ATLAS: Adaptive Text-to-SQL with Lifecycle-Aware Self-Maintaining Context

Qing Zhang[†]
Fudan University
Shanghai, China
23210240380@m.fudan.edu.cn

Shijing Hu
Fudan University
Shanghai, China
sjhu24@m.fudan.edu.cn

Zhihui Lu[*]
Fudan University
Shanghai, China
lzh@fudan.edu.cn

ABSTRACT

Deploying Text-to-SQL in production is hampered by context-window limits on large schemas, metadata that goes stale as schemas evolve, and infrastructure sprawl from external vector stores. We demonstrate **ATLAS**, which addresses all three by co-locating schema metadata, semantic annotations, and vector embeddings entirely within a single RDBMS with native vector-index support. ATLAS contributes: (1) *Unified in-database storage* with native vector indexes, requiring no external engine; (2) *Two-stage adaptive schema linking* that uses vector retrieval to narrow hundreds of tables to a compact candidate set before LLM refinement; (3) *Rich Context lifecycle* that automatically generates semantic descriptions, synonyms, sample values, business rules, and value mappings, bridging raw schema and LLM understanding; (4) *Agent-driven self-maintenance* via a coordinator–executor loop that detects DDL changes and regenerates annotations without manual intervention. We showcase three live scenarios on an enterprise database with 517 tables.

PVLDB Reference Format:
Qing Zhang, Shijing Hu, and Zhihui Lu. ATLAS: Adaptive Text-to-SQL with Lifecycle-Aware Self-Maintaining Context. PVLDB, 19(12): 4838 - 4841, 2026.
doi:10.14778/3827998.3828135

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/zqzqsb/atlas.

1 INTRODUCTION

Text-to-SQL systems translate natural language into SQL, enabling non-technical users to query databases. Recent LLM-based approaches [2, 9] achieve impressive benchmark accuracy, yet production fintech deployments reveal three persistent challenges: **(C1) Schema understanding at scale**—databases with hundreds of tables overwhelm LLM context windows, and existing methods either include all tables or use string matching; **(C2) Stale metadata**—production schemas evolve via DDL changes, causing manually curated metadata to fall out of sync; **(C3) Infrastructure sprawl**—current systems require separate vector databases, caches, and RDBMS, tripling operational complexity.

We present **ATLAS** (**A**daptive **T**ext-to-SQL with **L**ifecycle-**A**ware **S**elf-maintaining context), which addresses all three through a unified architecture where schema, context, and vectors live inside a single RDBMS with native vector support (MariaDB 12 in our prototype). ATLAS contributes: **(1) Unified in-database storage** (§2.1)—schema, semantic descriptions, and vector embeddings share dedicated context tables with native HNSW indexes, requiring no external store; **(2) Two-stage adaptive schema linking** (§2.2)—vector retrieval narrows large schemas to a compact candidate set before LLM refinement; **(3) Rich Context lifecycle** (§2.3)—a ReAct agent automatically generates semantic descriptions, synonyms, sample values, business rules, and value mappings, bridging the gap between raw schema and LLM understanding; **(4) Agent-driven self-maintenance** (§2.4)—a coordinator–executor loop detects DDL changes, marks stale context, and regenerates annotations without manual intervention. We demonstrate ATLAS through three live scenarios (§3) on a 517-table enterprise schema.

2 SYSTEM ARCHITECTURE

Our central design principle is that schema, semantic annotations, and vector embeddings should all be first-class citizens of the host RDBMS under ACID guarantees. Figure 1 shows the architecture: three pipelines—Onboarding, Inference, and Self-Maintenance—share a unified in-database storage layer.

2.1 Unified In-Database Storage

ATLAS stores all metadata in dedicated context tables (`rc_*`) within the host RDBMS. These cover table and column descriptions, foreign keys, business terminology, business rules and value mappings, vector embeddings with HNSW index, and a change audit trail—seven tables in total.

The key design is *co-locating vectors with relational data*. Native `VECTOR` columns and built-in distance functions enable a single SQL query to combine vector similarity with relational filters, eliminating external vector stores while preserving transactional consistency. Co-location also avoids cross-store sync lag and dual-write inconsistency: a DDL change and its embedding refresh commit in one transaction. Our prototype uses MariaDB 12 [8], but the design applies to any RDBMS with native vector support (e.g., pgvector [3]).

[*]Corresponding author.
[†]This work was done while the author was an intern at Tencent Cloud.

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828135

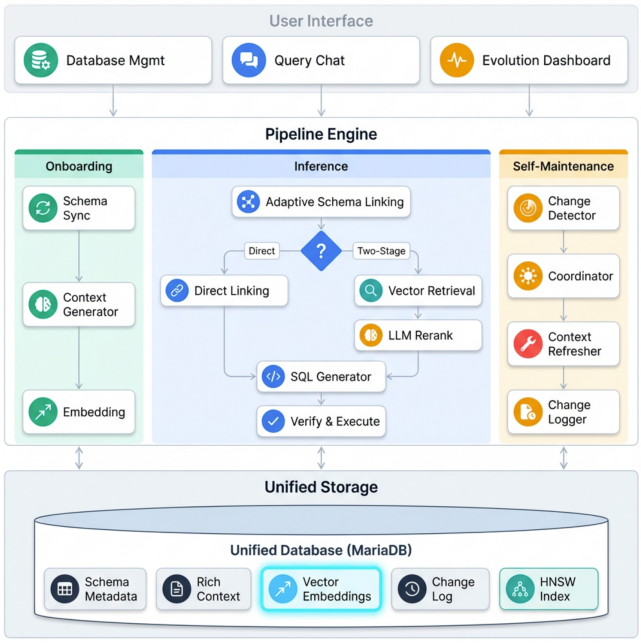

Figure 1: ATLAS architecture. Three pipelines share unified in-database storage.

2.2 Two-Stage Adaptive Schema Linking

Given a query q and a datasource with N tables, ATLAS selects a linking strategy adaptively based on schema scale:

Small-scale: When the full schema fits within the LLM context window, all table schemas (with Rich Context) are sent directly to the LLM agent.

Large-scale: A concurrent architecture (Figure 2) overlaps vector retrieval with LLM reasoning: a retrieval thread embeds q and searches `rc_embeddings` while the LLM agent begins reasoning; results are exchanged via an *atomic slot*—a lock-free shared pointer that the retrieval thread writes and the agent polls—hiding retrieval latency entirely.

2.3 Rich Context Lifecycle

Raw schema metadata alone is often insufficient for accurate Text-to-SQL: abbreviated names, implicit business rules, and vocabulary gaps between user language and physical names all cause LLM errors. ATLAS bridges this gap with *Rich Context*—LLM-generated annotations comprising (i) descriptions, (ii) synonyms, (iii) sample values, (iv) *business rules* (implicit domain constraints and conventions, e.g., that an order is active only while `status='A'`), and (v) *value mappings* (correspondences between user vocabulary or codes and physical column values)—all stored and versioned within the host RDBMS.

Rich Context passes through three lifecycle stages:

Onboarding. A ReAct agent [13] samples rows, queries metadata, and writes all five context types via a batch tool. For large schemas, a *forest-based chunked* strategy treats each connected component of the undirected FK graph as a subtree (batching FK-isolated tables together) and processes subtrees in parallel, with per-subtree iteration budgets scaled to their table count.

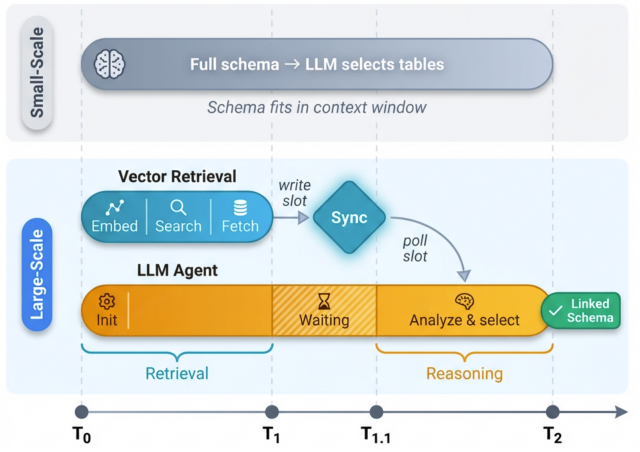

Figure 2: Two-stage adaptive schema linking with concurrent vector retrieval and LLM reasoning.

Inference. Grounded tables together with their Rich Context are injected into the LLM prompt; the agent generates SQL with iterative self-correction.

Evolution. DDL changes trigger the self-maintenance pipeline (§2.4).

2.4 Agent-Driven Self-Maintenance

A *coordinator–executor* architecture monitors schema evolution: (1) **DDL Detector** polls `information_schema` and diffs against context tables; (2) **Coordinator** marks affected entries as stale; (3) **Executor** invokes LLM to regenerate descriptions and re-embed; (4) **Change Logger** records all modifications. This closed-loop keeps Rich Context synchronized without manual intervention.

3 DEMONSTRATION

We demonstrate ATLAS as an interactive web application where attendees type natural language queries, trigger schema onboarding, and observe self-maintenance against live databases with up to hundreds of tables. Unlike existing Text-to-SQL demos that assume static, pre-curated metadata [11, 12], ATLAS demonstrates a *complete lifecycle*: Rich Context is generated, consumed, and autonomously refreshed within a single RDBMS.

Setup. ATLAS is deployed as three Docker containers (RDBMS + backend service + web frontend) with LLM and embedding endpoints configurable via OpenAI-compatible APIs. Figure 3 shows the interface.

Scenario 1: Onboarding a New Database. The user connects the 517-table TPC-H Enterprise database, clicks *Sync Schema* then *Generate Rich Context*, and observes the forest-chunked pipeline: the FK graph is decomposed into subtrees shown as a live treemap, each processed by a ReAct agent—requiring zero domain expertise.

Scenario 2: Two-Stage Adaptive Query. On the same database, the user asks a cross-domain analytical question. Vector search

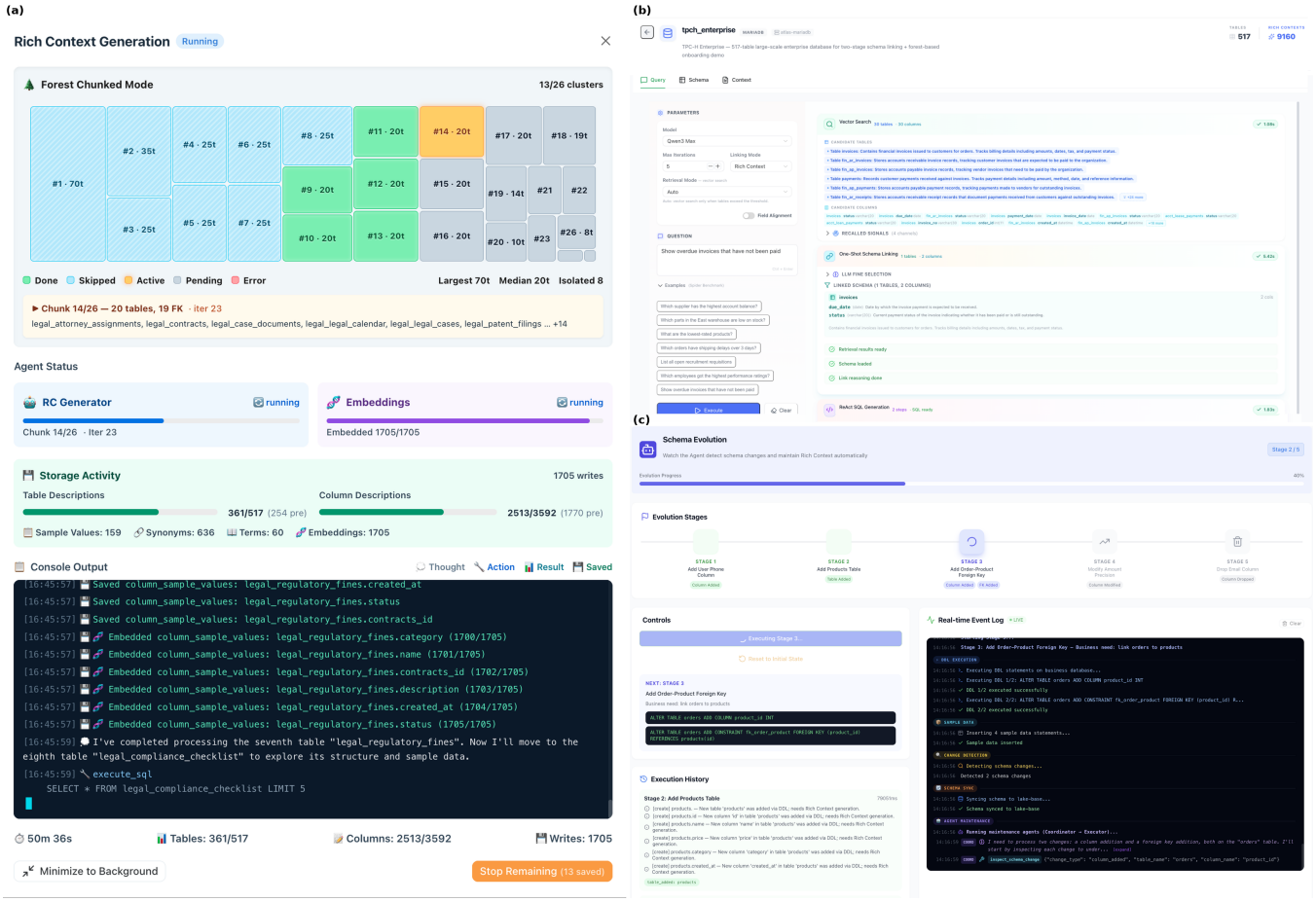

Figure 3: ATLAS demo interface. (a) Onboarding with forest-chunked treemap. (b) Adaptive query with schema linking and ReAct SQL generation. (c) Autonomous schema evolution with before/after diffs.

retrieves ~20 candidate tables in about one second; second-stage LLM refinement then selects the relevant set, with Rich Context supplying the semantic cues for disambiguation. ReAct generation with a `verify_sql` dry-run produces the final query, demonstrating the coarse-to-fine pipeline at enterprise scale.

Scenario 3: Autonomous Schema Evolution. Starting from a 2-table database, the user applies scripted DDL changes in stages. After each stage, clicking *Detect Changes* triggers the self-maintenance pipeline: detected diff → stale marking → LLM context refresh → embedding update → change log with before/after diffs—all without manual re-curation.

4 PRELIMINARY EVALUATION

We evaluate ATLAS from two angles: *SQL accuracy* on a public benchmark, and *system-level ablation* on a large-scale enterprise schema. All experiments use Qwen3-Max as the LLM backend and a 2,048-dimensional text embedding model for vector search; the system accepts any OpenAI-compatible endpoint.

SQL accuracy on BIRD. ATLAS achieves **76.40%** execution accuracy (EX) on the BIRD development set (11 databases, 1,534 questions) [7], ranking among the top entries on the public BIRD dev leaderboard, using a single LLM call per iteration and no ensemble or voting. A component-level ablation of Rich Context (semantic descriptions, synonyms, sample values, business rules, and value mappings) is provided in our public repository.

System-level ablation on TPC-H Enterprise. To evaluate ATLAS's system components under enterprise conditions, we construct a TPC-H Enterprise variant with over 500 tables spanning 21 business domains. Starting from the 8 standard TPC-H tables, we synthesize each domain (HR, finance, supply chain, etc.) as a hub-and-spoke schema under a fixed random seed; the generator is released for reproducibility. Table 1 reports the impact of removing each system component on a curated set of 30 cross-domain analytical questions.

Vector retrieval contributes the most to accuracy (−26.6 pp recall, −20 pp EX when removed); Rich Context affects both metrics (−13.3 pp recall, −16.7 pp EX), Adaptive Linking is indispensable (timeout without it), and disabling ReAct trades 13.3 pp EX for 2× lower latency.

Table 1: System-level ablation on TPC-H Enterprise (500+ tables).

Configuration	Recall@20	EX (%)	Latency (s)
Full ATLAS pipeline	**93.3**	**70.0**	4.8
– Adaptive Linking	—†	—†	timeout
– Vector Search	66.7	50.0	5.6
– ReAct Loop	93.3	56.7	2.3
– Rich Context	80.0	53.3	4.9

†Direct linking on 500+ tables exceeds the LLM context window and cannot produce results.

5 RELATED WORK

LLM-based Text-to-SQL. Recent surveys [10] categorize approaches into prompting and fine-tuning paradigms. DIN-SQL [9] decomposes the task with self-correction; DAIL-SQL [2] studies example selection; MAC-SQL [12] uses multi-agent collaboration; CHESS [11] enriches schema context with catalog information. These methods treat schema metadata as static input and do not address metadata maintenance or infrastructure consolidation. Unlike CHESS, which retrieves catalog descriptions from an external vector database built once during preprocessing, ATLAS generates and versions Rich Context inside the host RDBMS and self-maintains it as the schema evolves.

Schema linking and retrieval. Traditional linking relies on string matching [4] or learned rankers. RESDSQL [5] ranks schemas by cross-encoder scores; CodeS [6] uses coarse-to-fine BM25 retrieval. These require external retrieval services, whereas ATLAS performs linking entirely within the host RDBMS using native `VECTOR` columns and HNSW indexes.

Metadata management and native vector search. Metadata catalogs such as Apache Atlas [1] manage human-readable documentation but do not generate machine-readable annotations for LLM consumption; ATLAS's self-maintenance pipeline bridges this gap while keeping annotations human-readable and auditable via an interface. On the vector side, pgvector [3] adds vector support to PostgreSQL and MariaDB 12 [8] provides native `VECTOR` columns with HNSW indexing. ATLAS sits at the intersection of these three lines, unifying context-aware linking, in-database vector search, and automated metadata maintenance end-to-end without external dependencies.

6 CONCLUSION

We presented ATLAS, a Text-to-SQL system that co-locates schema metadata, Rich Context annotations, and vector embeddings within a single RDBMS, scaling to 500+ table enterprise databases while achieving 76.40% EX on BIRD (dev) [7] without ensemble or voting. Three live scenarios demonstrate the complete lifecycle from onboarding through query to schema evolution. Future work will incorporate query feedback to further refine Rich Context and support federation across multiple database instances, ensuring that the same in-database annotations stay consistent under schema evolution and remain available to additional agent-facing interfaces.

ACKNOWLEDGMENTS

This work was supported in part by Fudan University and conducted in collaboration with the TDSQL team at Tencent Cloud. This work powers the semantic layer of TDSQL Nexa, Tencent Cloud's AI-native data platform, and continues to evolve toward richer agent-facing capabilities. We also thank the Yangtze River Delta Science and Technology Innovation Community Joint Research Project (YDZX20233100004031) and the National Natural Science Foundation of China (72595845, 72595840). We thank the MariaDB community for early access to native vector search capabilities.

REFERENCES

[1] Apache Software Foundation. 2023. Apache Atlas – Data Governance and Metadata Framework. https://atlas.apache.org/. Accessed: 2026-06.
[2] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yue Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *Proceedings of the VLDB Endowment* 17, 5 (2024), 1132–1145.
[3] Andrew Kane. 2023. pgvector: Open-source vector similarity search for Postgres. https://github.com/pgvector/pgvector. Accessed: 2026-06.
[4] Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the Role of Schema Linking in Text-to-SQL. In *Proceedings of EMNLP*. Association for Computational Linguistics, Online, 6943–6954.
[5] Haoyang Li, Jing Hui, Jian Qu, Jian Cao, Jian Jiang, Feifei Gao, and Ce Yang. 2023. RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL. *Proceedings of AAAI* 37, 11 (2023), 13067–13075.
[6] Haoyang Li, Jing Hui, Jian Qu, Jian Cao, Jian Jiang, Feifei Gao, and Ce Yang. 2024. CodeS: Towards Building Open-source Language Models for Text-to-SQL. *Proceedings of SIGMOD* 3, 2 (2024).
[7] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2023. Can LLM Already Serve as a Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs. In *Proceedings of NeurIPS*. Curran Associates, New Orleans, LA.
[8] MariaDB Corporation. 2025. MariaDB 12.0 Vector Search. https://mariadb.com/kb/en/vector-overview/. Accessed: 2026-06.
[9] Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. *Proceedings of NeurIPS* 36 (2023).
[10] Liang Shi, Zhengju Tang, Nan Zhang, Xiaotong Zhang, and Zhi Yang. 2025. A Survey on Employing Large Language Models for Text-to-SQL Tasks. *Comput. Surveys* 58, 2 (2025), 54:1–54:37.
[11] 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* (2024).
[12] Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Linzheng Zhang, Zhao Yan, and Zhoujun Liu. 2024. MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL. *Findings of ACL* (2024).
[13] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. *Proceedings of ICLR* (2023).