Deferral, not capability: governed text-to-SQL with a small open-weights generator

ABSTRACT

Text-to-SQL systems deployed against governed enterprise data face a requirement that accuracy benchmarks do not measure: the system must be unable to read what the caller may not read, and must decline rather than answer when the data cannot support an answer. We describe mnemiq, a system that pairs a stochastic proposer with a deterministic decider, and report measurements over three public suites.

Our central finding is that the accuracy gap between a 14B open-weights model on a 24 GB card and a frontier model is dominated by **over-deferral rather than by capability**. A Qwen2.5-Coder-14B answering against a semantic contract reaches 17.0% execution accuracy while declining 73.9% of questions; three interventions that convert refusals into attempts each recover the same 51–52% band, and none of them improves the rate at which the model returns a wrong answer. Going from 14B to 32B costs 0.6 points. In the same pipeline and under the benchmark's own protocol, the 14B exceeds the published GPT-4 baseline for that dialect by 15.1 points.

We further report that a residual wrong-answer rate of 38–41% survives every calibration method we applied, that only verification moves it, and that verification is a cost ladder rather than a free improvement: the cheap deterministic layer catches 31 of 190 wrong answers for 2 correct ones, while an LLM judge catches 94 for 22. We find no transfer of these results to Spider 2.0-lite, where every local configuration sits in a 2–6% band, and we treat that as the boundary of the claim rather than as a detail.

1. Introduction

A text-to-SQL system that reaches a production database inherits obligations that a benchmark harness does not impose. It must not disclose a table the caller is not entitled to see. It must not execute a statement whose effects were never reviewed. And when the schema cannot answer the question asked, silence is a better outcome than a confident number, because a wrong figure delivered fluently is more expensive than no figure at all.

These obligations are not properties a model can be trained to have. A model can be trained to *usually* decline, and usually is not a guarantee. The design position taken here is that enforcement belongs outside the model: the model proposes, and a deterministic component decides, with the authority to refuse, rewrite, or approve.

This separation makes an empirical question available that is otherwise hard to ask. If enforcement is not the model's job, how much of the generation task can be moved to a small model that runs on commodity hardware? We report that the answer is more than the raw benchmark numbers suggest, and that the reason is a measurable behavioural difference rather than a capability difference.

Contributions.

1. A nine-stage architecture in which five stages are configuration surface and three are fixed by policy, with the enforcement points stated and their failure modes measured (§2).
2. An ablation over ten arms on BIRD mini-dev showing that over-deferral, not capability, accounts for the gap between a 14B model and a frontier model, and that three interventions converge on the same ceiling (§4.1–4.2).
3. A comparison against published baselines on the benchmark's own protocol, and against two commercial systems on three suites (§4.3–4.4).
4. A characterisation of verification as a cost ladder, with the price of each layer in correct answers deferred (§4.5).
5. A negative result: neither the interventions above nor the semantic layer transfers to Spider 2.0-lite, and the LLM semantic layer's own contribution does not separate on our evidence (§4.6, §5).

---

2. System

The system executes nine stages. Five of them are configuration surface that a deployment varies and a parameter search can explore; three are fixed by policy and are not reachable by any setting tuned for accuracy; one is not a stage at all but the artifact the second stage emits.

2.1 The semantic contract

Enrichment turns a bare schema into a **semantic contract**: columns with meanings, relationships, views, dimensions, metrics, and a job log recording what was attempted and what failed. The contract is content-addressed, so a replica can determine whether it holds the same one.

The pipeline runs in a fixed order, and the order carries the design.

Table 1. Enrichment phases, in execution order.

#	PHASE	WHAT IT DOES	LLM
1	enrich_structural	schema discovery and per-column profiling — distincts, nulls, types, relationships, views	no
2	ground_codes	resolves coded values against evidence already in the database	no
3	apply_certified	applies certified code schemes where one is declared	no
4	ontology grounding	embeds an ontology corpus and matches definitions	embeddings
5	enrich_semantic	the model proposes meanings for what remains, with protected terms withheld	yes
6	apply_dictionary	the operator's data dictionary — overrides everything above	no
7	enrich_facts, enrich_examples	optional table facts and worked examples	yes

Deterministic phases run first, the model in the middle, the operator last. The model never gets first say on a column whose meaning is already recoverable from the data, a certified scheme, or an ontology; and whatever it proposes, a human dictionary entry overrides. The consequence is auditable rather than merely orderly: for any column one can ask which layer decided its meaning, and the answer is a rank rather than an inference.

A column that could not be profiled is recorded as failed *with a cause*, never as an absence. The distinction matters because an absent value and a failed measurement are otherwise indistinguishable downstream, and they license different conclusions.

2.2 Retrieval and generation

Retrieval scores the contract against the question and returns k schema cards (k = 24 by default), scoped to the caller's identity. Ranking is over tables rather than columns.

This is the first of two independent access controls, and it is the stronger of the two in an important sense: a table the caller may not see is never placed in the context at all, so naming it in a question is not refused — it is inert. The model cannot leak what it was not shown.

Generation is the only stochastic component. It emits candidate SQL in a single dialect, which the system parses and executes against whatever source is attached.

2.3 The decider

Every candidate passes through a fixed sequence of ten checks before anything runs.

Table 2. The decider's checks, in order. Three are conditional on deployment configuration; a deployment with none of the three runs seven.

#	CHECK	OUTCOME ON FAILURE
1	`check_shape` — one statement, a SELECT, no DDL/DML, no `SELECT *`, a `LIMIT`	refuse
2	`check_access` — every table and column against what this identity may see	refuse
3	`check_cls` — denied columns; masked columns outside a bare projection	refuse
4	`check_views` — a view that reads a row-filtered table	refuse
5	`lint` — SQL that runs and answers wrongly	refuse, repairable
6	`check_values` — a filter literal that exists in no row of its column†	refuse, repairable
7	lineage — what the query, as asked, reads	recorded
8	`apply_row_and_mask` — injects row filters and column masks†	rewrite
9	`expand_tables` — federation qualification	rewrite
10	`prove` — the statement against the real source†	refuse

† conditional: 6 requires harvested column values, 8 a non-empty policy for the identity, 10 an attached adapter.

Shape is checked first because a destructive statement is destructive regardless of who asked. Access is re-checked here even though retrieval already scoped the context, because the model may name an object it was not shown; the database's own permission error is never relied upon as the control.

Two of the ten refusals are *repairable*: they return to the generator with the decider's reason attached, up to three attempts. The loop is closed by a deterministic checker rather than by the model reviewing its own work, which is what makes a repair a narrowing rather than a retry.

2.4 Verification

Verification is the only stage that sees the answer. It is a cascade that stops at the first layer producing a verdict: a deterministic sanity check (empty or null results), an optional grounding check over question literals, and an optional LLM judge with a confidence threshold.

The sanity layer is on by default; the other two ship off. Section 4.5 reports what each costs.

One property is worth stating because it is easy to assume otherwise: when the judge cannot be reached, the system is **fail-open**. An unreachable judge scores 1.0, which is the value approval returns, and the answer proceeds. What the record does in that case is decline to describe the answer as checked — the distinction between *approved* and *not evaluated* is preserved in the audit trail even though the answer is delivered either way.

2.5 Lineage and the audit record

Every answer emits the tables it read, the enrichment version it relied on, and a completeness marker with three values: complete, incomplete, or unknown. The third value exists because an empty list is ambiguous — it reads as *nothing was touched* when it may mean *we could not determine what was touched* — and those license different audit conclusions.

Disclosure of answer text, identity detail and result rows into the trace are three separate gates, all closed by default.

---

3. Experimental setup

3.1 Suites

We report over three public suites.

BIRD mini-dev, PostgreSQL dialect, n = 487 of 500. The thirteen excluded cases are dropped by the harness before the system sees them, when the gold result set exceeds a row cap; they are not refusals.

Spider 1 and **Spider 2.0-lite**. On Spider 2.0-lite the system runs the 135 local SQLite instances of 547; the remainder require BigQuery or Snowflake credentials. That subset is a quarter of the suite and results on it should not be read as results on the benchmark.

3.2 Grading

Grading is result-based. A query that returns the correct facts by a different route passes; we report both exact match (column count and order enforced) and a *got-the-facts* rate that permits additional columns. All arms in a comparison are graded under one rule — a figure graded under one rule and compared against a figure graded under another is not a comparison, and we do not present any such pair.

Denominators are stated per table. Rates are over *answerable* cases; a correct refusal is not scored as a failed attempt.

3.3 External knowledge

BIRD and Spider ship a human-written hint with each question, and every figure we report from those suites was produced with that field present — 485 of 487 BIRD cases and 13 of 135 Spider 2.0-lite cases carry one. No deployment receives such a hint.

This is the largest single effect the benchmark reports about itself: GPT-4 scores 54.89% with the field and 34.88% without (Li et al., 2023). Any figure quoted from these suites must state which side of that 20-point line it sits on, and comparisons that mix the two settings are invalid. Where we compare against published baselines (§4.3), both sides were produced with the field present.

4. Results

4.1 Over-deferral, not capability, is the gap

Table 3. BIRD mini-dev, PostgreSQL, n = 487. Enrichment ran on a hosted frontier model and was held constant across every arm; only generation varies. EX = exact match. EX, deferral and wrong are shares of all n and partition to 100%; wrong is therefore over every question, not over attempts.

GENERATOR	METHOD	EX	GOT-FACTS	DEFERRAL	WRONG
hosted gpt-5.5	—	50.7%	63.4%	1.0%	48.2%
Qwen2.5-Coder-7B	none	6.4%	6.6%	92.0%	1.6%
Qwen2.5-Coder-7B	guided decoding	33.9%	35.1%	26.7%	39.4%
Qwen2.5-Coder-14B	none	**17.0%**	17.7%	**73.9%**	9.0%
Qwen2.5-Coder-14B	assertive prompting	44.8%	47.8%	17.5%	37.8%
Qwen2.5-Coder-14B	guided decoding	**50.9%**	54.0%	10.3%	38.8%
Qwen2.5-Coder-14B	guided + self-consistency (N=5)	**52.3%**	57.0%	7.8%	39.9%
Qwen2.5-Coder-14B	QLoRA	51.1%	52.4%	10.3%	38.6%
Qwen2.5-Coder-32B	guided decoding	**50.3%**	53.6%	9.0%	40.7%
DeepSeek-V2-Lite 16B MoE	guided decoding	37.6%	41.9%	18.1%	44.4%

The 14B's baseline 17.0% sits under a 73.9% deferral rate. It was not answering incorrectly; it was not answering. Every intervention that improved the number did so by converting refusals into attempts.

The three arms that reduce deferral to approximately 10% — guided decoding (10.3%, 50.9%), QLoRA (10.3%, 51.1%) and guided decoding with self-consistency (7.8%, 52.3%) — land in the same 51–52% band. Guided decoding constrains the decoder; QLoRA modifies the weights; nothing is shared between them, and both arrive at exactly 10.3% deferral with 0.2 points between their accuracies. The third is not independent of the first — it is guided decoding plus five-sample voting, which adds 1.4 points to it. Assertive prompting stops at 17.5% deferral and falls six points short.

Among the local arms on this suite, how far deferral falls orders the outcome; *how* it falls does not. We note the limit of that observation in §5.

4.2 Parameters do not move it

14B with guided decoding reaches 50.9%; 32B with guided decoding reaches 50.3%. That is **0.6 points lost for 2.3× the parameters**, and the 32B does not fit on the 24 GB card the 14B runs on.

More consequentially, no arm moves the rate at which the system returns a wrong answer. Across every calibration method applied to the 14B — assertive prompting 37.8%, guided decoding 38.8%, guided with self-consistency 39.9%, QLoRA 38.6% — and at 2.3× the parameters (32B, 40.7%), the wrong-answer rate stays inside a 38–41% band. As a share of attempts rather than of all questions, the same band is 43–46%. Nothing in the generation stage moved it. Only verification did, and §4.5 reports its price.

4.3 Against published baselines

The benchmark repository publishes baseline execution accuracies per dialect. Both sides of Table 4 ran the standard protocol — schema plus the external-knowledge field — with the difference being the semantic contract. The reference figures are the repository's mini-dev PostgreSQL baselines, and its own run script sets the external-knowledge flag; they are not the full-dev SQLite figures quoted in §3.3, and the two are not comparable to each other.

Table 4. BIRD mini-dev, PostgreSQL column. Reference figures are the benchmark's published baselines; both mnemiq rows are enriched.

SYSTEM	EX
Llama3-70B	29.4%
GPT-4	35.8%
mnemiq, 14B + guided decoding	**50.9%**
mnemiq, 14B + guided + self-consistency	**52.3%**

This is **+15.1 points over the published GPT-4 figure** at the single-shot configuration, and +16.5 at five candidates. It is a claim about the pipeline, not about model quality, and it should not be phrased as a 14B outperforming GPT-4: the reference runs are unenriched and see a column listing, while the mnemiq rows carry a semantic contract built by a frontier model. The comparison isolates what the contract and the decider contribute, not what the generator does.

4.4 Against commercial systems

Three suites, three systems, three repeat runs of one configuration each; ranges are run-to-run variance on both sides rather than best-of. These figures come from an independent evaluation harness with its own denominators and are not comparable to Tables 3 and 4.

Table 5. Median (range) over three runs except where marked. mnemiq configurations: *structural* is deterministic enrichment only; *semantic* adds the LLM meaning-proposal phase. † carried from the tracker as a three-run mean; no per-run range is published for that row.

SUITE	SYSTEM	CONFIGURATION	N	EX	GOT-FACTS
BIRD	mnemiq	semantic	479–480	**52.3** (51.1–53.2)	**64.8** (64.7–65.3)
BIRD	mnemiq	structural	482	51.0 (50.0–51.5)	64.1 (63.3–64.1)
BIRD	Databricks Genie	minimal workspace	494–496	37.5–39.5	52.6–53.2
BIRD	Snowflake Cortex	autopilot views	494–495	33.6 (32.3–33.9)	49.0 (48.5–49.1)
Spider 1	mnemiq	structural	1022	**69.2** (68.4–69.5)	**80.2** (79.6–80.4)
Spider 1	mnemiq	semantic	1022	68.5 (68.0–68.7)	79.5 (79.4–79.6)
Spider 1	Databricks Genie	minimal workspace	1016–1022	63.1–63.6	76.4–77.4
Spider 1	Snowflake Cortex	autopilot views	1021	56.2 (56.2–56.9)	77.1 (76.9–77.2)
Spider 2	mnemiq	semantic	130–132	**36.6**†	**59.8**†
Spider 2	mnemiq	structural	130–132	34.6 (34.1–40.0)	56.9 (56.8–63.8)
Spider 2	Databricks Genie	minimal workspace	34–134	27.1–29.1	44.2–45.1
Spider 2	Snowflake Cortex	autopilot views	132	17.4 (one run)	24.2 (one run)

Graded denominators differ between systems because each harness excludes its own errored and unmapped items rather than scoring them wrong; on BIRD mnemiq is graded on some fifteen fewer cases than either product.

mnemiq's range clears both products on every suite and both metrics — six of six, with no overlap. The narrowest margin is Spider 1 got-facts, where the floor sits 2.0 points above the higher competitor's ceiling.

The condition these numbers were taken under belongs with them. All three systems saw the schema and the benchmark's own external-knowledge field, and no human-written semantic content beyond it. Cortex Analyst ran without its Verified Query Repository, and that is an asymmetry rather than a matching handicap: a verified query repository is human-confirmed question-to-SQL, and

mnemiq has no equivalent of it. Our own worked-example phase is model-generated, a different mechanism, and on this suite it measured worse. Populating a repository from the benchmark's gold SQL would be answer leakage rather than a fairer configuration; doing it properly requires a held-out split and was not done. **Every row should therefore be read as cold-start behaviour on schema alone, and none of them as any product's ceiling** — least of all that of the system whose design leans hardest on the mechanism that was switched off.

4.5 Verification is a cost ladder

Table 6. Verifier layers replayed over the 14B guided-decoding run, quoted on the grading in force when the replay was made. n = 487 answerable, 190 wrong, 232 correct. The catch counts have not been re-derived under the current grader, so the EX and wrong-rate anchors in this table are the replay's own and do not match Table 3.

LAYER	WRONG CAUGHT	CORRECT LOST	EX	WRONG RATE
sanity	31 / 190	**2 / 232**	47.6 → 47.2	39.0 → 32.6
grounding	28 / 190	16 / 232	47.6 → 44.4	39.0 → 33.3
sanity + grounding	49 / 190	18 / 232	47.6 → 43.9	39.0 → 29.0
judge @ 0.5	**94 / 190**	22 / 232	47.6 → 43.1	39.0 → **19.7**
judge @ 0.8	123 / 190	40 / 232	47.6 → 39.4	39.0 → 13.8
judge @ 0.9	144 / 190	53 / 232	47.6 → 36.8	39.0 → 9.4
sanity + judge @ 0.5	95 / 190	22 / 232	47.6 → 43.1	39.0 → 19.5

Read as a ladder rather than a ranking:

The deterministic layer is cheap, not best. Sanity catches 31 wrong answers for 2 correct ones and 0.4 points of exact match. Nothing else approaches that price, and it requires no model.

The judge is effective and not free. At the same cutoff it catches three times as many — 94 — and cuts the wrong-answer rate from 39.0% to 19.7%, for 22 correct answers deferred and one additional model call per question.

The layers do not compose. Running both catches 95, one more than the judge alone, for no additional correct answers lost: 30 of the 31 answers sanity catches, the judge already had. What sanity earns in a judged deployment is not accuracy but cost — the cascade stops at the first verdict, so the cases sanity defers never reach the judge, removing roughly 17% of judge calls.

4.6 What does not separate

Two negative results are worth reporting.

Neither guided decoding nor additional parameters transfers to Spider 2.0-lite. Every local arm sits in a 2–6% band against 35.6% for the hosted configuration on identical items — a 6–18× gap. Best-configuration against best-configuration among the local arms is 8 correct cases against 7 out of 135, and across four arms only 14 distinct cases are solved with just one solved by all four. That distribution is the signature of near-chance success, and we claim no ordering among those arms.

The LLM semantic layer's own contribution does not separate. The *structural* and *semantic* configurations in Table 5 differ only in the meaning-proposal phase, and were run on three suites with two metrics each. One of those six comparisons separates by the non-overlap criterion this section applies elsewhere — BIRD got-facts, where the semantic floor sits 0.6 above the structural ceiling. The other five overlap, and on Spider 1 both metrics favour the configuration *without* the semantic layer. A mechanism that leads on one suite and trails on another requires an explanation the run record does not supply, so we report the contribution as unresolved. This is a statement about this evidence and not about enrichment generally: the deterministic phases are present in both configurations and are not what varies.

---

5. Limitations

The deferral relation is bounded to this suite and these arms. Ranked by deferral, the nine local BIRD arms rise in accuracy at every step but two — the 32B, which defers less than either 10.3% arm and scores below both, and a tie at 10.3% where two arms differ by 0.2 points with no deferral difference to explain it. The relation does not hold across suites: on Spider 2.0-lite the guided 14B defers *less* than the unguided one (17.0% against 20.7%) and scores *lower* (4.4% against 5.9%), because there the recovered attempts are wrong. Nor across model families on the same suite: the hosted arm defers 1.0% and scores 50.7%, below a local arm at 7.8% deferral and 52.3%. Deferral orders local configurations of one suite; it is not a figure of merit.

Generation was localised; enrichment was not. In every measured arm the semantic contract was built by a hosted frontier model and held constant. The result is therefore that *generation* localises — the expensive, per-query, data-touching half — and not that the system runs on a 14B. A fully air-gapped configuration is a different setup and is not measured here.

The knowledge dimension is not isolated. Enrichment is held constant across the arms of Table 3, and a human-written hint is present on 99.6% of BIRD questions. That table isolates generator and method against a fixed knowledge background and says nothing about the value of the knowledge itself.

Spider 2.0-lite results cover a quarter of the suite (135 local instances of 547) and the remainder requires cloud warehouse credentials.

The wrong-answer floor is unexplained. A 38–41% wrong rate on BIRD survives every generation-stage method we applied and 2.3× the parameters. We can reduce it with verification at a measured cost in correct answers, but we cannot currently reduce the rate at which the generator produces confident errors.

---

6. Conclusion

The gap between an open-weights generator on a 24 GB card and a frontier model on governed text-to-SQL is, on the suite we measure most closely, not primarily a capability gap. It is a calibration gap, and it closes: three interventions each recover the same 51–52% band by converting refusals into attempts, and the resulting configuration exceeds the benchmark's published GPT-4 baseline for that dialect by 15.1 points under the benchmark's own protocol.

What does not close is the rate at which the generator is confidently wrong. That rate is insensitive to every generation-stage method we tried and to doubling the parameters, and the only instrument that moves it is verification, which trades correct answers for incorrect ones at a price we report rather than assume.

We take the practical implication to be that for this class of work the useful engineering is not a larger generator. It is the surrounding machinery — a semantic contract, a deterministic decider that can refuse, and a verifier whose cost is known. Those components are also the ones that make the system deployable against data that has an access policy, which is the setting the benchmarks do not measure and the deployment always has.

References

Li, J., Hui, B., Qu, G., et al. (2023). *Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs.* arXiv:2305.03111.

BIRD-bench. *mini_dev*: baseline execution accuracies by SQL dialect. `github.com/bird-bench/mini_dev`.

Yu, T., Zhang, R., Yang, K., et al. (2018). *Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL.* arXiv:1809.08887.

Lei, F., Chen, J., Ye, Y., et al. (2024). *Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows.* arXiv:2411.07763.