Journal of KIIT. Vol. 24, No. 7, pp. 81-93, Jul. 31, 2026. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2026.24.7.81

Improving SQL Generation on Industrial Time-Series Data with Structured EXPLAIN Feedback using a 4B SLM

Byounghoon Son*[1], Minsung Jung*[2], and Sangkeum Lee**

This research was supported by the Regional Innovation System & Education (RISE) program through the Daejeon RISE Center, funded by the Ministry of Education (MOE) and the Daejeon Metropolitan City, Republic of Korea. (2026-RISE-06-002)

Abstract

Iterative self-correction commonly improves LLM-generated SQL, yet feedback granularity remains unexplored. This study fine-tunes a 4B SLM with LoRA on industrial data (3M records, 8 sectors) and compares EXPLAIN-based correction at three feedback levels: generic failure, database error, and structured EXPLAIN with schema hints. Granularity governs convergence speed, not the accuracy ceiling: all three reach indistinguishable compliance (86-88%, McNemar $p \geq 0.75$), yet structured feedback needs 20% fewer retries (71 vs. 89 per 100 queries). Fine-tuning yields the largest single gain (38→80% ID, 34→69% OOT), and EXPLAIN retry adds +10/+8 pp to reach 90%/77% (ID/OOT) compliance. A learned 270M critic fails as a verifier, while Qwen-2.5 3B reproduces the OOT ceiling, indicating a data-constrained bound.

요 약

반복적 자기 수정은 LLM 기반 SQL 생성을 개선하는 일반적 전략이지만, 피드백의 세부 수준이 미치는 영향은 탐구되지 않았다. 본 연구는 LoRA로 파인튜닝한 4B 소형 언어모델로 산업용 전력 데이터(300만 건, 8개 업종)에서 세 가지 피드백 수준(일반 실패 신호, DB 오류 메시지, 스키마 힌트 포함 구조화된 EXPLAIN)의 EXPLAIN 기반 수정을 비교한다. 세부도는 정확도 상한이 아니라 수렴 속도를 결정한다. 세 수준 모두 구분 불가능한 준수율(86-88%, McNemar $p \geq 0.75$)에 도달하나, 구조화된 피드백은 재시도가 20% 더 적다(100쿼리당 71회 대 89회). 파인튜닝이 가장 큰 단일 성능 향상(38→80% ID, 34→69% OOT)을 제공하고 EXPLAIN 재시도가 +10/+8 백분율 포인트를 추가하여 90%/77%(ID/OOT) 준수율을 달성한다. 학습된 270M 크리틱은 검증기로서 실패하는 반면, Qwen-2.5 3B는 동일한 OOT 상한을 재현해 데이터에 제약된 한계를 시사한다.

Keywords

text-to-SQL, small language model, self-correction, feedback granularity, LoRA fine-tuning

* Dept. of Computer Engineering, Hanbat National University
- ORCID[1]: https://orcid.org/0009-0002-5625-1418
- ORCID[2]: https://orcid.org/0009-0002-3245-6785
** Assistant Professor, Dept. of Computer Engineering, Hanbat National University
- ORCID: https://orcid.org/0000-0001-6918-124X

· Received: Jun. 10, 2026, Revised: Jul. 13, 2026, Accepted: Jul. 16, 2026
· Corresponding Author: Sangkeum Lee
Dept. of Computer Engineering, Hanbat National University, 125 Dongseo-daero, Yuseong-gu, Daejeon, 34158, Korea
Tel.: +82-42-821-1205, Email: sangkeum@hanbat.ac.kr

Ⅰ. Introduction

Natural language database interfaces are attracting attention as a technology that can democratize data access for non-expert users. In industrial settings such as power consumption anomaly detection in manufacturing, data security requirements make cloud-based Large Language Model (LLM) APIs difficult to use, and small language models (SLMs, parameters $\leq$ 4B) become a realistic alternative. However, the baseline SQL generation accuracy of SLMs is low: in the experimental setting of this study, the 4B model achieves only a 38% format compliance rate. Prior work has attempted to address this problem through iterative self-correction [1]-[3]. When the generated SQL is invalid, the error is provided as feedback to request regeneration; the benefit of providing feedback has been consistently demonstrated. However, a key question remains unresolved: does the diagnostic richness of feedback matter, or is a simple failure signal sufficient?

This question has practical significance. SQL systems in enterprise environments face a design trade-off between feedback pipeline complexity and inference cost. If richer feedback achieves higher accuracy, the engineering investment is justified; however, if it only accelerates convergence, the decision reduces to a compute budget allocation problem. This study investigates this question using a Gemma-3 4B model fine-tuned with LoRA on Korean industrial electricity data (3 million records, 8 sectors). This study compares EXPLAIN-based iterative correction with three levels of feedback granularity: (A) a generic failure signal, (B) the database error message, and (C) a structured EXPLAIN output with schema hints.

A central finding of this study is that feedback granularity governs convergence speed but not the accuracy ceiling. All three feedback levels reach statistically indistinguishable final accuracy (86-88%, McNemar $p \geq 0.75$), but structured feedback reduces the inference calls required for convergence by 20% (71 vs. 89 calls). This suggests that feedback design is not an accuracy problem but a test-time compute efficiency problem [4].

The paper makes the following contributions. (1) A three-level feedback ablation study showing that feedback granularity affects convergence speed but not the accuracy ceiling - the first such comparative study in the Text-to-SQL self-correction literature. (2) An EXPLAIN-based correction pipeline that achieves a 90%/77% (ID/OOT) format compliance rate on industrial data with a 4B SLM. (3) A critic ablation study showing that a learned 270M critic cannot replace deterministic EXPLAIN verification. (4) Cross-model validation on Qwen-2.5 3B confirming the same OOT convergence point.

Ⅱ. Related Works

2.1 LLM self-correction

Self-Refine [5] and Reflexion [6] showed that LLMs can improve their outputs through iterative feedback loops. However, Huang et al. [7] showed that LLMs cannot self-correct reasoning without external signals, a finding corroborated by two surveys [8][9]. Self-Debug [1] uses execution results as external feedback for code generation, and CRITIC [10] incorporates tool interaction. Kumar et al. [11] train self-correction via reinforcement learning, but target mathematics and code rather than SQL. Critically, all prior work compares the presence versus absence of feedback; none systematically varies feedback granularity to measure its effect on convergence.

2.2 Text-to-SQL

The field has progressed from supervised models on Spider [12] and BIRD [13] to enterprise-scale

evaluations including Spider 2.0 [14] and LLM-based approaches such as DIN-SQL [15] and DAIL-SQL [16]. Recent systems increasingly integrate retry mechanisms: RetrySQL [2] trains on corrupted-SQL retry trajectories, MAGIC [3] generates self-correction guidelines from error patterns, and SHARE [17] leverages hierarchical SLM-based correction. CHASE-SQL [18] uses execution plans for reasoning at generation time, whereas this work applies EXPLAIN output as post-hoc correction feedback, offering a complementary approach. Concurrent work ErrorLLM [19] models SQL errors via AST-based structural representations, whereas the EXPLAIN-based approach used here leverages the database engine directly without an external parser. None of these systems ablates feedback granularity.

2.3 SLM fine-tuning and deployment

LoRA [20] and QLoRA [21] enable parameter-efficient fine-tuning of small models. Synthetic training data improves Text-to-SQL fine-tuning [22], and practical deployment systems such as SQLGenie [23] illustrate the growing demand for industrial SQL interfaces. This work applies LoRA to a 4B model targeting on-premises deployment, where cloud-based LLMs are impractical.

2.4 Learned verifiers

Step-by-step verification [24], execution-based re-ranking [25], and process reward models [26] have improved LLM outputs in mathematics and code domains. The critic ablation (Section 4.6) tests whether a learned verifier can replace deterministic SQL validation, and finds that a 270M model does not have sufficient capacity for reliable SQL discrimination.

III. Proposed Method

3.1 Task definition

Given a natural language query q about industrial electricity consumption patterns, the goal is to generate a syntactically valid SQL query s over a single-table SQLite database. The target schema consists of one table (consumption_data) with 28 columns: an industry classification identifier (industry), time fields (year, month, day), and 24 hourly electricity consumption values (hourly_00 ~ hourly_23, in kWh).

3.2 Fine-tuning

2,400 synthetic question-SQL pairs were generated via template-based generation across six query families (threshold, z-score, pattern detection, comparison, temporal spike, aggregation). For each sample, industry sector, year, hour, and relevant parameters were randomly selected, and all generated SQL was verified through database execution.

Gemma-3 4B was fine-tuned with LoRA (rank r=16, scaling α=32, dropout 0.05). Training used a learning rate of 2×10^(−4), cosine schedule, batch size 4, gradient accumulation 4, and 3 epochs. Evaluation uses two splits: In-Distribution (ID) queries (n=100) using the six training template families, and Out-of-Template (OOT) queries (n=100) using 10 unseen families that share the same schema but require SQL structures not seen during training.

3.3 EXPLAIN-based iterative correction

When the model generates candidate SQL s_i, it is validated by running EXPLAIN s_i on the target database. If EXPLAIN returns an error, a feedback prompt is constructed and a new generation s_{i+1} is requested. This loop continues until either K=5 retries are reached or EXPLAIN succeeds. Three feedback strategies of increasing diagnostic specificity are compared. Level A (generic): "This SQL is invalid.

Please regenerate." Level B (error message): "This SQL is invalid. Error: {e}" - {e} is the raw error string from EXPLAIN (e.g., "no such column: hour"). Level C (structured EXPLAIN): "This SQL is invalid. Error: {e}. Available columns: {cols}" - {cols} is the list of valid columns for the target table. This three-level design isolates the effect of feedback informativeness. All conditions provide the same signal (invalid SQL) but differ in the diagnostic content available for correction. Fig. 1 shows the overall system architecture, and Table 1 and Fig. 2 present the prompt structure of each level.

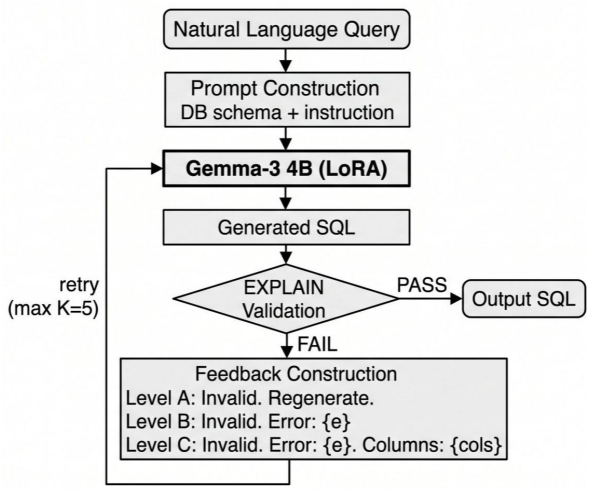

Fig. 1. System architecture

Table 1. Prompt structure of the three feedback levels

Slot	Content
System	"You are a SQL generator for table consumption_data."
User (q)	"{natural-language question}" (e.g., "What is the average power consumption of the food sector at 18:00 in 2021?")
Model output (s_i)	"SELECT … FROM consumption_data WHERE …"
Retry prompt appended on validation failure	
Level A	"This SQL is invalid. Please regenerate."
Level B	"This SQL is invalid. Error: {e}."
Level C	"This SQL is invalid. Error: {e}. Available columns: {cols}."

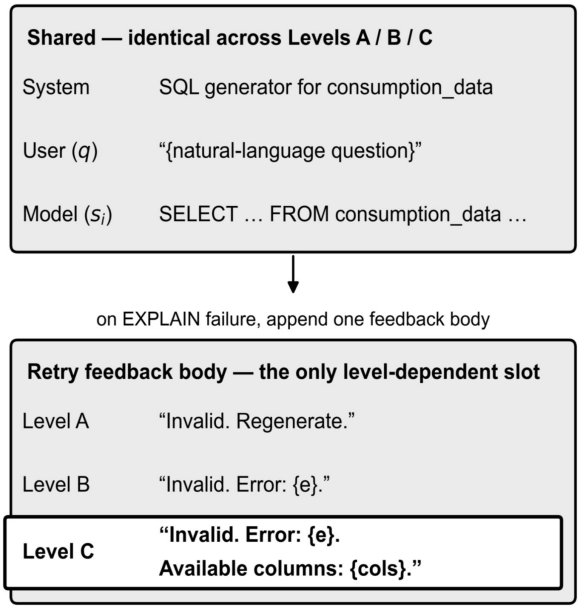

Fig. 2. Prompt structure of the three feedback levels

IV. Experiments and Results

4.1 Experimental setup

4.1.1 Dataset

Table 2. Time-Series characteristics of the dataset

Item	Value
Total columns	28
Hourly measurement columns (hourly_00 ~ hourly_23)	24 (85.7%)
Calendar time columns (year/month/day)	3 (10.7%)
Non-temporal categorical dimension (industry)	1 (3.6%)
Temporal resolution	1 hour
Total time span	2020-2022 (36 months)
Total rows	3,046,368
Industrial sectors	8
OOT categories referencing a temporal dimension	10/10 (100%)
(a) Point-in-time conditional	4/10
(b) Temporal window or sequence	3/10
(c) Time-unit aggregation or ranking	3/10

This study uses an industrial electricity consumption dataset covering eight Korean industrial sectors

(metals, chemicals, food, energy, ceramics, paper & wood, textiles, and other industry) from 2020 to 2022, comprising approximately 3 million hourly records. The data is stored in a single SQLite table (consumption_data) with 28 columns. Of the 28 columns, 24 (85.7%) are time-indexed hourly measurements; adding the three calendar fields, 27 (96.4%) map directly to a temporal dimension, indicating the strong time-series character of the dataset, as summarized in Table 2.

4.1.2 Evaluation queries

100 In-Distribution (ID) queries were constructed from the six training template families, together with 100 Out-of-Template (OOT) queries spanning 10 unseen categories: absolute threshold, composite aggregation, continuous pattern, cross-industry comparison, multi-condition, percentile, range difference, top-n, variance, and year-over-year. Each OOT category contains 10 queries.

4.1.3 Models

The primary generator is Gemma-3 4B. For the critic ablation, Gemma-3 270M was trained as a binary classifier. For cross-model validation, the same training data and LoRA configuration were applied to Qwen-2.5 3B. All experiments were run on a single NVIDIA RTX 5090 with FlashAttention-2.

4.1.4 Evaluation metrics

Format Compliance Rate (FCR): the proportion of generated SQL queries that pass syntactic validation via SQLite's EXPLAIN command. Because gold-standard SQL annotations are unavailable in this industrial setting, FCR is adopted in place of execution accuracy. For every FCR value, 95% Wilson confidence intervals are reported, and McNemar's exact test [27] is used for pairwise comparisons across feedback conditions.

4.2 Overall system performance

Table 3 presents the format compliance rate across five configurations, with 95% Wilson confidence intervals; FT denotes fine-tuning. Fine-tuning alone provides the largest single improvement, raising ID FCR from 38% to 80% (+42 pp) and OOT FCR from 34% to 69% (+35 pp). Adding EXPLAIN-based retry contributes an additional +10 pp on ID and +8 pp on OOT, reaching 90% and 77%, respectively.

The ID-OOT gap persists across all configurations: 4 pp on the base model, 11 pp after fine-tuning, and 13 pp after retry. This widening gap reflects the fact that residual OOT failures involve qualitatively different error types (unsupported functions, combinatorial complexity) that retry cannot resolve.

Table 3. Format compliance rate by configuration

Configuration	ID FCR (%) [95% CI]	OOT FCR (%) [95% CI]
Base	38 [29, 48]	34 [25, 44]
FT	80 [71, 87]	69 [59, 77]
FT + Critic (noisy)	33 [25, 43]	27 [19, 36]
FT + Critic (clean)	82 [73, 88]	65 [55, 74]
FT + EXPLAIN	90 [83, 94]	77 [68, 84]

4.3 Feedback granularity and convergence speed

The central experiment tests whether the content of correction feedback affects self-correction outcomes. The three feedback levels are applied to OOT queries (n=100, max 5 retries), starting from the fine-tuned baseline (69%). In Table 4, Calls is the total retry count across 100 queries, and Avg. Attempts is the mean attempts per query.

4.3.1 Accuracy ceiling is invariant to feedback

Table 4 shows that all three conditions reach

statistically indistinguishable final FCR: A 86%, B 88%, C 87%. McNemar's exact test yields $p \geq 0.75$ for every pairwise comparison (A vs. B: p=0.75; A vs. C: p=1.0; B vs. C: p=1.0), confirming that feedback granularity does not raise the accuracy ceiling.

Table 4. Feedback granularity ablation

Level	FCR (%)	95% CI	Avg. attempts	Calls
No retry	69	[59, 77]	1.00	0
A (generic)	86	[78, 91]	1.89	89
B (error msg)	88	[80, 93]	1.79	79
C (EXPLAIN)	87	[79, 92]	1.71	71

4.3.2 Convergence speed depends on feedback

Fig. 3 shows per-retry progression. Level C reaches 86% after one retry and stabilizes at 87% after two. Level B reaches 81% after retry 1 and 87% after retry 3. Level A rises gradually, recording 77%, 81%, 84%, and 86% over four retries. In total retry calls, Level C requires 71 calls versus 89 for Level A, achieving a 20% reduction in inference cost at the same final accuracy.

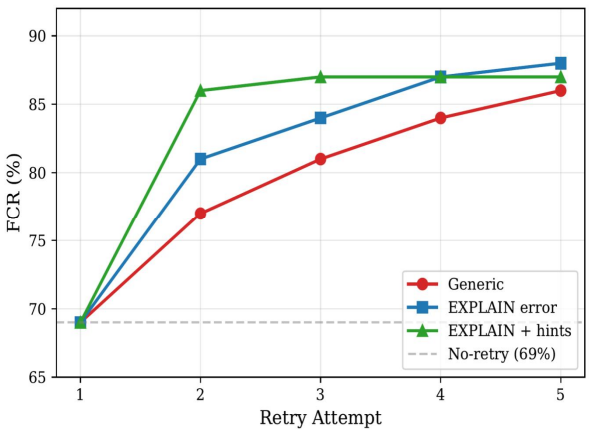

Fig. 3. FCR convergence curves by feedback granularity

The fact that Level B (error message) achieves a slightly higher point estimate (88%) than Level C (87%) reinforces that the benefit of structured feedback is confined to convergence speed rather than final accuracy. This finding characterizes feedback quality as a compute-efficiency factor rather than an accuracy factor, connecting to recent work on test-time compute scaling [4].

4.4 Per-Category OOT analysis

Aggregate OOT figures mask variation across query categories. Table 5 breaks FCR down by the 10 OOT query families (each n=10). Given the small sample size, three tentative observations are presented that warrant larger-scale confirmation. Categories resolved by fine-tuning: continuous pattern, cross-industry comparison, and percentile queries reach FCR 100% with fine-tuning alone, leaving no room for retry-based improvement.

Categories where EXPLAIN retry is effective: range difference and multi-condition queries show the largest retry gains (+40 pp and +20 pp, respectively). These categories generate SQL containing column-name errors that EXPLAIN feedback directly addresses. Categories resistant to correction: composite aggregation (40%) and variance (30%) queries remain low even after retry. Variance queries frequently generate VARIANCE(), which SQLite does not natively support, while composite aggregation queries require multi-stage CTEs and nested subqueries that exceed the model's combinatorial capacity.

Table 5. Per-Category OOT FCR

OOT category	Base (%)	FT (%)	+EXPLAIN (%)
Absolute threshold	80	80	90
Composite aggregation	10	40	40
Continuous pattern	40	100	100
Cross-industry comparison	60	100	100
Multi-condition	50	70	90
Percentile	50	100	100
Range difference	0	40	80
Top-n	30	70	70
Variance	0	30	30
Year-over-year	20	60	70
Overall	34	69	77

4.5 Error taxonomy

To understand what each pipeline stage resolves and what remains unresolved, all OOT failures were classified into mutually exclusive error types (Fig. 4).

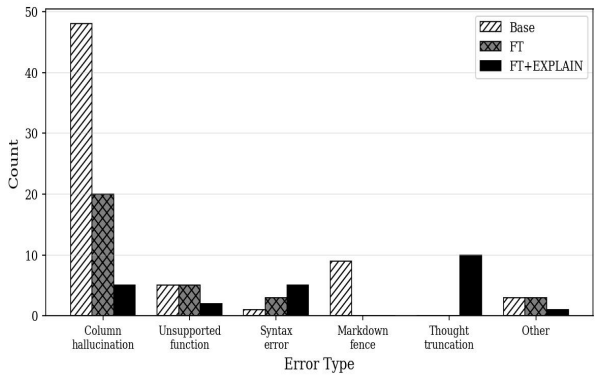

Fig. 4. OOT failure distribution by error type

Base-model failures (n=66) are dominated by column hallucinations (48/66, 73%). The most frequent sub-type is schema-similar variant hallucination (36 cases), where the model generates column names resembling but not matching the schema (e.g., hour_05 → hourly_05). Markdown-fence errors (9/66) indicate that the model emits code-fenced text instead of raw SQL.

Fine-tuning eliminates all markdown-fence errors (9 →0) and all schema-similar variant hallucinations (36 →0). Residual failures (n=31) shift to invented column hallucinations (20/31, 65%) - fully fictional column names bearing no resemblance to the schema - plus 5 unsupported-SQLite-function cases and 3 incomplete-SQL cases.

EXPLAIN retry further reduces column hallucinations from 20 to 5, but introduces a new dominant failure mode, thought truncation (10/23, 43% of residual failures). As the prompt grows with retries, the accumulated context exhausts the generation budget and produces incomplete SQL - a structural limitation of iterative correction under fixed generation length.

4.6 Critic ablation

Given that learned verifiers have shown promise in mathematical reasoning [24] and code generation [25], this study investigated whether a similar approach could replace deterministic EXPLAIN verification.

Training data and label noise: 14,400 SQL samples were collected from the generator's outputs with binary labels (PASS/FAIL). A post-hoc EXPLAIN audit revealed that 46.3% of FAIL-labeled samples were in fact valid SQL. These were relabeled with EXPLAIN to construct a clean dataset (11,059 samples), and a second critic was trained on the corrected data.

Results: Table 6 evaluates both critics on a leak-free test set (n=300, 0% overlap with training data). The noisy critic shows chance-level performance (fidelity 43%, noise rejection 52%). The clean critic achieves perfect fidelity (100%) but only 20% noise rejection, letting 80% of invalid SQL pass as valid.

This asymmetric failure is informative: the 270M model learns to recognize valid SQL patterns (high fidelity) but cannot distinguish invalid from valid SQL at sufficient resolution (low rejection). In contrast, EXPLAIN deterministically catches all syntactic errors and all invalid column references at no training cost, with zero false negatives.

Table 6. Verifier comparison on leak-free test set

Verifier	Fidelity (%)	Noise rejection (%)
Noisy critic	43 [36, 51]	52 [44, 60]
Clean critic	100 [97, 100]	20 [14, 27]
EXPLAIN	100	100

4.7 Cross-Model validation

To assess whether the findings are specific to Gemma, the fine-tuning pipeline was repeated on Qwen-2.5 3B with the same training data and LoRA configuration (Table 7).

Despite large gaps in base performance (Gemma 38%/34% vs. Qwen 10%/6%), both models converge to the same OOT FCR of 69% after fine-tuning. Gemma retains a small advantage on ID (80% vs.

77%). The shared OOT convergence point suggests that the generalization ceiling is constrained by training data (2,400 pairs over a single schema) rather than by model architecture. It should be noted, however, that two models are insufficient for a general claim; this result is therefore treated as suggestive.

Table 7. Cross-Model validation

Model	Base ID (%)	Base OOT (%)	FT ID (%)	FT OOT (%)
Gemma-3 4B	38	34	80	69
Qwen-2.5 3B	10	6	77	69

4.8 Semantic validation

FCR measures only syntactic validity, not whether the SQL returns meaningful results. Execution success was evaluated by running FCR-valid queries against the database with a 30-second timeout (Table 8).

The fine-tuned ID configuration shows the lowest execution rate (79%), attributable to 17 timeouts from z-score queries. OOT execution rates are consistently higher (92-99%), because OOT queries less frequently require expensive statistical operations. The FT+EXPLAIN configuration achieves 92% OOT execution rate, with 6 timeouts from complex window-function queries.

Table 8. Execution success rate of FCR-Valid queries

Configuration	Valid	Executed	Exec (%)	Timeouts
Base ID	38	37	97	1
Base OOT	34	32	94	2
FT ID	80	63	79	17
FT OOT	69	68	99	1
FT + EXPLAIN ID	90	73	81	17
FT + EXPLAIN OOT	77	71	92	6

V. Limitations

Scale: All experiments use n=100 queries per condition (10 per OOT category), yielding wide confidence intervals (±8-10 pp). The feedback-granularity differences (86-88%) fall within the overlapping CIs.

Schema diversity: This study evaluated only on a single table with a fixed schema. Real-world Text-to-SQL requires multi-table joins, schema linking, and diverse column types; generalization to such settings cannot be assumed without further validation.

FCR vs. semantic correctness: FCR measures only syntactic validity, not whether the SQL returns the intended result. Manual evaluation of output semantics is left as future work.

Benchmark coverage: No evaluation was performed on standard benchmarks such as Spider or BIRD. While questions have been raised about the ecological validity of these benchmarks [28][29], their absence limits direct comparison with existing systems.

VI. Discussion

Feedback is a compute allocation problem. The central finding, that feedback granularity governs convergence speed but not the accuracy ceiling, has direct implications for system design. In latency-constrained on-premises industrial monitoring, structured EXPLAIN feedback cuts expected inference calls by 20%, translating to proportional reductions in latency and compute cost. Conversely, when compute is cheap but engineering effort is scarce, a generic "retry" signal achieves equivalent accuracy with a simpler feedback pipeline.

The effective scope of EXPLAIN retry. The per-category analysis (Table 5) shows that EXPLAIN retry is most effective on column-level errors in structurally simple queries (range: +40 pp, multi-condition: +20 pp), because the EXPLAIN error message directly identifies the offending column and Level C provides valid alternatives. In contrast,

function-level errors (variance: unchanged at 30%) and structural complexity (composite aggregation: unchanged at 40%) involve semantic rather than syntactic errors, for which EXPLAIN feedback is insufficient.

Thought truncation: the ceiling of test-time compute. The error taxonomy shows that EXPLAIN retry corrects errors while simultaneously generating them: thought truncation accounts for 43% (10/23) of residual failures. Each retry appends the previous SQL and feedback to the prompt, consuming generation budget; after 3-4 retries, tokens become insufficient for a complete SQL statement. This exposes a fundamental tension of iterative correction - more retries raise the probability of success but also the probability of truncation. Longer context windows or summary-based feedback compression could mitigate this trade-off.

Label noise and verifier training. The 46.3% false-negative rate in the initial critic training data (discovered via EXPLAIN audit) highlights a broader problem: SQL-validity labels derived from execution-based evaluation are unreliable. Formatting artifacts, timeout limits, and evaluation-pipeline bugs can mislabel valid SQL as invalid. This finding suggests that Text-to-SQL systems relying on execution-based training signals [2] may benefit from EXPLAIN-based label verification as a preprocessing step.

Theoretical extension to other time-series domains. Although this work was validated on electricity data, the components of the proposed method (LoRA fine-tuning, the EXPLAIN0-based feedback loop, and the deterministic verifier) contain no domain-specific assumptions, so its portability to general industrial time-series domains can be discussed theoretically. Two conditions are sufficient for porting: (1) the target domain has a normalized schema composed of time-index and measurement columns, and (2) the database engine provides a deterministic EXPLAIN or an equivalent static check. When both hold, the pipeline structure itself ports without modification, and the domain-adaptation cost is limited to (a) regenerating synthetic training pairs that reflect the domain vocabulary and (b) reconstructing the column list in the schema hint.

Structural isomorphism with the schema of this study is illustrated for two domains. In manufacturing process monitoring, sensor time series (temperature, vibration, throughput) form the same structure as the time-aligned measurement columns of this study (hourly_00 ~ hourly_23), an equipment identifier corresponds to a non-temporal categorical dimension such as industry, and threshold-detection, moving-average, and pattern-matching queries are isomorphic to the OOT categories used here. In logistics demand forecasting, hourly shipment volumes correspond to measurement columns, a regional identifier to a categorical dimension, and queries of the same families (comparison, percentile, year-over-year) become the core analysis targets. In both cases, the central finding of this study (that feedback granularity governs convergence efficiency rather than accuracy) and the deterministic-verification-first principle are expected to apply unchanged. However, this is a theoretical possibility grounded in structural isomorphism; actual performance under differing domain vocabularies and query complexity requires separate empirical validation. This, together with extension to multi-table schemas, is left for future work.

Ⅶ. Conclusions

This work presents an empirical study of EXPLAIN-based iterative correction for SQL generation with a fine-tuned 4B small language model on industrial time-series data. Combining LoRA fine-tuning with structured retry, it achieves 90% ID and 77% OOT format compliance rates.

The central finding is that feedback granularity

governs convergence speed, not the accuracy ceiling: generic, error-only, and structured EXPLAIN feedback all reach equivalent final accuracy (86-88%, $p \geq 0.75$), but structured feedback requires 20% fewer inference calls. In this setting, this characterizes feedback design as a compute-optimization problem rather than an accuracy problem.

The per-category analysis shows that EXPLAIN retry is most effective on column-level errors in simple queries (range: +40 pp) but cannot address function-level gaps (variance) or combinatorial complexity (composite aggregation). Thought truncation driven by accumulating retry context emerges as the dominant residual failure mode, pointing to context management as a core bottleneck in iterative correction.

Minimum parameter scale for SQL validation: the asymmetric failure pattern of the learned 270M critic (achieving 100% fidelity when trained on clean data but only 20% noise rejection, letting 80% of invalid SQL pass) shows that SQL-validity discrimination requires a model's discriminative-reasoning capacity. The 270M model learns the statistical regularities of valid patterns but lacks the representational capacity to learn, in a generalizable form, the semantically diverse invalid cases (column hallucination, unsupported functions, syntax violations). This suggests that the minimum parameter scale for an SLM to serve as a simple classifier for SQL validation exceeds the 270M level of this study, and that where a database engine's static validation (such as deterministic EXPLAIN) is available at no cost, the incentive to introduce a learned critic is weak. At this model scale (<1B), deterministic verification is preferred over a learned critic; introducing a learned verifier would require re-evaluation at a larger scale (e.g., 1.5B or more) beyond the scope of this study.

The paper additionally shows that the fine-tuning gain reproduces across model families through a shared OOT convergence point (69%), indicating that the generalization ceiling is constrained by the amount and diversity of training data rather than by model architecture. These results, while requiring further validation on more complex schemas, suggest a direction for SLM-based SQL generation: invest in fine-tuning data quality, prefer deterministic verification over learned critics at this model scale, and treat feedback granularity as a compute-budget decision rather than an accuracy decision.

Future work can focus on (a) the effectiveness of EXPLAIN feedback under multi-table schemas, (b) feedback-compression techniques to mitigate thought truncation, and (c) the rejection-rate limits of critics at 1B-or-larger scale.

References

[1] X. Chen, M. Lin, N. Schärli, and D. Zhou, "Teaching large language models to self-debug", Proc. International Conference on Learning Representations (ICLR), Vienna, Austria, May 2024. https://doi.org/10.48550/arXiv.2304.05128.

[2] A. Rączkowska, R. Belluzzo, P. Zieliński, J. Baran, and P. Olszewski, "RetrySQL: Text-to-SQL training with retry data for self-correcting query generation", Proc. AAAI Conference on Artificial Intelligence, Singapore, Vol. 40, No. 39, pp. 32773-32781, Jan. 2026. https://doi.org/10.1609/aaai.v40i39.40556.

[3] A. Askari, C. Poelitz, and X. Tang, "MAGIC: Generating self-correction guideline for in-context text-to-SQL", Proc. AAAI Conference on Artificial Intelligence, Philadelphia, USA, Vol. 39, No. 22, pp. 23433-23441, Feb.-Mar. 2025. https://doi.org/10.1609/aaai.v39i22.34511.

[4] C. Snell, J. Lee, K. Xu, and A. Kumar, "Scaling LLM Test-Time Compute Optimally Can be More Effective than Scaling Parameters for Reasoning", Proc. International Conference on Learning Representations (ICLR), Singapore, Apr. 2025.

https://doi.org/10.48550/arXiv.2408.03314.

[5] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, and P. Clark, "Self-Refine: Iterative refinement with self-feedback", Proc. Advances in Neural Information Processing Systems (NeurIPS), New Orleans, USA, Vol. 36, pp. 46534-46594, Dec. 2023. https://doi.org/10.48550/arXiv.2303.17651.

[6] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning", Proc. Advances in Neural Information Processing Systems (NeurIPS), New Orleans, USA, Vol. 36, pp. 8634-8652, Dec. 2023. https://doi.org/10.48550/arXiv.2303.11366.

[7] J. Huang, X. Chen, S. Mishra, H. S. Zheng, A. W. Yu, X. Song, and D. Zhou, "Large language models cannot self-correct reasoning yet", Proc. International Conference on Learning Representations (ICLR), Vienna, Austria, May 2024. https://doi.org/10.48550/arXiv.2310.01798.

[8] R. Kamoi, Y. Zhang, N. Zhang, J. Han, and R. Zhang, "When can LLMs actually correct their own mistakes? A critical survey of self-correction of LLMs", Transactions of the Association for Computational Linguistics (TACL), Vol. 12, pp. 1417-1440, Nov. 2024. https://doi.org/10.1162/tacl_a_00713.

[9] L. Pan, M. Saxon, W. Xu, D. Nathani, X. Wang, and W. Y. Wang, "Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies", Transactions of the Association for Computational Linguistics (TACL), Vol. 12, pp. 484-506, May 2024. https://doi.org/10.1162/tacl_a_00660.

[10] Z. Gou, Z. Shao, Y. Gong, Y. Shen, Y. Yang, N. Duan, and W. Chen, "CRITIC: Large language models can self-correct with tool-interactive critiquing", Proc. International Conference on Learning Representations (ICLR), Vienna, Austria, May 2024. https://doi.org/10.48550/arXiv.2305.11738.

[11] A. Kumar, V. Zhuang, R. Agarwal, Y. Su, J. D. Co-Reyes, A. Singh, K. Baumli, S. Iqbal, C. Bishop, R. Roelofs, L. M. Zhang, K. McKinney, D. Shrivastava, C. Paduraru, G. Tucker, D. Precup, F. Behbahani, and A. Faust, "Training language models to self-correct via reinforcement learning", Proc. International Conference on Learning Representations (ICLR), Singapore, Apr. 2025. https://doi.org/10.48550/arXiv.2409.12917.

[12] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task", Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), Brussels, Belgium, pp. 3911-3921, Oct.-Nov. 2018. https://doi.org/10.18653/v1/D18-1425.

[13] J. Li, B. Hui, G. Qu, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Geng, N. Huo, X. Zhou, C. Ma, G. Li, K. Chang, F. Huang, R. Cheng, and Y. Li, "Can LLM already serve as a database interface? A BIg bench for large-scale database grounded text-to-SQLs", Proc. Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track, New Orleans, USA, Vol. 36, pp. 42330-42357, Dec. 2023. https://doi.org/10.48550/arXiv.2305.03111.

[14] F. Lei, J. Chen, Y. Ye, R. Cao, D. Shin, H. Su, Z. Suo, H. Gao, W. Hu, P. Yin, V. Zhong, C. Xiong, R. Sun, Q. Liu, S. Wang, and T. Yu, "Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows", Proc. International Conference on Learning Representations (ICLR), Singapore, Apr. 2025. https://doi.org/10.48550/arXiv.2411.07763.

[15] M. Pourreza and D. Rafiei, "DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction", Proc. Advances in Neural Information Processing Systems (NeurIPS), New Orleans, USA, Vol. 36, pp. 36339-36348, Dec. 2023. https://doi.org/10.48550/arXiv.2304.11015.

[16] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, and J. Zhou, "Text-to-SQL empowered by large language models: A benchmark evaluation", Proc. VLDB Endowment, Vol. 17, No. 5, pp. 1132-1145, Jan. 2024. https://doi.org/10.14778/3641204.3641221.

[17] G. Qu, J. Li, B. Qin, X. Li, N. Huo, C. Ma, and R. Cheng, "SHARE: An SLM-based hierarchical action CorREction assistant for text-to-SQL", Proc. 63rd Annual Meeting of the Association for Computational Linguistics (ACL) Long Papers, Vienna, Austria, pp. 11268-11292, Jul. 2025. https://doi.org/10.18653/v1/2025.acl-long.552.

[18] M. Pourreza, H. Li, R. Sun, Y. Chung, S. Talaei, G. T. Kakkar, Y. Gan, A. Saberi, F. Ozcan, and S. Arik, "CHASE-SQL: Multi-path reasoning and preference optimized candidate selection in text-to-SQL", Proc. International Conference on Learning Representations (ICLR), Singapore, Apr. 2025. https://doi.org/10.48550/arXiv.2410.01943.

[19] Z. Hong, H. Chen, Z. Yuan, Q. Zhang, L. Zhuang, Q. Liao, F. Huang, Y. Song, and X. Huang, "ErrorLLM: Modeling SQL errors for text-to-SQL refinement", arXiv preprint, Mar. 2026. https://doi.org/10.48550/arXiv.2603.03742.

[20] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models", Proc. International Conference on Learning Representations (ICLR), Virtual, Apr. 2022. https://doi.org/10.48550/arXiv.2106.09685.

[21] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs", Proc. Advances in Neural Information Processing Systems (NeurIPS), New Orleans, USA, Vol. 36, pp. 10088-10115, Dec. 2023. https://doi.org/10.52202/075280-0441.

[22] J. Yang, B. Hui, M. Yang, J. Yang, J. Lin, and C. Zhou, "Synthesizing text-to-SQL data from weak and strong LLMs", Proc. 62nd Annual Meeting of the Association for Computational Linguistics (ACL) Long Papers, Bangkok, Thailand, pp. 7864-7875, Aug. 2024. https://doi.org/10.18653/v1/2024.acl-long.425.

[23] P. Ghosh, A. Jain, and P. Yenigalla, "SQLGenie: A practical LLM based system for reliable and efficient SQL generation", Proc. 63rd Annual Meeting of the Association for Computational Linguistics (ACL) Industry Track, Vienna, Austria, pp. 1004-1012, Jul. 2025. https://doi.org/10.18653/v1/2025.acl-industry.71.

[24] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe, "Let's verify step by step", Proc. International Conference on Learning Representations (ICLR), Vienna, Austria, May 2024. https://doi.org/10.48550/arXiv.2305.20050.

[25] A. Ni, S. Iyer, D. Radev, V. Stoyanov, W. Yih, S. I. Wang, and X. V. Lin, "LEVER: Learning to verify language-to-code generation with execution", Proc. International Conference on Machine Learning (ICML), Honolulu, USA, PMLR 202, pp. 26106-26128, Jul. 2023. https://doi.org/10.48550/arXiv.2302.08468.

[26] P. Wang, L. Li, Z. Shao, R. Xu, D. Dai, Y. Li, D. Chen, Y. Wu, and Z. Sui, "Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations", Proc. 62nd Annual Meeting of the Association for Computational Linguistics (ACL) Long Papers, Bangkok, Thailand, pp. 9426-9439, Aug. 2024. https://doi.org/10.18653/v1/2024.acl-long.510.

[27] R. Dror, G. Baumer, S. Shlomov, and R.

Reichart, "The hitchhiker's guide to testing statistical significance in natural language processing", Proc. 56th Annual Meeting of the Association for Computational Linguistics (ACL) Long Papers, Melbourne, Australia, pp. 1383-1392, Jul. 2018. https://doi.org/10.18653/v1/P18-1128.

[28] S. Chang, J. Wang, M. Dong, L. Pan, H. Zhu, A. H. Li, W. Lan, S. Zhang, J. Jiang, J. Lilien, S. Ash, W. Y. Wang, Z. Wang, V. Castelli, P. Ng, and B. Xiang, "Dr.Spider: A diagnostic evaluation benchmark towards text-to-SQL robustness", Proc. International Conference on Learning Representations (ICLR), Kigali, Rwanda, May 2023. https://doi.org/10.48550/arXiv.2301.08881.

[29] X. Liu, S. Shen, B. Li, N. Tang, and Y. Luo, "NL2SQL-BUGs: A benchmark for detecting semantic errors in NL2SQL translation", Proc. ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), Toronto, Canada, pp. 5662-5673, Aug. 2025. https://doi.org/10.1145/3711896.3737427.

Authors

Byounghoon Son

2026. 2 : BS degree, Dept. of Computer Engineering, Hanbat National University

Research interests: Data Analysis, AI Agent, Energy Management

Minsung Jung

2021. 3 ~ Present : BS candidate, Dept. of Computer Engineering, Hanbat National University

Research interests: Data Analysis, Energy Management, Machine Learning, Deep Learning

Sangkeum Lee

2016. 8 : BS degree, Dept. of Electronics and Information Engineering, Korea University

2018. 2 : MS degree, CCS Graduate School of Green Transportation, Korea Advanced Institute of Science and Technology (KAIST)

2020. 8 : Ph.D. degree, CCS Graduate School of Green Transportation, KAIST

2020. 9 ~ 2021. 2 : Postdoctoral Researcher, Mechanical Engineering Research Institute, KAIST

2021. 3 ~ 2023. 8 : Senior Researcher, Energy & Environment ICT Research Department, Electronics and Telecommunications Research Institute (ETRI)

2023. 9 ~ 2026. 3 : Assistant Professor, Dept. of Computer Engineering, Hanbat National University

2026. 4 ~ Present : Associate Professor, Dept. of Computer Engineering, Hanbat National University

Research interests: Deep Learning, Reinforcement Learning, Sensor Networks, Smart Energy Management