Isolating Architectural Effects in Text-to-SQL: A Controlled Diagnostic Study Using the Spider Sandbox

Anonymous ACL submission

Abstract

When a Text-to-SQL pipeline is held strictly constant, how much does foundation model architecture influence reasoning capability? We evaluate three open-weight models – Gemma 4 31B (dense), Qwen3-Coder 30B (MoE, ∼3B active), and Qwen3-235B (MoE, ∼22B active) – through a fixed 11-step pipeline (ADAPT-SQL) on Spider 1.0, used as a *controlled diagnostic sandbox*. A robust pipeline equalizes performance on 89% of standard queries within 0.8 pp. On nested-complex compositional tasks, performance diverges by 4.4 pp: Mixture-of-Experts routing in Qwen3-235B activates richer reasoning circuits per token than a dense architecture of equivalent active scale. We release the fixed-pipeline protocol as a reproducible framework for architectural ablations.

1 Introduction

Text-to-SQL evaluations routinely co-optimize model, prompt, retrieval, and generation strategy simultaneously, conflating fundamental model capability with system-level engineering and making it impossible to isolate architectural effects.

Spider 1.0 as a controlled sandbox. While harder benchmarks such as BIRD (Li et al., 2023b) present more realistic challenges, their schema noise, evidence-grounded value retrieval, and large enterprise schemas introduce confounding variables that obscure pure compositional reasoning limits. Spider 1.0 (Yu et al., 2018)'s hand-curated annotations are sufficiently complex to stress-test reasoning (138 databases, correlated subqueries, multi-hop joins) while clean enough to attribute observed performance differences to model architecture – not annotation noise or retrieval quality. We use it as a controlled substrate, not a leaderboard.

We evaluate three open-weight models through the *identical* ADAPT-SQL pipeline – identical prompts, retrieval weights, and hyperparameters, no per-model tuning – to isolate architectural from engineering effects.

Core findings.

1. **Pipelines equalize models on standard queries.** On the 89% of Spider queries in the EASY and NON-NESTED classes, all three models score within 0.8 pp despite a tenfold difference in active parameters: pipeline engineering, not model scale, is the dominant factor.
2. **MoE routing unlocks compositional reasoning.** On NESTED-COMPLEX queries (10.7%), performance diverges by 4.4 pp. Qwen3-235B (22B active MoE) outperforms the dense Gemma 4 31B (31B active) – an ordering inconsistent with active-parameter count alone, consistent with MoE routing activating specialized reasoning circuits.
3. **A reproducible evaluation protocol.** Every component held fixed means any future model can be benchmarked under identical conditions, with score differences attributable to architecture alone.

2 Related Work

Schema linking and retrieval. RESDSQL (Li et al., 2023a) decouples schema ranking from skeleton generation. DIN-SQL (Pourreza and Rafiei, 2023) shows LLM-based schema linking outperforms rule-based approaches on cross-domain benchmarks. DAIL-SQL (Gao et al., 2023) demonstrates that structural similarity retrieval improves EX by 3–5 pp over semantic similarity alone. ADAPT-SQL chains lexical pre-filtering, LLM semantic linking, and FK-connectivity validation (97.3% table recall), then retrieves examples via FAISS (Johnson et al., 2021) with four-signal structural reranking over Nomic embeddings (Nussbaum et al., 2024).

Generation and fine-tuned systems. NatSQL (Gan et al., 2021) proposes a simplified intermediate representation reducing hallucinated joins. DIN-SQL (Pourreza and Rafiei, 2023) decomposes complex queries into sub-questions. XiYan-SQL (Xiang et al., 2024) ensembles diverse generation strategies; MAC-SQL (Wang et al., 2024) coordinates specialist agents. Among fine-tuned systems, SQL-R1 (Ma et al., 2025) reaches 88.7% via execution-based RL. Among prompt-only systems, CHESS (Talaei et al., 2024) reaches 87.2% via multi-agent contextual harnessing and AP-SQL (Tang et al., 2025) 89.7% via complexity-routed CoT/GoT with GPT-4o.

Multi-model comparisons. Prior comparisons (Gao et al., 2023; Xiang et al., 2024) co-optimize the pipeline per model, conflating capability with engineering. Our study holds every component fixed across architecturally diverse models – dense vs. MoE, 3B vs. 22B active parameters – enabling the first controlled measurement of how architecture alone determines SQL generation accuracy.

3 The ADAPT-SQL Pipeline

ADAPT-SQL translates a natural language question into SQL through eleven sequential steps in five phases (Figure 1, Table 1). **Every pipeline component – prompts, retrieval weights, validation logic, and hyperparameters – is held identical across all three models.**

3.1 Schema Linking (Step 1)

Three layers are applied in sequence. **Layer 1 (Lexical):** exact, prefix, and fuzzy string matching as a fast pre-filter. **Layer 2 (LLM Semantic):** the model identifies semantically relevant items including synonyms and paraphrases, providing the largest recall jump (+9.9 pp tables, +17.9 pp columns). **Layer 3 (FK Connectivity):** validates foreign-key paths between selected tables, adding bridging tables automatically (+3.2 pp recall). Final recall is 97.3% (tables) and 93.1% (columns) (Table 2).

3.2 Complexity Classification (Step 2)

Queries are classified as EASY, NON-NESTED-COMPLEX, or NESTED-COMPLEX using ten structural signals (subquery presence, set operations, aggregation diversity) handling ≈80% of queries at >95% accuracy, with an LLM fallback for the remaining 20%. Correct routing is critical: sending a NESTED query through the EASY path achieves only 34.2% EX on that sub-class vs. 84–88% with decomposition.

Step	Component	Design choice
S1	Schema Linking	Lexical → LLM semantic → FK connectivity
S2	Complexity Clf.	Rules (80%) + LLM fallback (20%)
S3	Prelim. SQL	Structural signature for retrieval
S4	Retrieval	FAISS + 4-signal structural reranking
S5	Routing	Complexity-driven strategy selection
S6	Generation	3 strategies (§3.4)
S6.5	Post-generation	Alias norm + set-op enforcement
S7	Validation	Deterministic 6-checker chain
S8	Retry	Error-feedback regen, ≤2 retries
S9	Normalization	Formatting consistency
S10	Execution	SQLite, 30 s timeout
S11	Evaluation	EX and EM (column-order normalized)

Table 1: The eleven pipeline steps and their key design choices.

Layer	Tbl Rec.	Col Rec.	Prec.	Avg
L1: Lexical only	84.2%	71.3%	91.4%	4.1
+L2: LLM Semantic	94.1%	89.2%	85.7%	6.3
+L3: FK Connectivity	**97.3%**	**93.1%**	84.2%	7.8
Gold schema	100%	100%	–	7.2

Table 2: Schema linking accuracy per layer (Qwen3-Coder on Spider dev).

3.3 Example Retrieval (Step 4)

ADAPT-SQL retrieves top-5 examples from a FAISS index of 6,962 training questions, then reranks by four signals: semantic similarity (45%, Nomic embeddings (Nussbaum et al., 2024)), SQL structural similarity (25%), style similarity (20%), and preliminary-SQL alignment (10%). Structural reranking recovers cases where semantic similarity misleads: "How many singers are there?" is semantically close to "What are all singer names?" but structurally close to "How many concerts are there?" – the more useful few-shot example.

3.4 Adaptive Generation (Steps 5–6)

Step 5 routes each query to one of three strategies. **Step 6a (Direct few-shot):** three retrieved examples plus the pruned schema, for EASY queries.

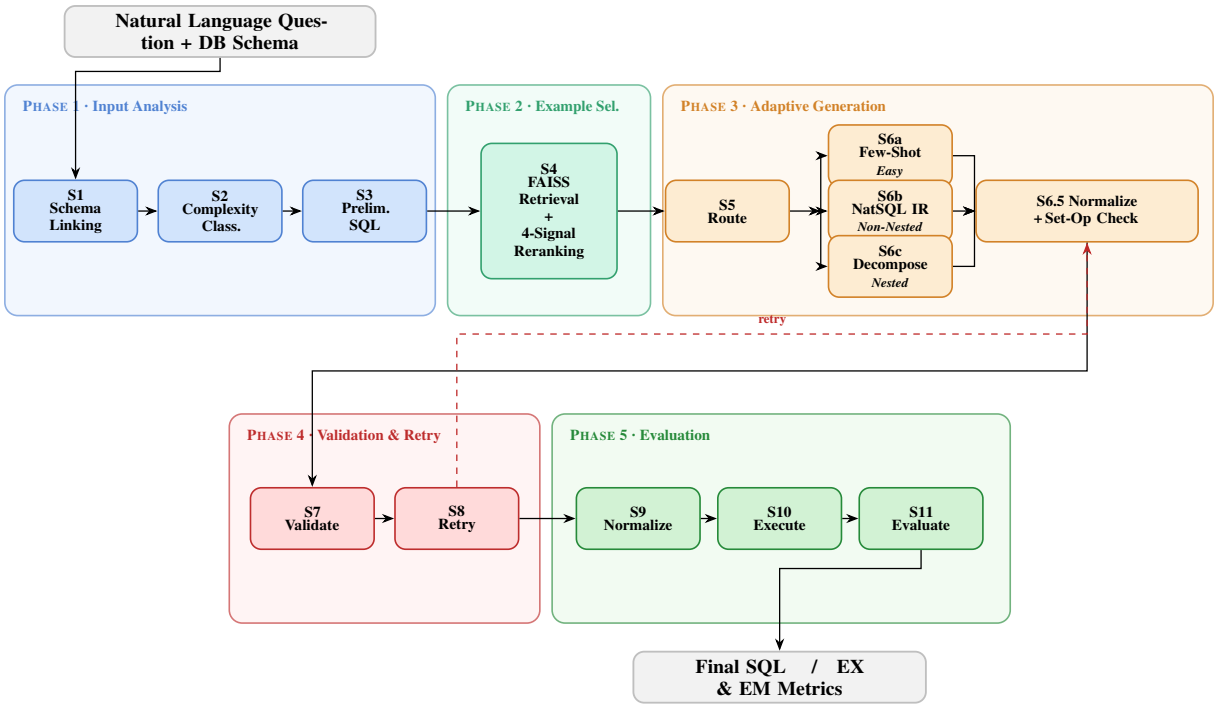

Figure 1: The ADAPT-SQL pipeline (11 steps, 5 phases). **PHASE 1** (blue): schema linking, complexity classification, preliminary SQL. **PHASE 2** (teal): FAISS retrieval with four-signal structural reranking. **PHASE 3** (orange): routing to one of three generation strategies plus normalization and set-op enforcement. **PHASE 4** (red): schema validation and error-feedback retry (dashed arc). **PHASE 5** (green): normalization, execution, and evaluation. All components held fixed across all models.

Step 6b (NatSQL IR): a simplified intermediate representation with implicit joins is generated first, then translated to standard SQL, reducing hallucinated joins in multi-table queries (Gan et al., 2021). **Step 6c (Sub-question decomposition):** the model generates the inner subquery first, then wraps it, following Pourreza and Rafiei (2023). **Step 6.5:** a deterministic checker enforces set-operation presence, recovering 6–7 queries per model that produce syntactically valid but semantically wrong SQL.

3.5 Validation, Retry, and Normalization (Steps 7–9)

Step 7 applies a six-checker chain: syntax, table/column existence, FK path validity, aggregation consistency, and set-op presence. Step 8 reruns generation with structured error feedback (e.g., *"Column 'num_of_seats' not found. Available: concert_id, concert_name..."*), enabling self-correction in ∼60% of triggered retries. Step 9 normalizes alias naming and clause ordering. A final deterministic pass (CAST-stripping and DISTINCT insertion) aligns results with the Spider EX metric without any LLM call.

Model	Architecture	Total	Active
gemma4-31b-it	Dense	31B	31B
qwen3-coder-30b-a3b	MoE	30B	∼3B
qwen3-235b-a22b	MoE	235B	∼22B

Table 3: Models evaluated. "Active" = parameters activated per forward pass. All are open-weight (Google DeepMind, 2026; Qwen Team, 2025).

4 Experimental Setup

Dataset. We evaluate on Spider (Yu et al., 2018): 1,034 dev queries (20 databases) and 2,147 test queries (40 databases) across 138 domains. Spider 1.0's hand-curated annotations provide a clean substrate: unlike BIRD (Li et al., 2023b), the absence of schema noise and evidence retrieval requirements means performance differences reflect compositional reasoning capacity rather than retrieval-augmentation skill, satisfying the controlled comparison design.

Models. Table 3 summarizes the three models. All are accessed via an OpenAI-compatible inference endpoint on an academic HPC cluster; temperature 0.0; no fine-tuning or gradient updates.

System	LM backbone	Dev EX	Test EX
Fine-tuned systems			
SQL-R1 (Ma et al., 2025)	Qwen2.5-Coder-7B	–	88.7%
ExCoT (Zhai et al., 2025)	LLaMA-3.1-70B	–	86.6%
LitE-SQL (Piao et al., 2026)	Fine-tuned (7B)	–	88.45%
RESDSQL (Li et al., 2023a)	T5-3B	84.1%	79.9%
Prompt-only systems (no fine-tuning)			
AP-SQL (Tang et al., 2025)	GPT-4o	–	89.7%
CHESS (Talaei et al., 2024)	DeepSeek-Coder-34B	–	87.2%
DAIL-SQL (Gao et al., 2023)	GPT-4	86.6%	86.6%
DIN-SQL (Pourreza and Rafiei, 2023)	GPT-4	82.8%	85.3%
ADAPT-SQL (this work): prompt-only, open-weight			
gemma4-31b	gemma4-31b-it	**91.0%**	**91.3%**
qwen3-235b	qwen3-235b-a22b	90.4%	89.7%
qwen3-coder-30b	qwen3-coder-30b	90.6%	87.4%

Table 4: Spider execution accuracy. All ADAPT-SQL variants use open-weight models via academic HPC (no proprietary API). **Bold** = best; underline = second best.

Model	Easy	Non-Nested	Nested	All
gemma4-31b	95.7%	**89.9%**	86.7%	**91.0%**
qwen3-coder	**95.9%**	89.1%	84.1%	90.6%
qwen3-235b	94.0%	89.3%	**88.5%**	90.4%
Max − Min	1.9 pp	0.8 pp	**4.4 pp**	0.6 pp
Query counts	251–316	605–670	113	1,034

Table 5: EX by complexity class. NESTED count is identical across models (structural detection); EASY/NON-NESTED vary slightly due to LLM fallback. Model differences compress to <2 pp on the first two classes but widen to 4.4 pp on NESTED.

5 Results and Analysis

5.1 Main Results

Table 4 reports execution accuracy alongside published baselines; baselines establish that the fixed pipeline operates in a competitive regime, while our analytical focus is on within-study contrasts.

The aggregate spread across our three models is only 0.6 pp (91.0% vs. 90.4%) on the dev set – within one binomial SE (≈0.93 pp), so no aggregate pairwise difference is statistically significant.[1] This compression is the central finding for standard queries, and it shifts attention to the 10.7% where the equalization breaks. On the test set, Qwen3-Coder drops 3.2 pp (90.6%→87.4%) while the other two remain stable (<1 pp); because all pipeline components are fixed, this is a clean architectural attribution.

5.2 Performance by Query Complexity

On EASY queries all models cluster within 1.9 pp (94.0–95.9%); on NON-NESTED-COMPLEX the spread narrows to 0.8 pp (89.1–89.9%), as the Nat-SQL IR constrains the generation space regardless of model. The pipeline's equalizing effect fully breaks on NESTED-COMPLEX (4.4 pp), revealing the architectural signal analyzed below.

[1] Clopper-Pearson 95% CIs: each ±1.8 pp.

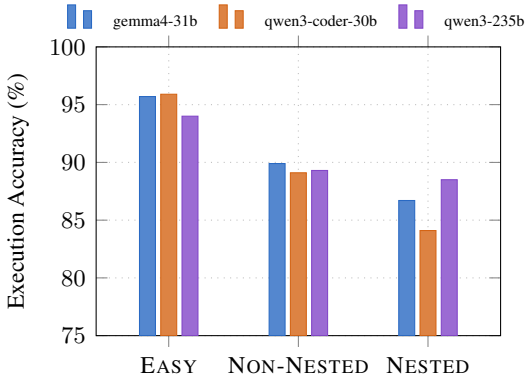

Figure 2: EX by complexity class. The fixed pipeline equalizes models on EASY and NON-NESTED (89% of Spider) but reveals a 4.4 pp architectural gap on NESTED: the MoE model Qwen3-235B (22B active) outperforms the dense Gemma 4 31B (31B active) despite *fewer* active parameters.

5.3 MoE Routing vs. Dense Architecture

On NESTED queries the ranking is: Qwen3-235B (22B active MoE, 88.5%) > Gemma 4 31B (31B dense, 86.7%) > Qwen3-Coder (3B active MoE, 84.1%). This ordering violates active-parameter count alone: Qwen3-235B has *fewer* active parameters per forward pass than Gemma 4 31B yet outperforms it by 1.8 pp on the hardest queries.

We interpret this as an architectural effect of MoE routing specialization. Decomposing a correlated subquery requires simultaneously tracking the outer predicate, the inner aggregation, and the join path connecting them. MoE routing can allocate distinct expert circuits to each sub-task within a single forward pass; a dense model of equivalent active scale must share its full capacity across all sub-tasks simultaneously. This qualitative difference in parameter *deployment*, not mere quantity, explains divergence at the compositional reasoning boundary and is only observable because all other variables are held constant.

6 Component Analysis

6.1 Component Contributions

Table 6 reports a controlled ablation on 300 Spider dev queries using Qwen3-Coder-30B with one component removed at a time.

Configuration	Dev EX	Δ
Full ADAPT-SQL	87.7%	–
– Validation + Retry (S7–8)	82.0%	−5.7 pp
– FK Connectivity (L3)	87.7%	±0.0 pp
– Structural Reranking (S4)	88.0%	+0.3 pp
– Set-Op Enforcement (S6.5)	88.3%	+0.6 pp

Table 6: Ablation on 300 Spider dev queries (Qwen3-Coder). Only validation + retry produces a significant drop. Other components show near-zero deltas within 1 SE (≈1.9 pp); their benefits are concentrated in sub-classes requiring full-scale evaluation.

Failure category	Count	%
Column/aggregation misidentification	34	35%
Wrong join path or table selection	24	25%
Missing or extra SELECT columns	15	15%
String literal case mismatch	10	10%
Unresolved set-op failures	7	7%
API/service errors (503)	4	4%
Other semantic errors	3	4%
Total	97	100%

Table 7: Failure categories for Qwen3-Coder on Spider dev after rescoring; distribution is similar across models.

Removing validation and retry (Steps 7–8) drops EX by **5.7 pp**, the single largest component contribution. Structured error feedback enables self-correction in ∼60% of retries. Other components – FK connectivity, structural reranking, set-op enforcement – show small or neutral deltas on this 300-query sample; their benefits are concentrated in specific sub-classes (join-failure queries; INTERSECT/EXCEPT queries) detectable only at full scale.

6.2 Failure Analysis

Column/aggregation misidentification (35%) is the dominant failure: NL terms ambiguous between schema elements and SQL functions (e.g., `average` as a column name vs. `avg()` as an aggregate) cannot be resolved by schema linking alone. **Wrong join path** (25%) occurs when the model routes through unnecessary intermediate tables, producing non-matching result sets despite syntactic validity. **String literal case mismatch** (10%) reflects SQLite case-sensitivity when stored values differ from model-generated literals; the pipeline cannot anchor string case without content retrieval.

7 Discussion

The pipeline as an architectural equalizer. A strong pipeline dominates model scale for standard SQL tasks: all three models achieve 89–96% EX on 89% of Spider with no model consistently leading. Qwen3-Coder at 3B active parameters matches Qwen3-235B at 22B active within measurement noise. The equalization disappears precisely at the compositional reasoning boundary – the 10.7% nested class – creating a clean threshold for model selection in deployment.

MoE routing as a signal amplifier for architecture. The 4.4 pp nested gap is visible only because all pipeline variables are held constant; in a co-optimized system this signal would be absorbed by engineering noise. The result motivates complexity-aware routing: directing NESTED queries to MoE models with high active-parameter counts while routing EASY/NON-NESTED queries to smaller alternatives could recover 2–4 pp over a uniform small-MoE deployment.

Reproducible diagnostic protocol. By not co-optimizing any component per model, any future open-weight model can be benchmarked under identical conditions with score differences attributable to architecture alone – a property leaderboard submissions do not provide. We release all code, prompts, retrieval indices, and checkpoints to support this use.

8 Limitations

We evaluate only three open-weight models; whether the equalizer effect holds for fine-tuned models or those already above 90% base EX is open. Extending to BIRD or Spider 2.0 would require additional pipeline modules (evidence ranking, content sampling) that introduce new degrees of freedom, compromising the single-variable isolation this study achieves. The Qwen3-Coder dev-to-test gap warrants further investigation across held-out schema partitions to separate distributional shift from model sensitivity.

9 Conclusion

We presented a controlled diagnostic study in which ADAPT-SQL, a fixed 11-step pipeline, isolates how model architecture shapes SQL generation accuracy on Spider 1.0. Three open-weight models spanning a 7× range in active parameters cluster within 0.6 pp overall yet diverge by 4.4 pp

on nested compositional queries. A strong pipeline equalizes models on 89% of standard SQL tasks; MoE routing specialization outperforms dense architectures of greater active-parameter count at the compositional boundary – an effect only observable under the controlled design. The fixed-pipeline evaluation protocol, with all artifacts released, provides a reproducible framework for future architectural ablations.

References

Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John Drake, and Qiaofu Zhang. 2021. Natural SQL: Making SQL easier to infer from natural language specifications. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2030–2042.

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

Google DeepMind. 2026. Gemma 4 model card. https://ai.google.dev/gemma/docs/core/model_card_4.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. RESDSQL: Decoupling schema linking and skeleton parsing for text-to-SQL. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13067–13075.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, and 1 others. 2023b. Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs. *arXiv preprint arXiv:2305.03111*.

Peixian Ma, Xialie Zhuang, Chengjin Xu, Xuhui Jiang, Ran Chen, and Jian Guo. 2025. SQL-R1: Training natural language to SQL reasoning model by reinforcement learning. *arXiv preprint arXiv:2504.08600*.

Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. 2024. Nomic embed: Training a reproducible long context text embedder. In *arXiv preprint arXiv:2402.01613*.

Shengmin Piao, Jieun Lee, and Sanghyun Park. 2026. LitE-SQL: A lightweight and efficient text-to-SQL framework with vector-based schema linking and execution-guided self-correction. *arXiv preprint arXiv:2510.09014*.

Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. In *Advances in Neural Information Processing Systems*, volume 36.

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

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

Zetong Tang, Qian Ma, and Di Wu. 2025. Auto prompt SQL: A resource-efficient architecture for text-to-SQL translation in constrained environments. *arXiv preprint arXiv:2506.03598*.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Lingxiao Zhang, Ran Yan, Duyu Su, Haiying Hu, and Zhoujun Li. 2024. MAC-SQL: A multi-agent collaborative framework for text-to-SQL. *arXiv preprint arXiv:2312.11242*.

Yingqi Xiang, Jieyu Ding, Shuofei Li, Qinglong Zhao, Shuai Shan, Jiahui Liu, Kang Zhu, Longxu Dou, and 1 others. 2024. XiYan-SQL: A multi-generator ensemble framework for text-to-SQL. *arXiv preprint arXiv:2411.08599*.

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. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Bohan Zhai, Canwen Xu, Yuxiong He, and Zhewei Yao. 2025. ExCoT: Optimizing reasoning for text-to-SQL with execution feedback. *arXiv preprint arXiv:2503.19988*.

A Complexity Classification Rules

The rule-based classifier applies the following signals in order:

1. **Subquery presence** in preliminary SQL (Step 3) → NESTED
2. **Set operation** keyword (UNION/INTERSECT/EXCEPT) → NESTED
3. **Correlated subquery** pattern (`WHERE col = outer.col`) → NESTED
4. **Simple single-table** with no aggregation and $\leq$1 condition → EASY
5. **Aggregation with GROUP BY** or **multiple tables** → NON-NESTED
6. **HAVING** clause → NON-NESTED

7. Remaining ambiguous cases → LLM with 3-shot CoT

The LLM fallback fires on ≈20% of queries and achieves >90% accuracy on held-out development examples. Because subquery detection is structural, all three models route exactly the same 113 queries to NESTED-COMPLEX.

B Example Pipeline Trace

Question: *"Find the name of the stadium where no concert was held."*
Database: `concert_singer` (tables: `stadium`, `singer`, `concert`, `singer_in_concert`)

S1 (Schema Linking): Layer 1 matches "stadium" exactly; Layer 2 adds `concert` semantically; Layer 3 confirms FK path. Pruned schema: {`stadium`(id, name, . . .), `concert`(concert_id, stadium_id, . . .)}.
S2 (Complexity): "no concert was held" implies EXCEPT/NOT IN. Rule fires: NESTED-COMPLEX.
S3 (Prelim. SQL): `SELECT name FROM stadium WHERE stadium_id NOT IN (SELECT stadium_id FROM concert)`
S4 (Retrieval): Retrieves examples with NOT IN subquery patterns.
S6c (Decompose): Inner: "Which stadiums hosted a concert?" → `SELECT DISTINCT stadium_id FROM concert`. Outer wraps with NOT IN.
Final SQL:

```
SELECT name FROM stadium
WHERE stadium_id NOT IN (
  SELECT stadium_id FROM concert
)
```

Result: Correct. EX = 1.