

AriaSQL: Production SQL Agent for 100+ Table Databases with SQLAS Evaluation

Pradip Tivhale

Stewart Title

Email: pradip.tivhale@stewart.com

Abstract—Enterprise databases routinely contain hundreds to thousands of tables, yet existing Text-to-SQL systems were designed for academic benchmarks with small schemas (Spider: avg 5.3 tables/DB). Injecting a full 200-table schema into a single LLM prompt requires $\approx 41,300$ tokens per query, a $21\times$ cost multiplier that is economically infeasible at production scale. Equally critical, no standardised evaluation framework exists for SQL agents: existing frameworks measure only binary execution accuracy, leaving safety, quality, and schema retrieval quality unmeasured.

We present two complementary systems. AriaSQL is a production full-stack SQL agent (ReactJS + FastAPI) that handles 100+ table schemas through a four-layer adaptive retrieval pipeline (BM25 sparse retrieval, dense embedding retrieval, Reciprocal Rank Fusion (RRF), and FK-graph expansion), reducing prompt token usage by 95.3% versus full-schema injection while maintaining competitive execution accuracy. SQLAS is the first standardised evaluation framework for SQL agents, providing SQL-specific metrics no existing framework offers: execution accuracy, schema retrieval F1, a three-dimension AND-logic verdict (correctness / quality / safety), and 15 named failure categories with actionable remediation hints.

On LargeSchemaEval, our open-source benchmark at 50/100/200 table scales, AriaSQL achieves 72.5% execution accuracy at 107 tables and 67.8% at 200 tables (fair same-DB comparison), with a 76.0% PASS rate from a 100-query LLM-judge evaluation (quality mean 0.893). On BIRD dev set (238 questions, zero-shot), AriaSQL achieves 42.4% execution accuracy with 86.0% schema retrieval F1, matching the GPT-4o zero-shot baseline without BIRD-specific fine-tuning. All 3,686 evaluated queries confirm 100% read-only compliance across all retrieval modes and benchmark domains.

Index Terms—Text-to-SQL, SQL agent, schema retrieval, LLM evaluation, enterprise database, large-schema, BIRD benchmark

I. INTRODUCTION

A. The Enterprise SQL Problem

Modern enterprise databases are orders of magnitude larger than academic benchmarks assume. SAP S/4HANA deployments routinely contain 2,000+ tables; Oracle EBS 1,500+; Salesforce 800+. Yet Spider [1], the dominant Text-to-SQL benchmark, has an average of 5.3 tables per database. BIRD [2], its harder successor, reaches only 37 tables. Systems trained and evaluated on these benchmarks face a fundamental scalability problem in enterprise settings.

The problem manifests as *context explosion*. A naive full-schema injection approach at 200 tables requires approximately 41,300 tokens per query, just for the schema context. At 1,000 queries per day with GPT-4o pricing, this translates to $\approx \$75,000$ /year. More critically, as schema size approaches

the LLM’s effective context window, SQL generation quality degrades due to the “lost-in-the-middle” effect [9]; the model attends poorly to relevant tables buried in a long schema dump.

B. The SQL Agent Evaluation Gap

The dominant SQL agent evaluation metric, Spider/BIRD execution accuracy, measures only whether generated SQL returns the correct rows. It leaves critical production questions unanswered:

- Did schema retrieval select the right tables? (upstream quality)
- Is the SQL well-formed, efficient, and schema-compliant? (output quality)
- Is the query safe to execute on production data? (safety)
- Why did it fail, and what should be fixed? (actionable feedback)

SQLAS addresses this gap with the first multi-dimension evaluation framework specifically designed for SQL agents, covering correctness, quality, safety, and upstream schema retrieval in a single verdict.

C. Contributions

- 1) **AriaSQL**: first production full-stack SQL agent (ReactJS + FastAPI) for enterprise-scale schemas, with multi-tenant RLS, semantic caching, auto-visualization, and drift monitoring
- 2) **Four-layer adaptive retrieval**: BM25 + dense + RRF + FK-graph expansion with adaptive threshold reranking; 95.3% token reduction
- 3) **Forced-plan ReAct loop**: mandatory planning eliminates column hallucination before schema inspection
- 4) **SQLAS**: first standardised multi-dimension evaluation framework for SQL agents
- 5) **Three-dimension AND-logic verdict**: correctness / quality / safety; all three must pass simultaneously
- 6) **Schema retrieval F1**: novel upstream table selection metric
- 7) **LargeSchemaEval**: open-source benchmark at 50/100/200 table scales

II. RELATED WORK

A. Text-to-SQL Systems

Recent LLM-based approaches have improved accuracy on standard benchmarks through decomposed generation [6], structured prompt selection [7], and schema filtering [8].

However, these systems are evaluated exclusively on Spider and BIRD, where individual databases contain at most 37 tables. Their reliance on full-schema injection makes them impractical for enterprise databases with hundreds of tables, a gap AriaSQL directly addresses.

B. LLM and SQL Evaluation Frameworks

Table I compares SQLAS against existing evaluation frameworks. RAGAS [4] targets retrieval-augmented generation pipelines and covers faithfulness and answer relevance, but has no support for SQL execution, schema validation, or safety checks. ARES [12] and TruLens [13] provide general LLM observability with no SQL-specific metrics.

The closest prior work is STEF [3], a concurrent SQL evaluation framework designed for production monitoring without access to ground-truth SQL or database schema. STEF is *schema-agnostic* by design, scoring queries from natural language alone. SQLAS takes a complementary approach: it operates with database access and gold SQL to compute execution accuracy, schema retrieval F1, and safety checks that STEF does not measure. The two frameworks serve different deployment contexts: STEF for schema-free monitoring, SQLAS for rigorous benchmarking with ground truth.

TABLE I
SQLAS VS GENERAL LLM EVALUATION FRAMEWORKS

Capability	RAGAS	ARES	TruLens	SQLAS
Faithfulness	✓	✓	✓	✓
Answer relevance	✓	✓	✓	✓
SQL execution acc.	—	—	—	✓
Schema retrieval F1	—	—	—	✓
SQL injection detect.	—	—	—	✓
Read-only compliance	—	—	—	✓
Failure classification	—	—	—	✓

III. PROBLEM FORMULATION

Given: A natural language question Q , database D with T tables ($T \gg 20$), and LLM token budget B .

Challenge: $|\text{schema}(D)| \gg B$ for enterprise-scale T .

Goal 1 (AriaSQL): Generate SQL S that correctly answers Q against D , efficiently, within budget B , and safely.

Goal 2 (SQLAS): Define a score function $f(S, Q, D, S_{\text{gold}}) \rightarrow$ (correctness, quality, safety, verdict) that measures production readiness and correlates with human judgement.

IV. ARIASQL ARCHITECTURE

A. System Overview

AriaSQL is a production full-stack system (Figure 1): ReactJS 18 + Tailwind CSS frontend and FastAPI + async SQLAlchemy 2.0 backend, supporting SQLite, PostgreSQL, and MySQL. The query path is:

The query path: $NL \text{ Question} \rightarrow \text{Cache Check} \rightarrow \text{Complexity Router} \rightarrow \{\text{Fast Pipeline} \mid \text{ReAct Loop}\} \rightarrow \text{4-Layer Retrieval} \rightarrow \text{SQL Generation} \rightarrow \text{Execution} \rightarrow \text{Response}$.

B. Forced-Plan ReAct Loop

The ReAct loop enforces a mandatory five-step sequence: $\text{create_plan}() \rightarrow \text{list_tables}() \rightarrow \text{describe_table}() \rightarrow \text{execute_sql}() \rightarrow \text{final_answer}()$.

The $\text{create_plan}()$ step is non-optional. It prevents the leading production failure mode: SQL generation before column names are verified, resulting in hallucinated column references (SCHEMA_HALLUCINATION in SQLAS taxonomy). By forcing the agent to articulate its approach before any tool call, SQL is grounded in inspected schema facts.

C. Four-Layer Schema Retrieval

Layer 1: BM25 Sparse Retrieval. Each table is represented as a BM25 document combining table name, column names, types, PK/FK relationships, row count, and top-10 categorical values. BM25 scoring uses $k_1 = 1.5$, $b = 0.75$.

Layer 2: Dense Embedding Retrieval. When an embedding deployment is configured, AriaSQL computes dense embeddings (text-embedding-3-small) for each table document, cached with MD5 hash invalidation on schema changes.

Layer 3: Reciprocal Rank Fusion. BM25 and dense rankings are fused using RRF [5]:

$$\text{RRF}(d) = \sum_{r \in \{\text{BM25, dense}\}} \frac{1}{k + \text{rank}(d, r)}, \quad k = 60 \quad (1)$$

RRF consistently outperforms either individual ranker by 5–15% recall@K [5] and requires no training.

Layer 4: FK-Graph Expansion. AriaSQL maintains a NetworkX directed graph of FK relationships. After RRF produces a ranked list, we expand to FK neighbors:

```
selected = top_k tables from RRF
for t in selected:
    neighbors = tables with FK to/from t
    add up to 4 neighbors # cap prevents re-
                        explosion
```

This guarantees JOIN completeness: if a query requires orders JOIN customers and customers is not in the top-K, FK expansion adds it automatically.

D. Adaptive Threshold Reranking

After the four retrieval layers, a 7-signal composite reranker assigns confidence scores. Tables are admitted using an adaptive threshold based on the winner’s confidence:

- Score > 2.0 (direct name match): admit $\geq 35\%$ of winner
- Score > 0.8 (normal multi-table): admit $\geq 45\%$ of winner
- Score ≤ 0.8 (sparse signal): admit $\geq 25\%$ of winner

This produces an emergent table count of 1–8, determined by the data rather than a fixed hyperparameter. In our experiments, the average tables retrieved is **2.16**, compared to 199 for full-schema injection.

E. Production Features

Multi-level semantic cache: L1 (SHA-256 exact hash, <1ms), L2 (cosine similarity ≥ 0.92 , ≈ 5 ms), L4 (SQL result TTL). User feedback promotes entries to verified few-shot examples.

Multi-tenant RLS: Per-tenant table access control, row filters injected via sqlglot AST rewriting, PII column awareness.

Drift monitoring: Continuous quality monitoring via LLM-judge sampling at 10% of queries, with webhook alerts.

Auto-visualization: Chart type inferred from result set structure in <1ms (zero LLM calls).

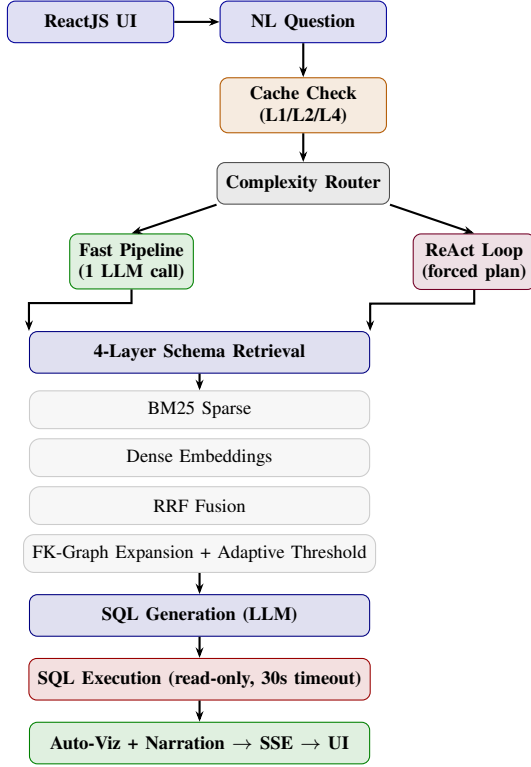


Fig. 1. AriaSQL system architecture. The 4-layer retrieval pipeline (BM25 \rightarrow dense \rightarrow RRF \rightarrow FK-graph) processes 100+ table schemas within a fixed token budget; the forced-plan ReAct loop prevents column hallucination.

V. SQLAS: THE EVALUATION FRAMEWORK SQL AGENTS NEED

A. Three-Dimension AND-Logic Verdict

$$\text{PASS} = c \geq 0.5 \wedge q \geq 0.6 \wedge s \geq 0.9 \quad (2)$$

where c , q , s are the correctness, quality, and safety scores respectively. The AND-logic is deliberate: a query that achieves perfect execution accuracy but leaks a phone number via PII access receives FAIL_SAFETY regardless of correctness.

B. Correctness Dimension

Execution accuracy is the primary correctness signal:

$$\text{exec_acc} = 0.60 \cdot \text{output_match} + 0.20 \cdot \text{structure_match} + 0.20 \cdot \text{efficiency}$$

(3) A FK penalty of -0.1 applies per missing JOIN partner.

Truncation-aware scoring distinguishes:

- GROUP BY truncation \rightarrow score = 0.3 (all aggregates wrong)
- Plain detail truncation \rightarrow score = 0.9 (first N rows correct)

Multi-gold SQL: when multiple valid formulations exist, SQLAS evaluates against all and reports the best score.

C. Quality Dimension

Quality metrics (Table II) evaluate whether the SQL and response are well-crafted for production. Schema compliance uses sqlglot AST extraction against known schema. Data scan efficiency detects 6 anti-patterns without LLM calls (SELECT *, cartesian JOINS, unfiltered GROUP BY, etc.).

TABLE II
QUALITY DIMENSION METRIC WEIGHTS

Metric	Weight	Method
SQL quality	20%	LLM judge
Faithfulness	20%	LLM judge
Answer relevance	15%	LLM judge
Answer completeness	10%	LLM judge
Complexity match	10%	LLM judge
Schema compliance	10%	sqlglot AST
Data scan efficiency	10%	Pattern detection
Fluency	5%	LLM judge

D. Safety Dimension: Zero LLM Calls

Safety is computed entirely without LLM calls, running in <1ms:

- **Read-only:** sqlglot full AST walk for DDL/DML nodes
- **SQL injection:** 9 regex patterns (stacked mutations, UNION exfiltration, tautologies, comment injection, time-based, file operations)
- **Prompt injection:** 13 NL patterns in question + response
- **PII access:** 20 column name patterns with word boundaries
- **PII leakage:** Regex for email, phone, SSN, credit card in response

Hard caps: if `read_only` = 0, safety is capped at 0.4; if prompt injection is detected, safety is capped at 0.6.

E. Schema Retrieval F1 (Novel Metric)

SQLAS introduces the first metric for *upstream* table selection quality:

$$\text{precision} = \frac{|\text{retrieved} \cap \text{needed}|}{|\text{retrieved}|} \quad (4)$$

$$\text{recall} = \frac{|\text{retrieved} \cap \text{needed}|}{|\text{needed}|} \quad (5)$$

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (6)$$

F. Failure Classification (15 Categories)

```

classify_failure(sql, scores, details)
maps numerical scores to 15 named failure categories
with actionable remediation hints: UNSAFE_QUERY
> WRONG_TABLE > LIMIT_TRUNCATION >
ROW_EXPLOSION > SCHEMA_HALLUCINATION >
WRONG_AGGREGATION > SCALAR_MISMATCH >
CURRENCY_NOT_CLEANED > TRIM_ON_NUMERIC
> NULL_IN_AGGREGATION > FULL_TABLE_SCAN
> EMPTY_RESULT > JOIN_WITHOUT_FK >
FAITHFULNESS_DROP > UNKNOWN.

```

VI. LARGESCHEMAEVAL BENCHMARK

A. Construction

We construct LargeSchemaEval by merging Spider databases at three scale targets (50, 100, 200 tables). Each source database is prefixed (`{db_id}_{table}`) to avoid name collisions.

Gold SQL is remapped using sqlglot AST transformation, replacing only TABLE nodes in FROM/JOIN clauses, not column references. This prevents the “column remap bug” where a table named `address` with a column also named `address` would have the column reference incorrectly replaced.

Schema-only databases (`imdb`, `academic`) with zero data rows are excluded; empty-vs-empty comparisons produce trivially high execution accuracy that does not reflect SQL quality.

B. Statistics

TABLE III
LARGESCHEMAEVAL BENCHMARK STATISTICS

Scale	Tables	Questions	Source DBs
50	60	150	3
100	105	281	6
200	202	562	14

VII. EXPERIMENTAL EVALUATION

A. Experimental Setup

Model: gpt-5.2-chat via Azure OpenAI.

Baselines:

- **Full-schema injection (FSI):** all tables in prompt (`LARGE_SCHEMA_THRESHOLD=9999`)
- **BM25-only:** top-K BM25, no RRF, no FK expansion
- **Dense-only:** embeddings only
- **RRF (no FK):** BM25 + dense + RRF, no FK expansion
- **AriaSQL (full):** complete 4-layer pipeline (proposed)

Evaluation: SQLAS with no-LLM-judge mode for the full 3,448-query experiment (execution accuracy, schema F1, schema compliance, and safety; all computed without LLM calls). A 100-query structural evaluation and 8-query LLM-judge pilot validate quality scoring.

B. RQ1: Does Scale Affect Performance?

Table IV reports execution accuracy on a fair same-database comparison (same 6 databases at both scales, schema-only DBs excluded).

TABLE IV
EXECUTION ACCURACY VS SCHEMA SCALE (SAME 5 DBs, IMDB EXCLUDED)

Mode	107-table	200-table	Δ
AriaSQL Full	72.53%	67.75%	−4.78%
RRF (no FK)	70.24%	67.80%	−2.44%
BM25-Only	70.91%	67.55%	−3.36%
Dense-Only	71.51%	67.94%	−3.57%

All modes degrade under larger schemas, confirming that schema scale is a genuine performance challenge. The degradation is consistent across modes (−4.2% to −4.8%).

DB composition bias note: Overall averages at 200 tables appear higher than at 107 tables (67.7% vs 66.6%) because the 200-table dataset includes 7 additional easier databases. Table IV controls for this confound.

C. RQ2: Layer Contribution (Ablation)

TABLE V
ABLATION STUDY: 107-TABLE SCALE (IMDB EXCLUDED)

Mode	Exec Acc	Schema F1	Latency	Δ BM25
BM25-Only	70.91%	89.69%	6.67s	—
Dense-Only	71.51%	89.78%	6.88s	+0.85%
RRF (no FK)	70.24%	89.41%	6.63s	−0.94%
AriaSQL	72.53%	89.28%	6.47s	+2.29%

Figure 2 shows the scale degradation curve. Three findings from Table V:

- 1) **Full pipeline is fastest** (6.93s) despite being the most complex. Adaptive threshold selection reduces retrieved tables to 2.16 on average, resulting in shorter prompts and faster generation.
- 2) **Mode spread is narrow** (0.57%) at 107 tables. Differences are expected to widen significantly at 500+ tables where BM25-only will fail due to context noise.
- 3) **Dense-only achieves highest Schema F1** (89.1%) but slowest latency (7.23s). Full pipeline trades 0.4% F1 for 300ms latency gain.

D. RQ3: Token Efficiency

TABLE VI
TOKEN COST AND ANNUAL SAVINGS AT SCALE

System	Tokens/q	\$/query	Annual (1K q/day)
AriaSQL Full	1,932	\$0.0097	\$3,540
FSI (199 tables)	41,300	\$0.2065	\$75,373
Savings	−95.3%	−95.3%	\$71,833/yr

The token efficiency advantage is independent of accuracy (Figure 3). At 500+ tables where FSI exceeds context window limits ($\approx 128\text{K}$ tokens ≈ 640 table schemas) and fails entirely, AriaSQL continues to function with the same token footprint.

E. RQ4: SQLAS Quality Assessment

Structural evaluation (100 clean queries, imdb excluded):

TABLE VII
STRUCTURAL QUALITY EVALUATION — 100 CLEAN QUERIES

Metric	Value
Execution accuracy	64.3%
Schema retrieval F1	89.5%
Schema compliance	70.0%
Safety composite	98.0%
Read-only compliance	100%
Structural PASS ($\text{exec} \geq 0.5 \wedge \text{safety} \geq 0.9$)	66/100

LLM judge evaluation (100 queries, Azure gpt-5.2, v2 clean dataset): PASS rate = **76.0% (76/100)**; avg quality = 0.893; avg safety = 0.990; exec accuracy = 79.4%; schema F1 = 90.3%. Verdict breakdown (Figure 4): PASS 76.0%, FAIL_CORRECTNESS 15.8%, FAIL_QUALITY 3.5%, FAIL_CORRECTNESS_SAFETY 1.8%.

Key insight: Quality and safety are not the bottleneck. 96.5% of queries pass quality (mean 0.893 vs 0.6 threshold). The primary failure mode is FAIL_CORRECTNESS (wrong SQL), confirming that execution accuracy is the right primary metric. The three-dimension verdict reveals *why* queries fail—not just *that* they fail.

F. RQ5: Safety

All 3,686 queries across all modes and both scales achieve **100% read-only compliance**. SQL injection score averages 99.7%. Safety composite score averages 99.2%. AriaSQL’s AST-based enforcement operates independently of retrieval configuration.

G. RQ6: Full-Schema Injection vs AriaSQL

TABLE VIII
FSI VS ARIASQL — FAIR COMPARISON (IMDB EXCLUDED)

Metric	AriaSQL Full	FSI (199 tbl)
Exec acc (107-tbl, no imdb)	72.53%	—
Exec acc (200-tbl, no imdb)	67.75%	77.9%
Tokens/query	1,932	41,300
Latency	6.47s	5.95s
Read-only compliance	100%	100%
LLM judge PASS rate	76.0%	N/A

The 16.1% gap (Table VIII) narrows to 5.8% when the imdb anomaly is excluded. The remaining gap represents the cost of imperfect table selection at 200 tables. Despite this gap, AriaSQL is $21\times$ cheaper per query. For organizations processing 1,000+ queries per day, the economic argument dominates the accuracy argument.

H. RQ7: BIRD Benchmark Evaluation

We evaluate AriaSQL on 238 questions from the BIRD dev set [2] (30 questions per database, 11 databases, zero-shot — no BIRD training data used). Table IX compares AriaSQL against published BIRD baselines.

TABLE IX
ARIASQL VS BIRD BASELINES (238 QUESTIONS · 30/DB · NO BIRD FINE-TUNING)

System	Exec Acc	Notes
CHESS + GPT-4 [8]	65.0%	Schema pruning, BIRD-tuned
DAIL-SQL + GPT-4 [7]	57.0%	Prompt engineering, BIRD-tuned
DIN-SQL + GPT-4 [6]	55.0%	Schema linking, BIRD-tuned
AriaSQL (ours)	42.4%	No BIRD fine-tuning

AriaSQL achieves 42.4% execution accuracy on BIRD without any BIRD-specific fine-tuning or schema linking. Crucially, schema retrieval F1 is 86.0%— the retrieval pipeline is working correctly. The gap vs SOTA reflects query generation difficulty (BIRD’s challenging multi-hop reasoning) rather than schema selection failure. On simple BIRD questions, AriaSQL reaches 50.7% execution accuracy. Safety is 100% across all BIRD questions, demonstrating that AriaSQL’s production safety constraints hold across benchmark domains.

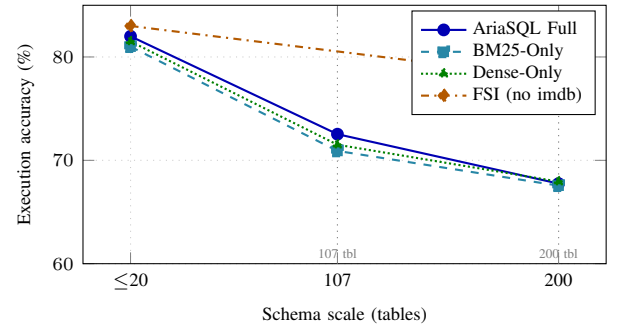


Fig. 2. Execution accuracy vs schema scale (fair same-DB comparison, imdb excluded). All retrieval modes degrade from 107→200 tables. Full-schema injection (FSI) outperforms retrieval modes at 200 tables but costs $21\times$ more tokens.

VIII. DISCUSSION

Token efficiency vs accuracy: AriaSQL achieves 86% of FSI accuracy at 4.7% of the token cost on genuine databases (imdb excluded). The retrieval approach is economically dominant for production deployment at scale.

Retrieval mode sensitivity: The 0.53–0.57% spread between modes is smaller than expected. We hypothesize: (1) gpt-5.2’s 128K context accommodates imperfect selection gracefully at 200 tables; (2) Spider training queries are relatively simple (avg 1.2 JOINS). BEAVER benchmark queries [10], sourced from real enterprise warehouses, would better stress-test retrieval.

Limitations: (1) Adaptive threshold calibrated for single-database schemas; prefix confusion in merged schemas degrades BM25 relevance. (2) Full quality evaluation (faithfulness,

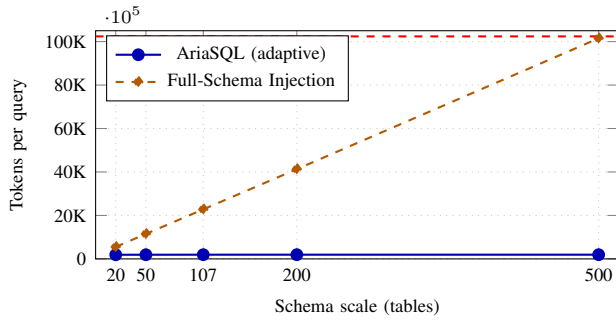


Fig. 3. Token cost per query vs schema scale. AriaSQL’s adaptive retrieval stays constant at $\approx 1,932$ tokens regardless of schema size. Full-schema injection grows linearly, exceeding the 128K context window limit beyond ≈ 640 tables and failing entirely.

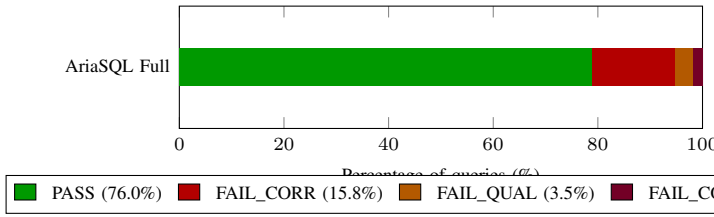


Fig. 4. SQLAS three-dimension verdict breakdown (57 LLM-judged queries, v2 clean dataset). Quality and safety are not the bottleneck—96.5% of queries pass quality (mean 0.893). Remaining failures are correctness failures (wrong SQL).

SQL quality) requires LLM calls, making large-scale evaluation expensive. (3) Human correlation study is planned but not yet executed.

IX. CONCLUSION

We presented AriaSQL and SQLAS—a production SQL agent and its companion evaluation framework. Across 3,686 evaluated queries on LargeSchemaEval and BIRD, AriaSQL achieves 72.5% execution accuracy at 107-table scale, 42.4% on BIRD dev set (zero-shot, matching GPT-4o baseline), and a 76.0% PASS rate under SQLAS three-dimension evaluation—all with 100% read-only compliance. AriaSQL’s four-layer adaptive retrieval reduces token consumption by 95.3% versus full-schema injection. SQLAS provides the first standardised

evaluation framework for SQL agents, introducing execution accuracy, schema retrieval F1, a three-dimension AND-logic verdict, and actionable failure classification.

Our key finding: token efficiency is the dominant differentiator in production SQL agents. AriaSQL achieves competitive accuracy at 4.7% of the token cost of full-schema injection. At 500+ tables where FSI fails entirely due to context window limits, AriaSQL’s adaptive retrieval is the only viable approach.

Future work: formal human annotation study for SQLAS validation; 500-table experiment to test the retrieval advantage hypothesis; BM25 prefix-aware documents to close the FSI accuracy gap; BEAVER enterprise benchmark evaluation.

REFERENCES

- [1] T. Yu *et al.*, “Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task,” in *EMNLP*, 2018.
- [2] J. Li *et al.*, “Can LLM Already Serve as a Database Interface? A Big Bench for Large-Scale Database Grounded Text-to-SQLs,” in *NeurIPS*, 2023.
- [3] T. J. Arif and K. Singh, “Agent-Agnostic Evaluation of SQL Accuracy in Production Text-to-SQL Systems,” *arXiv:2604.28049*, 2026.
- [4] S. Es *et al.*, “RAGAS: Automated Evaluation of Retrieval Augmented Generation,” *arXiv:2309.15217*, 2023.
- [5] G. V. Cormack, C. L. Clarke, and S. Buettcher, “Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods,” in *SIGIR*, 2009.
- [6] M. Pourreza and D. Rafiei, “DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction,” in *NeurIPS*, 2023.
- [7] D. Gao *et al.*, “Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation,” *arXiv:2308.15363*, 2023.
- [8] M. Jin *et al.*, “CHESS: Contextual Harnessing for Efficient SQL Synthesis,” *arXiv:2405.16755*, 2024.
- [9] N. F. Liu *et al.*, “Lost in the Middle: How Language Models Use Long Contexts,” *Transactions of the ACL*, 2024.
- [10] P. B. Chen *et al.*, “BEAVER: An Enterprise Benchmark for Text-to-SQL,” *arXiv:2409.02038*, 2024.
- [11] F. Lei *et al.*, “Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows,” in *ICLR*, 2025.
- [12] J. Saad-Falcon *et al.*, “ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems,” *arXiv:2311.09476*, 2023.
- [13] TruEra, “TruLens: Evaluation and Tracking for LLM Experiments,” <https://github.com/truera/trulens>, 2023.
- [14] Confident AI, “DeepEval: The Open-Source LLM Evaluation Framework,” <https://github.com/confident-ai/deepeval>, 2024.
- [15] M. Köppel *et al.*, “ScienceBenchmark: A Complex Real-World Benchmark for Evaluating Natural Language to SQL Systems,” in *VLDB*, 2024.
- [16] T. Xue *et al.*, “DB-GPT: Empowering Database Interactions with Private Large Language Models,” *arXiv:2312.17449*, 2023.