FROM SYNTACTIC TO SEMANTIC: CONTROLLING TEXT-TO-SQL COMPLEXITY VIA SEMANTICALLY ANCHORED TARGETED SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Text-to-SQL failures are often driven not by surface SQL syntax, but by semantic structures that govern how queries are scoped, composed and constrained. We introduce a semantically grounded Depth-Width complexity coordinate, where Depth measures the longest chain of cross-scope dependencies and Width characterizes within-scope combinatorial semantic burden. To operationalize this coordinate for controlled analysis and synthesis, we define structural motifs as atomic, detectable, verifiable, and synthesizable patterns that anchor semantic complexity. We cast synthesis as a generate-and-verify loop with multi-stage quality validation to filter decorative complexity. On Spider-dev and BIRD-dev, our motif taxonomy captures real-world structure with high coverage, reaching 70.6% and 88.0% overall, respectively, and achieving near-complete coverage on high-complexity subsets. Finally, we propose a complexity-controlled targeted synthesis pipeline that injects motifs into SQL skeletons, instantiates parameters via data probing, and produces executable training data annotated with Depth, Width, and motif labels for expansion toward structurally complex semantic regions. Training on validated synthetic data yields consistent execution-accuracy gains, with improvements concentrated in high-complexity regions and in Depth-Width buckets corresponding to these failure-prone structures.

1 INTRODUCTION

Text-to-SQL tasks aim to automatically translate natural language questions into structured query language (SQL) commands. However, while Large Language Models (LLMs) have significantly enhanced average performance, failure cases within these tasks are not uniformly distributed; instead, errors remain heavily concentrated on queries that demand complex semantic structural reasoning (Liu et al., 2025; Gan et al., 2022). Such failures are specifically manifested in intricate patterns such as cross-scope dependencies, join-predicate coupling, and multi-branch conditional logic. Consequently, simply scaling training data without explicitly targeting these underlying semantic structures (Yu et al., 2019) leads to diminishing returns.

Most existing work characterizes complexity using surface-level heuristics such as query length, abstract syntax tree depth (Yu et al., 2018a), or the number of components (Yu et al., 2018b), which are highly sensitive to semantics-preserving rewrites (Fig. 1). They fail to distinguish between semantic complexity, which stems from decision-critical structural choices that influence execution behavior, and syntactic complexity, which often reflects superficial constructs that can be rewritten away under semantics-preserving transformations (Gan et al., 2021b). As a result, data synthesis pipelines guided by such proxies tend to generate queries that look complex but introduce little additional semantic burden, providing weak or misaligned supervision with respect to actual model failure modes.

We introduce a semantically grounded complexity coordinate, **Depth-Width**, where Depth measures cross-scope dependency chains and Width captures within-scope combinatorial burden. Structural motifs—atomic, verifiable patterns such as correlated subqueries and join-predicate coupling—anchor this coordinate in concrete synthesis targets. An LLM generates queries conditioned on target motifs and Depth-Width buckets; executability, motif consistency, and redundancy checks re-

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

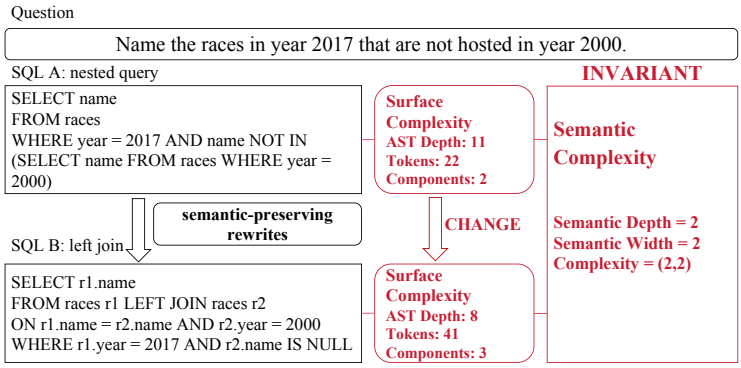

Figure 1: Rewrite sensitivity of surface-level complexity versus invariance of semantic complexity. Two semantically equivalent SQL queries, obtained via common query rewrites, exhibit substantially different surface metrics while their Depth-Width semantic structure remains unchanged.

tain queries whose motifs affect execution results. The accepted instances carry Depth, Width, and motif labels for targeted training.

On SPIDER and BIRD, the motifs cover the vast majority of high-complexity queries. Training on validated synthetic data improves execution accuracy, with larger gains on high-complexity subsets.

This work makes three key contributions: (1) a stable **Depth-Width complexity coordinate** that captures structural reasoning burden beyond surface syntax; (2) a compact set of **verifiable structural motifs** that anchor abstract complexity to concrete, synthesizable SQL patterns; and (3) a **generate-and-verify synthesis pipeline** that produces non-trivial training data with explicit structural complexity annotations.

2 RELATED WORK

Modeling Structural Difficulty. Spider-style hardness labels and component counts (Yu et al., 2018b) are convenient but *rewrite-sensitive* and confound decorative and decision-bearing structure (Gan et al., 2021a). Skeleton-based abstractions (Dong & Lapata, 2018) improve controllability but they primarily capture syntactic shape and do not guarantee that the instantiated query introduces a non-trivial semantic decision.

Structure-Aware Text-to-SQL Models. Structure-aware parsers encode schema relations and compositional structure (Wang et al., 2020; Rubin & Berant, 2021; Xu et al., 2022), and synthetic pretraining can inject grammar priors (Yu et al., 2021). For LLM generation, constrained decoding reduces invalid SQL (Scholak et al., 2021) and other methods mitigate hallucination or enforce constraints (Wang et al., 2018; Raspanti et al., 2025); prompting pipelines further improve in-context performance via task decomposition and systematic prompt design (Pourreza & Rafiei, 2023; Gao et al., 2023). Recent methods use multi-agent collaboration (Wang et al., 2025; Yang et al., 2025) and retrieval augmentation (Thorpe et al., 2024; Li et al., 2025b; Zhang et al., 2023). However, these lines typically ensure *validity/consistency* or improve prompting efficiency, rather than verifying that a produced query embodies a specific semantic-bearing motif.

Data Synthesis for Text-to-SQL. Grammar- or skeleton-driven synthesis improves structural coverage (Yu et al., 2021; Ji et al., 2025), and verification-oriented pipelines improve reliability (Guo et al., 2025). Yet existing synthesis is rarely targeted to empirically observed failure regions with traceable structural labels, and often lacks a non-triviality check that filters executable-but-decorative samples.

3 PRELIMINARIES

3.1 PROBLEM FORMULATION

Under the standard execution-based setting (Zhong et al., 2017) with dataset $\mathcal{D} = \{(x, s, q)\}$ and model $f_\theta(x, s) = \hat{q}$, we aim to design a complexity representation $\mathcal{C}(q)$ that is (i) **semantically aligned** with execution-level structural decisions, (ii) **operational** for stratification and comparison, and (iii) **actionable** as a control signal for generating verifiable training data.

3.2 SEMANTIC COMPLEXITY

We define a semantically anchored complexity coordinate $\mathcal{C}(Q) = (D(Q), W(Q))$ for a SQL query Q.

Depth $D(Q)$ measures cross-scope dependency depth: the longest chain of semantic scopes (subqueries, CTEs, correlated references) that must be resolved sequentially. Formally, we construct a scope-dependency DAG over all `SELECT`-scopes and define

$$D(Q) = \max_{\pi \in \mathrm{Paths}(G_D)} |\pi|.$$

Width $W(Q)$ measures within-scope combinatorial burden: the largest coupled decision space at any single semantic level, arising from join coupling, Boolean fanout, `CASE` branching, or aggregation grouping. We take the max across sub-dimensions and scopes:

$$W(Q) = \max_{s \in \mathcal{S}(Q)} \max_{k \in \{\mathrm{bool,join,case,agg}\}} W_k(s).$$

Max aggregation isolates the dominant bottleneck rather than inflating complexity by accumulating decorative components. Empirically, max-based Width correlates more strongly with model failure rate than sum-based alternatives (Pearson r: $W_{\max}$=0.253 vs. W_{sum}=0.202 on BIRD-dev, $p < 10^{-20}$).

Both dimensions are anchored at the `SELECT`-scope level, reducing sensitivity to semantics-preserving rewrites. On 2,847 output-equivalent rewrite pairs, 87.3% retain identical (D, W) values (D unchanged in 93.1%, W in 89.6%), confirming practical stability. Full algorithmic details (scope-graph construction, longest-path computation, width sub-dimensions) are given in Appendix A. We validate the coordinate's predictive power (correlation with failure rate) and its utility as a synthesis control signal in §5.

3.3 STRUCTURAL MOTIFS

To connect semantic complexity with observable failures, we introduce structural motifs: minimal semantics-bearing patterns that anchor diagnosis, targeted synthesis, and verification. Each motif corresponds to an execution-semantic decision unit, whose errors typically surface as exec_wrong cases, that is, executable queries with incorrect denotations under database semantics.

Motif	Executor Family	ΔD	ΔW	Axis
SUBQUERY	Scan	1.03	0.09	D
PREDICATE_JOIN	Join	0.01	0.12	W
AGGREGATION	Materialization	0.00	0.01	W
QUALIFY	Materialization	1.00	0.50	D/W
DISTINCT	Materialization	0.00	0.00	W
OR	Control	0.01	0.58	W
CASE	Control	0.00	0.73	W

Table 1: Structural motifs grouped by executor-operator families, with ablation-induced shifts on the Depth-Width coordinate (§4.1). Axis indicates the primary complexity dimension associated with each motif.

We organize motifs into scan, join, materialization, and control families following the GaussDB(DWS) executor-operator taxonomy (Huawei Cloud Computing Technology Co., 2025). Our experiments use single-node SQLite and exclude distributed physical operators.

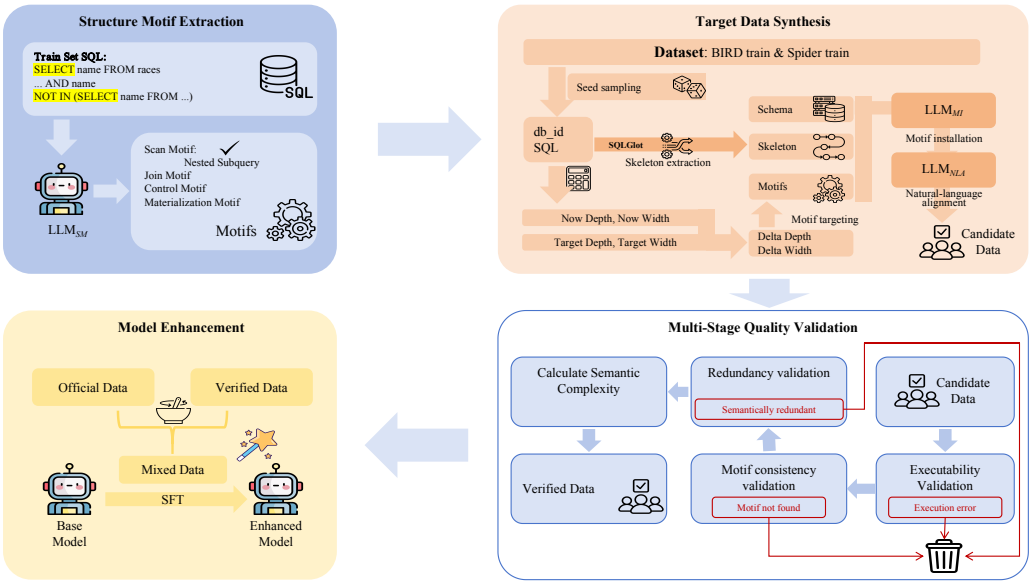

Figure 2: Overview of our method: (1) structure motif extraction, (2) targeted data synthesis, (3) multi-stage quality validation, and (4) model enhancement via fine-tuning on mixed data.

Criteria for fundamental motifs. A motif m is considered *fundamental* only if it represents an execution-semantic decision whose presence can alter the query denotation on at least some database instances, rather than a purely syntactic or stylistic variation. Moreover, m must admit a deterministic and automatable recognition rule in the form of a structural or semantic signature derivable from static analysis of the parsed query. Finally, m is required to be non-redundant at the instance level: for any accepted query instance, weakening or removing m must change the observable query output under the evaluation protocol. These criteria define the properties a motif must satisfy in principle; Stage 3 of our validation pipeline (§4.3) enforces the non-redundancy criterion on each synthesized instance at generation time.

Table 1 summarizes the motif tags used in this work together with their deterministic structural or semantic signatures. These tags are consumed by the targeted synthesis pipeline (§4.2) and by the failure attribution analyses (§5). Due to space limits, the full motif dictionary including formal definitions, deterministic detection signatures, and canonical SQL examples is provided in Appendix B.

4 Method

4.1 Structure Motif Extraction

For closed-loop analysis and intervention, we prefer a small motif vocabulary that still covers most structurally hard queries. Specifically, we use a large language model to perform candidate motif ablation: given a SQL query containing a candidate motif, the LLM rewrites the query with that motif removed or weakened. If the removal leads to a change in semantic Depth or Width, we include the motif in the candidate set. Note that this LLM-assisted ablation is used only during the one-time motif vocabulary construction phase; at synthesis and validation time, motif *detection* is performed by the deterministic rule-based recognizer described in §4.3, while the LLM serves as a proposal mechanism for SQL generation (§4.2) and rewrite generation (Stage 3b). To get a small motif vocabulary, we apply a greedy maximum-coverage (set cover) procedure: starting from an empty set, we iteratively add the motif that covers the largest number of previously uncovered queries in $\mathcal{Q}_{\text{high}}$ until the marginal gain saturates. Empirically, the marginal gain quickly saturates (Fig. 3), and a small core set of motifs reaches near-complete coverage on $\mathcal{Q}_{\text{high}}$. The remaining uncovered portion is treated as *residual* and analyzed as a roadmap for extending the motif inventory,

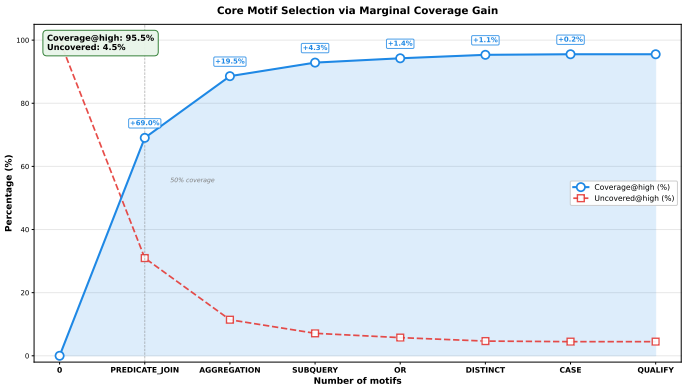

Figure 3: Greedy maximum-coverage selection of structural motifs on the high-complexity subset $\mathcal{Q}_{\text{high}}$: marginal gain quickly saturates, yielding a small core motif set with high coverage.

including schema linking (Li et al., 2023; Guo et al., 2019; Cao et al., 2021), value reasoning, and NULL/multiplicity nuances.

We perform coverage selection on $\mathcal{Q}_{\text{high}}$ rather than the full distribution to prevent frequent low-complexity patterns from dominating the motif inventory and to keep the anchors aligned with high-complexity failure cases.

We relate motifs to the Depth-Width coordinate through *motif-complexity attribution via ablation*. For queries containing a motif o, we weaken or remove o and measure the resulting changes in ΔD and ΔW, preserving executability whenever possible. Table 1 reports the observed deltas.

Notably, some motifs may exhibit near-zero ΔD or ΔW under this procedure. This is expected given our abstraction choices in §3.2: D measures cross-scope dependency depth (longest path), and W uses a max-aggregation to highlight the dominant within-scope bottleneck. Therefore, a motif can be decision-bearing yet show limited marginal change when (i) it is dominated by a larger bottleneck in the same query, or (ii) it strongly co-occurs with already-present motifs. We retain such motifs as anchors because they remain indispensable for targeted synthesis control and fine-grained failure attribution, even when their marginal effect on (D, W) is small.

4.2 Targeted Data Synthesis

We present a controlled synthesis pipeline that generates training instances whose difficulty is (i) *bucket-controlled* by a target $(D^\star, W^\star)$, (ii) *motif-consistent*, and (iii) *non-redundant*. The pipeline uses an LLM as a proposal mechanism while the database engine and validation predicates (§4.3) serve as oracles rejecting spurious candidates. The exact prompt template is shown in Appendix C.

Given a seed instance $(\texttt{db_id}, x, Q_{\text{seed}})$ sampled from real training data, the pipeline outputs a traceable record containing the synthesized pair $(\hat{x}, Q)$, detected motifs, the assigned (D, W) label, and per-stage validation outcomes.

Synthesis procedure. For a target bucket $(D^\star, W^\star)$, we synthesize one sample via the following steps.

1. **Seed sampling.** Sample a real instance $(\texttt{db_id}, x, Q_{\text{seed}})$ from the training split.
2. **Skeleton extraction.** We use SQLGlot to parse Q_{seed} into an AST and replace all identifiers (tables, columns, literals) with typed placeholders, yielding a structural skeleton.
3. **Motif targeting.** Sample a target motif set $\mathcal{M}^\star$ from the motif inventory, conditioned on the desired complexity direction (Depth-driving vs. Width-driving).
4. **Motif installation.** We prompt the LLM with (i) the database schema and integrity constraints, (ii) the seed skeleton, and (iii) the target motifs $\mathcal{M}^\star$. We provide value candidates and joinability hints from lightweight database probing, including frequent categorical val-

ues, min/max ranges, and non-empty join keys. The LLM is restricted to selecting constants and join paths from these admissible sets to generate the new SQL Q.

5. **Natural-language alignment.** For each validated SQL Q, we generate its paired question $\hat{x}$ conditioned on the schema and Q, and apply lightweight coverage checks to ensure that $\hat{x}$ references only schema elements actually used by Q.

4.3 Multi-stage Quality Validation

We apply a multi-stage validation pipeline (Figure 2, Stage 3) to filter semantically spurious candidates (Wang et al., 2023).

Stage 1: Executability. Q must execute without errors on the target DBMS and pass non-degeneracy checks (no trivial constants, no all-`NULL` columns, controlled empty-result rate).

Stage 2: Motif Consistency. A deterministic rule-based recognizer extracts $\mathrm{Motif}(Q)$ via static analysis of the parsed AST. We accept Q only if $\mathcal{M}^\star \subseteq \mathrm{Motif}(Q)$.

Stage 3: Redundancy. We check non-redundancy at two levels: **(a)** For each injected motif m, we ablate it to obtain Q^{-m} and re-execute; if outputs are identical, m is redundant and the sample is rejected. **(b)** We generate semantics-preserving rewrites and reject Q if any output-equivalent rewrite Q' satisfies $D(Q') \leq D(Q)$ and $W(Q') \leq W(Q)$ with at least one strict inequality (Pareto dominance).

4.4 Model Enhancement

We enhance Text-to-SQL models by supervised fine-tuning (SFT) (Li et al., 2025a; Feng et al., 2021; Sun et al., 2023) on verified synthetic instances from our pipeline (§4.2–§4.3). The training corpus combines the original benchmark training split and validated synthetic samples: $\mathcal{D}_{\text{mix}} = \mathcal{D}_{\text{orig}} \cup \mathcal{D}_{\text{syn}}$.

Complexity-bucket sampling. Rather than sampling uniformly, we partition $\mathcal{D}_{\text{mix}}$ into disjoint buckets $\mathcal{B}_{d,w} = \{(x, s, Q) \mid D = d,\, W = w\}$ and draw examples according to

$$P(\mathcal{B}_{d,w}) \propto \alpha_{d,w}, \qquad \sum_{d,w} \alpha_{d,w} = 1,$$

where $\alpha_{d,w}$ controls the relative contribution of each complexity region. In our experiments, we set $\alpha_{d,w}$ proportional to the natural frequency of each bucket in $\mathcal{D}_{\text{mix}}$, which up-weights high-complexity buckets relative to a uniform-over-instances baseline since synthetic data is concentrated there. Total training updates are kept fixed across settings for fair comparison.

Training objective. We use standard next-token negative log-likelihood on the target SQL sequence, conditioned on a serialized prompt encoding the schema and natural-language question. Each synthetic instance additionally carries (D, W) and motif labels, enabling complexity-aware curriculum learning (Zhu et al., 2025).

5 Experiments

We evaluate our framework through two research questions that correspond to the closed-loop formulation in §3.1: **(RQ1)** *Coverage:* Do the seven core motifs cover real-world structurally complex queries? **(RQ2)** *Actionability:* Can motifs and complexity-controlled synthesis deliver attributable training gains, especially on high-complexity buckets?

5.1 Experimental Setup

Benchmarks. We use two execution-centric Text-to-SQL benchmarks. **Spider-dev** serves as a canonical cross-domain evaluation for complexity modeling and attribution analysis. **BIRD-dev** introduces noisier schema linking and more realistic database interfaces, enabling a robustness check under harder schema conditions. Table 2 reports dataset statistics together with the distribution of queries across Depth-Width buckets.

Complexity and the high-complexity subset. We compute semantically-anchored structural complexity $\mathcal{C}(Q) = (D(Q), W(Q))$ as defined in §3.2. To avoid analyses being dominated by frequent low-complexity instances, we define the high-complexity subset

$$\mathcal{Q}_{\text{high}} = \{Q \mid D(Q) \geq 2 \text{ or } W(Q) \geq 2\}. \tag{1}$$

We report `coverage@high` on $\mathcal{Q}_{\text{high}}$, select core motifs via high-complexity coverage in §4.1, and analyze training gains by bucket in §5.3.

Models and training. We evaluate Qwen2.5-Coder-3B-Instruct and Qwen2.5-Coder-7B-Instruct (Hui et al., 2024) as the backbone LLMs for our synthesis pipeline. The 3B model is trained via full fine-tuning, while the 7B model uses LoRA adaptation. All settings share the same evaluation protocol and decoding constraints unless noted otherwise.

Metric and failure taxonomy. Our primary metric is execution accuracy (EX): a prediction is correct if the predicted SQL executes successfully and returns the same result as the gold query on the benchmark database. For diagnosis, each incorrect prediction is assigned to exactly one of: `invalid_sql` (parse/compile failure), `exec_error` (runtime error), or `exec_wrong` (executable but semantically incorrect), in this priority order to ensure exclusivity.

Statistic	BIRD-dev	Spider-dev
Samples	1,534	1,034
Databases	11	20
Avg. tables/DB	6.8	4.0
Avg. columns/DB	72.5	21.9
Avg. SQL tokens	42.0	22.5
Avg. tables/query	1.95	1.51
Avg. joins/query	0.93	0.50
Avg. Depth (D)	1.09	1.08
Avg. Width (W)	1.66	1.13

Table 2: Dataset statistics.

5.2 RQ1: Coverage of Real Structural Complexity

We first test whether the seven core motif tags are sufficiently prevalent in real benchmark queries to serve as semantic anchors. We report overall coverage (fraction of queries containing at least one detected motif) and Coverage@high (restricted to $\mathcal{Q}_{\text{high}}$).

Table 3: Coverage of the seven core motifs on real benchmarks. High: $D \geq 2$ or $W \geq 2$. Average is weighted by the corresponding sample counts.

Dataset	N	Coverage	N_{high}	Coverage@high
BIRD-dev	1,534	88.01%	899	98.42%
Spider-dev	1,034	70.60%	196	100.00%
Average	—	**81.00%**	—	**98.70%**

Coverage is higher on the high-complexity subsets of both benchmarks, supporting the use of these motifs as anchors for complexity-controlled synthesis.

5.3 RQ2: Targeted Synthesis and Training Gains

Failure attribution as motivation. To identify which motifs should be prioritized as synthesis targets, we measure the relative risk of each motif: $\text{RiskRatio}(o) = \Pr(\text{fail} \mid o \in Motif(Q))/\Pr(\text{fail})$. As shown in Figure 4, a small subset of motifs exhibits substantially elevated risk ratios, confirming that execution-level failures concentrate on specific decision-bearing structures. We therefore target these high-risk motifs in our synthesis pipeline.

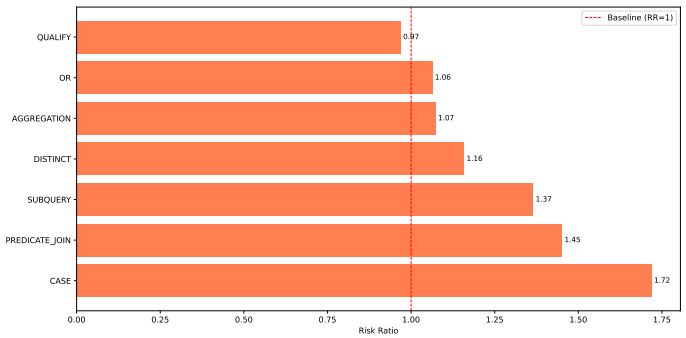

Figure 4: Risk Ratio and Error Mass for top motifs. A small subset of motifs accounts for disproportionate failure mass.

Training settings. We compare: **Baseline** (original training data only) and **Controlled Synth (Ours)**. All settings match synthesis scale, training budget, and evaluation protocol.

	(Depth,Width) buckets						Aggregate	
Setting	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)	(2,4)	All	High
N	838	115	2	65	12	2	1034	196
3B (FFT)								
Baseline	69.9	46.1	50.0	49.2	16.7	50.0	65.3	45.4
Ours	**71.5**	**53.0**	50.0	**52.3**	**25.0**	50.0	**67.6**	**51.0**
Δ (Ours $-$ Base)	+1.6	+6.9	+0.0	+3.1	+8.3	+0.0	+2.3	+5.6
7B (LoRA)								
Baseline	82.1	60.9	0.0	64.6	41.7	0.0	77.9	59.7
Ours	**83.8**	**70.4**	0.0	**76.9**	**58.3**	0.0	**81.2**	**70.4**
Δ (Ours–Base)	+1.7	+9.5	+0.0	+12.3	+16.6	+0.0	+3.3	+10.7

Table 4: Results on Spider-dev. Execution accuracy (%) by Depth-Width buckets. High: $(d \geq 2) \vee (w \geq 2)$.

Results on Spider. Controlled synthesis improves overall EX for both backbones, with larger gains on high-complexity buckets (Table 4).

Comparison with synthesis baselines. To isolate the contribution of Depth-Width control from simply having more validated data, we compare with recent synthesis methods under the same 7B LoRA setting, 7k-sample budget, and evaluation protocol.

Method	**All**	**High**
Original Data	77.9	59.7
OmniSQL (Li et al., 2025a)	79.3	61.2
Skeleton-based (Ji et al., 2025)	**81.6**	64.3
Ours	81.2	**70.4**

Table 5: Comparison with synthesis baselines on Spider-dev (7B LoRA). Our method achieves competitive overall EX while delivering substantially larger gains on high-complexity queries.

Table 5 shows that while overall performance is competitive with strong baselines, our method delivers significantly larger gains in the high-complexity region (+10.7pp vs. +4.6pp for Skeleton-based), confirming that improvements arise from targeted complexity control rather than data volume alone. The slight overall gap with Skeleton-based (81.2 vs. 81.6) reflects our deliberate allocation of synthesis budget toward high-complexity buckets, which constitute only 19% of evaluation queries; gains on this minority are diluted in the aggregate metric.

Results on BIRD. We report BIRD-dev results using Qwen2.5-Coder-3B-Instruct with full fine-tuning.

	$W=1$	$W=2$	$W=3$	$W=4$	$W=5$
$D=1$	N=635 58.4→59.8 **+1.4**	N=646 34.8→36.1 **+1.2**	N=115 32.2→36.5 **+4.3**	N=14 35.7→50.0 **+14.3**	N=2 0.0→0.0 **+0.0**
$D=2$	N=53 39.6→47.2 **+7.5**	N=50 24.0→34.0 **+10.0**	N=6 0.0→16.7 **+16.7**	N=2 50.0→50.0 **+0.0**	–
$D=3$	N=3 0.0→0.0 **+0.0**	N=4 0.0→25.0 **+25.0**	N=1 0.0→0.0 **+0.0**	N=3 33.3→33.3 **+0.0**	–

Table 6: Results on BIRD-dev. Execution accuracy (%) by Depth-Width buckets. Each cell reports N, Baseline→Ours, and absolute improvement (Δ).

BIRD-dev results extend the gains to noisier schemas, with improvements on both All and High (Table 6).

5.4 Ablation: Executability Is Not Enough

We conduct a fine-grained ablation to isolate the contribution of each validation stage in controlled synthesis. We incrementally add stages to an **Exec-only** baseline (Validation 1 only): **+ Motif Consistency** (Stage 2) and **+ Redundancy** (Stage 3, full pipeline). All settings match synthesis scale, training budget, and evaluation protocol on Spider-dev using Qwen2.5-Coder-3B-Instruct with full fine-tuning.

Table 7 shows that Exec-only synthesis reduces EX from 65.28 to 62.67 on All and from 45.40 to 40.80 on High. Adding motif consistency raises EX to 64.80 All / 45.60 High; redundancy validation further improves it to 67.60 All / 51.02 High, exceeding the original-data baseline.

Metric	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)	(2,4)	**All**	**High**
N	838	115	2	65	12	2	1034	196
Exec-only (EX)	67.78	46.09	50.00	36.92	8.33	50.00	62.67	40.80
+ Motif Consistency	69.33	49.57	50.00	44.62	16.67	50.00	64.80	45.60
+ Redundancy (Full)	71.48	53.04	50.00	52.31	25.00	50.00	67.60	51.02

Table 7: Fine-grained ablation on Spider-dev (3B FFT). Each row adds one validation stage.

6 Conclusion

We introduce a semantically grounded Depth–Width coordinate and verifiable structural motifs for complexity-controlled Text-to-SQL synthesis. Experiments on Spider-dev and BIRD-dev show that training on validated synthetic data yields larger gains on high-complexity subsets. Validation ablations show that motif consistency and redundancy checks improve training outcomes over executability checks alone.

Limitations

The motif inventory excludes residual errors involving schema linking, value reasoning, and NULL semantics. Experiments use SQLite; the Depth-Width coordinate and motifs may require adaptation to engines such as PostgreSQL or Spark SQL. Evaluation is limited to 3B and 7B models; transfer to larger or closed-source models remains untested.

AI USE STATEMENT

Generative AI tools were used to assist with writing and language polishing, to propose SQL motif ablations and validation rewrites, and to generate synthetic Text-to-SQL training examples through the controlled synthesis pipeline described in Section 4 and Appendix C. The generated examples are filtered using the multi-stage quality validation procedure described in the paper.

REPRODUCIBILITY STATEMENT

Code and reproduction instructions are provided in the supplementary material. Appendix A specifies the Depth–Width computation, Appendix B lists the motif dictionary, and Appendix C provides the synthesis prompt. The experimental setup is described in Section 5.1, and synthesis costs are reported in Appendix D.

REFERENCES

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

Li Dong and Mirella Lapata. Coarse-to-fine decoding for neural semantic parsing. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 731–742, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1068. URL https://aclanthology.org/P18-1068/.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. A survey of data augmentation approaches for NLP. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 968–988, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.84. URL https://aclanthology.org/2021.findings-acl.84/.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. Towards robustness of text-to-SQL models against synonym substitution. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 2505–2515, Online, August 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.195. URL https://aclanthology.org/2021.acl-long.195/.

Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John Drake, and Qiaofu Zhang. Natural SQL: Making SQL easier to infer from natural language specifications. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 2030–2042, Punta Cana, Dominican Republic, November 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.174. URL https://aclanthology.org/2021.findings-emnlp.174/.

Yujian Gan, Xinyun Chen, Qiuping Huang, and Matthew Purver. Measuring and improving compositional generalization in text-to-SQL via component alignment. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 831–843, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.62. URL https://aclanthology.org/2022.findings-naacl.62/.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yuzhe Qian, Bolin Ding, and Jingren Zhou. Text-to-sql empowered by large language models: A thorough evaluation. *arXiv preprint arXiv:2308.15363*, 2023.

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

Yu Guo, Dong Jin, Shenghao Ye, Shuangwu Chen, Jianyang Jianyang, and Xiaobin Tan. Sqlforge: Synthesizing reliable and diverse data to enhance text-to-sql reasoning in llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 8441–8452, 2025.

Ltd. Huawei Cloud Computing Technology Co. *GaussDB(DWS) Technical Principles Analysis*, 2025. URL https://res-static.hc-cdn.cn/cloudbu-site/china/zh-cn/liuzhongzhengdewenjian/1715322842700303739.pdf.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Yuchen Ji, Bo Xu, Jie Shi, Jiaqing Liang, Deqing Yang, Yu Mao, Hai Chen, and Yanghua Xiao. Skeletons matter: Dynamic data augmentation for text-to-query. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1214–1236, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.64. URL https://aclanthology.org/2025.emnlp-main.64/.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 13067–13075, 2023.

Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, Hong Chen, and Cuiping Li. Omnisql: Synthesizing high-quality text-to-sql data at scale. *ArXiv*, abs/2503.02240, 2025a. URL https://api.semanticscholar.org/CorpusID:276774742.

Xiuwen Li, Qifeng Cai, Yang Shu, Chenjuan Guo, and Bin Yang. Aid-sql: Adaptive in-context learning of text-to-sql with difficulty-aware instruction and retrieval-augmented generation. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, pp. 3945–3957. IEEE, 2025b.

Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuxin Zhang, Ju Fan, Guoliang Li, Nan Tang, and Yuyu Luo. A survey of text-to-sql in the era of llms: Where are we, and where are we going? *IEEE Transactions on Knowledge and Data Engineering*, 2025.

Mohammadreza Pourreza and Davood Rafiei. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems*, 36:36339–36348, 2023.

Federico Raspanti, Tanir Ozcelebi, and Mike Holenderski. Grammar-constrained decoding makes large language models better logical parsers. In Georg Rehm and Yunyao Li (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 485–499, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-288-6. doi: 10.18653/v1/2025.acl-industry.34. URL https://aclanthology.org/2025.acl-industry.34/.

Ohad Rubin and Jonathan Berant. SmBoP: Semi-autoregressive bottom-up semantic parsing. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 311–324, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.29. URL https://aclanthology.org/2021.naacl-main.29/.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 9895–9901, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.779. URL https://aclanthology.org/2021.emnlp-main.779/.

Ruoxi Sun, Sercan Ö Arik, Alex Muzio, Lesly Miculicich, Satya Gundabathula, Pengcheng Yin, Hanjun Dai, Hootan Nakhost, Rajarishi Sinha, Zifeng Wang, et al. Sql-palm: Improved large language model adaptation for text-to-sql (extended). *arXiv preprint arXiv:2306.00739*, 2023.

Dayton G Thorpe, Andrew J Duberstein, and Ian A Kinsey. Dubo-sql: Diverse retrieval-augmented generation and fine tuning for text-to-sql. *arXiv preprint arXiv:2404.12560*, 2024.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 7567–7578, 2020.

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 Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 540–557, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.36/.

Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. Robust text-to-sql generation with execution-guided decoding. *arXiv preprint arXiv:1807.03100*, 2018.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13484–13508, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.754. URL https://aclanthology.org/2023.acl-long.754/.

Kuan Xu, Yongbo Wang, Yongliang Wang, Zihao Wang, Zujie Wen, and Yang Dong. SeaD: End-to-end text-to-SQL generation with schema-aware denoising. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 1845–1853, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.141. URL https://aclanthology.org/2022.findings-naacl.141/.

Haolin Yang, Jipeng Zhang, Zhitao He, and Yi R Fung. Mars-sql: A multi-agent reinforcement learning framework for text-to-sql. *arXiv preprint arXiv:2511.01008*, 2025.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1653–1663, Brussels, Belgium, October-November 2018a. Association for Computational Linguistics. doi: 10.18653/v1/D18-1193. URL https://aclanthology.org/D18-1193/.

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. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018*

Conference on Empirical Methods in Natural Language Processing, pp. 3911–3921, Brussels, Belgium, October-November 2018b. Association for Computational Linguistics. doi: 10.18653/v1/D18-1425. URL https://aclanthology.org/D18-1425/.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. SParC: Cross-domain semantic parsing in context. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4511–4523, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1443. URL https://aclanthology.org/P19-1443/.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Michihiro Yasunaga, Dragomir Xue, et al. Grappa: Grammar-augmented pre-training for table semantic parsing. In *International Conference on Learning Representations (ICLR)*, 2021.

Kun Zhang, Xiexiong Lin, Yuanzhuo Wang, Xin Zhang, Fei Sun, Cen Jianhe, Hexiang Tan, Xuhui Jiang, and Huawei Shen. ReFSQL: A retrieval-augmentation framework for text-to-SQL generation. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 664–673, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.48. URL https://aclanthology.org/2023.findings-emnlp.48/.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

Xiaohu Zhu, Qian Li, Lizhen Cui, and Yuntao Du. Learning SQL like a human: Structure-aware curriculum learning for text-to-SQL generation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 3545–3559, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.190. URL https://aclanthology.org/2025.findings-emnlp.190/.

A Depth-Width Computation Details

This appendix formalizes the computation of the *Depth* and *Width* metrics used throughout the paper. Both metrics are defined over the execution-semantic structure of a SQL query rather than surface syntax, and are computed deterministically from the abstract syntax tree (AST) produced by a standard SQL parser.

Preliminaries. Given a SQL query Q, we parse it into an AST and identify all `SELECT` scopes, including: (i) the outermost query, (ii) bodies of Common Table Expressions (CTEs), and (iii) bodies of scalar, predicate, or derived-table subqueries. Let $S = \{s_0, s_1, \ldots, s_{n-1}\}$ denote the set of scopes, with s_0 the outermost scope.

Depth. We construct a *scope dependency graph* $G = (V, E)$, where each node corresponds to a scope in S. A directed edge $(u \to v)$ indicates that scope v semantically depends on the result of scope u. Two dependency types are considered: (1) *subquery nesting*, where an inner query must be evaluated before its enclosing scope; and (2) *CTE references*, where a scope references a named CTE defined elsewhere. By SQL semantics, G is a directed acyclic graph.

The Depth $D(Q)$ is defined as the length of the longest path in G (measured in number of scopes), capturing the maximum chain of cross-scope dependencies that must be resolved sequentially.

Width. Width measures the maximum *within-scope* combinatorial decision burden. For each scope $s \in S$, we compute five orthogonal width sub-dimensions: (i) boolean predicate fanout (`AND`/`OR`), (ii) conditional branching via `CASE`, (iii) grouping dimensionality (`GROUP BY`), and (iv) join-graph degree. The Width of a query is the maximum value across all sub-dimensions and all scopes, with a minimum of 1.

Algorithm 1: ComputeDepth

Input: SQL query Q
Output: Depth D

1. Parse Q into AST and collect all SELECT scopes S
2. Build dependency graph $G = (V, E)$ from subquery nesting and CTE references
3. Initialize $\text{dist}(v) = 1$ for all $v \in V$
4. Perform topological sort on G
5. For each edge $(u \to v)$ in topological order:
   $\text{dist}(v) \leftarrow \max(\text{dist}(v), \text{dist}(u) + 1)$
6. Return $D = \max_{v \in V} \text{dist}(v)$

Algorithm 2: ComputeWidth

Input: SQL query Q
Output: Width W

1. Parse Q into AST and collect all SELECT scopes S
2. Initialize $W \leftarrow 1$
3. For each scope $s \in S$:
   Compute $W_{\text{bool}}(s)$ from WHERE/HAVING boolean fanout
   Compute $W_{\text{case}}(s)$ from CASE branch counts
   Compute $W_{\text{group}}(s)$ as number of GROUP BY columns
   Compute $W_{\text{join}}(s)$ as max degree of join graph
   $W \leftarrow \max\big(W, W_{\text{bool}}, W_{\text{case}}, W_{\text{group}}, W_{\text{join}}\big)$
4. Return W

B Complete Motif Dictionary

This appendix presents the complete dictionary of *structural motifs* used in our Depth–Width framework.

M1: SUBQUERY This motif captures the introduction of a new `SELECT` scope nested within an outer query, including correlated subqueries, `IN/NOT IN`, `EXISTS`, and scalar subqueries. Detection flags the presence of any `Subquery` or `Exists` node in the AST. Each subquery introduces a new semantic scope whose result must be resolved before the outer predicate, contributing primarily to Depth.

```
SELECT e.name
FROM employees e
WHERE e.salary >
  (SELECT AVG(s.salary)
   FROM employees s
   WHERE s.dept_id = e.dept_id);
```

M2: PREDICATE_JOIN This motif identifies queries that combine one or more joins with filtering predicates within the same scope. Detection requires at least one `JOIN` and a `WHERE` clause in the same `SELECT`. The complexity arises from jointly reasoning over join topology and predicate interactions, contributing to Width via the join-graph degree.

```
SELECT *
FROM A JOIN B ON A.id = B.id
WHERE A.x > 10 AND B.y < 5;
```

M3: AGGREGATION This motif covers aggregate computation patterns, including `GROUP BY`, `HAVING`, and nested aggregates. Detection triggers on aggregate functions or grouping clauses. Ag-

gregation compresses row-level data into summary values and may introduce mild Width through grouping dimensions.

```
SELECT dept_id, COUNT(*)
FROM employees
GROUP BY dept_id
HAVING COUNT(*) > 10;
```

M4: QUALIFY This motif captures the two-stage pattern where a window function is computed in an inner scope (typically a CTE or subquery) and filtered in an outer scope. Detection identifies window functions inside a nested scope whose outputs are referenced by an enclosing `WHERE` clause. This pattern necessarily increases Depth and often increases Width through partitioning and ordering dimensions.

```
WITH r AS (
  SELECT
    ROW_NUMBER() OVER
    (PARTITION BY dept
    ORDER BY salary DESC)
    AS rn,
    *
  FROM employees
)
SELECT *
FROM r
WHERE rn <= 3;
```

M5: OR This motif identifies disjunctive predicates connected by `OR`. Detection scans `WHERE` and `HAVING` expressions for `OR` nodes. Each branch represents an independent semantic path, expanding the parallel decision space and increasing Width.

```
SELECT * FROM employees
WHERE dept_id = 1 OR
salary > 10000 OR title = 'Manager';
```

M6: CASE This motif captures conditional logic expressed through `CASE WHEN` constructs in projection, filtering, or ordering. Detection flags any `CASE` node in the AST. Each `WHEN` branch introduces an additional decision branch, contributing to Width.

```
SELECT CASE
  WHEN salary > 15000 THEN 'Senior'
  WHEN salary > 8000 THEN 'Mid'
  ELSE 'Junior'
END
FROM employees;
```

M7: DISTINCT This motif identifies explicit duplicate elimination via `DISTINCT`. Detection checks the `DISTINCT` flag on `SELECT` nodes. While DISTINCT rarely affects Depth or Width, it encodes a non-trivial semantic requirement and is included as a semantic marker rather than a structural complexity driver.

```
SELECT DISTINCT dept_id FROM employees;
```

C Prompt Templates for Controlled Data Synthesis

D Computational Overhead of Synthesis Pipeline

Our pipeline produced 5,000 validated instances from 17,538 attempts (28.5% overall pass rate). Table 8 reports per-stage pass rates and cost types.

Prompt Template for Controlled Data Synthesis

```
You are a SQL expert. Please evolve the given SQL into a more complex form.
## Database Schema
{schema_text}

## Seed SQL
{seed_sql}

## Seed Question
{seed_question}

## Evolution Target
Current complexity: Depth={current_depth}, Width={current_width}
Target complexity: Depth={target_depth}, Width={target_width}

## Required Operators
{list of required operators with internal mutation instructions}

## Constraints
1. Use only the required operators specified above
2. Use only tables and columns defined in the schema
3. Ensure the SQL is syntactically correct and executable on SQLite
4. Do not introduce decorative or vacuous conditions
5. Respect foreign key relationships defined in the schema

## Semantic Constraints
- Do not aggregate identifier or date/time fields
- Do not apply numeric aggregation to string-valued columns
- Aggregations must be semantically meaningful

## Output Format (JSON)
{"sql": "...", "question": "..."}
```

Figure 5: Structured prompt template used for controlled SQL synthesis.

Stage	Pass Rate	Cost Type
Executability	65.6%	1 DB exec/attempt
Motif Consistency	77.6%	Static AST (zero cost)
Motif-level Redundancy	90.8%	1 DB exec/candidate
Rewrite-based Redundancy	61.7%	$\sim$3 LLM + $\sim$4 DB/candidate

Table 8: Per-stage pass rates and computational costs.

Per valid instance, this corresponds to approximately 8.4 LLM calls and 10 database executions. Database executions are sub-second in our SQLite setup ($<$100ms on average), so runtime is dominated by LLM inference. Since each instance is independent, the pipeline is naturally parallelizable.