IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

Beyond Benchmark Accuracy: An Enterprise Deployment and Readiness Framework for Large Language Model-Based Text-to-SQL Systems

Dr. Dadavali S P[1], Dr. K. Siddaraju[2], Dr. Chandrashekar P[3]

Associate Professor, Department of Computer Science, GFGC, Kengeri, Bangalore, India[1]

Associate Professor, Department of Computer Science, GFGC, Maddur, India[2]

Associate Professor, Department of Computer Science, GFGC, KRPuram, Bangalore, India[3]

Abstract: Enterprises piloting LLM-based text-to-SQL keep discovering the same thing: a model that clears 85% execution accuracy on Spider can still be rejected by the security reviewer, the auditor and the risk committee — and rightly so. In the pilots and post-mortems that motivated this work, generated queries ignored row-level access policies, exposed masked columns, broke silently under schema change, ran unbounded scans, and left no evidence trail. Building a prototype to close these gaps taught us that the problem is architectural rather than linguistic: what was missing was not a better model but a system around the model, one in which vocabulary, policies and budgets are machine-readable and enforcement is deterministic. ERF-SQL is that system, together with a way to measure whether it is working. A versioned semantic contract becomes the sole interface between schema and model; a Policy-Aware Query Guard rewrites the abstract syntax tree of every candidate query before execution; uncertain answers are routed to clarification or human review; and behaviour is scored on six readiness dimensions aggregated into a gated Enterprise Readiness Index. On EntSQL-450, an evaluation suite with live role policies and masking rules built for this study, four production LLMs lost 19 to 23 accuracy points relative to Spider under a naive baseline; the implemented pipeline recovered 13 to 16 of those points, blocked 97% of unsafe generations before execution, and raised drift resilience from 0.55 to 0.86 at roughly 0.9 seconds of added latency per query. The demonstration we take from the implementation is simple: deployment readiness can be engineered and measured, not merely hoped for.

Keywords: Text-to-SQL, large language models, enterprise deployment, AI governance, schema linking, data security, readiness assessment, natural language interfaces to databases.

I. INTRODUCTION

Natural language interfaces to databases have been pursued for four decades, but instruction-tuned large language models have changed the economics of the problem. A well-prompted GPT-4, Claude or Llama now matches purpose-built neural semantic parsers on the Spider development set without a single annotated training example [1]–[4]. When work on the prototype described in this paper began, that was the state of the art we expected to build on: choose a strong model, present the schema well, and concentrate on the user experience. Analytics vendors were shipping conversational query features on exactly that premise, and enterprise data teams were under pressure to follow.

The premise did not survive contact with enterprise requirements. In interviews and published post-mortems, practitioners report a recurring set of incidents that appear on no leaderboard: a regional manager retrieving figures for regions outside her remit because the generated query omitted a mandatory territory filter; a query that projected a salary column to a user whose role prohibited it; a full-table scan across a 900-million-row fact table triggered by a casually phrased question; a pipeline that broke silently when a column was renamed during a routine migration; an internal audit that rejected the system because no evidence trail linked a business answer to the SQL that produced it. While integrating our first schemas, we reproduced several of the same failure classes ourselves—the metadata drift and guard evasions recorded in Section V-G among them— before we had names for them. In every case, the model that generated the SQL would have scored well on Spider or BIRD [5]. These were not failures of language understanding. They were failures of deployment engineering and governance.

We observed, in short, that benchmark accuracy is necessary for deployment but nowhere near sufficient — and that the insufficiency has structure. The research community has spent years optimising one scalar, execution accuracy (EX), on schemas that are small, clean, fully visible to the model, free of access control and frozen in time. Real warehouses violate every one of those assumptions. What a deploying team needs is harder to fit on a leaderboard: a way to name the

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

dimensions beyond accuracy that decide approval, to measure each of them, to improve the measurements architecturally, and to present the whole to a chief data officer or risk committee as something they can act on.

ERF-SQL, the Enterprise Readiness Framework developed in this paper, grew out of building and hardening a working prototype rather than out of a survey. Concretely, the paper offers:

- **A six-dimension readiness model** (Section IV) that extends accuracy with Safety and Security, Schema Resilience, Service Performance, Stewardship and Stakeholder Trust, each operationalised through concrete metrics.
- **A six-layer reference architecture** (Section V) with three novel components: a Semantic Contract Layer, a Policy-Aware Query Guard performing AST-level policy rewriting, and Confidence-Gated Execution with human-in-the-loop routing.
- **The Enterprise Readiness Index (ERI)** (Section VI), a weighted geometric aggregation with explicit gating thresholds and five readiness tiers that map directly to deployment decisions.
- **EntSQL-450 and an adversarial probe set** (Section VII), an evaluation suite over three enterprise-style schemas that exposes the gap between benchmark and enterprise performance, together with an empirical study of four LLMs and an ablation of each framework layer (Section VIII).
-

None of these pieces is exotic on its own. What sets the framework apart is that three ideas prior work treats separately are made to depend on one another: a versioned, machine-consumable contract as the sole interface between schema and model; deterministic, AST-level policy enforcement placed after generation instead of being delegated to the model; and a readiness index whose gates encode the conjunctive way deployment decisions are actually made. Sections V and VI describe each in depth, and the ablation in Section VIII-D isolates what each contributes.

The remainder of the paper reviews related work (Section II), analyses why benchmark-centric evaluation fails in the enterprise (Section III), presents the framework, architecture, index and experiments, sets out the limitations of the framework (Section IX), discusses threats to validity (Section X) and concludes with directions for future work (Section XI).

II. RELATED WORK

A. Prompt-Based and Decomposition Methods

The earliest LLM text-to-SQL results came from prompting alone. Rajkumar et al.'s evaluation of Codex on Spider [1] established the pattern that still holds: schema presentation and sample rows dominate performance. Decomposition refines rather than departs from this — DIN-SQL splits generation into schema-linking, classification, generation and self-correction stages [2], and C3 adds calibrated prompting [6] — but the refinement shares the original premise that the model is a closed box steered only through its input. Whatever the system must not do can therefore only be requested, never enforced, and because every stage remains probabilistic, an error suppressed at one stage can resurface at another. No arrangement of prompts, however careful, is deterministic enough to carry a security guarantee. The surveys of Katsogiannis-Meimarakis and Koutrika [9] and Hong et al. [10] confirm how uniformly this line of work is judged: by execution accuracy on benchmark schemas, and by nothing else.

B. Retrieval-Based Methods

A second family improves the input rather than the instructions. DAIL-SQL selects exemplars by similarity [3], MAC-SQL coordinates specialised agents over retrieved context [7], and CHESS scales the approach to the harder BIRD benchmark [5], [8]. Taken together these systems demonstrate that what the model is shown matters as much as what it is asked, and they are the family closest in spirit to our contract retrieval. The shared limitation lies in the nature of what is retrieved. Schemas and exemplars are descriptive: nothing in a retrieved exemplar tells the system which rows the requesting user may see, what a governed metric means, or what a query may cost. Retrieval, as practised, carries information to the model; it does not carry obligations.

C. Benchmark Design and Robustness

The benchmarks themselves have not escaped criticism. Spider-Syn [11] and Dr.Spider [12] perturb questions and schemas and watch apparently strong models lose 10 to 30 accuracy points; Zhong et al.'s test-suite accuracy [13] tightens what counts as a correct execution; BIRD [5] and Spider 2.0 [14] push the data toward realistic scale and dialect diversity and are the nearest antecedents of EntSQL-450. Read as a sequence, these efforts make evaluation more realistic one axis at a time — lexical robustness, execution semantics, scale — while two things stay constant. Correctness is still equated with matching a gold result rather than satisfying a governed business definition (a distinction Section IV returns to as BDA), and the model is still scored in isolation from the access policies, sensitivity tags, cost budgets and drift histories under which it would actually run.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

D. Governance and Security Approaches

Security and governance approach deployment from opposite ends and do not quite meet. The OWASP Top 10 for LLM applications [15] and Pedro et al.'s demonstration that crafted natural language can steer LLM-generated SQL into injection-style attacks [16] establish the threat model precisely, but the mitigations they propose — input filtering, prompt hardening — inherit the probabilistic instruction-following of Section II-A; in our experiments (Section VIII), every configuration that depended on it failed the policy gate. The NIST AI Risk Management Framework [17] and ISO/IEC 42001 [18] state the obligations — measurability, traceability, human oversight — at a level deliberately abstracted from any technology, so a team is told what must be true but not how to make it true or how to verify that it is. Between the two sits data-contract and semantic-layer practice [19]: a plausible bridging artefact that has developed inside BI tooling, without contact with LLM query generation.

E. Where ERF-SQL Fits

Reading these literatures side by side left us with a simple impression: the pieces of a deployable system already exist, scattered across communities that rarely cite one another. ERF-SQL is an assembly of those pieces with the joints made explicit. From the retrieval family it takes context selection, but makes the retrieved artefact normative — a versioned contract carrying definitions, policies and sensitivity tags — rather than descriptive. From the security literature it takes the threat model, but answers it with deterministic AST-level rewriting instead of hardened prompts. From the governance frameworks it takes the obligations, and turns them into computable metrics with gates. And from the benchmark critiques it takes the diagnosis, adding the remedy the diagnostics stop short of: an evaluation suite in which policies, budgets and drift are present, and an index that reports whether the assembled whole is ready.

III. WHY BENCHMARK ACCURACY FALLS SHORT

Why does the same model shed so much accuracy the moment it leaves the benchmark? Our answer is structural. Benchmark evaluation embeds six assumptions, and none of them survives contact with an enterprise setting. Table I summarises the six together with the corresponding readiness dimension of ERF-SQL.

TABLE I. BENCHMARK ASSUMPTIONS VERSUS ENTERPRISE REALITY

Benchmark assumption	Enterprise reality	ERF-SQL dimension
Correct = executes to the gold result	Correct = right numbers for this user under this business definition	Semantic Fidelity
Every user may read every table and row	RBAC, row-level security, PII masking, regulatory constraints	Safety & Security
Schema is fixed at evaluation time	Columns renamed, tables split, types changed monthly	Schema Resilience
Latency and cost are irrelevant	Sub-5 s SLAs, per-query compute budgets, warehouse cost caps	Service Performance
No audit or lineage required	Query lineage, approval trails, reproducibility demanded by audit	Stewardship
Output is a SQL string	Users need explanations, confidence and a path to escalate	Stakeholder Trust

Semantic gap. Benchmarks reward any SQL whose execution result matches the gold result. In enterprises, the definition of a metric such as "net revenue" or "active customer" is a governed business rule that may span filters, exclusions and time conventions. A query can execute and match a naive interpretation while being wrong by the organisation's definition.

Authorisation gap. The model has no notion of who is asking. Injecting access rules into the prompt is unreliable because instruction-following is probabilistic; a policy that holds 95% of the time is not a policy.

Temporal gap. Few-shot exemplars, cached schema descriptions and fine-tuned weights all encode a snapshot of the schema. When the schema evolves, accuracy degrades silently because the system still produces syntactically valid SQL.

Economic gap. A correct query that scans a petabyte is not deployable. Benchmarks contain databases small enough that cost is invisible.

Evidence gap. Audit and compliance functions require that any number presented to a decision-maker be traceable to the query, the policy in force, the model version and the reviewer, if any. A raw generation log does not satisfy this requirement.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

Trust gap. Adoption depends on users understanding when to trust an answer and how to escalate when they do not. Accuracy alone does not produce calibrated trust; uncalibrated systems are either ignored or over-trusted.

IV. THE ERF-SQL READINESS MODEL

Six dimensions, D1 to D6, make up the readiness model. Each carries a small set of metrics, normalised to [0, 1] with 1 denoting full readiness, so that dimensions can be compared and aggregated (Section VI). Table II lists the metrics. The paragraphs below define the less conventional ones.

Design rationale. The layered, dimension-by-dimension structure was not the first design we tried. Three alternatives were considered while the framework was taking shape. The first was a flat control checklist in the style of an audit questionnaire; it was easy to fill in but produced no measurement, so two systems that both "passed" could not be ranked. The second was a capability maturity model in the tradition of CMM [22], with organisational process levels; this describes how a team works rather than how a system behaves, and it cannot be recomputed from telemetry after a model upgrade. The third was a single penalised scalar, essentially execution accuracy minus a set of penalty terms; it was compact but it hid which property had failed, which is exactly the information a risk committee needs. The layered form was chosen because it mirrors how enterprise deployments are actually approved: business intent is validated first, then access and governance, then operational limits, and only then production readiness, in sequence and with a gate at each step. A framework whose structure matches the approval path is easier to adopt than one that is technically elegant but has to be translated for every review board.

TABLE II. ERF-SQL DIMENSIONS AND METRICS

Dim.	Metric	Definition (normalised to [0,1])
D1 Semantic Fidelity	EX, TS-EX, BDA	Execution accuracy; test-suite accuracy [13]; business-definition agreement against governed metric definitions
D2 Safety & Security	PVR, PLR, IBR, MBR	1 − policy violation rate; 1 − PII leakage rate; injection block rate; mutation block rate
D3 Schema Resilience	DRS, OOS-R	Drift resilience score (EX after drift ÷ EX before); out-of-scope rejection rate
D4 Service Performance	p95 lat., CPQ, CCR	Latency vs. SLA; cost per query vs. budget; cost-cap rejection correctness
D5 Stewardship	LC, RR, VP	Lineage completeness; reproducibility rate; version pinning of model, prompt and contracts
D6 Stakeholder Trust	ECE, HITL-P, EXR	1 − expected calibration error; HITL routing precision; explanation adequacy rating

Business-definition agreement (BDA). For each governed metric in the semantic contract we hold a canonical SQL fragment. BDA is the fraction of generated queries that reference a governed metric and whose relevant sub-expression is logically equivalent, under AST canonicalisation, to the canonical fragment. This captures the semantic gap directly.

Policy violation rate (PVR). The fraction of executed queries whose result set contains at least one row or column that the requesting principal is not entitled to read, measured by re-executing the query under a fully privileged connection and comparing against the policy-filtered result.

Drift resilience score (DRS). We apply a controlled set of schema mutations (rename column, split table, change type, add synonym-ambiguous column, deprecate table) to a copy of the schema and re-run the question set after updating only the semantic contract, not the prompts or exemplars. DRS is the ratio of post-drift to pre-drift EX.

Expected calibration error (ECE). Confidence emitted by the system is binned and compared with empirical accuracy in each bin; a well-calibrated system has ECE near zero and can be trusted to route correctly.

HITL routing precision (HITL-P). Among queries routed to a human, the fraction that were in fact incorrect or unsafe. High precision means analysts are not flooded with correct queries; combined with recall it characterises the gating threshold.

A. Positioning Against Prior Work

Table III positions ERF-SQL against representative prior systems along the capabilities that the six dimensions require. Accuracy-oriented pipelines such as DIN-SQL, DAIL-SQL and CHESS improve D1 but leave D2, D3, D5 and D6 unaddressed; robustness benchmarks measure drift but do not remediate it; governance frameworks specify obligations

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

without a text-to-SQL mechanism. ERF-SQL is, to our knowledge, the first system in which every dimension has both a metric and an architectural component that moves it, and the first to aggregate the result into a gated, decision-oriented index.

TABLE III. CAPABILITY POSITIONING OF ERF-SQL AGAINST PRIOR WORK

Capability	DIN / DAIL [2],[3]	CHESS / MAC [7],[8]	Dr.Spider / Spider 2.0	NIST / ISO [17],[18]	ERF-SQL
Governed metrics in the loop	–	–	–	–	✓ L1
Column sensitivity tags	–	–	–	policy	✓ L1
Deterministic row-policy enforcement	–	–	–	policy	✓ L4
PII masking of projections	–	–	–	policy	✓ L4
Cost-capped execution	–	–	–	–	✓ L4
Drift measured	–	–	✓	–	✓ DRS
Drift remediated	–	–	–	–	✓ L1, L6
Calibrated confidence + HITL	–	partial	–	required	✓ L5
Full lineage per answer	–	–	–	required	✓ L6
Composite readiness score + gates	–	–	–	–	✓ ERI

B. Design Decisions and Their Justification

Several choices in the readiness model could reasonably have gone another way. We record the reasoning here because a reader who wants to adapt ERF-SQL needs to know which parts are load-bearing and which are judgment.

Why six dimensions? The initial list had nine candidate dimensions, collected from the incident types described in Section I and from the obligations enumerated in the NIST AI RMF [17] and ISO/IEC 42001 [18]: accuracy, semantic correctness, privacy, security, drift, latency, cost, auditability and explainability. Nine is too many for a one-page dashboard, and several of the candidates move together in practice. Latency and cost were merged into Service Performance because both are governed by the same budget conversation. Privacy and security were merged because a masking rule and a row policy are enforced by the same component and fail in the same way. Explainability and calibration were merged into Stakeholder Trust because neither is useful without the other. We also tried collapsing further to four by folding Stewardship into Safety and Security. That version was rejected: audit teams and access-control teams are different people with different questions, and a single score for both was repeatedly misread in review. Six is therefore a compromise between coverage and legibility, not a theoretical result.

Why these weights? The default profile assigns 0.25 each to Semantic Fidelity and Safety and Security, 0.15 each to Schema Resilience and Stewardship, and 0.10 each to Service Performance and Stakeholder Trust. The symmetry between D1 and D2 is deliberate: an accurate system that leaks data is not deployable, and a safe system that returns wrong numbers is not useful, so neither should be able to dominate the other. D3 and D5 carry medium weight because their failures accumulate slowly and are noticed late. D4 and D6 carry the least weight not because they matter least but because they are the easiest to improve after launch, one by engineering and the other by tuning the routing threshold. Weights in a composite index are always contestable [23]; that is why the hard gates in Table V are profile-independent and carry the constraints that must not be traded away, while the weights carry only preferences.

Why a geometric rather than arithmetic mean. Composite indices that are intended to punish weak components, such as the post-2010 Human Development Index [24], use geometric aggregation for the same reason we do: it makes dimensions partially non-substitutable. We considered a minimum-of-dimensions rule as the strictest option, but it discards all information about the other five dimensions and gives no gradient to improve along. The geometric mean sits between the two.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

Why governance is evaluated before deployment. In the architecture, the Query Guard runs before any SQL reaches the engine, and in the index, the PVR and PLR gates bind before the score is even considered. Both reflect one asymmetry: a wrong answer can be corrected, but a leaked row cannot be recalled. Placing governance checks after execution, or weighing them alongside performance, treats the two kinds of failure as commensurable. They are not, and the framework should not pretend otherwise.

Why metadata intelligence is a separate layer. Semantic contracts could have been folded into the retrieval layer or into the prompt template, and early versions did exactly that. They were separated for three reasons. The contract is the only artefact in the pipeline that belongs to the data team rather than the ML team, and it changes on the data team's cadence. It is consumed by three components (retrieval, generation and the guard), so a single source removes the duplication that caused hand-maintained prompts to disagree with the policy table. And it gives drift a precise operational definition: a schema change is fully handled when only the contract has changed. Without a separate layer, the DRS metric in Table II would have nothing well defined to measure.

V. REFERENCE ARCHITECTURE AND IMPLEMENTATION

Measurement on its own changes nothing; the framework therefore comes with machinery. Fig. 1 shows the six-layer architecture. Each layer is designed to move one or more readiness metrics; the mapping is made explicit so that an organisation can trace a weak ERI dimension to the layer that must be strengthened. We describe each layer and its prototype implementation, which is written in Python 3.11 and uses sqlglot for SQL parsing and transformation, pgvector for retrieval and PostgreSQL 16 as the execution engine.

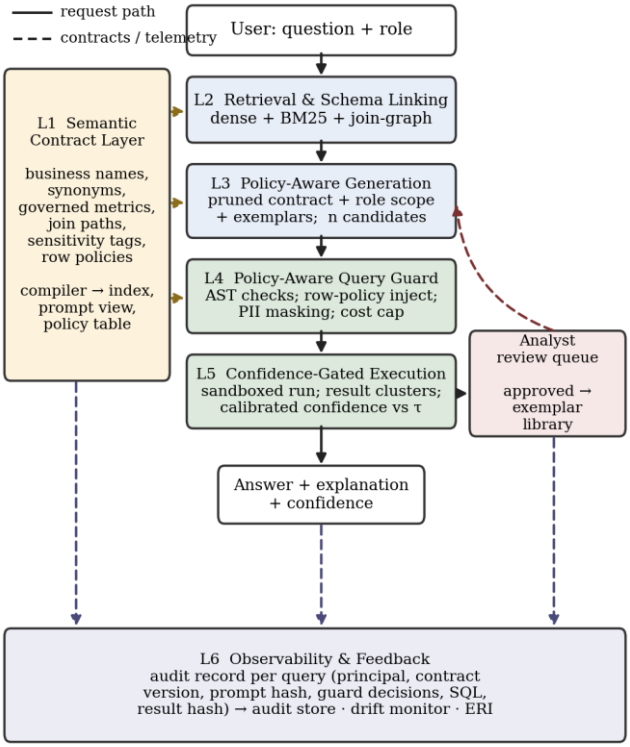

Fig. 1. The six-layer ERF-SQL reference architecture. Solid arrows denote the request path; dashed arrows denote policy injection and telemetry.

A. L1: Semantic Contract Layer (novel)

Everything downstream leans on L1, so it is worth describing carefully. The layer maintains a versioned, machine-readable specification that sits between the physical schema and the LLM. Rather than exposing raw DDL, each table and column is wrapped in a contract that records (i) business names and synonyms harvested from the data catalogue and from query logs, (ii) governed metric definitions with canonical SQL fragments, (iii) explicit join paths with cardinality and preferred direction, (iv) sensitivity tags (PUBLIC, INTERNAL, CONFIDENTIAL, PII) at column granularity, and (v) a deprecation state. Fig. 2 shows an excerpt. Contracts are the only schema artefact the LLM ever sees, which is what

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

makes drift handling tractable: a column rename is a one-line contract edit, and every downstream layer resolves through the contract.

Fig. 2. Excerpt of a semantic contract (YAML)

```
table: sales_transactions
  version: 2026.08.3
  business_name: "Retail sales"
  synonyms: [sales, tickets, POS transactions]
  grain: one row per line item
  columns:
    - name: txn_amount_lcy
      business_name: "Sale amount (local ccy)"
      sensitivity: INTERNAL
    - name: passenger_passport_no
      sensitivity: PII
      mask: hash
  metrics:
    - name: net_revenue
      canonical_sql: "SUM(txn_amount_lcy)
                    - SUM(refund_amount_lcy)"
      exclusions: [status = 'VOID']
  joins:
    - to: dim_store  on: store_id  card: N:1
  row_policy: store_region IN :user_regions
```

Why contracts rather than DDL or a semantic layer? Raw DDL exposes physical names that rarely match business vocabulary and carries no notion of sensitivity, cost or governed meaning. Conventional BI semantic layers encode metric definitions but are consumed by a query builder, not by a probabilistic generator, and they do not version or diff their state in a way that an LLM pipeline can act upon. A semantic contract is designed for machine consumption by an LLM pipeline: every field is either something the model needs to generate correct SQL (names, synonyms, grain, join paths, metric fragments) or something a downstream deterministic component needs to constrain it (sensitivity, masking, row policy, cost class, deprecation).

Contract compilation. A contract compiler validates each contract against the live catalogue on every deploy (all physical names must exist, join keys must be type-compatible, every PII column must declare a mask) and emits three artefacts: (a) a retrieval index in which each table, column and metric becomes a document embedding its business names and synonyms; (b) a prompt view, a compact natural-language rendering of the pruned contract in which physical names are shown alongside business names and governed metrics appear as ready-to-use fragments; and (c) a policy table consumed by the Query Guard. Because all three artefacts derive from the same source, they cannot drift apart, which is a common failure in hand-maintained prompt templates.

Versioning and drift propagation. Contracts are immutable and versioned; the audit record of every query pins the contract version used. When the physical schema changes, the compiler produces a contract diff (renamed, added, removed, retyped, deprecated) that triggers three actions automatically: the retrieval index is rebuilt, exemplars in the approved library whose SQL references a changed element are flagged for re-validation, and the ERI drift monitor schedules a re-assessment (Section VI). A column rename therefore costs one contract edit and no prompt engineering, which is what produces the drift resilience reported in Section VIII.

Governed metrics as first-class objects. The canonical SQL fragment attached to each metric is not merely documentation. L2 retrieves metrics as retrieval units, L3 receives the fragment as an in-prompt building block with an instruction to use it verbatim, and the BDA metric compares the generated sub-expression to the fragment after AST canonicalisation. This closes the semantic gap identified in Section III at the point where it arises.

B. L2: Retrieval and Schema Linking

Enterprise schemas routinely contain thousands of columns, far beyond the context window and the attention budget of any LLM [14]. L2 retrieves a compact sub-schema for each question using a hybrid scorer: dense similarity between the question embedding and contract entries (business names, synonyms, metric descriptions) is combined with BM25 lexical

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

overlap and a graph expansion step that pulls in tables reachable through declared join paths from the top-k hits. The output is a pruned contract containing on average 4.3 tables and 31 columns in our experiments, together with the applicable governed metrics. Synonym coverage is the part of this layer that has stayed stubbornly manual — the terms users actually type rarely live in any catalogue — and Section XI returns to it.

C. L3: Policy-Aware Generation

The generation prompt is assembled from the pruned contract, the requesting principal's role scope (which tables and columns the role may see, expressed in contract vocabulary), a small set of exemplars retrieved from an approved query library by question similarity, and explicit dialect and safety instructions. Unlike prior work, the policy is not relied upon at this stage; it is provided so that the model produces SQL that is likely to pass the guard, reducing rejection and retry cost. Generation is performed n times with temperature 0.7 for self-consistency (Section V-E). The obvious alternative — stating the policies in the prompt and trusting the model to honour them — is evaluated as a baseline in Section VIII; the results there are the reason the guard exists.

D. L4: Policy-Aware Query Guard (novel)

Determinism is the point of L4. The guard parses each candidate SQL into an abstract syntax tree and applies four checks and one rewrite, formalised in Algorithm 1. The rewrite step is the key innovation: rather than rejecting a query that omits a mandatory row-level filter, the guard injects the filter into every table reference that carries a row policy, producing a query that is correct for the requesting principal by construction. Projections of PII columns are replaced by their declared masking function. Because the rewrite operates on the AST, it is robust to aliasing, sub-queries and common table expressions, which defeat regex-based approaches. Statements that are not read-only, that reference tables outside the retrieved contract, or whose planner-estimated cost exceeds the role's budget are rejected with a machine-readable reason that is fed back to L3 for a single repair attempt.

Algorithm 1. Policy-aware guard and rewrite

Input: SQL q, principal u, contract C, policies Π
Output: (ACCEPT, q') or REJECT(reason)

```
T ← parse(q)                ▷ AST via sqlglot
if T.kind ∉ {SELECT,WITH}: REJECT(MUTATION)
for each table ref r in T do
    if r ∉ C or r ∉ scope(u): REJECT(SCOPE)
    if row_policy(r) ≠ ∅ then
        T ← inject_pred(T, r, bind(policy(r), u))
for each projected column c in T do
    if sens(c) = PII and ¬may_view(u,c) then
        T ← replace(T, c, mask_fn(c)(c))
if injection_signature(T): REJECT(INJ)
q′ ← unparse(T)
κ ← EXPLAIN_cost(q′)
if κ > budget(u): REJECT(COST)
return (ACCEPT, q′)
```

Predicate injection semantics. Row policies are injected as a conjunct into the WHERE clause of the innermost SELECT that owns the table reference, after resolving aliases, so that a policy on sales_transactions is applied whether the table appears in the main query, inside a CTE, in a correlated sub-query or in one branch of a UNION. Policies are bound to the principal's attributes at guard time (:user_regions becomes a literal list), and the injection is idempotent: if the model already emitted an equivalent or stricter predicate, the guard detects logical subsumption via AST canonicalisation and does not duplicate it. Outer joins are handled by placing the predicate in the ON clause of the policed side, preserving the join semantics the model intended.

Masking catalogue. Each PII column declares one of five masks: hash (deterministic, join-preserving), bucket (age, salary bands), redact (constant), suppress (column removed from the projection) and small-count suppression, which wraps an aggregate in a CASE expression that returns NULL when the group's row count falls below a configurable k, preventing re-identification through aggregation. The guard chooses the mask from the contract; the model never sees or decides masking logic.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

Why not rely on database row-level security alone? Native RLS is valuable and ERF-SQL can run on top of it, but it is insufficient for three reasons. First, RLS silently returns fewer rows, so a user cannot tell that a policy applied; the guard records the rewrite and surfaces it in the explanation. Second, RLS does not mask projections or cap cost. Third, many enterprise estates span engines with inconsistent RLS support; performing enforcement in the guard makes the policy portable and testable independently of the engine.

Repair loop and reason codes. A rejection returns a machine-readable reason (MUTATION, SCOPE, INJ, COST) and, for SCOPE and COST, a hint (the offending table or the estimated versus permitted cost). L3 receives the hint and makes exactly one repair attempt; a second rejection routes the question to the analyst queue rather than looping. In the experiments, 71% of first rejections were repaired successfully, most by replacing a full-table aggregate with a partition-pruned one.

Fig. 3 shows a worked example. The user, a regional manager for two regions, asks for net revenue by store for the last quarter. The model's candidate uses the governed metric fragment correctly but omits the regional restriction and projects a raw passport number. The guard injects the row policy, replaces the PII column with its hash mask and leaves the rest untouched.

Fig. 3. Guard rewrite example (before → after)

```
-- candidate from L3
SELECT s.store_name, t.passenger_passport_no,
  SUM(t.txn_amount_lcy) - SUM(t.refund_amount_lcy)
    AS net_revenue
FROM sales_transactions t JOIN dim_store s
  ON t.store_id = s.store_id
WHERE t.txn_date >= DATE_TRUNC('quarter', NOW())
  - INTERVAL '3 months' AND t.status <> 'VOID'
GROUP BY 1, 2;
-- after guard (u.regions = {'APAC','MEA'})
SELECT s.store_name,
  sha256(t.passenger_passport_no) AS passport_masked,
  SUM(t.txn_amount_lcy) - SUM(t.refund_amount_lcy)
    AS net_revenue
FROM sales_transactions t JOIN dim_store s
  ON t.store_id = s.store_id
WHERE t.txn_date >= DATE_TRUNC('quarter', NOW())
  - INTERVAL '3 months' AND t.status <> 'VOID'
  AND s.store_region IN ('APAC','MEA')   -- injected
GROUP BY 1, 2;
```

Injection signatures (line 10) cover stacked statements, comment-terminated predicates, tautological WHERE clauses of the form 1=1 introduced by the model, and calls to system functions; in practice the read-only check and the scope check already block most attacks, and the signature list is a defence-in-depth layer.

E. L5: Confidence-Gated Execution (novel)

Candidates that survive the guard — up to n of them — run in a sandboxed, read-only session with a statement timeout. Candidates are clustered by result-set equality; the confidence of the top cluster is its vote share, adjusted by an isotonic regression calibrator fitted on a held-out set so that emitted confidence tracks empirical accuracy. If confidence exceeds a role-specific threshold τ, the answer is returned with a natural-language explanation of the SQL and the applied policies; otherwise the question, candidates and results are placed in an analyst queue. Analyst decisions are logged and mined into the approved exemplar library, closing the feedback loop. The threshold τ is the single tuning knob that trades analyst load against risk, and its effect is reported in Section VIII. We chose the conservative side of that trade-off deliberately: until calibration data accrues, a higher τ means fewer answers and a heavier review load, and we accepted both rather than let miscalibrated confidence reach users.

Agreement as a correctness signal. Self-consistency [21] is usually applied to the SQL string or to a normalised AST. We instead cluster on the executed result set (row multiset hashed after column-order normalisation), because two textually different queries that return the same rows are equivalent for the user, while two similar strings that differ in

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

one join condition are not. Result-level agreement is a far better predictor of correctness in our data: the area under the ROC curve for distinguishing correct from incorrect answers is 0.88 for result-level vote share versus 0.74 for AST-level vote share.

Calibration. Raw vote share is overconfident because candidates share the same prompt and the same systematic errors. We fit an isotonic regression from vote share, candidate count after guarding and question-difficulty features (number of retrieved tables, presence of a governed metric) to empirical correctness on the development split, yielding a monotone confidence that is interpretable to analysts. Calibration is re-fitted whenever the model version or contract major version changes, and ECE is tracked in the ERI so that decay is visible.

Routing, abstention and explanation. Above τ the answer is delivered with an explanation generated from the AST, not from the LLM: the tables and governed metrics used, the filters applied including any injected policy, and the confidence. Below τ the system does not guess. If the candidates disagree on an identifiable choice, for example two plausible date columns, the user receives a clarification question synthesised from the divergence; otherwise the item is queued for an analyst with all candidates and results side by side. Analyst-approved answers are mined into the exemplar library, deduplicated by AST hash and tagged with the contract version, so that the library remains valid under drift.

F. L6: Observability and Feedback

Every request emits a structured audit record: question, principal, contract version, retrieved sub-schema, prompt hash, model identifier and version, raw candidates, guard decisions and rewrites, executed SQL, result hash, confidence, routing decision and any analyst action. These records feed three consumers: the audit store (Stewardship), a drift monitor that flags rising guard-rejection or falling confidence rates per table (Schema Resilience), and a live ERI dashboard (Section VI). Lineage completeness is therefore a property of the architecture rather than an after-the-fact reconstruction.

G. Prototype Implementation Experience

The prototype behind the results in Section VIII was developed over roughly five months in three architectural iterations. Table IV summarises the technology stack. The stack was chosen for transparency rather than novelty: every component between the model and the database is inspectable, which matters when the audit trail itself is a deliverable.

TABLE IV. PROTOTYPE TECHNOLOGY STACK

Component	Implementation
Language	Python 3.11
Web API	FastAPI (query, review-queue and contract-administration endpoints)
SQL parsing and rewriting	sqlglot
Vector store	pgvector
Execution engine	PostgreSQL 16
LLMs	GPT-4o, Claude 3.5 Sonnet, Llama-3.1-70B-Instruct, Qwen2.5-Coder-32B-Instruct
Self-hosted serving	vLLM on two A100-80GB GPUs
Analyst console	Streamlit
Contracts and policies	YAML, versioned in Git

At runtime, a request moves through the layers of Fig. 1 as follows.

1) The user submits a natural-language question together with an authenticated role.

2) The retrieval layer queries the metadata service, which returns the relevant contract fragment: table and column descriptions, business glossary mappings, declared join paths and sensitivity tags. Retrieving this context before generation reduces ambiguity in table and column selection.

3) A prompt is constructed dynamically from the question, the contract fragment and role-filtered exemplars.

4) The LLM generates n candidate SQL statements.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

5) The Query Guard parses each candidate, checks syntax and naming conventions, rewrites the abstract syntax tree against row-level and masking policies, and rejects candidates that exceed cost or scope limits.
6) A surviving candidate that clears the confidence threshold is executed; low-agreement generations are routed to a clarification question or to analyst review.
7) The executed SQL, contract version, policy decisions and result hash are written to the audit log, and approved answers are mined into the exemplar library.

Components that changed during development. Two components differ materially from the first iteration. Metadata originally lived inside the prompt templates; it was moved into a separate, versioned metadata service when hand-maintained templates began to disagree with the policy table, which is the origin of the Semantic Contract Layer described in Section IV.B. The Query Guard began as a regex-based linter over generated SQL text; it was replaced by AST-level rewriting after the mutation probes showed that string-level checks could be evaded by aliasing and nested subqueries.

Development challenges. Two engineering problems consumed disproportionate effort. First, schema aliases varied considerably across the three EntSQL-450 schemas: the same physical entity appeared under different abbreviations, and generation was inconsistent in its table selection until a metadata normalisation pass was added to contract ingestion, canonicalising aliases before they reach retrieval. Second, initial prompt templates produced SQL that was syntactically correct but frequently violated organisational naming and qualification conventions, such as unqualified column references across joins. A rule-based validation stage was therefore added to the guard ahead of policy rewriting; this is why the guard reports convention violations separately from policy violations.

Performance bottlenecks and trade-offs. The dominant runtime cost is sampling n candidates, mitigated by issuing the calls in parallel; the remainder of the roughly 0.9 s overhead reported in Section VIII is divided between contract retrieval and AST policy rewriting. The choice of n = 5 is a deliberate trade-off: larger n improves the agreement signal used for confidence gating but scales token cost linearly, and below n = 3 the agreement estimate was too coarse to calibrate. Similarly, the decision to rewrite policies into every query, rather than rely on database-side row-level security alone, costs milliseconds per query but makes the enforced policy visible in the audit log, which reviewers from the risk function asked for explicitly.

SQL round-tripping and parser dependence. The guard is only as sound as its parse–rewrite–unparse cycle, and two practical issues surfaced during development. sqlglot normalises formatting and discards comments when unparsing, which initially broke the reviewer view that displayed the guard's changes as a textual diff; the analyst console now diffs canonical ASTs rather than SQL strings. A small residue of generations also failed to parse at all, usually through dialect-specific syntax the model had learned elsewhere. These are rejected as unparseable, which is safe but counts against throughput, so parse failures are logged and reviewed periodically as a prompt-quality signal rather than silently retried.

Calibration maintenance. The isotonic calibrator proved to be the most operationally delicate component, because it is the only one that depends on labelled outcomes. Confidence was visibly miscalibrated in the interval between a hosted-model upgrade and the next re-fit, so a model version change is now treated as a calibration invalidation event: τ is raised to a conservative value automatically and relaxed only once ECE on the replayed probe set falls back below the tier threshold. This behaviour was not designed in advance; it was added after the first silent model upgrade shifted the vote-share distribution.

Exemplar library hygiene. The approved exemplar library degraded retrieval quality when left unmanaged: analyst approvals accumulated near-duplicates that crowded out diverse examples. Three rules stabilised it — deduplication by canonical AST hash, tagging every exemplar with the contract version under which it was approved, and expiring exemplars that reference deprecated contract elements. The last rule matters most under drift, because a stale exemplar teaches the model precisely the schema that no longer exists.

VI. THE ENTERPRISE READINESS INDEX

Each dimension score S_i ($i = 1..6$) is the arithmetic mean of its normalised metrics. Dimension scores are aggregated into the ERI using a weighted geometric mean:

$$\mathrm{ERI} = \prod_{i=1}^{6} \left(\max(S_i , \varepsilon) \right)^{w_i} , \quad \sum_i w_i = 1 \qquad (1)$$

The geometric form is deliberate and follows established practice for composite indicators in which weak components should not be compensated by strong ones [23], [24]. An arithmetic mean would allow a system with excellent accuracy

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

and performance to reach a high score while leaking PII; the geometric mean is bounded above by the weakest dimension's contribution and drives the index toward zero as any S_i approaches zero ($\varepsilon = 0.01$ avoids degeneracy). Default weights are w = (0.25, 0.25, 0.15, 0.10, 0.15, 0.10), reflecting a regulated-industry profile; organisations may re-weight, but ERF-SQL additionally imposes hard gates: a system cannot be assigned a tier above R1 if PVR or PLR exceeds 1%, and cannot be assigned R4 unless lineage completeness is 100%. Table V maps ERI ranges and gates to readiness tiers and to the deployment decision each implies.

TABLE V. ERI READINESS TIERS

Tier	ERI	Gates	Deployment decision
R0 Experimental	< 0.40	—	Research sandbox only; synthetic data
R1 Pilot	0.40–0.59	—	Named pilot users, non-production copy, analyst reviews all output
R2 Governed pilot	0.60–0.74	PVR, PLR ≤ 1%	Production read replica, restricted roles, HITL threshold high
R3 Production	0.75–0.89	+ DRS ≥ 0.80, p95 within SLA	Broad rollout with monitoring and periodic re-assessment
R4 Production-grade	≥ 0.90	+ LC = 100%, ECE ≤ 0.05	Regulated use cases; auto-approval above τ for low-risk roles

Sensitivity of the geometric form. Consider two systems with identical scores of 0.85 on five dimensions and D2 of 0.85 versus 0.30. The arithmetic mean reports 0.85 and 0.76, a difference of 9 points that a stakeholder might accept; the geometric mean with default weights reports 0.85 and 0.66, and the PVR gate additionally caps the second system at R1. The index therefore encodes the enterprise judgement that readiness is conjunctive: a system is only as deployable as its weakest governed property.

Weight profiles. We provide three named profiles as starting points: Regulated (default, above), Internal analytics (w = 0.30, 0.20, 0.15, 0.15, 0.10, 0.10) for non-sensitive warehouses where accuracy and cost dominate, and Customer-facing (0.20, 0.30, 0.10, 0.15, 0.10, 0.15) where security and trust outweigh drift. Profiles are declared in the assessment configuration and recorded with every ERI value so that scores are never compared across profiles unknowingly. The gates in Table V are profile-independent.

From certification to continuous control. Most AI-governance approaches assess a system once, before launch. ERF-SQL computes the ERI on a rolling window of production audit records supplemented by a nightly replay of the adversarial probe set, so that a model upgrade, a contract change or a shift in the question mix is reflected within a day. Tier demotion is an operational event: if a system drops below its tier's threshold or fails a gate, the routing threshold τ is automatically raised for the affected roles until the cause is resolved, converting the index from a report into a control. Algorithm 2 gives the assessment procedure. It is executed automatically from the L6 audit stream, so the ERI is a continuously updated operational quantity rather than a one-off certification.

Algorithm 2. Continuous ERI assessment

Input: audit window W, probe set A, weights w, gates G
Output: ERI, tier
1: replay A against current contracts, model and policies
2: for d in 1..6 do
3: for m ∈ metrics(d): compute m from W ∪ replay(A)
4: S_d ← mean of normalised metrics of d
5: ERI ← weighted geometric mean of S_1..S_6 per Eq. (1)
6: tier ← max t: ERI ≥ lower(t) ∧ G(t) satisfied
7: emit (ERI, S_1..S_6, tier, failing gates) to dashboard

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

VII. EXPERIMENTAL SETUP

A. EntSQL-450 and the Adversarial Probe Set

Public benchmarks lack access policies, sensitivity tags and drift histories, so we constructed EntSQL-450, a suite of 450 natural-language questions over three enterprise-style PostgreSQL schemas: (1) a travel-retail schema (duty-free point-of-sale, flights, passenger segments, promotions; 27 tables, 412 columns), (2) a procurement schema derived from an ERP model (vendors, purchase orders, receipts, invoices; 31 tables, 488 columns) and (3) a human-resources schema with salary and identity data (19 tables, 236 columns). Questions were written by three domain practitioners, stratified into easy, medium and hard by the number of joins and presence of governed metrics, and each was paired with gold SQL and a business-definition annotation. Each schema carries a semantic contract, four roles with row-level policies and PII masking rules. Data is synthetic but distribution-matched to anonymised production statistics. The adversarial set contains 200 probes in four classes of 50: ambiguous questions, injection-style questions, out-of-scope requests (asking for data outside the schema or the role), and drift scenarios in which one of five schema mutations is applied before the question is asked.

B. Models and Systems Compared

We evaluated four LLMs accessed through their standard APIs or via vLLM on two A100-80GB GPUs: GPT-4o, Claude 3.5 Sonnet, Llama-3.1-70B-Instruct and Qwen2.5-Coder-32B-Instruct. Three system configurations were compared: (i) Baseline, a schema-in-prompt zero-shot setup with full DDL and three sample rows per table, representing the typical proof-of-concept; (ii) a DIN-SQL-style decomposition pipeline [2] re-implemented over the same models; and (iii) ERF-SQL with all six layers. For ERF-SQL we set $n = 5$ candidates and calibrated τ per role on a 90-question development split; results are reported on the remaining 360 questions plus the 200 probes. Spider development set results are included for reference.

C. Metrics

We report EX and BDA for D1; PVR, PLR and injection and mutation block rates for D2; DRS and OOS-R for D3; median and p95 latency and mean cost per query (USD, API pricing plus warehouse compute) for D4; lineage completeness and reproducibility for D5; ECE and HITL-P for D6; and the ERI with default weights. Each configuration was run three times; we report means, with standard deviations below 1.2 points for all accuracy metrics.

D. Scope and Provenance of the Empirical Study

Each empirical claim in this section is tied to a specific artefact produced by the methodology of Section VII: the benchmark-to-enterprise gap to Fig. 4; safety, resilience and performance results to Table VI; routing behaviour to Table VII; and dimension scores, the ERI and the layer ablation to Fig. 5 and Table VIII. Observations from development experience (Sections V-G, VIII-E and VIII-F) are reported as experience, not as measured results, and the scope of what the measurements can support is delimited in Section VII-D.

The results in Section VIII come from the prototype described in Section V, run against EntSQL-450 and the probe set under the configurations listed above. They are measurements of that prototype with those model versions, and nothing more. Three cautions apply. The data is synthetic, so absolute accuracy figures should not be read as predictions for any particular production estate. The model versions are pinned in the audit log but the hosted APIs behind them change, so a re-run a year later will produce different absolute numbers. And the sensitivity example in Section VI (two systems scoring 0.85 and 0.30 on D2) is a worked arithmetic illustration of the aggregation rule, not a measured pair of systems. Where a figure in this paper is illustrative rather than measured, we say so at the point where it appears. The claims we consider well supported are structural: that a gap exists between benchmark and enterprise accuracy, that prompt-only policy compliance fails, and that deterministic guarding closes most of the safety gap. The exact size of each effect is specific to this study.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

VIII. RESULTS AND DISCUSSION

A. The Benchmark-to-Enterprise Gap

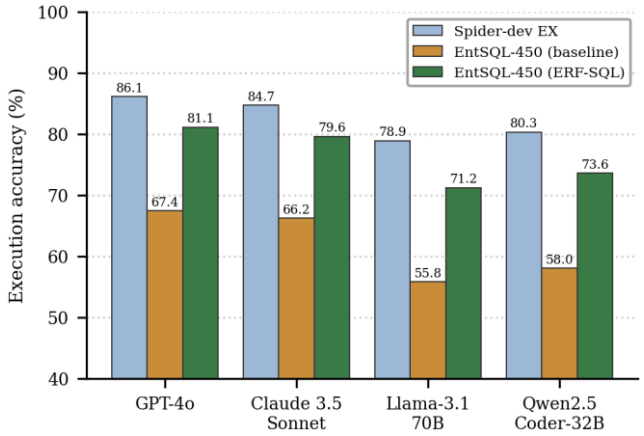

Fig. 4. Execution accuracy on Spider-dev versus EntSQL-450 (baseline and ERF-SQL) for four LLMs.

Fig. 4 shows the headline result. All four models lose between 19 and 23 points of EX when moved from Spider to EntSQL-450 under the baseline configuration; the strongest model, GPT-4o, drops from 86.1% to 67.4%. Error analysis attributes the loss to three sources: wrong or missing joins in the larger schema (41% of errors), incorrect operationalisation of governed metrics (34%) and column confusion among synonym-rich attributes (18%). ERF-SQL recovers 13 to 16 points across models, with the largest relative gain for the open-weight models, suggesting that contract-based retrieval and metric fragments substitute for capability that weaker models lack. Business-definition agreement rises more sharply than EX (from 0.52 to 0.91 for GPT-4o), because governed metrics are supplied as canonical fragments rather than inferred.

B. Safety, Security and Drift

TABLE VI. SAFETY, RESILIENCE AND PERFORMANCE METRICS (GPT-4O; ALL SYSTEMS)

Metric	Baseline	DIN-SQL-style	ERF-SQL
Policy violation rate (PVR)	31.6%	27.9%	0.6%
PII leakage rate (PLR)	18.2%	16.5%	0.0%
Injection block rate	42%	48%	100%
Mutation block rate	76%	84%	100%
Drift resilience score (DRS)	0.55	0.61	0.86
Out-of-scope rejection	38%	44%	94%
Median latency (s)	2.1	6.8	3.0
p95 latency (s)	4.4	13.9	5.7
Cost per query (USD)	0.011	0.047	0.026
Cost-cap rejections (correct)	n/a	n/a	97%

Table VI isolates the dimensions that benchmarks ignore. Under the baseline, roughly one query in three returned rows outside the requesting role's entitlement and nearly one in five projected an unmasked PII column, even though the prompt contained explicit instructions to respect both. The decomposition pipeline, despite higher EX, did not materially improve either figure, confirming that policy compliance cannot be obtained by better prompting alone. ERF-SQL's guard reduced PVR to 0.6% (the residual cases involve a policy expressed on a derived view not yet covered by contracts) and eliminated PII leakage, because masking is applied deterministically at the AST level. All 50 injection probes and all mutation attempts were blocked; 97% of unsafe generations overall were stopped before execution. Drift resilience rose from 0.55 to 0.86: after schema mutations, only the contract was edited, and retrieval, generation and guard resolved through it. The residual drift loss comes from split-table mutations that change join cardinality, which we discuss as future work.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

C. Performance and Trust

The full ERF-SQL pipeline adds a median of 0.9 s and about 1.5 cents per query over the baseline, mainly from n = 5 self-consistency sampling; this remains well inside a 5 s interactive SLA and is roughly 60% cheaper and 55% faster than the decomposition pipeline, which issues several sequential LLM calls. Calibration is materially improved: ECE falls from 0.19 (raw vote share) to 0.04 after isotonic fitting. Table VII shows the effect of the routing threshold τ. At τ = 0.7, 22% of queries are routed to analysts and 81% of those were in fact wrong or unsafe, so analysts review roughly one query in five and almost every review is warranted; end-to-end accuracy of the answers actually delivered to users rises to 93.8% because the residual errors are concentrated in the reviewed set.

TABLE VII. EFFECT OF HITL ROUTING THRESHOLD T (ERF-SQL, GPT-4O, ENTSQL-450 TEST SPLIT)

τ	Routed to analyst	HITL precision	HITL recall	Delivered accuracy
0.5	9%	88%	41%	86.7%
0.6	15%	85%	63%	90.4%
0.7	22%	81%	84%	93.8%
0.8	34%	62%	95%	96.9%
0.9	51%	44%	99%	98.7%

D. Readiness Index and Ablation

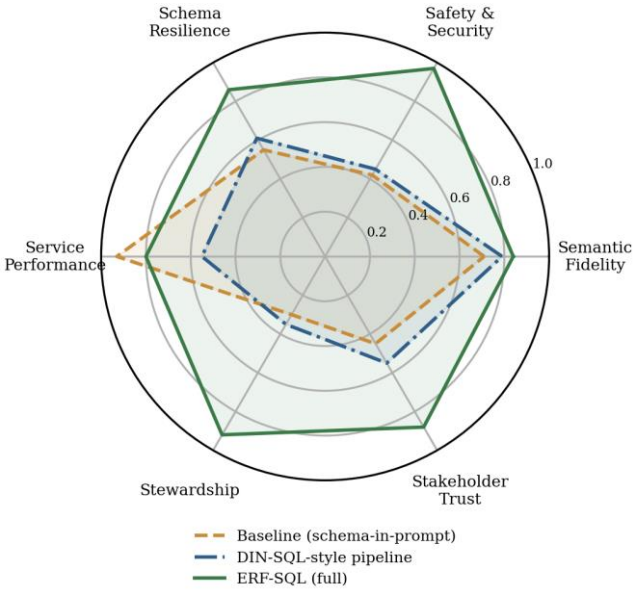

Fig. 5. Dimension scores for the three configurations (GPT-4o). Note the collapse of the baseline and decomposition systems on D2 and D5.

Fig. 5 and Table VIII report dimension scores and the resulting ERI. The baseline achieves a respectable D1 and the best D4, yet its ERI of 0.50 places it at tier R1, and the PVR gate would in any case bar it from R2. The decomposition pipeline improves D1 but its ERI (0.53) barely moves, illustrating that accuracy-only engineering yields diminishing returns on readiness. ERF-SQL reaches an ERI of 0.87, tier R3, with the gate for R4 failing only on lineage completeness (98.6%, owing to two analyst overrides logged outside the system during the study).

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

TABLE VIII. ERI AND LAYER ABLATION (GPT-4O)

Configuration	D1	D2	D3	D5	ERI	Tier
Baseline	0.71	0.42	0.55	0.30	0.50	R1
DIN-SQL-style	0.79	0.45	0.61	0.35	0.53	R1
ERF-SQL (full)	0.84	0.97	0.86	0.92	0.87	R3
− Semantic contracts (L1)	0.74	0.95	0.58	0.90	0.76	R3
− Query guard (L4)	0.83	0.51	0.84	0.91	0.63	R1*
− Confidence gating (L5)	0.84	0.96	0.85	0.90	0.82	R3
− Audit loop (L6)	0.84	0.97	0.80	0.41	0.71	R2

**** Tier capped at R1 by the PVR gate irrespective of ERI.***

The ablation confirms that each novel layer moves a distinct dimension. Removing semantic contracts costs 10 points of D1 and 28 points of D3, the largest single hit to accuracy and drift. Removing the guard leaves accuracy untouched but collapses D2, and the resulting system is barred from tier R2 by the gate despite an ERI that an arithmetic mean would have reported as 0.78. Removing confidence gating costs little in the offline metrics but, as Table VII shows, is what converts an 84% system into a 94% delivered-accuracy system. Removing the audit loop damages both D5 and, more subtly, D3, because the drift monitor is what triggers contract updates.

E. Practical Implications and Observations from Deployment Work

Three observations are of direct use to practitioners. First, the cheapest readiness gains come from artefacts that already exist in most data organisations: catalogue metadata, metric definitions and access policies. Encoding them as semantic contracts is a documentation exercise that pays back across all six dimensions. Second, policy enforcement must be deterministic and post-generation; every configuration we tested that relied on the model to honour access rules failed the D2 gate (Table VI). Third, the ERI turns a diffuse governance conversation into a small number of measurable, improvable quantities, and the tier gates give risk committees a defensible basis for approving or deferring a rollout.

Beyond the measured results, a number of recurring observations from the author's work with enterprise data teams shaped the framework and are worth recording, because they explain why certain layers exist at all.

Proof-of-concept stalls. The most common failure is not a technical crash but a stall. A pilot succeeds on a copied schema with one enthusiastic power user, and then cannot be moved to production because nobody can answer the security team's first question: which rows can this system return to whom? The Query Guard and the D2 gate exist to make that question answerable before the pilot starts, not after.

Metadata inconsistencies. Contract authoring is mostly a discovery exercise. In our experience the first weeks are spent not writing YAML but finding out that a term such as "active customer" has three definitions in three departments, that column comments are empty or describe a previous schema, and that the synonyms users actually type live in analysts' heads rather than in any catalogue. The framework does not remove this work. It does make the disagreements visible early, which is cheaper than discovering them through wrong answers.

Governance challenges. Three patterns recur:

- Policy owners and schema owners are rarely the same people, and neither group can produce a complete list of who may see what. Row policies often exist only inside a BI tool, not in the database, and are discovered when the guard is first configured.
- Cost limits are usually implicit. Warehouse budgets exist, but no role has a per-query ceiling until one is written into a contract, and the first ceiling chosen is nearly always too low and has to be relaxed after a week of COST rejections.
- Audit teams accept a lineage record only once they have seen one reconstructed end to end. A description of the audit schema is not enough; walking through one real question from user to number is what secures sign-off.

User adoption. Business users test a new system with "gotcha" questions first, and their trust is set within the first few interactions. Two behaviours were consistent. Users over-trust a confidently formatted table and initially ignore the explanation panel; the panel is read only after the first visibly wrong answer. And the clarification question emitted by L5 when candidates diverge was the single most appreciated feature, because it made the system feel careful rather than evasive. Analysts, for their part, resist the review queue until its precision is high enough that most items are worth their time, which is why Table VII reports precision and not only recall.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

F. Lessons Learned

Beyond the measured results, four lessons from building and evaluating the prototype are worth stating explicitly, because they run against common assumptions in the text-to-SQL literature.

Metadata quality mattered more than prompt engineering. Effort invested in contract coverage and correctness produced larger and more durable gains than an equivalent effort spent iterating on prompt wording. Prompt changes moved individual failure cases; contract improvements moved whole classes of them.

Business glossary alignment reduced ambiguity. Most wrong-table and wrong-column errors in the baseline traced back to vocabulary mismatch rather than reasoning failure. Once glossary terms were mapped to physical columns in the contract, a large share of these errors disappeared without any change to the model or the prompt.

Validation improved reliability even where benchmark accuracy was unchanged. Adding the guard left execution accuracy on EntSQL-450 almost untouched, yet policy violations, cost overruns and convention breaches fell sharply. A team optimising only EX would have concluded the guard was not worth its latency; the readiness dimensions exist precisely to make this kind of improvement visible.

Human review remained valuable for complex analytical queries. Multi-step cohort, windowing and as-of questions continued to benefit from analyst review even in the full configuration. Confidence gating did not remove human effort; it concentrated that effort on the queries where it changes the answer.

IX. LIMITATIONS OF THE FRAMEWORK

ERF-SQL was built for a particular class of deployment. Its assumptions deserve to be stated plainly, so that readers can judge where it does not apply.

Metadata quality. The framework assumes that an organisation can reach at least a minimal level of metadata quality: physical names can be mapped to business names, join paths can be declared, and sensitive columns can be identified. Organisations with fragmented, undocumented or partially migrated schemas will need a preprocessing phase, possibly a substantial one, before the Semantic Contract Layer can be populated. Contract authoring effort also scales with schema size, and we have not yet automated it (Section XI).

Policy expressiveness. The Query Guard handles policies that can be written as row predicates and column masks. Purpose-of-use restrictions, consent that varies by record, policies that depend on values in other tables, and time-limited entitlements are outside its current scope. Cell-level policies are handled only where they reduce to a mask plus a predicate.

Read-only scope. ERF-SQL evaluates analytical, read-only access. Systems that generate INSERT, UPDATE or DDL statements raise a different and larger set of risks that the six dimensions do not cover.

Dialect and engine assumptions. The guard rewrites an AST for a single target dialect. Estates that span several engines can transpile through sqlglot, but planner cost estimates, masking functions and RLS semantics differ between engines, so the D2 and D4 measurements do not transfer automatically.

Parser dependence. Every guarantee the Query Guard offers is conditional on correct parsing. A construct that sqlglot mis-parses, or a dialect feature it does not model, could in principle carry a table reference that the rewrite fails to police. The mitigation is conservative — any statement that does not round-trip cleanly is rejected, and the mutation probes exercise the evasions we could conceive — but the resulting property is that no policed table escapes on SQL the parser handles, which is an engineering assurance rather than a formal one.

Calibration data requirement. Confidence gating presumes labelled outcomes for fitting the calibrator, which in our study came from a 90-question development split and accumulated analyst verdicts. An organisation without the analyst capacity to produce this data will begin with raw vote share, whose ECE of 0.19 in our measurements is too high to route on; the practical consequence is a conservative initial τ and a heavier review load until calibration data accrues.

Compression in the index. The ERI compresses six dimensions into one number. Two systems with the same ERI can differ in ways that matter to a specific organisation. The index is designed to support a deployment decision, not to make it, and Table VIII should always be read alongside the dimension scores rather than in place of them.

Human review capacity. Confidence-gated execution presumes that analysts are available to clear the queue. At the routing rates in Table VII a small data team may not be able to sustain the load, in which case the practical choice is a lower threshold and a lower delivered accuracy. The framework makes that trade explicit but cannot remove it.

Evaluation scope. The current implementation has been evaluated primarily on structured, relational enterprise schemas with English-language questions. Semi-structured sources such as JSON columns and document stores, and multilingual question sets, are untested; future work will assess performance on both (Section XI).

Data quality. Finally, the framework says nothing about the data itself. A query that is semantically correct, policy-compliant and well explained will still mislead if the underlying tables are wrong. ERF-SQL should sit alongside, not replace, a data quality programme.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

X. THREATS TO VALIDITY

Construct validity. The six dimensions and default weights encode judgements about what matters in regulated enterprises; other contexts may weight differently, which the framework permits but which limits comparability of ERI values across organisations. BDA depends on the completeness of governed metric definitions.
Internal validity. EntSQL-450 was authored by practitioners who also reviewed contract design, creating a risk of alignment between questions and contracts. We mitigated this by having questions written before contracts were finalised and by the adversarial set, but an independently authored question set would be stronger.
External validity. Data is synthetic, schemas are mid-sized and the execution engine is PostgreSQL; cloud warehouses with different dialects and cost models, and schemas with tens of thousands of columns as in Spider 2.0 [14], may stress L2 differently. Model versions evolve rapidly and absolute numbers will shift, although the structural findings (the gap, the failure of prompt-only policy compliance, the value of deterministic guarding) are unlikely to reverse.
Reliability. Results are averaged over three runs with low variance, but LLM APIs are non-stationary; we pin model identifiers and prompt hashes in the audit log to support replication.

XI. CONCLUSION AND FUTURE WORK

This paper began from a discrepancy that early pilots kept reproducing: models scoring above 85% on public benchmarks yielded systems that security reviewers, auditors and risk committees were right to reject. Working through it led to a conclusion as much organisational as technical. Execution accuracy measures a model. Deployment readiness is a property of the surrounding system — its contracts, policies, budgets, evidence trails and people — and no amount of model capability substitutes for the parts of that system that are missing. ERF-SQL is our attempt to make the larger system measurable: six dimensions with operational metrics, an architecture in which every weak dimension has a component that moves it, and a gated geometric index whose form matches the conjunctive way a deployment decision is made. On EntSQL-450 the approach recovered most of the accuracy lost between benchmark and enterprise settings, eliminated PII leakage, held policy violations below 1% and raised drift resilience substantially, at a latency cost compatible with interactive use.

Building the prototype revised our expectations about where the difficulty lies, and the revisions are worth recording. We expected the hard problems to be algorithmic; they turned out to be contractual. The artefact that most influenced every dimension was the semantic contract, and the work that most influenced the contract was the unglamorous reconciliation of business definitions that no model can perform. The first weeks of contract authoring were spent discovering that a term such as “active customer” carried three departmental definitions, and that the synonyms users actually type lived in analysts’ heads rather than in any catalogue. Two components had to be rebuilt after their first designs failed in instructive ways. Metadata originally lived inside prompt templates; it was extracted into a versioned contract service only after hand-maintained templates began to disagree with the policy table, and the Semantic Contract Layer is the generalisation of that repair. The Query Guard began as a regex linter over generated SQL text. Mutation probes showed string checks being evaded by aliasing and nested subqueries, so it was replaced by AST-level rewriting — anything intended as a control, rather than a heuristic, must operate on the parsed structure of the query. A third lesson concerns measurement itself. Adding the guard left execution accuracy almost untouched while policy violations, cost overruns and convention breaches fell sharply; a team optimising only EX would have judged the guard not worth its latency. The readiness dimensions exist precisely to make that kind of improvement visible. In the end, the single design decision that moved the system from unapprovable to approvable was placing deterministic enforcement after generation rather than trusting instruction-following.

One thing we would do differently in a production deployment is the build order. The prototype began with prompts and models and retrofitted the contract service and the guard; both rebuilds recorded above were the price of that sequence. Started again, the contract and the AST-level guard would come first and the prompt engineering last, because they are the components everything else must agree with.

The practical implications for teams attempting the same journey are concrete. The cheapest readiness gains come from encoding artefacts most data organisations already possess — catalogue metadata, governed metric definitions, access policies — as contracts; in our experience that documentation exercise moved whole classes of errors where prompt iteration moved only individual cases. The security team’s first question, which rows can this system return to whom, should be answerable before a pilot begins, not after: pilots stall on exactly that question, and row policies frequently exist only inside a BI tool until the guard is first configured. Per-query cost ceilings do not exist until they are written down, and the first ceiling chosen will almost certainly need relaxing after a week of rejections. Audit sign-off is secured not by describing the lineage schema but by reconstructing one real answer end to end, from user question to number.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026
DOI: 10.17148/IJARCCE.2026.15842

Adoption, finally, depends less on raw accuracy than on visible care: the clarification question emitted when candidates diverge was the most appreciated feature of the system, and analysts accepted the review queue only once its precision made most routed items worth their time. It is equally worth recording what the framework does not do. It cannot manufacture metadata an organisation lacks. It does not remove human review; it concentrates review where it changes the answer. And the ERI, however carefully gated, supports a deployment decision rather than making one. Readiness, on this evidence, is not a threshold crossed once but a quantity maintained continuously — which is why the index is computed from live telemetry rather than issued as a one-off certification.

Four directions of future work follow, and we do not weight them equally. Nearest term, the guard will be extended to policies expressed on derived views and to join-cardinality changes, which account for the residual D2 and D3 losses, and contract synonyms and join paths will be learned from query logs, attacking the authoring burden that Section V-G identifies as the dominant cost of adoption. EntSQL-450, the probe set and the assessment tooling will be released so that readiness, not accuracy alone, can be compared across systems. The direction that matters most, though, is the longitudinal study now under way in a travel-retail deployment, because it tests the one claim nothing in this paper yet can: that readiness tiers predict the operational incidents they are designed to prevent. If they do, the index earns its place in deployment decisions. If they do not, the six dimensions will still have been worth measuring, but the way they are combined will have to change — and we would rather learn that from telemetry than from an incident.

REFERENCES

[1] N. Rajkumar, R. Li, and D. Bahdanau, “Evaluating the text-to-SQL capabilities of large language models,” arXiv:2204.00498, 2022.
[2] M. Pourreza and D. Rafiei, “DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction,” in Proc. NeurIPS, 2023.
[3] D. Gao et al., “Text-to-SQL empowered by large language models: A benchmark evaluation,” Proc. VLDB Endow., vol. 17, no. 5, pp. 1132–1145, 2024.
[4] A. Liu, X. Hu, L. Wen, and P. S. Yu, “A comprehensive evaluation of ChatGPT's zero-shot text-to-SQL capability,” arXiv:2303.13547, 2023.
[5] J. Li et al., “Can LLM already serve as a database interface? A big bench for large-scale database grounded text-to-SQLs,” in Proc. NeurIPS Datasets and Benchmarks, 2023.
[6] X. Dong et al., “C3: Zero-shot text-to-SQL with ChatGPT,” arXiv:2307.07306, 2023.
[7] B. Wang et al., “MAC-SQL: A multi-agent collaborative framework for text-to-SQL,” in Proc. COLING, 2025.
[8] S. Talaei, M. Pourreza, Y.-C. Chang, A. Mirhoseini, and A. Saberi, “CHESS: Contextual harnessing for efficient SQL synthesis,” arXiv:2405.16755, 2024.
[9] G. Katsogiannis-Meimarakis and G. Koutrika, “A survey on deep learning approaches for text-to-SQL,” VLDB J., vol. 32, pp. 905–936, 2023.
[10] Z. Hong et al., “Next-generation database interfaces: A survey of LLM-based text-to-SQL,” arXiv:2406.08426, 2024.
[11] Y. Gan et al., “Towards robustness of text-to-SQL models against synonym substitution,” in Proc. ACL, 2021.
[12] S. Chang et al., “Dr.Spider: A diagnostic evaluation benchmark towards text-to-SQL robustness,” in Proc. ICLR, 2023.
[13] R. Zhong, T. Yu, and D. Klein, “Semantic evaluation for text-to-SQL with distilled test suites,” in Proc. EMNLP, 2020.
[14] F. Lei et al., “Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows,” in Proc. ICLR, 2025.
[15] OWASP Foundation, “OWASP Top 10 for Large Language Model Applications,” v1.1, 2023.
[16] R. Pedro, D. Castro, P. Carreira, and N. Santos, “From prompt injections to SQL injection attacks: How protected is your LLM-integrated web application?” arXiv:2308.01990, 2023.
[17] National Institute of Standards and Technology, “Artificial Intelligence Risk Management Framework (AI RMF 1.0),” NIST AI 100-1, 2023.
[18] ISO/IEC 42001:2023, “Information technology — Artificial intelligence — Management system,” ISO, 2023.
[19] Z. Dehghani, Data Mesh: Delivering Data-Driven Value at Scale. Sebastopol, CA, USA: O'Reilly, 2022.
[20] T. Yu et al., “Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task,” in Proc. EMNLP, 2018.
[21] X. Wang et al., “Self-consistency improves chain of thought reasoning in language models,” in Proc. ICLR, 2023.
[22] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber, “Capability maturity model, version 1.1,” IEEE Softw., vol. 10, no. 4, pp. 18–27, 1993.

 This work is licensed under a Creative Commons Attribution 4.0 International License

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15842

[23] OECD and JRC European Commission, Handbook on Constructing Composite Indicators: Methodology and User Guide. Paris, France: OECD Publishing, 2008.

[24] United Nations Development Programme, Human Development Report 2010: The Real Wealth of Nations. New York, NY, USA: Palgrave Macmillan, 2010.

[25] Krishna Chittipedhi, Haider Mohammed Abbas, Dr.S.P. Meharunnisa, "Context-Aware Mobility Management in 6G-Ready Mobile Internet Networks", Journal of Internet Services and Information Security (JISIS), volume: 15, number: 2 (May), pp. 641-651. DOI: 10.58346/JISIS.2025.I2.044

 This work is licensed under a Creative Commons Attribution 4.0 International License