s)
Affiliation
Address
email

Abstract

Large language models have substantially improved text-to-SQL across tasks that utilize relational database systems, but applications to real-world clinical queries remain unreliable. We argue that the bottleneck is not simply a lack of context, but a failure to *manage* context. We introduce *schema guidance* as a first-class representation of managed context for agentic SQL generation: a structured, updateable object that encodes task-relevant schema knowledge such as key filters, common joins, anti-patterns, example query patterns, and critical notes. Unlike standard prompting approaches, we treat context as a structured object that explicitly constrains the model's search space. We formalize schema guidance as both (i) a compressed representation that motivates the approach and (ii) a search-space reduction mechanism that accounts for improved decoding performance. We construct guidance via a fixed pipeline that is shared across databases. We show empirically that schema guidance improves construction of SQL queries in non-clinical SPIDER datasets, open EHRSQL MIMIC-III and eICU datasets, and a closed institutional dataset with representative complex clinical queries. These results suggest that learning the right context representation is a critical and underexplored axis for reliable agentic SQL generation in real-world clinical settings.

1 Introduction

Text-to-SQL has become one of the clearest demonstrations of how large language models (LLMs) can turn natural language into executable programs. For electronic health records, which primarily use relational database management systems, recent LLM-based text-to-SQL systems have shown improvements using retrieval augmented generation [Ziletti and D'Ambrosi, 2024, 2025], ensemble LLM approaches [Gundabathula and Kolar, 2024], and seq2seq based methods Wang et al. [2020]. Other approaches have focused on modifying database structure entirely, either creating their own database structures Katz et al. [2024]or adopting graph databases [Thio et al., 2026].

However, reliability remains brittle in real clinical databases, where query generation must cope with highly complex logic with deeply nested heterogeneous schemas, ambiguous or noisy database structures, and domain-specific business rules. The result is a familiar pattern: the model often identifies the general task correctly, but fails on the exact filter, wrong join path, spurious table, or clinically invalid value mapping.

The central problem may not merely be *insufficient context*; it is *poorly managed context*.[Liu et al., 2023, Modarressi et al., 2025] Most existing systems increase context quantity by retrieving more schema text, values, or exemplars. [Hong et al., 2024, Caferoğlu et al., 2025] Some improve context quality through retrieval augmentation.[Shen et al., 2023, Toteja et al., 2025] An important question is what *representation of context* an agent should maintain as a stable object over time. This question

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

is especially important in clinical SQL generation, where raw schema dumps are noisy and expensive, while clinically correct querying depends on compact but high-value knowledge such as: which table is the canonical anchor for a cohort, which dimensions are mandatory to disambiguate a request, which joins are common and which are anti-patterns, and which filters should never be omitted.

This paper proposes *schema guidance* as the missing abstraction layer. Schema guidance is a structured representation attached to schema elements to guide query construction rather than merely describe tables. A guidance entry can encode key filters, commonly joined tables, critical notes, and `do_not` rules. This shifts the locus of optimization away from prompt wording alone and toward the *context object* that shapes planning and decoding.

We operationalize guidance via a schema-guidance construction pipeline. Its distinctive design choice is *information asymmetry*: a guidance optimizer can inspect full schemas and aggregate benchmark failures, but the runtime SQL generator receives only compressed guidance. This makes guidance a distillation target rather than just retrieved text. Conceptually, we view guidance as a compressed sufficient structure for generation; mechanistically, we view it as a search-space shaping device that prunes irrelevant hypotheses while aiming to keep clinically valid programs reachable.

Clinical text-to-SQL Benchmarks Single-domain benchmarks for text-to-SQL tasks in the clinical domain have relied on question-answer pairs derived from surveys generated by providers Lee et al. [2024], Wang et al. [2020]. These are built primarily on open source EHR databases (MIMIC-III and eICU) and tailored to answer tasks requiring reliable text-to-SQL over electronic health records. These do not reflect the broad range of clinical questions that may lie outside of ICU-heavy databases, with queries that span operational as well as clinical tasks. We provide results on existing open datasets but contribute a real-world closed institutional dataset, illustrating the potential of our approach to improve performance across multiple complex clinical settings.

Our contributions are:

1. **Schema guidance as a database-invariant construction.** We formalize *schema guidance* as a structured context object produced by a fixed three-stage construction (Section 3.3): input normalization, a join-neighborhood graph, per-table generation, deterministic validation with bounded repair, and budget-constrained summarization with a $(1-1/e)$ coverage guarantee under the token budget. Only the per-database schema and configuration vary; the construction itself is shared across databases. We call this property *construction invariance* and adopt it as our operational notion of *database-agnostic*. The formal policy definition and the submodular coverage argument are deferred to Appendices A and B.
2. **Information-asymmetric generation as search-space reduction.** The guidance optimizer may inspect the full schema, but the SQL generator sees only the compressed, budget-feasible guidance object. We frame this asymmetry as a search-space reduction at decoding time. A recall-preserving formal statement appears in Appendix B.1.
3. **Empirical evaluation across non-clinical, open clinical, and closed clinical settings.** We apply the same construction to multiple open-source clinical databases and a real-world closed institutional benchmark, and compare against an unguided baseline given the raw schema truncated to the same token budget. The held-out evaluation tests construction invariance as the operational test of generalization: gains accrue to the shared guidance construction rather than to static prompting or one-shot retrieval.

2 Problem Setting

Let $x \in \mathcal{X}$ denote a natural-language clinical query, $\mathcal{S} \in \mathfrak{S}$ a relational schema, and $y^\star$ a target SQL program. A text-to-SQL system is a stochastic mapping

$$p_f(\hat{y} \mid x, c)$$

that produces SQL hypotheses conditioned on a context c. Two choices for c are of interest: an unguided baseline that serializes the raw schema and truncates it to fit a fixed token budget B (formally $\pi_{g_0}(\mathcal{S}) = \mathrm{Trunc}_B(\mathcal{S})$; see Appendix A), and a derived guidance object $c = g$ obtained by a structured transformation of $\mathcal{S}$. We write

$$\hat{y} \sim \mathrm{Decode}\big(p_f(\cdot \mid x, c)\big)$$

for a fixed decoding rule (e.g., greedy, beam, or best-of-k with execution feedback), held constant across all conditions compared in this paper.

Definition 2.1 (Schema-guidance policy)**.** *Let Θ be a space of database-specific configuration parameters (e.g., key/time naming heuristics, slowly-changing-dimension flag handling, sensitivity rules). A* schema-guidance policy *is a parameterized mapping*

$$\pi_g \;:\; \mathfrak{S} \times \Theta \;\to\; \mathcal{G}_B, \qquad g \;=\; \pi_g(\mathcal{S}; \theta),$$

producing a structured context object containing key filters, commonly joined tables, field notes, example query patterns, and critical notes.

We treat $\pi_g(\,\cdot\,;\theta)$ as a *procedure* that is invariant across databases: only the inputs $\mathcal{S}$ and the configuration θ vary, while the construction itself is shared. Section 3.3 specifies the procedure; Section 4 reports its instantiation across the clinical databases studied as $g^{(d)} = \pi_g(\mathcal{S}^{(d)}; \theta^{(d)})$. We refer to this property as *construction invariance* and treat it as our operational notion of *database-agnostic*; we make no formal categorical claim.

Empirical objectives. We evaluate guidance policies on a held-out validation set

$$\mathcal{D}_{\mathrm{val}} \;=\; \{(x_i, \mathcal{S}_i, y_i^\star)\}_{i=1}^{n}.$$

Let $u(x, y^\star, \hat{y}) \in [0, 1]$ be a bounded utility (e.g., execution match, Code F1, Table F1) with $u(x, y^\star, y^\star) = 1$. The empirical utility of a policy is

$$\hat{J}(\pi_g) \;=\; \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\hat{y}_i \sim \mathrm{Decode}(p_f(\,\cdot\,|x_i,\, \pi_g(\mathcal{S}_i;\theta)))} \big[\, u(x_i, y_i^\star, \hat{y}_i) \,\big].$$

Joint reporting For each database d, we report $\hat{J}(\pi_g)$ for the guidance policy π_g, alongside an unguided baseline π_{g_0} (raw schema, truncated to B). Improvements due to guidance are assessed through pairwise comparisons.

3 Method

3.1 Schema guidance as runtime context

The operational role of schema guidance follows directly from the problem setting (Section 2): the generator $p_f(\hat{y} \mid x, c)$ has two natural choices for c, namely the raw schema ($c = \mathcal{S}$) and a derived guidance object ($c = g \in \mathcal{G}_B$). The unguided choice $c = \mathcal{S}$ is generally infeasible at clinical scale once $|\mathcal{S}| > B$; truncation produces the baseline policy π_{g_0} used throughout our experiments. Schema guidance is the alternative: a structured context object produced by a construction π_g that is shared across databases (Section 3.3). Our approach augments runtime context by $g = \pi_g(\mathcal{S}; \theta)$, the schema-derived guidance. The schema-derived component g is the only part of c produced by π_g.

3.2 Information asymmetry between construction and inference

We enforce an explicit asymmetry over which arguments each stage may inspect:

- the construction π_g may read the full schema $\mathcal{S}$ and the configuration θ, both of which are construction-time inputs and do not depend on $\mathcal{D}_{\mathrm{val}}$;
- the generator $p_f(\hat{y} \mid x, c)$ at initialization receives only the guidance object $c = g$ with $|g| \le B$; any further per-table information acquired during decoding is mediated by tool calls (Appendix D), and the raw schema $\mathcal{S}$ is never delivered as a flat artifact.

We do not feed validation outcomes from $\mathcal{D}_{\mathrm{val}}$ back into the construction of π_g or $\theta^{(d)}$; held-out databases are constructed from $\mathcal{S}^{(d)}$ and $\theta^{(d)}$ alone, so the asymmetry concerns what each side may inspect about the schema, not what either side knows about validation labels. The asymmetry results in the generator's admissible hypothesis space at inference time to be the support of $\mathrm{Decode}(p_f(\,\cdot \mid x, g))$. Because π_g is a procedure shared across databases (Section 3.3), improving θ or any stage of ϕ generalizes across schemas without retraining f. We therefore treat π_g as the optimization surface and f as fixed throughout the paper.

3.3 Schema-guidance construction pipeline

We construct $g^{(d)} = \pi_g(\mathcal{S}^{(d)}; \theta^{(d)})$ via a fixed three-stage pipeline whose only per-database inputs are the raw schema and the configuration $\theta^{(d)}$. All stages are evaluated empirically. Full operator signatures and the validator specification are given in Appendix B.

Stage 1: Input normalization. We canonicalize raw schema metadata for each database into a uniform per-database JSON of tables, columns, data types, optional textual descriptions, and foreign-key relationships (extracted from declarations, or inferred from naming conventions when absent).

Stage 2: Schema graph and join neighborhoods. We build a foreign-key graph over tables and define each table's *one-hop join neighborhood*; per-table guidance is conditioned on its neighborhood so that join structure is reflected locally. Multi-hop structure is recovered indirectly via overlapping neighborhoods at the summarization step.

Stage 3: Per-table guidance generation. For each table T_j, a per-table operator ϕ produces a structured artifact g_j with slots for indexed columns, performance filters, common joins, field notes, and example query patterns. The operator composes (i) a deterministic *heuristic skeleton* that fills index/filter/join slots from column names, types, and declared relationships; (ii) an *LLM enrichment* step conditioned on the table, its neighborhood, and any domain hints in $\theta^{(d)}$; and (iii) an optional *curation merge* that overlays subject-matter-expert annotations when available.

Per-database instantiation. The complete policy is the composition

$$\pi_g(\mathcal{S}^{(d)}; \theta^{(d)}) \;=\; \mathrm{Summarize}\Big(\big\{\, \phi\big(T_j, \mathcal{N}(T_j); \theta^{(d)}\big) \,\big\}_{j=1}^{m_d};\, B \Big),$$

applied independently to each database d. The procedure $\pi_g(\,\cdot\,;\,\cdot\,)$ is identical across all databases studied; only the inputs $(\mathcal{S}^{(d)}, \theta^{(d)})$ change. Whether this construction invariance suffices for transfer to unseen schemas is decided empirically by the held-out database evaluation.

3.4 Guidance as compressed sufficient structure and search-space reduction view

Compressed sufficient structure. We do not claim sufficiency in the information-theoretic sense. Instead, we use *compressed sufficient structure* as a working notion of what guidance should encode. Let $\mathcal{H}(\mathcal{S}, x)$ denote the SQL programs compatible with the full schema $\mathcal{S}$ for query x, and let $\mathcal{H}_g(\mathcal{S}, x) \subseteq \mathcal{H}(\mathcal{S}, x)$ denote the support effectively explored by the generator under guidance g. We call g *task-sufficient* for $(x, \mathcal{S}, y^\star)$ when $y^\star \in \mathcal{H}_g(\mathcal{S}, x)$, and *recall-preserving on a distribution* $\mathcal{D}$ when

$$\Pr_{(x,\mathcal{S},y^\star)\sim\mathcal{D}}\big[\, y^\star \in \mathcal{H}_g(\mathcal{S}, x) \,\big] \;\geq\; 1 - \delta$$

for some small δ. Recall-preservation is a *necessary* condition for guidance to help: if $y^\star \notin \mathcal{H}_g$, decoding under g strictly loses regardless of what is pruned. Whether π_g achieves recall-preservation in practice on real clinical schemas is an empirical question; the construction's Stage 4 validator certifies well-formedness of *what is kept*, not coverage of unseen queries (Appendix B.1).

Search-space reduction. Decoding under context c produces $\hat{y}$ from a distribution supported on $\mathcal{H}_c(\mathcal{S}, x)$. We adopt one working assumption about LLM decoders: that the probability of decoder failure is upper bounded by a non-decreasing function $\varepsilon(\,\cdot\,)$ of the realized candidate-set size. Under this assumption, recall-preserving and precision-improving guidance gives

$$\Pr[\hat{y} \neq y^\star \mid g] \;\leq\; \varepsilon\big(|\mathcal{H}_g(\mathcal{S}, x)|\big) \;\leq\; \varepsilon\big(|\mathcal{H}(\mathcal{S}, x)|\big).$$

This is the formal content of the search-space-reduction view: gains scale with how much $\mathcal{H}_g$ shrinks $\mathcal{H}$ *without losing the gold program* (Appendix B.1). The same logic predicts *where* gains should concentrate. Spurious table selection and structurally invalid joins are exactly the failure modes whose support is large in $\mathcal{H}(\mathcal{S}, x)$ when the schema is large; pruning them tightens the bound and should manifest in *structural* metrics (Code F1, Table F1) before answer-level metrics.

Connection to agentic context. This perspective is useful beyond text-to-SQL. An agent does not only retrieve context; it must choose which context to *maintain* as a stable representation between steps. We argue that for SQL generation, the right stable representation encodes query-construction knowledge—joins, anti-patterns, mandatory filters, do-not rules—rather than raw schema text. Schema guidance is the operational instantiation: the construction pipeline (Section 3.3) produces the representation.

Empirical signature. The mechanism makes a falsifiable prediction: if guidance primarily compresses *structural* hypothesis support while preserving gold reachability, gains in Code F1 and Table F1 should exceed gains in execution-match. Our experiments (Section 4) are consistent with this prediction.

4 Experimental Setup

4.1 SPIDER

To evaluate our method on widely studied text-to-SQL tasks, we adopt a unified test set constructed from multiple classical benchmarks, following the preprocessing provided in prior work.[Yu et al., 2019] Test examples are aggregated from eight text-to-SQL datasets across multiple domains; each example consists of a natural language query, a corresponding ground-truth SQL query, and auxiliary metadata. The SQL queries are rewritten to follow a standardized format consistent with the SPIDER dataset, enabling uniform evaluation across datasets. Additional details in Appendix C

4.2 MIMIC-III

In the EHRSQL benchmark, MIMIC-III serves as one of the underlying relational databases over which natural-language questions are paired with executable SQL queries. This makes it particularly suitable for evaluating text-to-SQL systems in the clinical domain, where questions often require temporal reasoning, aggregation, and multi-table joins [Lee et al., 2024]. In our work, we follow the benchmark setting provided by EHRSQL rather than reconstructing the database schema independently.

4.3 eICU

Within EHRSQL, eICU complements MIMIC-III by providing a more heterogeneous clinical environment for evaluating text-to-SQL generalization [Lee et al., 2024]. Using both MIMIC-III and eICU is important because it reduces the risk that performance reflects overfitting to a single hospital-specific schema or documentation style. Therefore, the benchmark offers a more realistic test bed for clinical question answering over relational EHR databases.

4.4 Closed Institutional Benchmark of Real World Queries

From a tertiary medical center 554 SQL queries tied to clinical requests to Information Technology for data extraction were collated. Each SQL query was linked to a natural language requests. If multiple queries were linked to the original request, natural language queries were derived from the individual SQL query using zero shot gemini-2.5-pro.

5 Benchmark and Evaluation Metrics

5.1 Benchmark construction

We construct a SQL-centric benchmark from a corpus of clinically validated queries organized as a hierarchy of task folders. Each folder corresponds to a high-level clinical task, described by a human-authored *solution description*, and contains one or more SQL queries representing different data extraction components (e.g., demographics, labs, medications).

Each SQL query is paired with a natural language *task description* derived from the query using a preprocessing pipeline. This results in a dataset of approximately 554 query-level test cases, each

defined as:

$$(x_i, y_i^\star),$$

where x_i is a natural language task description and $y_i^\star$ is the corresponding gold SQL query.

Multiple queries within a folder may share overlapping logic (e.g., cohort definitions and join patterns), which is intentional and reflects real-world analytical workflows. This allows evaluation of both independent query generation and implicit structural consistency.

5.2 Evaluation setup

For each test case, an agent receives a natural language task description x_i along with clinical concept descriptions a_i (e.g., value-set aligned concepts). The agent generates SQL $\hat{y}_i = f(x_i, a_i)$, which is executed against the database alongside the gold SQL $y_i^\star$.

Evaluation is primarily execution-based, supplemented by structural metrics computed from generated and gold SQL queries.

Model and decoding configuration. We use gemini-2.5-pro via the Vertex AI GenAI API (`vertexai=True`) using the Google `genai` client, with the model identifier (e.g., `gemini-2.5-pro`) and Google Cloud `project` and `location` supplied at initialization.

For comparison experiments, we additionally evaluate gemini-2.5-flash via the same Vertex AI API, gpt-5.2 served through an enterprise API gateway (Apigee) using an Azure OpenAI–compatible interface, and medgemma-1.5-4b-it deployed locally via vLLM.

Decoding uses `temperature`=0.0 for API-based models to encourage deterministic SQL generation; other decoding hyperparameters (e.g., top-p, top-k, maximum output tokens) are not explicitly set and therefore follow API defaults. For MedGemma, decoding uses `temperature`=0.2 and `max_tokens`=2048, with other parameters following vLLM defaults.

For agent control, we allow up to five function-calling rounds per query. Transient API failures are handled with up to two retries and a five-second delay between retry attempts.

Compute setup. API-based models are accessed through external inference APIs from Google Cloud Workbench notebooks and are not executed locally.

Medgemma-1.5-4b-it is deployed locally on a Google Cloud VM with a g2-standard-16 configuration (16 vCPUs, 64 GB RAM) and a single NVIDIA L4 GPU (24 GB VRAM). Each dataset evaluation requires on the order of hours, depending on the number of queries.

5.3 Per-test-case metrics

Code F1. We compute F1 over extracted clinical codes (e.g., ICD-10) to measure correctness of inclusion criteria.

Table F1. We compute F1 over referenced tables to evaluate schema selection.

Column Recall. Recall over expected output columns: $|\mathrm{Col}_{\text{agent}} \cap \mathrm{Col}_{\text{gold}}|/|\mathrm{Col}_{\text{gold}}|$.

5.4 Composite score

We focus on the metrics with signal in the uploaded summary: Composite score, Code F1, Table F1, and Column Recall. The exported patient-level, row-count, and null-quality metrics were uniformly zero across all evaluated settings, and are therefore excluded from the analysis. Composite score serves as the main end-to-end benchmark objective. We place weights to reflect the relative importance of cohort identification (code) along with correctly selected tables and columns in the SQL query. We do not normalize weights to sum to 1; the composite score should be interpreted as a weighted utility rather than a probability-like metric. We plan to incorporate additional downstream evaluation dimensions—such as patient counts returned, null-value handling, and column datatype validation—that are currently excluded due to the inability to execute queries on the requisite databases. Metrics are aggregated across test cases using mean, min, and max statistics.

$$\text{Score} = 0.20 \cdot \text{Code F1} + 0.15 \cdot \text{Table F1} + 0.15 \cdot \text{Column Recall}$$

Comparison Protocol We compare *with query guidance* against *without query guidance but access to schemas*. The CSV includes multiple runs/models per dataset; our tables report averages (across 3 runs) and models per dataset. This comparison isolates the value of *guidance as managed context*. The baseline is not deprived of schema access entirely; instead, it has schema access without the curated guidance layer.

More precisely, the with-guidance condition pre-loads the global `query_guidance_index` and the `MASTER_DESCRIPTIONS.json` summary into the agent's initial system prompt, while the without-guidance condition omits these. Both conditions retain access to the same `load_table_schemas` tool, which returns per-table `query_guidance` fields when present in the JSON artifacts. The comparison therefore isolates the value of *eager* guidance—guidance resident in the initial context window—relative to *lazy*, on-demand retrieval through tool calls (Appendix D.2). Improvements therefore support the claim that *how* and *when* schema information is represented matters, not only whether it is present.

6 Results

6.1 Guidance Improves Aggregate Performance

Table 1 reports aggregate results across all dataset–model pairs, averaged over three runs per configuration, with clinical datasets reported separately from the SPIDER dataset.

Across clinical datasets, schema guidance improves Composite score from 0.220 ± 0.017 to 0.296 ± 0.018, Code F1 from 0.691 ± 0.036 to 0.880 ± 0.016, Table F1 from 0.290 ± 0.055 to 0.451 ± 0.064, and Column Recall from 0.253 ± 0.029 to 0.351 ± 0.039. The largest relative improvement is observed in Table F1 ($+55.5\%$), consistent with the search-space reduction view that guidance helps the model select the correct structural backbone before fine-grained decoding. We provide additional results about failure mode analysis in the appendixE

Setting	Composite	Code F1	Table F1	Column Recall
Clinical datasets				
With guidance	$\mathbf{0.296 \pm 0.018}$	$\mathbf{0.880 \pm 0.016}$	$\mathbf{0.451 \pm 0.064}$	$\mathbf{0.351 \pm 0.039}$
Without guidance	0.220 ± 0.017	0.691 ± 0.036	0.290 ± 0.055	0.253 ± 0.029
SPIDER				
With guidance	$\mathbf{0.178 \pm 0.012}$	$\mathbf{0.651 \pm 0.040}$	$\mathbf{0.258 \pm 0.022}$	$\mathbf{0.062 \pm 0.003}$
Without guidance	0.056 ± 0.008	0.215 ± 0.029	0.062 ± 0.014	0.027 ± 0.004

Table 1: Aggregate results (mean $\pm$ standard error) reported separately for clinical datasets and the SPIDER dataset. Results are averaged over dataset–model pairs and three runs per configuration.

6.2 Consistent Gains Across Datasets

Table 2 reports results by dataset. The Composite score improves under guidance on every dataset: from 0.133 ± 0.007 to 0.180 ± 0.005 on the closed institutional benchmark, from 0.243 ± 0.044 to 0.419 ± 0.005 on EHRSQL-eICU, from 0.289 ± 0.013 to 0.320 ± 0.012 on EHRSQL-MIMIC-III, and from 0.056 ± 0.008 to 0.178 ± 0.012 on SPIDER. Code F1 likewise improves across all datasets: $0.591 \pm 0.034 \rightarrow 0.799 \pm 0.018$ (closed), $0.584 \pm 0.088 \rightarrow 0.961 \pm 0.011$ (eICU), $0.870 \pm 0.012 \rightarrow 0.901 \pm 0.025$ (MIMIC-III), and $0.215 \pm 0.029 \rightarrow 0.651 \pm 0.040$ (SPIDER).

The largest relative gains are observed on SPIDER, where the Composite score increases more than $3\times$ and Table F1 rises from 0.062 ± 0.014 to 0.258 ± 0.022, and on EHRSQL-eICU, where Table F1 improves from 0.453 ± 0.105 to 0.839 ± 0.016 and Column Recall increases substantially ($0.389 \pm 0.071 \rightarrow 0.677 \pm 0.014$). In contrast, EHRSQL-MIMIC-III shows the smallest absolute gain, consistent with its stronger unguided baseline.

6.3 Significance of Improvement in Structural Metrics

The concentration of gains in structural metrics (Code F1 and Table F1) supports the search-space reduction hypothesis: guidance is most effective when it suppresses structurally implausible hypotheses.

Dataset	N	Comp. (G)	Comp. (No G)	Code F1 (G)	Code F1 (No G)	Table F1 (G)	Table F1 (No G)	Column Recall (G)	Column Recall (No G)
Closed institutional benchmark	554	$\mathbf{0.180 \pm 0.005}$	0.133 ± 0.007	$\mathbf{0.799 \pm 0.018}$	0.591 ± 0.034	$\mathbf{0.007 \pm 0.001}$	0.004 ± 0.000	$\mathbf{0.129 \pm 0.012}$	0.094 ± 0.006
EHRSQL-eICU	1204	$\mathbf{0.419 \pm 0.005}$	0.243 ± 0.044	$\mathbf{0.961 \pm 0.011}$	0.584 ± 0.088	$\mathbf{0.839 \pm 0.016}$	0.453 ± 0.105	$\mathbf{0.677 \pm 0.014}$	0.389 ± 0.071
EHRSQL-MIMIC-III	1198	$\mathbf{0.320 \pm 0.012}$	0.289 ± 0.013	$\mathbf{0.901 \pm 0.025}$	0.870 ± 0.012	$\mathbf{0.606 \pm 0.043}$	0.454 ± 0.081	$\mathbf{0.328 \pm 0.004}$	0.310 ± 0.006
SPIDER	3509	$\mathbf{0.178 \pm 0.012}$	0.056 ± 0.008	$\mathbf{0.651 \pm 0.040}$	0.215 ± 0.029	$\mathbf{0.258 \pm 0.022}$	0.062 ± 0.014	$\mathbf{0.062 \pm 0.003}$	0.027 ± 0.004

Table 2: Per-dataset results (mean $\pm$ standard error) averaged over three runs and models available in both guidance conditions. Clinical datasets are reported alongside the SPIDER dataset benchmark for comparison.

Model	Comp. (G)	Comp. (No G)	Code F1 (G)	Code F1 (No G)	Table F1 (G)	Table F1 (No G)	Column Recall (G)	Column Recall (No G)
gemini-2.5-flash	$\mathbf{0.280 \pm 0.029}$	0.170 ± 0.032	$\mathbf{0.838 \pm 0.033}$	0.491 ± 0.076	$\mathbf{0.443 \pm 0.095}$	0.277 ± 0.076	$\mathbf{0.306 \pm 0.070}$	0.198 ± 0.045
gemini-2.5-pro	$\mathbf{0.301 \pm 0.031}$	0.244 ± 0.038	$\mathbf{0.897 \pm 0.026}$	0.712 ± 0.072	$\mathbf{0.493 \pm 0.107}$	0.403 ± 0.105	$\mathbf{0.320 \pm 0.077}$	0.276 ± 0.069
gpt-5.2	$\mathbf{0.267 \pm 0.033}$	0.140 ± 0.029	$\mathbf{0.810 \pm 0.056}$	0.470 ± 0.079	$\mathbf{0.406 \pm 0.098}$	0.163 ± 0.068	$\mathbf{0.293 \pm 0.066}$	0.144 ± 0.031
medgemma-1.5-4b-it	$\mathbf{0.201 \pm 0.023}$	0.185 ± 0.016	0.729 ± 0.013	$\mathbf{0.776 \pm 0.035}$	$\mathbf{0.186 \pm 0.081}$	0.001 ± 0.001	0.183 ± 0.055	$\mathbf{0.197 \pm 0.060}$

Table 3: Performance comparison across models (mean $\pm$ standard error), averaged over datasets where both guidance conditions are available and over three runs per configuration.

In clinical SQL, errors in table selection are often fatal to downstream interpretation, and guidance directly targets this failure mode.

One caveat is the closed institutional benchmark, where Table F1 remains near zero in both conditions (0.007 ± 0.001 with guidance, 0.004 ± 0.000 without). This likely reflects the use of views, CTEs, and schema-qualified or aliased table names in the reference queries, which are not fully captured by the table-name matching metric. Furthermore, complex queries sometimes integrated various databases where the schemas are separate from the current schemas used in this work. We therefore report Table F1 for completeness but rely primarily on Code F1 and Composite score as more reliable indicators on this dataset.

6.4 Relevance to Clinical Applications

The closed institutional benchmark results are especially important because they suggest the method is not limited to public benchmark schemas. Real clinical warehouses often contain many tables, multiple near-synonymous dimensions, and institutional conventions that are not obvious from schema names alone. Guidance provides a place to encode those conventions explicitly. Precisely this type of knowledge is difficult to recover from one-shot prompting but well suited to a structured, updateable context object.

7 Related Work

Large Language Model Text-to-SQL Systems Modern text-to-SQL systems increasingly rely on large language models, with a range of enhancements improving performance across domains. Initial work focused on single LLM enhancements with prompt-level methods such as in-context learning Sun et al. [2023], Zhang et al. [2023], Pourreza and Rafiei [2023] and prompt engineering Pandey et al. [2025], as well as retrieval-augmented pipelines [Shen et al., 2024] and reinforcement-learning-enhanced retrieval [Toteja et al., 2025]. Other approaches directly modify model weights through fine-tuning Chafik et al. [2026], Roberson et al. [2024], augment capabilities by incorporating expert knowledge Hong et al. [2024] or synthetic data Caferoğlu et al. [2025], or seek to introduce alternative query representations Eyal et al. [2023].

With the increasing capability of agentic systems, system-level approaches are emerging that incorporate multiple LLMs via multi-step planning, with or without agentic control or reinforcement learning Chen et al. [2024], Pham et al. [2026], Xu et al. [2025], Wang et al. [2025], Shao et al. [2025], or leverage ensemble methods Shen et al. [2023].

8 Conclusion

Our results suggest that agentic SQL systems benefit from treating context as something to be *constructed* rather than only retrieved. In the clinical setting, raw schemas exceed practical context windows and a structured guidance object that is explicit, inspectable, and budget-feasible offers a

usable middle ground between flat retrieval and opaque latent memory. The empirical evidence here is limited to the clinical databases we evaluate and the specific construction in Section 3.3.

Limitations Our empirical evaluation is based on exported benchmark summaries rather than raw per-example traces. This restricts the depth of error analysis and ablation we can provide since we do not provide it at the query level. Future iterations should include run-level paired comparisons and error analysis. While we frame schema guidance as the primary contribution, the system is to be implemented within a broader agentic pipeline that includes planning, value resolution, validation, and repair. Improvements may arise in part from system-level redundancy, error correction, or architectural choices rather than the guidance representation alone. A key issue is the analysis primarily on the SQL queries, and not the returned data; the system may produce syntactically and structurally plausible SQL that is nonetheless clinically incorrect, incomplete, or misleading. This emphasis on structural metrics (e.g. Code F1 and Table F1) without assessment of clinically meaningful correctness of the data returned is primarily due issues with accessing identifiable data for the closed institutional dataset. However, we have built in downstream metrics (e.g., patient-level accuracy, row-count agreement, null-quality) with future plans to use these metrics to assess real-world utility. Finally, the approach is evaluated primarily on structured relational databases with relatively well-defined schemas. Its applicability to noisier real-world settings—such as partially documented schemas, missing or implicit foreign keys, heavy reliance on free-text fields, or hybrid data models (e.g., FHIR, event streams)—is not established. More broadly, the system does not address key deployment concerns in clinical environments, including uncertainty estimation, interpretability, auditability, and human oversight. These limitations are critical for real-world adoption and remain open directions for future work.

Broader Impact This work lays a foundation for iterating from static guidance construction toward a future of *interactive, clinician-in-the-loop context refinement*. Schema guidance is naturally interpretable and editable, which creates an opportunity for domain experts to directly inspect and revise the context that governs generation. Incorporating such feedback into the guidance optimization loop would allow the system to iteratively refine its representation of clinically relevant structure, transforming guidance from a fixed, dataset-level artifact into a continuously evolving knowledge layer. We anticipate integration of this into an agentic harness that incorporates deterministic validators that check structural properties, a SQL fixer that repairs failed outputs; and an execution-aware gate using dry-run validation. These can be further extended beyond single-query generation into time-varying *longitudinal phenotyping* across clinical databases. Finally, we envision applications to downstream tasks in clinical research and care, particularly *deep phenotyping and outcome ascertainment*. By enabling more reliable and interpretable cohort definitions, schema guidance could support the construction of high-quality observational datasets and clinically meaningful endpoints. More broadly, this work points toward agentic systems that do not merely generate queries, but maintain and refine structured representations of domain knowledge that can be reused across tasks, datasets, and institutions. Despite these benefits, there are important risks to consider. Incorrect or incomplete SQL generation may lead to misleading clinical analyses if used without expert validation, particularly in high-stakes settings. Misuse of such systems for automated cohort construction without appropriate oversight could introduce bias or propagate erroneous assumptions. In addition, applying schema guidance to sensitive clinical data raises privacy and governance concerns. Mitigation strategies include maintaining human-in-the-loop review, enforcing validation checks prior to execution, and restricting deployment to controlled environments with appropriate data governance.

References

Hasan Alp Caferoğlu, Mehmet Serhat Çelik, and Özgür Ulusoy. Sing-sql: A synthetic data generation framework for in-domain text-to-sql translation, 2025. URL https://arxiv.org/abs/2509.25672.

Salmane Chafik, Saad Ezzini, and Ismail Berrada. Lego-code: Can modular curriculum learning advance complex code generation? insights from text-to-sql, 2026. URL https://arxiv.org/abs/2604.18254.

Ziru Chen, Michael White, Raymond Mooney, Ali Payani, Yu Su, and Huan Sun. When is tree search useful for llm planning? it depends on the discriminator, 2024. URL https://arxiv.org/abs/2402.10890.

Ben Eyal, Amir Bachar, Ophir Haroche, and Michael Elhadad. Semantic parsing for complex data retrieval: Targeting query plans vs. sql for no-code access to relational databases, 2023. URL https://arxiv.org/abs/2312.14798.

Satya K Gundabathula and Sriram R Kolar. Promptmind team at ehrsql-2024: Improving reliability of sql generation using ensemble llms, 2024. URL https://arxiv.org/abs/2405.08839.

Zijin Hong, Zheng Yuan, Hao Chen, Qinggang Zhang, Feiran Huang, and Xiao Huang. Knowledge-to-sql: Enhancing sql generation with data expert llm, 2024. URL https://arxiv.org/abs/2402.11517.

Alistair E. W. Johnson, Tom J. Pollard, Lu Shen, Li-Wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3:160035, 2016. doi: 10.1038/sdata.2016.35.

B. Ross Katz, Abdul Khan, James York-Winegar, and Alexander J. Titus. Nhanes-gcp: Leveraging the google cloud platform and bigquery ml for reproducible machine learning with data from the national health and nutrition examination survey, 2024. URL https://arxiv.org/abs/2401.06967.

Gyubok Lee, Sunjun Kweon, Seongsu Bae, and Edward Choi. Overview of the EHRSQL 2024 shared task on reliable text-to-sql modeling on electronic health records. In *Proceedings of the 6th Clinical Natural Language Processing Workshop*, pages 644–654, Mexico City, Mexico, jun 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.clinicalnlp-1.62. URL https://aclanthology.org/2024.clinicalnlp-1.62/.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023. URL https://arxiv.org/abs/2307.03172.

Ali Modarressi, Hanieh Deilamsalehy, Franck Dernoncourt, Trung Bui, Ryan A. Rossi, Seunghyun Yoon, and Hinrich Schütze. Nolima: Long-context evaluation beyond literal matching, 2025. URL https://arxiv.org/abs/2502.05167.

Hari Mohan Pandey, Anshul Gupta, Subham Sarkar, Minakshi Tomer, Schneider Johannes, and Yan Gong. Gemma-sql: A novel text-to-sql model based on large language models, 2025. URL https://arxiv.org/abs/2511.04710.

Minh Tam Pham, Trinh Pham, Tong Chen, Hongzhi Yin, Quoc Viet Hung Nguyen, and Thanh Tam Nguyen. Av-sql: Decomposing complex text-to-sql queries with agentic views, 2026. URL https://arxiv.org/abs/2604.07041.

Tom J. Pollard, Alistair E. W. Johnson, Joseph D. Raffa, Leo A. Celi, Roger G. Mark, and Omar Badawi. The eicu collaborative research database, a freely available multi-center database for critical care research. *Scientific Data*, 5:180178, 2018. doi: 10.1038/sdata.2018.178.

Mohammadreza Pourreza and Davood Rafiei. Din-sql: Decomposed in-context learning of text-to-sql with self-correction, 2023. URL https://arxiv.org/abs/2304.11015.

Richard Roberson, Gowtham Kaki, and Ashutosh Trivedi. Analyzing the effectiveness of large language models on text-to-sql synthesis, 2024. URL https://arxiv.org/abs/2401.12379.

Zhihui Shao, Shubin Cai, Rongsheng Lin, and Zhong Ming. Enhancing text-to-sql with question classification and multi-agent collaboration. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4340–4349, Albuquerque, New Mexico, apr 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.245. URL https://aclanthology.org/2025.findings-naacl.245/.

Ran Shen, Gang Sun, Hao Shen, Yiling Li, Liangfeng Jin, and Han Jiang. Spsql: Step-by-step parsing based framework for text-to-sql generation, 2023. URL https://arxiv.org/abs/2305.11061.

Zhili Shen, Pavlos Vougiouklis, Chenxin Diao, Kaustubh Vyas, Yuanyi Ji, and Jeff Z. Pan. Improving retrieval-augmented text-to-sql with ast-based ranking and schema pruning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7865–7879, Miami, Florida, USA, nov 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.449. URL https://aclanthology.org/2024.emnlp-main.449/.

Ruoxi Sun, Sercan Ö. Arik, Rajarishi Sinha, Hootan Nakhost, Hanjun Dai, Pengcheng Yin, and Tomas Pfister. Sqlprompt: In-context text-to-sql with minimal labeled data, 2023. URL https://arxiv.org/abs/2311.02883.

Samuel Thio, Matthew Lewis, Spiros Denaxas, and Richard J. B. Dobson. Unlocking electronic health records: a hybrid graph rag approach to safe clinical ai for patient qa. *Frontiers in Digital Health*, 8, March 2026. ISSN 2673-253X. doi: 10.3389/fdgth.2026.1780700. URL http://dx.doi.org/10.3389/fdgth.2026.1780700.

Rishit Toteja, Arindam Sarkar, and Prakash Mandayam Comar. In-context reinforcement learning with retrieval-augmented generation for text-to-sql. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 10390–10397, Abu Dhabi, UAE, jan 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.692/.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, LinZheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. Mac-sql: A multi-agent collaborative framework for text-to-sql. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 540–557, Abu Dhabi, UAE, jan 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.36/.

Ping Wang, Tian Shi, and Chandan K. Reddy. Text-to-sql generation for question answering on electronic medical records, 2020. URL https://arxiv.org/abs/1908.01839.

Zekun Xu, Siyu Xia, Chuhuai Yue, Jiajun Chai, Mingxue Tian, Xiaohan Wang, Wei Lin, Haoxuan Li, and Guojun Yin. Mtir-sql: Multi-turn tool-integrated reasoning reinforcement learning for text-to-sql, 2025. URL https://arxiv.org/abs/2510.25510.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task, 2019. URL https://arxiv.org/abs/1809.08887.

Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. Act-sql: In-context learning for text-to-sql with automatically-generated chain-of-thought, 2023. URL https://arxiv.org/abs/2310.17342.

Angelo Ziletti and Leonardo D'Ambrosi. Retrieval augmented text-to-sql generation for epidemiological question answering using electronic health records. In *Proceedings of the 6th Clinical Natural Language Processing Workshop*, pages 47–53, Mexico City, Mexico, jun 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.clinicalnlp-1.4. URL https://aclanthology.org/2024.clinicalnlp-1.4/.

Angelo Ziletti and Leonardo D'Ambrosi. Generating patient cohorts from electronic health records using two-step retrieval-augmented text-to-sql generation, 2025. URL https://arxiv.org/abs/2502.21107.

A Formal problem setting

This appendix gives the formal version of the problem setting summarized in Section 2. Notation is consistent with the body; we restate the objects here so that the construction in Appendix B and the search-space view in Appendix B.1 can be read self-containedly.

Inputs and generator. Let $x \in \mathcal{X}$ denote a natural-language clinical query, $\mathcal{S} \in \mathfrak{S}$ a relational schema drawn from a space of schemas $\mathfrak{S}$, and $y^\star$ a target SQL program. A text-to-SQL system is a stochastic mapping

$$p_f(\hat{y} \mid x, c)$$

that produces SQL hypotheses conditioned on a context c. Two choices for c are of interest: the raw schema $c = \mathcal{S}$ and a derived guidance object $c = g$. Hypotheses are sampled as

$$\hat{y} \;\sim\; \mathrm{Decode}\big(p_f(\,\cdot \mid x, c)\big),$$

where $\mathrm{Decode}(\cdot)$ is a fixed decoding rule (e.g., greedy, beam, or best-of-k with execution feedback) held constant across all conditions compared in the paper. The generator f is also held fixed: all comparisons reported below vary c only.

Definition A.1 (Schema-guidance policy)**.** *Let Θ be a space of database-specific configuration parameters (e.g., key/time naming heuristics, slowly-changing-dimension flag handling, sensitivity rules). A* schema-guidance policy *is a parameterized mapping*

$$\pi_g \;:\; \mathfrak{S} \times \Theta \;\longrightarrow\; \mathcal{G}_B, \qquad g \;=\; \pi_g(\mathcal{S}; \theta),$$

producing a structured context object containing key filters, commonly joined tables, field notes, example query patterns, and critical notes.

We treat $\pi_g(\,\cdot\,; \theta)$ as a *procedure* that is invariant across databases: only the inputs $\mathcal{S}$ and the configuration θ vary, while the construction itself is shared. The pipeline realizing π_g is specified in Appendix B, and its per-database instantiation is written $g^{(d)} = \pi_g(\mathcal{S}^{(d)}; \theta^{(d)})$. We refer to this property as *construction invariance* and treat it as our operational notion of *database-agnostic*; we make no formal categorical claim.

Unguided baseline The unguided baseline π_{g_0} is defined as raw schema text serialized and truncated to fit $|g| \le B$:

$$\pi_{g_0}(\mathcal{S}; \theta) \;=\; \mathrm{Trunc}_B(\mathcal{S}),$$

where Trunc_B is a deterministic, schema-independent serialization-and-truncation rule. This isolates the value of *managed* context: π_g and π_{g_0} consume the same budget B and the same generator f, and differ only in how schema information is represented.

Empirical objective We evaluate guidance policies on a held-out validation set

$$\mathcal{D}_{\mathrm{val}} \;=\; \{(x_i, \mathcal{S}_i, y_i^\star)\}_{i=1}^{n}.$$

Let $u(x, y^\star, \hat{y}) \in [0, 1]$ be a bounded utility (e.g., execution match, Code F1, Table F1) with $u(x, y^\star, y^\star) = 1$. The empirical utility of a policy π_g under fixed θ is

$$\hat{J}(\pi_g) \;=\; \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\hat{y}_i \sim \mathrm{Decode}(p_f(\,\cdot\,|x_i,\, \pi_g(\mathcal{S}_i;\theta)))}\big[\, u(x_i, y_i^\star, \hat{y}_i) \,\big].$$

For each database d we report $\hat{J}(\pi_g)$ against $\hat{J}(\pi_{g_0})$, and the improvement $\Delta(d) = \hat{J}(\pi_g^{(d)}) - \hat{J}(\pi_{g_0}^{(d)})$ quantifies the value of guidance on that database. Construction invariance is then operationalized as the requirement that a single procedure π_g yields $\Delta(d) > 0$ across the held-out databases in Section 4.

What is held fixed For all comparisons in the body, the following are held constant across π_g and π_{g_0}: the generator f, the decoding rule $\mathrm{Decode}(\cdot)$, the validation set $\mathcal{D}_{\mathrm{val}}$, and the utility u. The only quantity that varies is the context $c \in \{\, \pi_g(\mathcal{S}; \theta),\ \pi_{g_0}(\mathcal{S}; \theta) \,\}$ delivered to f at inference time.

B Detailed construction pipeline

We construct $g^{(d)} = \pi_g(\mathcal{S}^{(d)}; \theta^{(d)})$ via a pipeline whose shape is fixed across datasets and whose only per-database inputs are the raw schema $\mathcal{S}^{(d)}$ and the configuration $\theta^{(d)}$. A single approximation guarantee is invoked for the summarization step; all other stages are evaluated empirically.

Stage 1: Input normalization. For each database d, we normalize raw schema metadata into a canonical representation

$$\mathcal{S}^{(d)} \;=\; \{T_1, \ldots, T_m\}, \qquad T_j \;=\; \big\{(C_{j\ell}, \tau_{j\ell}, \delta_{j\ell})\big\}_{\ell=1}^{n_j},$$

where $C_{j\ell}$ is a column name, $\tau_{j\ell}$ its data type, and $\delta_{j\ell}$ an optional textual description. Foreign-key relationships $\mathcal{R} \subseteq \binom{\bigcup_j T_j}{2}$ are extracted from declarations when available and inferred from naming conventions otherwise. The output is a unified per-database JSON artifact with identical shape across all d.

Stage 2: Schema graph and join neighborhoods. We construct a schema graph $G^{(d)} = (V, E)$ with $V = \{T_1, \ldots, T_m\}$ and edges induced by $\mathcal{R}$. Per-table guidance is generated over a one-hop join neighborhood

$$\mathcal{N}(T_j) \;=\; \{\, T_k \in V \;:\; (T_j, T_k) \in E \,\},$$

so that each table's guidance is conditioned on the tables it can be joined with directly. Multi-hop join structure is recovered indirectly via overlapping neighborhoods at the summarization stage; we report the effect of this locality choice.

Stage 3: Per-table guidance generation. We apply a per-table operator

$$\phi \;:\; \big(T_j, \mathcal{N}(T_j); \theta^{(d)}\big) \;\longmapsto\; g_j \in \mathcal{G}_{\mathrm{tab}},$$

producing a structured artifact g_j with slots for indexed columns, performance filters, common joins, field notes, and example query patterns. The operator ϕ is the composition of three sub-operators applied in sequence:

1. **Heuristic skeleton.** Index, filter, and join slots are populated by deterministic rules over column names, types, and declared relationships (e.g., key/time naming patterns, shared-column join inference). This stage is content-free regarding clinical semantics and depends only on $\theta^{(d)}$.
2. **LLM enrichment.** A language model conditioned on T_j, $\mathcal{N}(T_j)$, a global relationship summary, and any domain hints in $\theta^{(d)}$ proposes natural-language descriptions, recommendations, and example query patterns for the qualitative slots left empty by Stage 3.1.
3. **Curation merge.** When subject-matter-expert annotations exist for $\mathcal{S}^{(d)}$, they are merged into g_j with precedence over the LLM-generated content.

B.1 Search-space reduction view

Let $\mathcal{H}(\mathcal{S})$ denote the hypothesis class of SQL programs compatible with the full schema, and let $\mathcal{H}_g(\mathcal{S}, x) \subseteq \mathcal{H}(\mathcal{S})$ denote the subset that the generator effectively explores at inference time when conditioned on the guidance object g. Guidance is intended to bias the generator away from irrelevant tables, implausible joins, missing filters, and known anti-patterns, so $\mathcal{H}_g$ is typically strictly smaller than the unguided support.

We adopt a deliberately abstract model of decoding error: let $\varepsilon(m)$ be a non-decreasing function of the size m of the candidate program set. This captures the common situation in which decoding becomes harder as more spurious hypotheses remain available; we treat this monotonicity as a working assumption about LLM decoders, not as a theorem.

Remark B.1 (Search-space reduction under recall-preserving guidance)**.** *Suppose, for an example $(x, \mathcal{S}, y^\star)$, that (A1) $y^\star \in \mathcal{H}_g(\mathcal{S}, x)$ (*recall-preserving*), (A2) $|\mathcal{H}_g(\mathcal{S}, x)| < |\mathcal{H}(\mathcal{S}, x)|$ (*precision-improving*), and (A3) the decoder error is upper bounded by a non-decreasing function $\varepsilon(\,\cdot\,)$ of candidate-set size. Then*

$$\Pr[\hat{y} \neq y^\star \mid g] \;\le\; \varepsilon(|\mathcal{H}_g(\mathcal{S}, x)|) \;\le\; \varepsilon(|\mathcal{H}(\mathcal{S}, x)|)\,.$$

We treat Remark B.1 as a *decomposition* rather than a substantive result: the inequality is an immediate consequence of (A1)–(A3), and the substantive content lies in whether those assumptions hold for our construction. (A3) is a working assumption about LLM decoders that we adopt to organize the decomposition. Together, (A1) and (A2) motivate two distinct empirical questions—*recall* of guidance (does g contain the schema elements required by $y^\star$?) and *precision* of guidance (does g suppress structurally implausible alternatives?)—and predict that gains should concentrate in structural metrics (Code F1, Table F1) when both hold. Section 4 reports the structural-metric pattern predicted by this decomposition; a direct empirical decomposition of guidance recall vs. precision is left to future work with run-level traces.

C Additional Dataset details

Licensing and data usage MIMIC-III and eICU are publicly available de-identified clinical datasets accessed under credentialed PhysioNet data use agreements and used in accordance with their respective terms. The SPIDER and related text-to-SQL benchmark datasets are used under their original research licenses. The EHRSQL dataset is released under the CC-BY-4.0 license. The closed institutional dataset is not publicly released due to privacy and data governance constraints. All datasets are used strictly for research purposes and in compliance with their respective licenses and terms of use.

C.1 SPIDER

Each data point includes (i) a database identifier (`db_id`), (ii) a ground-truth SQL query, (iii) extracted constants and their associated schema-level context, (iv) a set of executable test databases for denotation-based evaluation, and (v) the corresponding natural language utterance. This unified format facilitates consistent benchmarking across heterogeneous datasets and reduces variability caused by dataset-specific SQL styles. The SPIDER dataset is released under the Apache-2.0 license and is used in compliance with its terms.

C.2 MIMIC-III

MIMIC-III is a publicly available, de-identified critical care database developed by the MIT Laboratory for Computational Physiology and collaborators. It contains detailed electronic health records associated with intensive care unit (ICU) admissions at Beth Israel Deaconess Medical Center, including demographics, laboratory measurements, medications, charted observations, and clinical events [Johnson et al., 2016]. The database covers adult and neonatal ICU stays collected between 2001 and 2012, and has become a standard resource for clinical machine learning and medical NLP research.

C.3 eICU Collaborative Database

The eICU Collaborative Research Database is a large-scale, de-identified multi-center ICU database collected from hospitals across the United States [Pollard et al., 2018]. Unlike MIMIC-III, which is derived from a single medical center, eICU provides broader institutional diversity, including variations in documentation practices, patient populations, and treatment patterns. The database contains high-granularity information such as vital signs, diagnoses, treatment records, and care documentation.

C.4 Closed Institutional Benchmark Dataset

The closed institutional dataset is not publicly released due to privacy and data governance constraints and is used in accordance with institutional data use and privacy policies. All data are de-identified or handled under appropriate governance procedures, with Institutional Review Board (IRB) approval number 26-002996.

D Runtime agent and prompt

D.1 Runtime agent implementation

At inference time, we instantiate an agentic SQL generator that performs structured text-to-SQL generation under schema constraints. Unless stated otherwise, the configuration includes the full `query_guidance_index` summarized in (ii) below; the without-query-guidance baseline uses the same agent and pipeline but omits that pre-loaded index, as detailed when we describe the simplified prompt template. The agent consumes schema guidance as its primary context and maintains a dynamically constructed system prompt that integrates (i) compact schema descriptions, (ii) a query guidance index summarizing indexed columns, key filters, and join patterns (omitted in the baseline), and (iii) domain-specific hints for mapping user queries to relevant schema components.

Given a natural language query x, the language model is prompted to decide whether additional schema detail is needed, using the task text together with high-level guidance (e.g., the query-guidance summary and domain hints). When additional detail is required, it emits a structured call to `load_table_schemas` with tables identified in `SCHEMA.TABLE` form. The tool returns table definitions drawn from the curated JSON schema artifacts: column names, data types, and—when present in the file—optional `query_guidance` fields (indexed columns, performance-oriented filters, and common join patterns). That payload is serialized into the dialogue and supplied to the next decoding pass together with the standing system instructions and prior turns.

SQL generation follows two design rules communicated to the model: (i) in executable SQL, only simple table names are used (no dataset or schema prefixes), and (ii) column identifiers are restricted to those present in the loaded schema JSON (including outputs returned by `load_table_schemas`), rather than being invented or copied from unrelated examples.

These rules enforce schema-grounded generation through prompting, rather than via an external syntactic validator. The model is instructed to return structured JSON with fields such as the SQL string, a short explanation, tables used, assumptions, and optional notes.

In the minimal inference configuration evaluated here, `load_table_schemas` is the only callable tool for retrieving detailed column definitions from the JSON artifacts; optional routines that sample or summarize rows in a live database, when enabled, belong to an extended SQLite-connected variant and are outside the core schema-only agent.

The agent supports multi-turn behavior with a fixed cap on function-calling rounds. Within a single user query, previously loaded table definitions are cached so repeated `load_table_schemas` requests for the same file avoid redundant I/O. Across independent benchmark instances, our evaluation harness calls an explicit reset after each test case, clearing conversation history and the schema cache so that no context carries over between examples. The same agent object can be reused without reset for multi-turn interaction, in which case prior dialogue and cached schema files persist; this behavior is determined by the caller (e.g., whether the benchmark adapter invokes reset), not by a separate named configuration flag.

D.2 Simplified runtime prompt template

The following is a simplified version of the dynamically generated runtime prompt used in both conditions. In the with-guidance condition, `<<<DYNAMIC_QUERY_GUIDANCE_SUMMARY>>>` is populated from `query_guidance_index`. In the schema-only condition, the agent is initialized directly from schema JSON files without `MASTER_DESCRIPTIONS.json`, leaving `query_guidance_index` empty and removing all pre-loaded guidance summaries from the system prompt. As a result, the model receives only raw schema information in its initial context. Any per-table `query_guidance` fields, if present in the schema artifacts, are only accessible after explicit calls to `load_table_schemas` and are therefore not part of the initial prompt context.

```
You are an expert SQL query generator for a SQLite healthcare database.

[SQLite Configuration]
This database uses SQLite. All SQL queries must follow these rules:
- Use simple table names (e.g., FROM FACT_DIAGNOSIS, FROM DIM_PATIENT)
- Do not use schema prefixes or BigQuery-style paths
```

```
- Use double quotes for identifiers if necessary

[Naming Conventions]
There are two distinct conventions:
1. Schema loading (for load_table_schemas):
   - Use SCHEMA.TABLE format
   - Example: "FACT_CLINICAL_DOCS_MCC.FACT_CLINICAL_DOCUMENTS"
2. SQL generation:
   - Use simple table names only (no schema prefixes)

[Schema Context]
<<<DYNAMIC_JSON_SCHEMA_DESCRIPTIONS>>>
<<<DYNAMIC_QUERY_GUIDANCE_SUMMARY>>>  # empty in schema-only
<<<DYNAMIC_SCHEMA_NOTE>>>
[Instructions]


1. Query Guidance Usage:
   - If query guidance is provided, review it before loading schemas
   - If query guidance is absent, select tables using schema descriptions
     and domain hints only

2. Schema Loading:
   - Use load_table_schemas when detailed schema information is required
   - Always use SCHEMA.TABLE format when loading schemas

3. SQL Generation:
   - Use simple table names (e.g., FACT_DIAGNOSIS)
   - Include appropriate JOIN conditions
   - Add LIMIT clauses (e.g., LIMIT 100)
   - Use ROW_CURRENT_INDICATOR = 'Y' where applicable

4. Column Validation:
   - Only use columns that exist in the loaded schemas
   - Do not infer or assume column names
   - Verify all columns in SELECT, WHERE, JOIN, and ORDER BY clauses

5. Response Format:

For SQL queries:
{
    "sql_query": "...",
    "explanation": "...",
    "tables_used": [...],
    "assumptions": [...],
    "notes": [...]
}

For conversational responses:
{
    "response": "..."
}

6. Follow-up Handling:
   - Maintain conversation context
   - Reuse previously loaded schemas when possible

7. Example Workflow:
   Step 1: Identify relevant tables
```

```
    Step 2: Load schemas using SCHEMA.TABLE format
    Step 3: Generate SQL query using SQLite syntax

Example:
SELECT *
FROM FACT_CLINICAL_DOCUMENTS f
JOIN FACT_DIAGNOSIS d ON f.patient_dk = d.patient_dk
```

E Failure Mode Analysis and Demonstrated Benefit of Query Guidance

Across public datasets without SME curation (eICU, MIMIC-III, SPIDER), guidance consistently reduces non-SQL errors and improves exact match—primarily by decreasing hallucinated tables and columns—demonstrating that gains arise from improved task framing rather than reliance on curated annotations, even as incorrect table selection remains the dominant residual failure mode.

Model	No SQL ↓	Wrong Table ↓	Exact Match ↑	Code F1 ↑
Closed Institutional Benchmark				
gpt-5.2 (No G)	∼0.20	∼0.60	∼0.00	0.56
gpt-5.2 (G)	∼0.05	∼0.80–0.90	∼0.01	0.82
gemini-2.5-flash (No G)	∼0.40	∼0.60	∼0.00	0.42
gemini-2.5-flash (G)	∼0.05	∼0.95+	∼0.006	0.84
gemini-2.5-pro (No G)	∼0.10	∼0.90	∼0.00	0.68
gemini-2.5-pro (G)	∼0.01	∼0.95+	∼0.007	0.85
medgemma-1.5-4b-it (No G)	∼0.15	∼0.84	∼0.00	0.70
medgemma-1.5-4b-it (G)	∼0.18	∼0.81	∼0.00	0.70
EHRSQL-eICU				
gpt-5.2 (No G)	0.92–0.96	∼0.30	∼0.02	0.28
gpt-5.2 (G)	0.20–0.30	∼0.30	0.50+	0.98
gemini-2.5-flash (No G)	0.55–0.65	∼0.25	∼0.05	0.56
gemini-2.5-flash (G)	∼0.05–0.10	∼0.20	0.70+	0.91
gemini-2.5-pro (No G)	0.10–0.20	∼0.30	0.60	0.89
gemini-2.5-pro (G)	< 0.01	∼0.25	0.75+	0.98
EHRSQL-MIMIC-III				
gpt-5.2 (No G)	∼0.90+	∼0.10–0.20	∼0.05	0.84–0.86
gpt-5.2 (G)	∼0.20–0.30	∼0.40+	0.50+	0.95+
gemini-2.5-flash (No G)	∼0.40–0.50	∼0.30	∼0.20	0.84
gemini-2.5-flash (G)	∼0.05–0.10	∼0.30	0.70+	0.90+
gemini-2.5-pro (No G)	∼0.20–0.30	∼0.30	∼0.30	0.93–0.94
gemini-2.5-pro (G)	< 0.05	∼0.30	0.75+	0.98
medgemma-1.5-4b-it (No G)	∼0.12	∼0.88	∼0.00	0.85
medgemma-1.5-4b-it (G)	∼0.17	∼0.21	∼0.00	0.76
SPIDER				
gpt-5.2 (No G)	0.80–0.90	∼0.15	∼0.05–0.10	0.17
gpt-5.2 (G)	0.15–0.25	∼0.25	0.55+	0.65+
gemini-2.5-flash (No G)	0.85–0.90	∼0.10	∼0.05	0.14
gemini-2.5-flash (G)	∼0.05–0.10	∼0.20	0.70+	0.85+
gemini-2.5-pro (No G)	0.70–0.80	∼0.20	∼0.10–0.15	0.33
gemini-2.5-pro (G)	< 0.05	∼0.25	0.75+	0.90+

Table 4: Failure mode analysis across datasets comparing models with and without query guidance (G). Guidance reduces non-SQL generation failures, especially on eICU, MIMIC-III, and SPIDER, but residual errors remain dominated by incorrect table selection. This indicates that guidance improves task framing and generation reliability, while schema grounding remains a bottleneck.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction consistently describe schema guidance as a structured representation for managing context in agentic SQL generation and as a mechanism for search-space reduction. The stated contributions, including the schema-guidance construction pipeline, its interpretation as a compressed context representation, and its role in improving SQL generation, are aligned across sections. The empirical scope—covering SPIDER, open clinical benchmarks (MIMIC-III and eICU), and a closed institutional dataset—is consistently reflected in both the abstract and introduction. Differences in emphasis do not affect the accuracy or scope of the stated claims.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated Limitations paragraph that explicitly discusses key assumptions and constraints of the proposed approach. These include the reliance on benchmark summaries rather than query-level traces, which limits detailed error analysis; the dependence on structural SQL metrics without full validation of clinically meaningful outputs; and potential system-level confounds arising from the broader agentic pipeline. The discussion also reflects on the scope of empirical evaluation, including limitations related to structured relational schemas and the lack of validation in noisier real-world settings (e.g., incomplete schemas, free-text-heavy databases, or hybrid data models). Additionally, the authors acknowledge important deployment challenges such as interpretability, auditability, and human oversight. Overall, the paper provides a clear and transparent account of the main factors that may affect performance and generalization.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.

- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper states its theoretical claims with explicit assumptions and cross-references them to the appendix. The submodular coverage guarantee is stated in Remark **??** with the required monotonicity, non-negativity, submodularity, and budget assumptions, and references the standard greedy approximation result. The search-space reduction result is stated as a proposition in Appendix B.1, with the recall-preservation and monotone decoder-error assumptions made explicit.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the schema-guidance construction pipeline, runtime agent behavior, datasets, evaluation metrics, comparison protocol, model configurations, decoding settings, and compute setup. The appendix further details the runtime prompt structure and agentic SQL generation process, including tool-calling behavior, schema-loading mechanisms, caching, and reset behavior between benchmark examples. Additional implementation details, including fully instantiated prompts and generated guidance artifacts, are provided in the supplemental material and accompanying code.

   While the institutional dataset cannot be released due to privacy and governance constraints, the same evaluation protocol is applied to publicly available benchmarks (SPIDER, MIMIC-III, and eICU), enabling independent verification of the main claims.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper provides anonymized code, prompts, and configuration details in the supplemental material to support reproducibility. The submission includes the schema and query guidance construction pipeline, runtime agent implementation, and evaluation scripts. All experiments are conducted on publicly available datasets (SPIDER, MIMIC-III, and eICU), for which access instructions are provided. The institutional dataset cannot be released due to privacy and governance constraints, but it is used only as an additional evaluation setting.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies datasets, benchmark construction, evaluation metrics, comparison protocols, model configurations, decoding settings, agent control parameters, retry policies, and compute setup. As the system is evaluated in an inference-only setting, no training procedure or optimizer is required. The appendix further details runtime agent behavior, schema-loading tools, and prompt structure across examples. Additional implementation details, including fully instantiated prompts and generated guidance artifacts, are provided in the supplemental material and accompanying code.

   Some low-level implementation details, such as exact fully instantiated prompts and generated guidance artifacts, are summarized rather than exhaustively enumerated. These details do not prevent interpretation of the reported results, and additional implementation details will be provided with the released code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports results as mean $\pm$ standard error across aggregated evaluations over datasets and runs, as shown in the tables. Each evaluation corresponds to a repeated inference under identical experimental conditions, capturing variability arising from model inference and agent execution.

   The reported error bars correspond to the standard error of the mean (SEM), computed as the sample standard deviation divided by $\sqrt{n}$, where n is the number of aggregated dataset–run evaluations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).

- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Experiments were conducted using API-based inference for GPT and Gemini models via Google Cloud Workbench notebooks; these models were accessed through external cloud services and were not executed locally. As a result, hardware specifications for these models are abstracted by the provider, and compute is primarily determined by API latency and throughput.

   In contrast, MedGemma-1.5-4B-IT was executed locally on a Google Cloud VM instance with a g2-standard-16 configuration (16 vCPUs, 64 GB RAM) and a single NVIDIA L4 GPU (24 GB VRAM).

   Each dataset evaluation required on the order of hours, depending on the number of queries. These specifications reflect the primary computational requirements needed to reproduce the reported experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research complies with the NeurIPS Code of Ethics. The study uses de-identified or publicly available datasets where applicable, and the closed institutional dataset is handled in accordance with appropriate privacy and data governance practices. The work does not involve human subject experimentation or deployment in a real-world clinical setting.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses positive societal impacts, including more reliable and interpretable cohort construction, deep phenotyping, and outcome ascertainment in clinical research. It also discusses potential negative impacts, including misleading clinical analyses from incorrect SQL generation, misuse of automated cohort construction without oversight, bias propagation, and privacy or governance risks when applied to sensitive clinical data. The

paper further identifies mitigation strategies such as human-in-the-loop review, validation checks before execution, and deployment within controlled environments with appropriate data governance.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release a high-risk model, pretrained language model, image generator, or scraped dataset requiring special safeguards. The closed institutional clinical dataset is not publicly released, and the work focuses on a schema-guidance construction method evaluated with existing models rather than releasing a deployable model artifact.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper cites the original sources for the existing datasets used, including MIMIC-III, eICU, EHRSQL, and Spider-style text-to-SQL benchmarks. Appendix C

explicitly describes licensing and data usage terms: Spider is released under Apache-2.0, EHRSQL under CC-BY-4.0, MIMIC-III and eICU are accessed under credentialed PhysioNet data use agreements, and the closed institutional dataset is not publicly released due to privacy and governance constraints. All datasets are used for research purposes in compliance with their respective licenses and terms of use.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not release new assets such as a public dataset, codebase, or model checkpoint. The closed institutional benchmark is used for evaluation but is not publicly released due to privacy and data governance constraints, as described in Appendix C.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The study does not involve crowdsourcing or prospective experiments with human participants (e.g., no participant recruitment, instructions, or compensation). It is based on retrospective analysis of de-identified clinical and/or publicly available datasets. The use of institutional data was reviewed and approved by an Institutional Review Board (IRB), and all analyses were conducted in compliance with applicable data governance and privacy regulations.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: The study involves retrospective analysis of clinical data. Institutional Review Board (IRB) approval was obtained (IRB No. 26-002996). All data were handled in compliance with relevant privacy and data protection regulations.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Large language models (LLMs) are a core component of both the schema guidance construction pipeline and the runtime SQL generation process. In the construction pipeline, LLMs are used for enrichment during per-table guidance generation. At inference time, LLMs are used within the agent to generate SQL queries conditioned on schema guidance. In addition, LLMs are used for zero-shot derivation of natural language task descriptions in preprocessing of the closed institutional dataset. These uses are described in the Method and Experimental Setup sections.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.